CHAPTER

PROBABILISTIC GRAPHICAL MODELS: PART II 16

CHAPTER OUTLINE

16.1 Introduction 795
16.2 Triangulated Graphs and Junction Trees 796
16.2.1 Constructing a Join Tree 799
16.2.2 Message-Passing in Junction Trees 801
16.3 Approximate Inference Methods 804
16.3.1 Variational Methods: Local Approximation 804
Multiple-Cause Networks and the Noisy-OR Model 805
The Boltzmann Machine 807
16.3.2 Block Methods for Variational Approximation 809
The Mean Field Approximation and the Boltzmann Machine 810
16.3.3 Loopy Belief Propagation 813
16.4 Dynamic Graphical Models 816
16.5 Hidden Markov Models 818
16.5.1 Inference 821
16.5.2 Learning the Parameters in an HMM 825
16.5.3 Discriminative Learning 828
16.6 BEYOND HMMs: A DISCUSSION 829
16.6.1 Factorial Hidden Markov Models 829
16.6.2 Time-Varying Dynamic Bayesian Networks 832
16.7 Learning Graphical Models 833
16.7.1 Parameter Estimation 833
16.7.2 Learning the Structure 837
Problems 838
References 840

16.1 INTRODUCTION

This is the follow-up to Chapter 15 and it builds upon the notions and models introduced there. The emphasis of this chapter is on more advanced topics for probabilistic graphical models. It wraps up the topic of exact inference in the context of junction trees and then moves on to introduce approximate inference techniques. This establishes a bridge with Chapter 13. Then, dynamic Bayesian networks are introduced with an emphasis on hidden Markov models (HMM). Inference and training of HMMs is seen as a special case of the message-passing algorithm and the EM scheme discussed in Chapter 12. Finally, the more general concept of training graphical models is briefly discussed.

16.2 TRIANGULATED GRAPHS AND JUNCTION TREES

In Chapter 15, we discussed three efficient schemes for exact inference in graphical entities of a tree structure. Our focus in this section is to present a methodology that can transform an arbitrary graph into an equivalent one having a tree structure. Thus, in principle, such a procedure offers the means for exact inference in arbitrary graphs. This transformation of an arbitrary graph to a tree involves a number of stages. Our goal is to present these stages and explain the procedure more via examples and less via formal mathematical proofs. A more detailed treatment can be obtained from more specialized sources, for example, [32, 45].

We assume that our graph is undirected. Thus, if the original graph was a directed one, then it is assumed that the moralization step has previously been applied.

Definition 16.1. An undirected graph is said to be *triangulated* if and only if for *every* cycle of length greater than three the graph possesses a *chord*. A chord is an edge joining two *nonconsecutive* nodes in the cycle.

In other words, in a triangulated graph, the largest "minimal cycle" is a triangle. Figure 16.1a shows a graph with a cycle of length $n = 4$ and Figures 16.1b and c show two triangulated versions; note that the process of triangulation does not lead to unique answers. Figure 16.2a is an example of a graph with a cycle of $n = 5$ nodes. Figure 16.2b, although it has an extra edge joining two nonconsecutive nodes, is not triangulated. This is because there still remains a chordless cycle of four nodes ($x_2 - x_3 - x_4 - x_5$). Figure 16.2c is a triangulated version. There are no cycles of length $n > 3$ without a chord. Note that by joining nonconsecutive edges in order to triangulate a graph, we divide it into cliques (Section 15.4); we will appreciate this very soon. Figure 16.1b, c comprise two (three-node) cliques and Figure 16.2 c comprises three cliques. This is not the case with Figure 16.2b, where the subgraph (x_2, x_3, x_4, x_5) is not a clique.

Let us now see how the previous definition relates to the task of variable elimination, which underlies the message-passing philosophy. In our discussion on such algorithmic schemes, we "just" picked a node and marginalized out the respective variable (e.g., in the sum-product algorithm); as a matter of

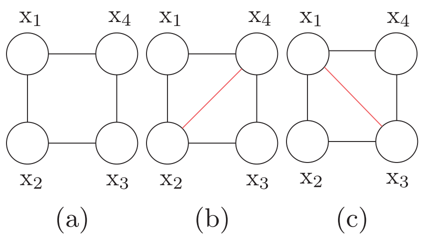


FIGURE 16.1

(a) A graph with a cycle of length $n = 4$. (b), (c) Two possible triangulated versions.

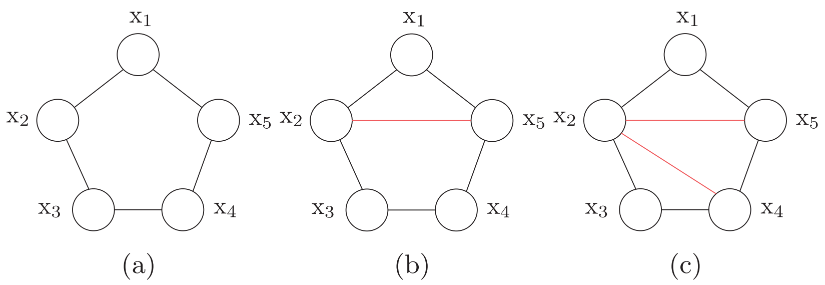


FIGURE 16.2

(a) A graph of cycle of length $n = 5$. (b) Adding one edge still leaves a cycle of length $n = 4$ chordless. (c) A triangulated version; there are no cycles of length $n > 3$ without a chord.

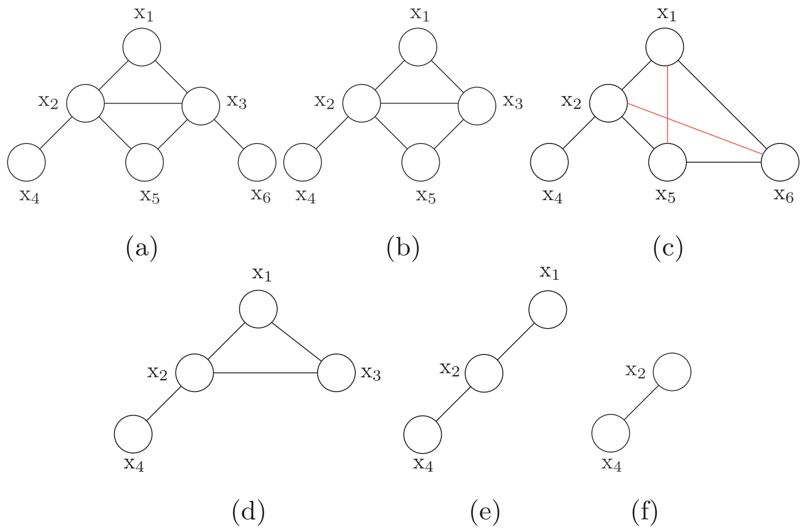


FIGURE 16.3

(a) An undirected graph with potential (factor) functions $\psi_1(x_1)$, $\psi_2(x_1, x_2)$, $\psi_3(x_1, x_3)$, $\psi_4(x_2, x_4)$, $\psi_5(x_2, x_3, x_5)$, $\psi_6(x_3, x_6)$. (b) The graph resulting after the elimination of x_6. (c) The graph that would have resulted if the first node to be eliminated were x_3. Observe the fill-in edges denoted by red. (d), (e), (f) are the graphs that would result if the elimination process had continued from the topology shown in (b) and sequentially removing the nodes: x_5, x_3, and finally x_1.

fact, this is not quite true. The message-passing was initialized at the leaves of the tree graphs; this was done on purpose, although not explicitly stated there. We will soon realize why.

Consider Figure 16.3 and let

$$P(\boldsymbol{x}) := \psi_1(x_1)\psi_2(x_1,x_2)\psi_3(x_1,x_3)\psi_4(x_2,x_4)\psi_5(x_2,x_3,x_5)\psi_6(x_3,x_6), \tag{16.1}$$

assuming that $Z = 1$.

Let us eliminate x_6 first, that is,

$$\sum_{x_6} P(\boldsymbol{x}) = \psi^{(1)}(x_1, x_2, x_3, x_4, x_5) \sum_{x_6} \psi_6(x_3, x_6)$$
$$= \psi^{(1)}(x_1, x_2, x_3, x_4, x_5)\psi^{(3)}(x_3), \tag{16.2}$$

where the definitions of $\psi^{(1)}$ and $\psi^{(3)}$ are self-explained, by comparing Eqs. (16.1) and (16.2). The result of elimination is equivalent with a new graph, shown in Figure 16.3b with $P(\boldsymbol{x})$ given as the product of the same potential functions as before with the exception of ψ_3, which is now replaced by the product $\psi_3(x_1, x_3)\psi^{(3)}(x_3)$. Basically, $\psi^{(3)}(\cdot)$ is the message passed to x_3.

In contrast, let us now start by eliminating x_3 first. Then, we have

$$\sum_{x_3} P(\boldsymbol{x}) = \psi^{(2)}(x_1, x_2, x_4) \sum_{x_3} \psi_3(x_1, x_3)\psi_5(x_2, x_3, x_5)\psi_6(x_3, x_6)$$
$$= \psi^{(2)}(x_1, x_2, x_4)\tilde{\psi}^{(3)}(x_1, x_2, x_5, x_6).$$

Note that this summation is more difficult to perform. It involves four variables (x_1, x_2, x_5, x_6) besides x_3, which requires many more combination terms be computed than before. Figure 16.3c shows the resulting equivalent graph, after eliminating x_3. Due to the resulting factor $\tilde{\psi}^{(3)}(x_1, x_2, x_5, x_6)$ *new* connections implicitly appear, known as *fill-in edges*. This is not a desired situation, as it introduces factors depending on new combinations of variables. Moreover, the new factor depends on four variables, and we know that the larger the number of variables, or the *domain* of the factor as we say, the larger the number of terms involved in the summations, which increases the computational load.

Thus, the choice of the sequence of elimination is very important and far from innocent. For example, for the case of Figure 16.3a an elimination sequence that does not introduce fill-ins is the following: $x_6, x_5, x_3, x_1, x_2, x_4$. For such an elimination sequence, every time a variable is eliminated, the new graph results from the previous one by just removing one node. This is shown by the sequence of graphs in Figures 16.3a, b, d, e, f, for the case of the previously given elimination sequence. An elimination sequence that *does not* introduce fill-ins is known as a *perfect elimination sequence*.

Proposition 16.1. An undirected graph is triangulated if and only if it has a perfect elimination sequence, for example, [32].

Definition 16.2. A tree, T, is said to be a *join tree* if (a) its nodes correspond to the cliques of an (undirected) graph, G, *and* (b) the intersection of *any* two nodes, $U \cap V$ is contained in every node in the *unique* path between U and V. The latter property is also known as the *running intersection property*.

Moreover, if a probability distribution p factorizes over G so that each of the product factors (potential functions) is attached to a clique (i.e., depends only on variables associated with the nodes in the clique) then the join tree is said to be a *junction tree* for p [7].

Example 16.1. Consider the triangulated graph of Figure 16.2c. It comprises three cliques, namely (x_1, x_2, x_5), (x_2, x_3, x_4), and (x_2, x_4, x_5). Associating each clique with a node of a tree, Figure 16.4 presents three possibilities. The trees in Figure 16.4a, b are not join trees. Indeed, the intersection $\{x_1, x_2, x_5\} \cap \{x_2, x_4, x_5\} = \{x_2, x_5\}$ does not appear in node (x_2, x_3, x_4). Similar arguments hold true for the case of Figure 16.4b. In contrast, the tree in Figure 16.4c is a join tree, because the intersection $\{x_1, x_2, x_5\} \cap \{x_2, x_3, x_4\} = \{x_2\}$ is contained in (x_2, x_4, x_5). If, now, we have a distribution such as

$$p(\boldsymbol{x}) = \psi_1(x_1, x_2, x_5)\psi_2(x_2, x_3, x_4)\psi_3(x_2, x_4, x_5)$$

the graph in Figure 16.4c is a junction tree for $p(\boldsymbol{x})$

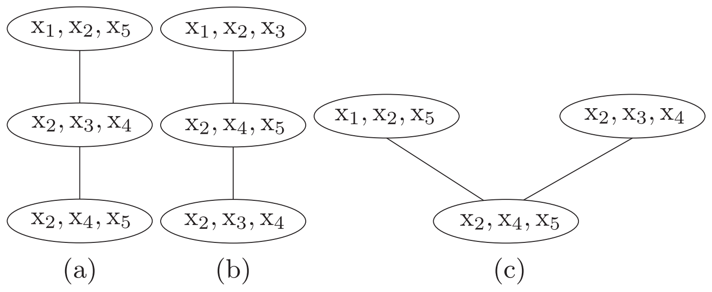


FIGURE 16.4

The graphs resulting from Figure 16.2c and shown in (a) and (b) are not join trees. (c) This is a join tree, because the node in the path from (x_1, x_2, x_5) to (x_2, x_3, x_4) contains their intersection, x_2.

We are now ready to state the basic theorem of this section; the one that will allow us to transform an arbitrary graph into a graph of a tree structure.

Theorem 16.1. *An undirected graph is triangulated if and only if its cliques can be organized into a join tree (Problem 16.1).*

Once a triangulated graph, which is associated with a factorized probability distribution, $p(\boldsymbol{x})$, has been transformed into a junction tree, then any of the message-passing algorithms, described in Chapter 15, can be adopted to perform exact inference.

16.2.1 CONSTRUCTING A JOIN TREE

Starting from a triangulated graph, the following algorithmic steps construct a join tree ([32]):

- Select a node in a maximal clique of the triangulated graph, which is not shared by other cliques. Eliminate this node and keep removing nodes from the clique, as long as they are *not* shared by other cliques. Denote the set of the remaining nodes of this clique as S_i, where i is the number of the nodes eliminated so far. This set is called a *separator*. Use V_i to denote the set of all the nodes in the clique, prior to the elimination process.
- Select another maximal clique and repeat the process with the index counting the node elimination starting from i.
- Continue the process until all cliques have been eliminated. Once the previous peeling off procedure has been completed, join together the parts that have resulted, so that each separator, S_i, is joined to V_i on one of its sides and to a clique node (set) V_j, $(j > i)$, such that $S_i \subset V_j$. This is in line with the running intersection property. It can be shown that the resulting graph is a join tree (part of proof in Problem 16.1).

An alternative algorithmic path to construct a join tree, once the cliques have been formed, is the following: Build an undirected graph having as nodes the maximal cliques of the triangulated graph. For each pair of linked nodes, V_i, V_j, assign a weight, w_{ij}, on the respective edge equal to the cardinality of $V_i \cap V_j$. Then run the *maximal spanning tree* algorithm (e.g., [43]) to identify a tree in this graph such as the sum of weights is maximal [41]. It turns out that such a procedure guarantees the running intersection property.

Example 16.2. Consider the graph of Figure 16.5, which is described in the seminal paper [44]. Smoking can cause lung cancer or bronchitis. A recent visit to Asia increases the probability of tuberculosis. Both, tuberculosis and cancer can result in a positive X-ray finding. Also, all three diseases can cause difficulty in breathing (dyspnea). In the context of the current example, we are not interested in the values of the respective probability table, and our goal is to construct a join tree, following the previous algorithm. Figure 16.6 shows a triangulated graph that corresponds to Figure 16.5.

The elimination sequence of the nodes in the triangulated graph is graphically illustrated in Figure 16.7. First, node A is eliminated from the clique (A, T) and the respective separator set comprises T. Because only one node can be eliminated $(i = 1)$, we indicate the separator as S_1. Next, node T is eliminated from the clique (T, L, E) and the S_2 $(i = i + 1)$ separator comprises L, E. The process

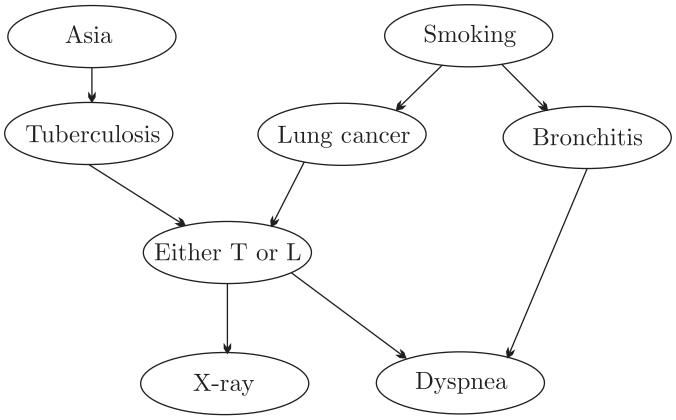


FIGURE 16.5

The Bayesian network structure of the example given in [44].

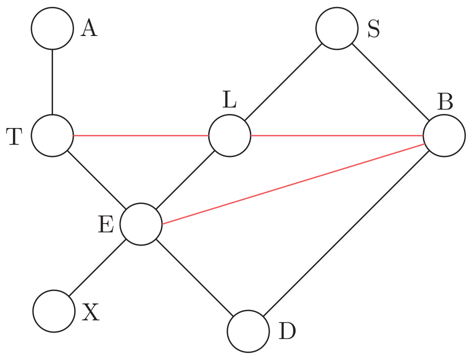


FIGURE 16.6

The graph resulting from the Bayesian network structure of Figure 16.5, after having been moralized and triangulated. The inserted edges are drawn in red.

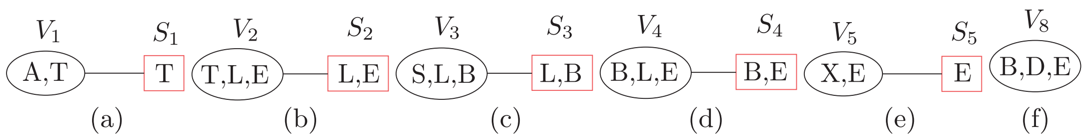


FIGURE 16.7

The sequence of elimination of nodes from the respective cliques of Figure 16.6 and the resulting separators.

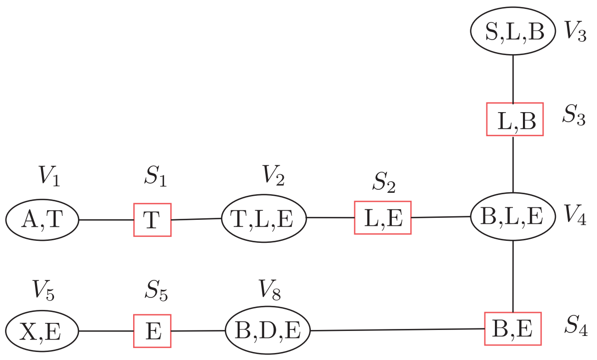


FIGURE 16.8

The resulting join tree from the graph of Figure 16.6. A separator, S_i, is linked to a clique V_j, $(j > i)$ and so that $S_i \subset V_j$.

continues till clique (B, D, E) is the only remaining one. It is denoted as V_8, as all three nodes can be eliminated sequentially (hence, $8 = 5 + 3$); there is no other neighboring clique. Figure 16.8 shows the resulting junction tree. Verify the running intersection property.

16.2.2 MESSAGE-PASSING IN JUNCTION TREES

By its definition, a junction tree is a join tree where we have associated a factor, say ψ_c, of a probability distribution, p, with each one of the cliques. Each factor can be considered as the product of all potential functions, which are defined in terms of the variables associated with the nodes of the corresponding clique; hence, the domain of each one of these potential functions is a subset of the variables-nodes comprising the clique. Then, focusing on the discrete probability case, we can write

$$P(\boldsymbol{x}) = \frac{1}{Z} \prod_c \psi_c(\boldsymbol{x}_c), \tag{16.3}$$

where c runs over the cliques and $\boldsymbol{x}_c$ denotes the variables comprising the respective clique. Because a junction tree is a graph with a tree structure, exact inference can take place in the same way as we have already discussed in Section 15.7, via a message-passing rationale. A two-way message-passing is also required here. There are, however, some small differences. In the case of the factor graphs, which we have considered previously, the exchanged messages were functions of one variable. This is not necessarily the case here. Moreover, in this case, after the bidirectional flow of the messages has been completed, what is recovered from each node of the junction tree is the *joint probability of the variables associated with the clique, $P(\boldsymbol{x}_c)$*. The computation of the marginal probabilities for individual variables requires extra summations in order to marginalize with respect to the rest.

Note that in the message-passing, the following take place:

- A separator receives messages and passes their product to one of its connected cliques, depending on the direction of the message-passing flow, that is,

$$\boxed{\mu_{S\to V}(\boldsymbol{x}_S) = \prod_{v\in\mathcal{N}(S)\setminus V} \mu_{v\to S}(\boldsymbol{x}_S),} \tag{16.4}$$

  where $\mathcal{N}(S)$ is the index set of the clique nodes connected to S, and $\mathcal{N}(S)\setminus V$ is this set excluding the index for clique node V. Note that the message is a function of the variables comprising the separator.
- Each clique node performs marginalization and passes the message to each one of its connected separators, depending on the direction of the flow. Let V be a clique node and $\boldsymbol{x}_V$ the vector of the involved variables in it, and let S be a separator node connected to it. The message passed to S is given by

$$\boxed{\mu_{V\to S}(\boldsymbol{x}_S) = \sum_{\boldsymbol{x}_V\setminus\boldsymbol{x}_S} \psi_V(\boldsymbol{x}_V) \prod_{s\in\mathcal{N}(V)\setminus S} \mu_{s\to V}(\boldsymbol{x}_s).} \tag{16.5}$$

By $\boldsymbol{x}_S$ we denote the variables in the separator S. Obviously, $\boldsymbol{x}_S \subset \boldsymbol{x}_V$ and $\boldsymbol{x}_V\setminus\boldsymbol{x}_S$ denotes all variables in $\boldsymbol{x}_V$ excluding those in $\boldsymbol{x}_S$. $\mathcal{N}(V)$ is the index set of all separators connected to V and $\boldsymbol{x}_s$, the set of variables in the respective separator ($\boldsymbol{x}_s \subset \boldsymbol{x}_V,\ s\in\mathcal{N}(V)$). $\mathcal{N}(V)\setminus S$ denotes the index set of all separators connected to V excluding S. This is basically the counterpart of Eq. (15.50). Figure 16.9 shows the respective configuration.

Once the two-way message-passing has been completed, marginals in the clique as well as the separator nodes are computed as (Problem 16.3)

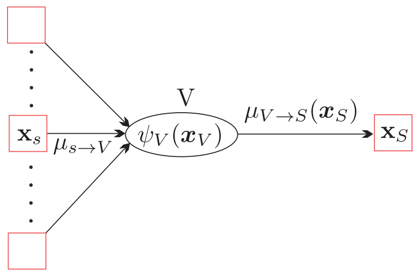


FIGURE 16.9

Clique node V "collects" all incoming messages from the separators it is connected with (except S); then, it outputs a message to S, after the marginalization performed on the product of $\psi_V(\boldsymbol{x}_V)$ with the incoming messages.

- Clique nodes:

$$P(\boldsymbol{x}_V) = \frac{1}{Z}\psi_V(\boldsymbol{x}_V) \prod_{s\in\mathcal{N}(V)} \mu_{s\to V}(\boldsymbol{x}_s) \tag{16.6}$$

- Separator nodes: Each separator is connected only to clique nodes. After the two-way message-passing, every separator has received messages from both flow directions. Then, it is shown that

$$P(\boldsymbol{x}_S) = \frac{1}{Z} \prod_{v\in\mathcal{N}(S)} \mu_{v\to S}(\boldsymbol{x}_S). \tag{16.7}$$

An important by-product of the previous message-passing algorithm in junction trees concerns the joint distribution of all the involved variables, which turns out to be independent of Z, and it is given by (Problem 16.4)

$$\boxed{P(\boldsymbol{x}) = \frac{\prod_v P_v(\boldsymbol{x}_v)}{\prod_s [P_s(\boldsymbol{x}_s)]^{d_s-1}},} \tag{16.8}$$

where $\prod_v$ and $\prod_s$ run over the sets of clique nodes and separators, respectively, and d_s is the number of the cliques separator S is connected to.

Example 16.3. Let us consider the junction tree of Figure 16.8. Assume that $\psi_1(A,T)$, $\psi_2(T,L,E)$, $\psi_3(S,L,B)$, $\psi_4(B,L,E)$, $\psi_5(X,E)$, and $\psi_6(B,D,E)$ are known. For example,

$$\psi_1(A,T) = P(T|A)P(A),$$

and

$$\psi_3(S,L,B) = P(L|S)P(B|S)P(S).$$

The message-passing can start from the leaves, (A,T) and (X,E), toward (S,L,B); once this message flow has been completed, message-passing takes place in the reverse direction. Some examples of message computations are given below.

The message received by node (T,L,E) is equal to

$$\mu_{S_1\to V_2}(T) = \sum_A \psi_1(A,T).$$

Also,

$$\mu_{V_2\to S_2}(L,E) = \sum_T \psi_2(T,L,E)\mu_{S_1\to V_2}(T) = \mu_{S_2\to V_4}(L,E),$$

and

$$\mu_{V_4\to S_3}(L,B) = \sum_E \psi_4(B,L,E)\mu_{S_2\to V_4}(L,E)\mu_{S_4\to V_4}(B,E).$$

The rest of the messages are computed in a similar way.

For the marginal probability, $P(T,L,E)$, of the variables in clique node V_2, we get

$$P(T,L,E) = \psi_{V_2}(T,L,E)\mu_{S_2\to V_2}(L,E)\mu_{S_1\to V_2}(T)$$

Observe that in this product, all other variables, besides T, L, and E, have been marginalized out. Also,

$$P(L,E) = \mu_{V_4 \to S_2}(L,E)\mu_{V_2 \to S_2}(L,E).$$

Remarks 16.1.

Note that a variable is part of more than one node in the tree. Hence, if one is interested in obtaining the marginal probability of an individual variable, this can be obtained by marginalizing over different variables in different nodes. The properties of the junction tree guarantee that all of them give the same result (Problem 16.5).

We have already commented that there is not a unique way to triangulate a graph. A natural question is now raised: Are all the triangulated versions equivalent from a computational point of view? Unfortunately, the answer is No. Let us consider the simple case where all the variables have the same number of possible states, k. Then the number of probability values for each clique node depends on the number of variables involved in it, and we know that this dependence is of an exponential form. Thus, our goal while triangulating a graph should be to implement it in such a way that the resulting cliques are as small as possible with respect to the number of nodes-variables involved. Let us define the size of a clique, V_i, as $s_i = k^{n_i}$, where n_i denotes the number of nodes comprising the clique. Ideally, we should aim at obtaining a triangulated version (or equivalently an elimination sequence) so that the total size of the triangulated graph, $\sum_i s_i$, where i runs over all cliques, to be minimum. Unfortunately, this is an NP-hard task [1]. One of the earliest algorithms proposed to obtain low-size triangulated graphs is given in [71]. A survey of related algorithms is provided in [39].

16.3 APPROXIMATE INFERENCE METHODS

So far, our focus has been to present efficient algorithms for exact inference in graphical models. Although such schemes form the basis of inference and have been applied in a number of applications, often one encounters tasks where exact inference is not practically possible. At the end of the previous section, we discussed the importance of small-sized cliques. However, in a number of cases, the graphical model may be so densely connected that it renders the task of obtaining cliques of a small size impossible. We will soon consider some examples.

In such cases, resorting to methods for tractable approximate inference is the only viable alternative. Obviously, there are various paths to approach this problem and a number of techniques have been proposed. Our goal in this section is to discuss the main directions that are currently popular. Our approach will be more on the descriptive side than that of rigorous mathematical proofs and theorems. The reader who is interested in delving deeper into this topic can refer to more specialized references, which are given in the text below.

16.3.1 VARIATIONAL METHODS: LOCAL APPROXIMATION

The current and the next subsections draw a lot of their theoretical basis on the variational approximation methods, which were introduced in Chapter 13 and in particular Sections 13.2 and 13.8.

The main goal in variational approximation methods is to replace probability distributions with computationally attractive bounds. The effect of such *deterministic* approximation methods is that it simplifies the computations; as we will soon see, this is equivalent to simplifying the graphical structure. Yet, these simplifications are carried out in the context of an associated optimization process. The functional form of these bounds is very much problem-dependent, so we will demonstrate the methodology via some selected examples.

Two main directions are followed: the *sequential* one and the *block* one [34]. The former will be treated in this subsection and the latter in the next one.

In the sequential methods, the approximation is imposed on individual nodes in order to modify the functional form of the local probability distribution functions. This is the reason we called them *local methods*. One can impose the approximation on some of the nodes or to all of them. Usually, some of the nodes are selected, whose number is sufficient so that exact inference can take place with the remaining ones, within practically acceptable computational time and memory size. An alternative viewpoint is to look at the method as a sparsification procedure that removes nodes so as to transform the original graph to a "computationally" manageable one. There are different scenarios on how to select nodes. One way is to introduce approximation to one node at a time until a sufficiently simplified structure occurs. The other way is to introduce approximation to all the nodes and then reinstate the exact distributions one node at a time. The latter of the two has the advantage that the network is computationally tractable all the way; see, for example, [30]. Local approximations are inspired by the method of bounding convex/concave functions in terms of their conjugate ones, as discussed in Section 13.8. Let us now unveil the secrets behind the method.

Multiple-cause networks and the noisy-OR model

In the beginning of Chapter 15 (Section 15.2) we presented a simplified case from the medical diagnosis field, concerning a set of diseases and findings. Adopting the so called noisy-OR model, we arrived at Eqs. (15.4) and (15.5), which are repeated here for convenience.

$$P(f_i = 0|\boldsymbol{d}) = \exp\left(-\sum_{j\in \mathrm{Pa}_i} \theta_{ij} d_j\right), \tag{16.9}$$

$$P(f_i = 1|\boldsymbol{d}) = 1 - \exp\left(-\sum_{j\in \mathrm{Pa}_i} \theta_{ij} d_j\right), \tag{16.10}$$

where we have exploited the experience we have gained so far and we have introduced in the notation the set of the parents Pa_i of the ith finding. The respective graphical model belongs to the family of multiple-cause networks (Section 15.3.6) and it is shown in Figure 16.10a. We will now pick up a specific node, say the ith node, assume that it corresponds to a positive finding ($f_i = 1$), and demonstrate how the variational approximation method can offer a way out from the "curse" of the exponential dependence of the joint probability on the number of the involved terms; recall that this is caused by the form of Eq. (16.10).

Derivation of the Variational Bound: The function $1 - \exp(-x)$ belongs to the so-called *log-concave* family of functions, meaning that

$$f(x) = \ln(1 - \exp(-x)), \quad x > 0,$$

is concave (Problem 16.9). Being a concave function, we know from Section 13.8 that it is upper bounded by

$$f(x) \leq \xi x - f^*(\xi),$$

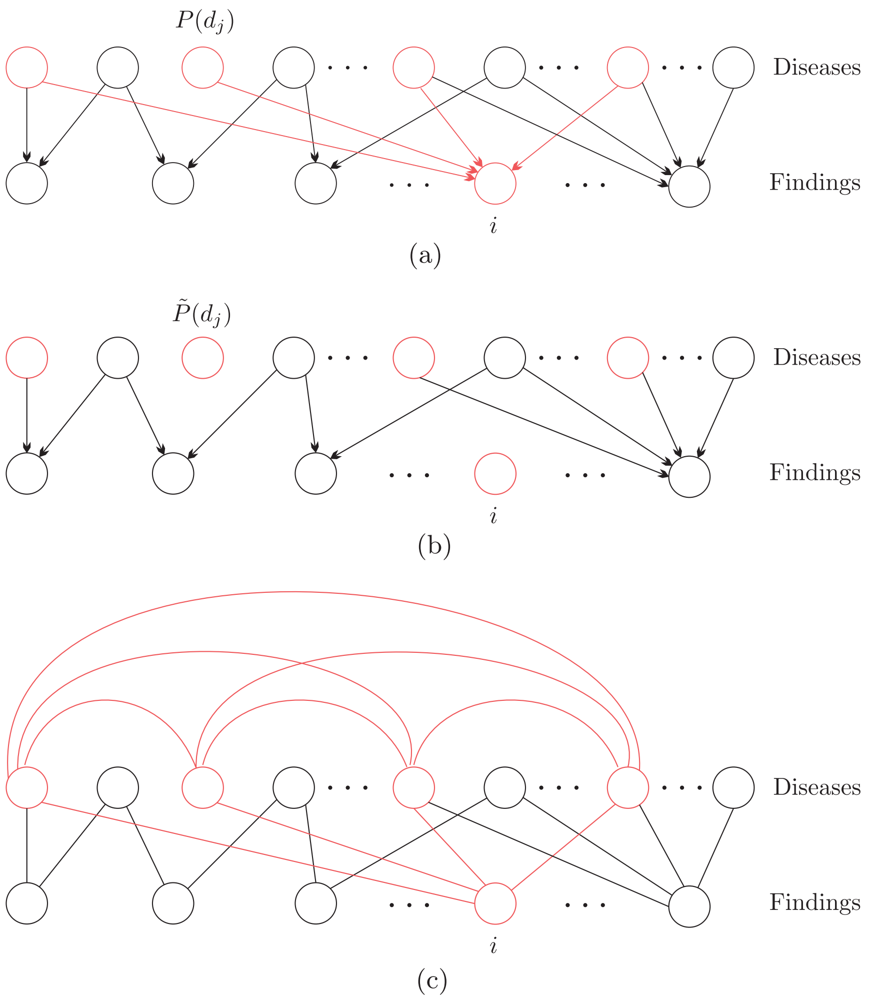


FIGURE 16.10

(a) A Bayesian network for a set of findings and diseases. The node associated with the *i*th finding, together with its parents and respective edges, are shown in red; it is the node on which variational approximation is introduced. (b) After the variational approximation is performed for node *i*, the edges joining it with its parents are removed. At the same time, the prior probabilities of the respective parent nodes change values. (c) The graph that would have resulted after the moralization step, focusing on node, *i*.

where $f^*(\xi)$ is its conjugate function. Tailoring it to the needs of Eq. (16.10) and using ξ_i in place of ξ, to explicitly indicate the dependence on the node i, we obtain

$$P(f_i = 1|\boldsymbol{d}) \leq \exp\left(\xi_i \left(\sum_{j\in \mathrm{Pa}_i} \theta_{ij} d_j\right) - f^*(\xi_i)\right), \tag{16.11}$$

or

$$P(f_i = 1|\boldsymbol{d}) \leq \exp\big(-f^*(\xi_i)\big) \prod_{j\in \mathrm{Pa}_i} \big(\exp(\xi_i\theta_{ij})\big)^{d_j}, \tag{16.12}$$

where (Problem 16.10)

$$f^*(\xi_i) = -\xi_i \ln(\xi_i) + (\xi_i + 1)\ln(\xi_i + 1), \quad \xi_i > 0.$$

Note that usually, a constant θ_{i0} is also present in the linear terms ($\sum_{j\in \mathrm{Pa}_i} \theta_{ij} d_j + \theta_{i0}$) and in this case the first exponent in the upper bound becomes $\exp(-f^*(\xi_i) + \xi_i\theta_{i0})$.

Let us now observe Eq. (16.12). The first factor on the right-hand side is a constant, once ξ_i is determined. Moreover, each one of the factors, $\exp(\xi_i\theta_{ij})$, is also a constant raised in d_j. Thus, substituting Eq. (16.12) in the products in Eq. (15.1), in order to compute, for example, Eq. (15.3), each one of these constants can be absorbed by the respective $P(d_j)$, that is,

$$\tilde{P}(d_j) \propto P(d_j)\exp(\xi_i\theta_{ij}d_j), \quad j \in \mathrm{Pa}_i.$$

Basically, from a graphical point of view, we can equivalently consider that the ith node is delinked and its influence on any subsequent processing is via the modified factors associated with its parent nodes, see Figure 16.10b. In other words, the variational approximation *decouples* the parent nodes. In contrast, for exact inference, during the moralization stage, all parents of node i are connected. This is the source of computational explosion, see Figure 16.10c. The idea is to remove a sufficient number of nodes, so that the remaining network can be handled using exact inference methods.

There is still a main point to be addressed: how the various ξ_is are obtained. These are computed to make the bound as tight as possible, and any standard optimization technique can be used. Note that this minimization corresponds to a convex cost function (Problem 16.11). Besides the upper bound, a lower bound can also be derived [27]. Experiments performed in [30] verify that reasonably good accuracies can be obtained in affordable computational times. The method was first proposed in [27].

The Boltzmann machine

The Boltzmann machine, which was introduced in Subsection 15.4.2, is another example where any attempt for exact inference is confronted with cliques of sizes that make the task computationally intractable [28].

We will demonstrate the use of the variational approximation in the context of the computation of the normalizing constant Z. Recall from Eq. (15.23) that

$$\begin{aligned} Z &= \sum_{\boldsymbol{x}} \exp\left(-\sum_i \left(\sum_{j>i} \theta_{ij} x_i x_j + \theta_{i0} x_i\right)\right) \\ &= \sum_{\boldsymbol{x}\backslash x_k} \sum_{x_k=0}^{1} \exp\left(-\sum_i \left(\sum_{j>i} \theta_{ij} x_i x_j + \theta_{i0} x_i\right)\right), \end{aligned} \tag{16.13}$$

where we chose node x_k to impose variational approximation. We split the summation into two, one with regard to x_k and one with regard to the rest of the variables; $\boldsymbol{x}\backslash x_k$ denotes summation over all variables excluding x_k. Performing the inner sum in Eq. (16.13) (terms different to x_k and $x_k = 0, x_k = 1$), we get

$$Z = \sum_{\boldsymbol{x}\backslash x_k} \exp\left(-\sum_{i\neq k}\left(\sum_{i<j\neq k} \theta_{ij} x_i x_j + \theta_{i0} x_i\right)\right)\left(1 + \exp\left(-\sum_{i\neq k} \theta_{ki} x_i - \theta_{k0}\right)\right)$$

where $i < j \neq k$ indicates that both i and j are different to k.

Derivation of the Variational Bound: The function $1 + \exp(-x)$, $x \in \mathbb{R}$ is *log-convex* (Problem 16.12); thus, following similar arguments to those adopted for the derivation of the bound in Eq. (16.12), but for convex instead of concave functions, we obtain

$$\begin{aligned} Z \geq & \sum_{\boldsymbol{x}\backslash x_k} \exp\left(-\sum_{i\neq k}\left(\sum_{i<j\neq k} \theta_{ij} x_i x_j + \theta_{i0} x_i\right)\right) \\ & \times \exp\left(\xi_k\left(\sum_{i\neq k} \theta_{ki} x_i + \theta_{k0}\right) - f^*(\xi_k)\right), \end{aligned} \tag{16.14}$$

where $f^*(\xi)$ is the respective conjugate function (Problem 16.12). Note that the second exponential in the bound can be combined with the first one and we can write

$$Z \geq \exp\left(-f^*(\xi_k) + \xi_k \theta_{k0}\right) \sum_{\boldsymbol{x}\backslash x_k} \exp\left(-\sum_{i\neq k}\left(\sum_{i<j\neq k} \tilde{\theta}_{ij} x_i x_j + \tilde{\theta}_{i0} x_i\right)\right),$$

where

$$\tilde{\theta}_{ij} = \theta_{ij}, \quad i,j \neq k,$$

and

$$\tilde{\theta}_{i0} = \theta_{i0} - \xi_k \theta_{ki}, \quad i \neq k.$$

In other words, if from now on we replace Z with the bound, it is as if node x_k has been removed and the remaining network is a Boltzmann machine with one node less and the respective parameters modified, compared to the original ones. The value of ξ_k can be obtained via optimization so as to make the bound as tight as possible.

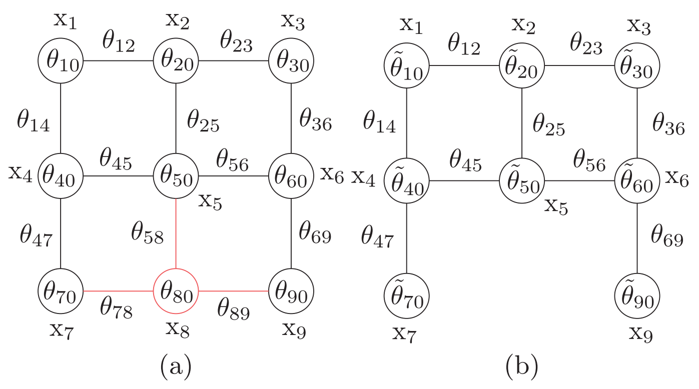


FIGURE 16.11

(a) An MRF corresponding to a Boltzmann machine. (b) The resulting MRF after removing x_8 via variational approximation. Note that if x_5 is removed next, then the remaining graphical model is a chain.

Figure 16.11 illustrates the effect of applying variational approximation to the node x_8. For the case of this figure, note that if x_5 is removed next (after x_8) the remaining graphical structure becomes a chain and exact inference can be carried out.

Following exactly similar arguments and employing the conjugate of the sigmoid function (Problem 13.18), one can apply the technique to the sigmoidal Bayesian networks, discussed in Section 15.3.4; see also [27].

16.3.2 BLOCK METHODS FOR VARIATIONAL APPROXIMATION

In contrast to the previous approach, where the approximation is introduced on selected nodes individually, here the approximation is imposed on a set of nodes. Once more, a derived bound of the involved probability distribution is optimized. In principle, the method is equivalent to imposing a specific graphical structure on the nodes, which can then be addressed by tractable exact inference techniques. In the sequel, the family of distributions, which can be factorized over this simplified substructure, is optimized with respect to a set of variational parameters. The method builds upon the same arguments as those used in Section 13.2. We will retain the same notation and we will provide the related formulas for the discrete variable case, to be used later on in our selected examples.

Let $\mathcal{X}$ be the set of observed and $\mathcal{X}^l$ the set of the latent random variables associated with the nodes of a graphical structure; in the graphical model terminology we can refer to them as evidence and hidden nodes, respectively. Define

$$\mathcal{F}(Q) = \sum_{x \in \mathcal{X}^l} Q(\mathcal{X}^l) \ln \frac{P(\mathcal{X}, \mathcal{X}^l)}{Q(\mathcal{X}^l)}, \tag{16.15}$$

where Q is any probability function. Then, from Eq. (16.15), we readily obtain that

$$\mathcal{F}(Q) = \ln P(\mathcal{X}) + \sum_{x \in \mathcal{X}^l} Q(\mathcal{X}^l) \ln \frac{P(\mathcal{X}^l|\mathcal{X})}{Q(\mathcal{X}^l)},$$

or

$$\ln P(\mathcal{X}) = \mathcal{F}(Q) + \sum_{x \in \mathcal{X}^l} Q(\mathcal{X}^l) \ln \frac{Q(\mathcal{X}^l)}{P(\mathcal{X}^l|\mathcal{X})}. \tag{16.16}$$

Note that the second term in Eq. (16.16) is the Kullback-Leibler divergence between $P(\mathcal{X}^l|\mathcal{X})$ and $Q(\mathcal{X}^l)$. Because KL divergence is always nonnegative (Problem 12.12), we can write

$$\ln P(\mathcal{X}) \geq \mathcal{F}(Q),$$

and the lower bound is maximized if we minimize the KL divergence.

Let us now return to our goal; that is, given the evidence, to perform inference on the graph associated with $P(\mathcal{X}^l|\mathcal{X})$. If this cannot be performed in a tractable way, the method adopts an approximation, $Q(\mathcal{X}^l)$, of $P(\mathcal{X}^l|\mathcal{X})$ and at the same time imposes a specific factorization on $Q(\mathcal{X}^l)$ (which equivalently induces a specific graphical structure) so that exact inference techniques can be employed. From the adopted family of distributions, we choose the one that minimizes the KL divergence between $P(\mathcal{X}^l|\mathcal{X})$ and $Q(\mathcal{X}^l)$. Such a choice guarantees the maximum lower bound for the log-evidence function. Among the different ways of factorization, the so-called *mean field* factorization is the simplest and, possibly, the most popular. The imposed structure on the graph has no edges, which leads to a complete factorization of $Q(\mathcal{X}^l)$, that is,

$$\boxed{Q(\mathcal{X}^l) = \prod_{i:x_i \in \mathcal{X}^l} Q_i(x_i) : \quad \text{Mean Field Factorization.}} \tag{16.17}$$

The mean field approximation and the Boltzmann machine

As we already know, the joint probability for the Boltzmann machine is given by

$$P(\mathcal{X}, \mathcal{X}^l) = \frac{1}{Z} \exp\left(- \sum_i \left(\sum_{j>i} \theta_{ij} x_i x_j + \theta_{i0} x_i \right) \right),$$

where some of x_i (x_j) belong to $\mathcal{X}$ and some to $\mathcal{X}^l$. Our first goal is to compute $P(\mathcal{X}^l|\mathcal{X})$ so as to use it in the KL divergence. Note that if both $x_i, x_j \in \mathcal{X}$ their contribution results to a constant, which is finally absorbed by the normalizing factor Z. If one is observed and the other one is latent, then the product contribution becomes linear with regard to the hidden variable and it is absorbed by the respective linear term. Then we can write that

$$P(\mathcal{X}^l|\mathcal{X}) = \frac{1}{\tilde{Z}} \exp\left(- \sum_{i:x_i \in \mathcal{X}^l} \left(\sum_{x_j \in \mathcal{X}^l : j>i} \theta_{ij} x_i x_j + \tilde{\theta}_{i0} x_i \right) \right). \tag{16.18}$$

We now turn our attention into the form of $Q(\cdot)$. Due to the (assumed) binary nature of the variables, a sensible completely factorized form of Q is ([34])

$$Q(\mathcal{X}^l; \boldsymbol{\mu}) = \prod_{i:x_i \in \mathcal{X}^l} \mu_i^{x_i} (1 - \mu_i)^{(1-x_i)}, \tag{16.19}$$

where the dependence on the variational parameters, $\boldsymbol{\mu}$, is explicitly shown. Also, due to the adopted Bernoulli distribution for each variable, $\mathbb{E}[\mathrm{x}_i] = \mu_i$ (Chapter 2). The goal now is to optimize the KL divergence with respect to the variational parameters. Plugging Eqs. (16.18) and (16.19) into

$$KL(Q||P) = \sum_{x_i \in \mathcal{X}^l} Q(\mathcal{X}^l; \boldsymbol{\mu}) \ln \frac{Q(\mathcal{X}^l; \boldsymbol{\mu})}{P(\mathcal{X}^l|\mathcal{X})}, \tag{16.20}$$

we obtain (Problem 16.13, [34]),

$$KL(Q||P) = \sum_i \left(\mu_i \ln \mu_i + (1-\mu_i)\ln(1-\mu_i) + \sum_{j>i} \theta_{ij}\mu_i\mu_j + \tilde{\theta}_{i0}\mu_i \right) + \ln \tilde{Z},$$

whose minimization with regard to μ_i finally results in (Problem 16.13)

$$\boxed{\mu_i = \sigma\left(-\left(\sum_{j\neq i} \theta_{ij}\mu_j + \tilde{\theta}_{i0} \right)\right) : \quad \text{Mean Field Equations,}} \tag{16.21}$$

where $\sigma(\cdot)$ is the sigmoid link function; recall from the definition of the Ising model that $\theta_{ij} = \theta_{ji} \neq 0$, if x_i and x_j are connected and it is zero otherwise. Plugging the values μ_i into Eq. (16.19), an approximation of $P(\mathcal{X}^l|\mathcal{X})$ in terms of $Q(\mathcal{X}^l; \boldsymbol{\mu})$ has been obtained.

Equation (16.21) is equivalent to a set of coupled equations known as *mean field equations* and they are used in a recursive manner to compute a solution fixed point set, assuming that one exists. Eq. (16.21) is quite interesting. Although we assumed independence among hidden nodes, imposing minimization of the KL divergence, information related to the (true) mutually dependent nature of the variables (as this is conveyed by $P(\mathcal{X}^l|\mathcal{X})$) is "embedded" into the mean values with respect to $Q(\mathcal{X}^l|\boldsymbol{\mu})$; the mean values of the respective variables are *interrelated*. Eq. (16.21) can also be viewed as a message-passing algorithm, see Figure 16.12. Figure 16.13 shows the graph associated with a Boltzmann machine prior to and after the application of the mean field approximation.

Note that what we have said before is nothing but an instance of the variational EM algorithm, presented in Section 13.2; as a matter of fact, Eq. (16.21) is the outcome of the E-step for each one of the factors of Q_i, assuming the rest are fixed.

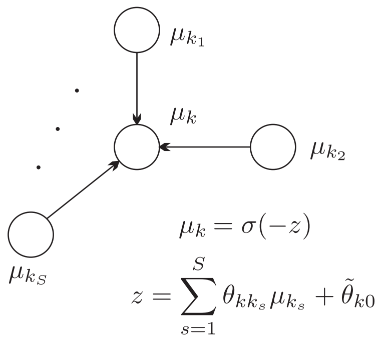

μ_{k_1}
μ_k
μ_{k_2}
μ_{k_S}
$\mu_k = \sigma(-z)$
$z = \sum_{s=1}^{S} \theta_{kk_s}\mu_{k_s} + \tilde{\theta}_{k0}$

FIGURE 16.12

Node k is connected to S nodes and receives messages from its neighbors; then, passes messages to its neighbors.

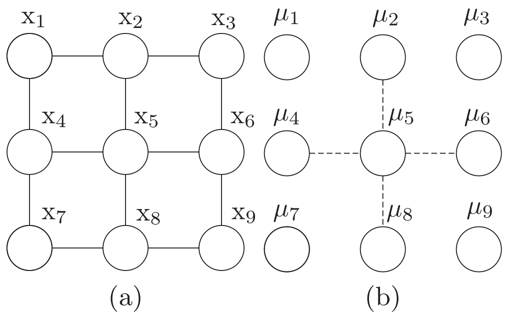


FIGURE 16.13

(a) The nodes of the graph representing a Boltzmann machine. (b) The mean field approximation results in a graph without edges. The dotted lines indicate the deterministic relation that is imposed among nodes, which were linked prior to the approximation with node x_5.

Thus far in the chapter, we have not mentioned the important task of how to obtain estimates of the parameters describing a graphical structure; in our current context, these are the parameters θ_{ij} and θ_{i0}, comprising the set $\boldsymbol{\theta}$. Although the parameter estimation task is discussed at the end of this chapter, there is no harm in saying a few words at this point. Let us give the dependence of $\boldsymbol{\theta}$ explicitly and denote the involved probabilities as $Q(\mathcal{X}^l; \boldsymbol{\mu}, \boldsymbol{\theta})$, $P(\mathcal{X}, \mathcal{X}^l; \boldsymbol{\theta})$, and $P(\mathcal{X}^l|\mathcal{X}; \boldsymbol{\theta})$. Treating $\boldsymbol{\theta}$ as an unknown parameter vector, we know from the variational EM, that this can be iteratively estimated by adding the M-step in the algorithm and optimizing the lower bound, $\mathcal{F}(Q)$, with regard to $\boldsymbol{\theta}$, fixing the rest of the involved parameters; see, for example, [29, 69].

Remarks 16.2.

- The mean field approximation method has also been applied in the case of sigmoidal neural networks, defined in Section 15.3.4 (see, e.g., [69]).
- The mean field approximation involving the completely factorized form of Q is the simplest and crudest approximation. More sophisticated attempts have also been suggested, where Q is allowed to have a richer structure while retaining its computational tractability (see, e.g., [17, 31, 80]).
- In [82], the mean field approximation has been applied to a general Bayesian network, where, as we know, the joint probability distribution is given by the product of the conditionals across the nodes,

$$p(\boldsymbol{x}) = \prod_i p(x_i|\text{Pa}_i).$$

Unless the conditionals are given in a structurally simple form, exact message-passing can become computationally tough. For such cases, the mean field approximation can be introduced in the hidden variables.

$$Q(\mathcal{X}^l) = \prod_{i:x_i \in \mathcal{X}^l} Q_i(x_i),$$

which are then estimated so as to maximize the lower bound $\mathcal{F}(Q)$ in (16.15). Following the arguments that were introduced in Section 13.2, this is achieved iteratively starting from some initial estimates and at each iteration step optimization takes place with respect to a single factor, holding the rest fixed. At the $(j+1)$ step, the mth factor is obtained as (Eq. (13.14)

$$\begin{aligned}\ln Q_m^{(j+1)}(x_m^l) &= \mathbb{E}\left[\ln \prod_i p(x_i|\mathrm{Pa}_i)\right] + \text{constant}\\ &= \mathbb{E}\left[\sum_i \ln p(x_i|\mathrm{Pa}_i)\right] + \text{constant},\end{aligned}$$

where the expectation is with respect to the currently available estimates of the factors, *excluding* Q_m, and x_m^l is the respective hidden variable. When restricting the conditionals within the conjugate-exponential family, the computations of expectations of the logarithms become tractable. The resulting scheme is equivalent to a message-passing algorithm, known as *variational message-passing*, and it comprises passing moments and parameters associated with the exponential distributions. An implementation example of the variational message-passing scheme in the context of MIMO-OFDM communication systems is given in [6, 23, 38].

16.3.3 LOOPY BELIEF PROPAGATION

The message-passing algorithms, which were considered previously for exact inference in graphs with a tree structure, can also be used for approximate inference in general graphs with cycles (loops). Such schemes are known as *loopy belief propagation* algorithms.

The idea of using the message-passing (sum-product) algorithm with graphs with cycles goes back to Pearl [57]. Note that algorithmically, there is nothing to prevent us from applying the algorithm in such general structures. On the other hand, if we do it, there is no guarantee that the algorithm will converge in two passes and, more important, that it will recover the true values for the marginals. As a matter of fact, there is no guarantee that such a message propagation will ever converge. Thus, without any clear theoretical understanding, the idea of using the algorithm in general graphs was rather forgotten. Interestingly enough, the spark for its comeback was ignited by a breakthrough in coding theory, under the name *turbo codes* [5]. It was empirically verified that the scheme can achieve performance very close to the theoretical Shannon limit.

Although, in the beginning, such coding schemes seemed to be unrelated to belief propagation, it was subsequently shown, [49], that the turbo decoding is just an instance of the sum-product algorithm, when applied to a graphical structure that represents the turbo code. As an example of the use of the loopy belief propagation for decoding, consider the case of Figure 15.19. This is a graph with cycles. Applying belief propagation on this graph, we can obtain after convergence the conditional probabilities $P(x_i|y_i)$ and, hence, decide on the received sequence of bits. This finding revived interest in loopy belief propagation; after all, it "may be useful" in practice. Moreover, it initiated activity on theoretical research in order to understand its performance as well as its more general convergence properties.

In [83], it is shown, on the basis of pairwise connected MRF's (undirected graphical models with potential functions involving at most pairs of variables, e.g., trees), that whenever the sum-product algorithm converges on loopy graphs, the fixed points of the message-passing algorithm are actually stationary points of the so-called *Bethe free energy* cost. This is directly related to the KL divergence, between the true and an approximating distribution (Section 12.5.2). Recall from Eq. (16.20) that one of the terms in KL divergence is the negative entropy associated with Q. In the mean field approximation, this entropy term can be easily computed. However, this is not the case for more general structures with cycles and one has to be content to settle for an approximation. The so-called Bethe entropy approximation is employed, which in turn gives rise to the Bethe free energy cost function. To obtain the Bethe entropy approximation, one "embeds" into the approximating distribution, Q, a structure that is in line with (16.8), which holds true for trees. Indeed, it can be checked out (try it) that for (singly connected) trees, the product in the numerator runs over all pairs of connected nodes in the tree, and let us denote it as $\prod_{(i,j)} P_{ij}(x_i, x_j)$. Also, d_s is equal to the number of nodes that node s is connected with. Thus, we can write the joint probability as

$$P(\boldsymbol{x}) = \frac{\prod_{(i,j)} P_{ij}(x_i, x_j)}{\prod_s [P_s(x_s)]^{d_s - 1}}.$$

Note that nodes that are connected to only one node have no contribution in the denominator. Then, the entropy of the tree, that is,

$$E = -\mathbb{E}[\ln P(\boldsymbol{x})],$$

can be written as

$$\boxed{E = -\sum_{(i,j)} \sum_{x_i} \sum_{x_j} P_{ij}(x_i, x_j) \ln P_{ij}(x_i, x_j) + \sum_s (d_s - 1) \sum_{x_s} P_s(x_s) \ln P_s(x_s).} \tag{16.22}$$

Thus, this expression for the entropy is exact for trees. However, for more general graphs with cycles, this can only hold approximately true, and it is known as the *Bethe approximation of the entropy*. The closer to a tree a graph is, the better the approximation becomes; see [83] for a concise, related introduction.

It turns out that in the case of trees, the sum-product algorithm leads to the true marginal values because no approximation is involved and minimizing the free energy is equivalent to minimizing the KL divergence. Thus, from this perspective, the sum-product algorithm gets an optimization flavor. In a number of practical cases, the Bethe approximation is accurate enough, which justifies the good performance that is often achieved in practice by the loopy belief algorithm (see, e.g., [52]). The loopy belief propagation algorithm is not guaranteed to converge in graphs with cycles, so one may choose to minimize the Bethe energy cost directly; although such schemes are slower compared to message-passing, they are guaranteed to converge (e.g., [85]).

An alternative interpretation of the sum-product algorithm as an optimization algorithm of an appropriately selected cost function is given in [75, 77]. A unifying framework for exact, as well as approximate, inference is provided in the context of the exponential family of distributions. Both the mean field approximation as well as the loopy belief propagation algorithm are considered and viewed as different ways to approximate a convex set of realizable mean parameters, which are associated with the corresponding distribution. Although we will not proceed in a detailed presentation, we will provide a few "brush strokes," which are indicative of the main points around which this theory develops. At the same time, this is a good excuse for us to be exposed to an interesting interplay among the notions of convex duality, entropy, cumulant generating function, and mean parameters, in the context of the exponential family.

The general form of a probability distribution in the exponential family is given by (Section 12.4.1),

$$p(\boldsymbol{x};\boldsymbol{\theta}) = C\exp\left(\sum_{i\in I}\theta_i u_i(\boldsymbol{x})\right)$$
$$= \exp\left(\boldsymbol{\theta}^T\boldsymbol{u}(\boldsymbol{x}) - A(\boldsymbol{\theta})\right),$$

with

$$A(\boldsymbol{\theta}) = -\ln C = \ln\int \exp\left(\boldsymbol{\theta}^T\boldsymbol{u}(\boldsymbol{x})\right)\,\mathrm{d}\boldsymbol{x},$$

where the integral becomes summation for discrete variables. $A(\boldsymbol{\theta})$ is a convex function and it is known as the *log-partition* or *cumulant generating function* (Problem 16.14, [75, 77]). It turns out that the conjugate function of $A(\boldsymbol{\theta})$, denoted as $A^*(\boldsymbol{\mu})$, is the *negative entropy* function of $p(\boldsymbol{x};\boldsymbol{\theta}(\boldsymbol{\mu}))$, where $\boldsymbol{\theta}(\boldsymbol{\mu})$ is the value of $\boldsymbol{\theta}$ where the maximum (in the definition of the conjugate function) occurs given the value of $\boldsymbol{\mu}$; we say that $\boldsymbol{\theta}(\boldsymbol{\mu})$ and $\boldsymbol{\mu}$ are dually coupled (Problem 16.15). Moreover,

$$\mathbb{E}[\boldsymbol{u}(\mathbf{x})] = \boldsymbol{\mu},$$

where the expectation is with respect to $p(\boldsymbol{x};\boldsymbol{\theta}(\boldsymbol{\mu}))$. This is an interesting interpretation of $\boldsymbol{\mu}$ as a mean parameter vector; recall from Section 12.4.1 that these mean parameters define the respective exponential distribution. Then

$$A(\boldsymbol{\theta}) = \max_{\boldsymbol{\mu}\in\mathcal{M}}\left(\boldsymbol{\theta}^T\boldsymbol{\mu} - A^*(\boldsymbol{\mu})\right). \qquad (16.23)$$

$\mathcal{M}$ is the set that guarantees that $A^*(\boldsymbol{\mu})$ is finite, according to the definition of the conjugate function in (13.77). It turns out that in graphs of a tree structure, the sum-product algorithm is an iterative scheme of solving a Lagrangian dual formulation of (16.23), [75, 77]. Moreover, in this case, the set $\mathcal{M}$, which can be shown to be a convex one, is possible to be characterized explicitly in a straightforward way and the negative entropy $A^*(\boldsymbol{\mu})$ has an explicit form. These properties are no more valid in graphs with cycles. The mean field approximation involves an inner approximation of the set $\mathcal{M}$; hence, it restricts optimization to a limited class of distributions, for which the entropy can be recovered exactly. On the other hand, the loopy belief algorithm provides an outer approximation and, hence, enlarges the class of distributions; entropy can only approximately be recovered, which for the case of pairwise MRFs can take the form of the Bethe approximation.

The previously summarized theoretical findings have been generalized to the case of junction trees, where the potential functions involve more than two variables. Such methods involve the so-called *Kikuchi* energy, which is a generalization of the Bethe approximation [77, 84]. Such arguments have their origins in statistical physics [37].

Remarks 16.3.

- Following the success of loopy belief propagation in turbo decoding, further research verified its performance potential in a number of tasks such as low-density parity check codes [15, 47], network diagnostics [48], sensor network applications [24], and multiuser communications [70]. Furthermore, a number of modified versions of the basic scheme have been proposed. In [74], the so-called tree reweighted belief propagation is proposed. In [26], arguments from information geometry are employed and in [78], projection arguments in the context of information geometry are used. More recently, the belief propagation algorithm and the mean field approximation are proposed to be optimally combined to exploit their respective advantages [65]. A related review can be found in [76]. In a nutshell, this old scheme is still alive and kicking!

- In Section 13.11, the expectation propagation algorithm was discussed in the context of parameter inference. The scheme can also be adopted in the more general framework of graphical models, if the place of parameters is taken by the hidden variables. Graphical models are particularly tailored for this approach because the joint pdf is factorized. It turns out that if the approximate pdf is completely factorized, corresponding to a partially disconnected network, the expectation propagation algorithm turns out to be the loopy belief propagation algorithm [50]. In [51], it is shown that a new family of message-passing algorithms can be obtained by utilizing a generalization of the KL divergence as the optimizing cost. This family encompasses a number of previously developed schemes.
- Besides the approximation techniques that were previously presented, another popular pool of methods is the Markov chain Monte Carlo (MCMC) framework. Such techniques were discussed in Chapter 14; see, for example, [25] and the references therein.

16.4 DYNAMIC GRAPHICAL MODELS

All the graphical models that have been discussed so far were developed to serve the needs of random variables whose statistical properties remained fixed over time. However, this is not always the case. As a matter of fact, the terms *time adaptivity* and *time variation* are central for most parts of this book. Our focus in this section is to deal with random variables whose statistical properties are not fixed but are allowed to undergo changes. A number of time series as well as sequentially obtained data fall under this setting with applications ranging from signal processing and robotics to finance and bioinformatics.

A key difference here, compared to what we have discussed in the previous sections of this chapter, is that now observations are sensed sequentially and the *specific sequence* in which they occur carries important information, which has to be respected and exploited in any subsequent inference task. For example, in speech recognition, the sequence in which the feature vectors result is very important. In a typical speech recognition task, the raw speech data are *sequentially* segmented in short (usually overlapping) time windows and from each window a feature vector is obtained (e.g., DFT of the samples in the respective time slot). This is illustrated in Figure 16.14. These feature vectors constitute the observation sequence. Besides the information that resides in the specific values of these observation vectors, the sequence in which the observations appear discloses important information about the word that is spoken; our language and spoken words are highly structured human activities. Similar arguments hold true for applications such as learning and reasoning concerning biological molecules, for example, DNA and proteins.

Although any type of graphical model has its dynamic counterpart, we will focus on the family of *dynamic Bayesian networks* and, in particular, a specific type known as *hidden Markov models*.

A very popular and effective framework to model sequential data is via the so-called *state-observation* or *state-space* models. Each set of random variables, $\mathbf{y}_n \in \mathbb{R}^l$, which are observed at time n, is associated with a corresponding hidden/latent random vector $\mathbf{x}_n$ (not necessarily of the same dimensionality as that of the observations). The system dynamics are modeled via the latent variables and observations are considered to be the output of a measuring *noisy* sensing device. The so-called *latent Markov models* are built around the following two independence assumptions:

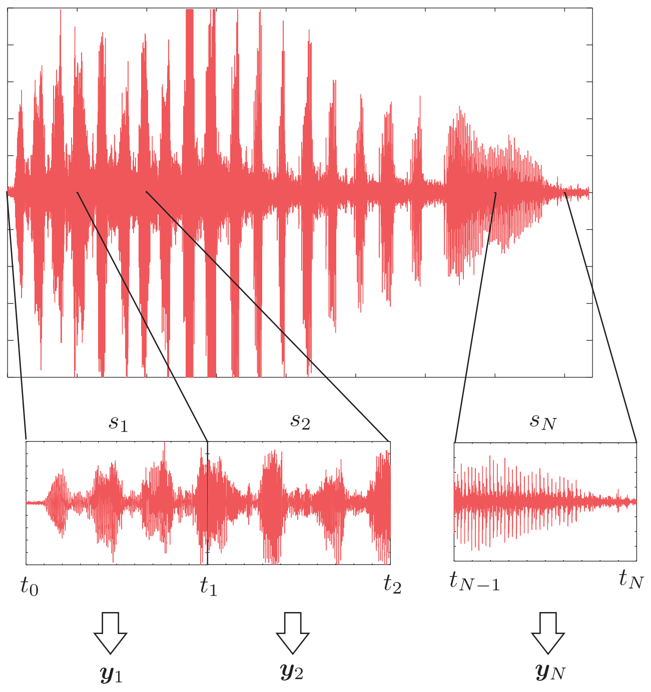


FIGURE 16.14

A speech segment and N time windows, each one of length equal to 500 ms. They correspond to time intervals [0, 500], [500, 1000], and [3500, 4000], respectively. From each one of them, a feature vector, $\boldsymbol{y}$, is generated. In practice, an overlap between successive windows is allowed.

$$(1)\ \mathbf{x}_{n+1} \perp (\mathbf{x}_1, \ldots, \mathbf{x}_{n-1}) | \, \mathbf{x}_n \tag{16.24}$$

$$(2)\ \mathbf{y}_n \perp (\mathbf{x}_1, \ldots, \mathbf{x}_{n-1}, \mathbf{x}_{n+1}, \ldots, \mathbf{x}_N) | \, \mathbf{x}_n, \tag{16.25}$$

where N is the total number of observations. The first condition defines the system dynamics via the transition model

$$p(\boldsymbol{x}_{n+1}|\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n) = p(\boldsymbol{x}_{n+1}|\boldsymbol{x}_n), \tag{16.26}$$

and the second one the observation model

$$p(\boldsymbol{y}_n|\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N) = p(\boldsymbol{y}_n|\boldsymbol{x}_n). \tag{16.27}$$

In words, *the future is independent of the past given the present, and the observations are independent of the future and past given the present.*

The previously stated independencies are graphically represented via the graph of Figure 16.15. If the hidden variables are of a discrete nature, the resulting model is known as a hidden Markov model. If on the other hand, both hidden as well as observation variables are of a continuous nature, the resulting model gets rather involved to deal with. However, analytically tractable tools can be and have been

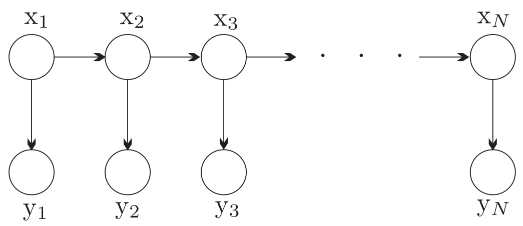


FIGURE 16.15

The Bayesian network corresponding to a latent Markov model. If latent variables are of a discrete nature, this corresponds to an HMM. If both observed as well as latent variables are continuous and follow a Gaussian distribution, this corresponds to a linear dynamic system (LDS). Note that the observed variables comprise the leaves of the graph.

developed for some special cases. In the so-called *linear dynamic systems* (LDS), the system dynamics and the generation of the observations are modeled as

$$\mathbf{x}_n = F_n \mathbf{x}_{n-1} + \boldsymbol{\eta}_n, \tag{16.28}$$

$$\mathbf{y}_n = H_n \mathbf{x}_n + \boldsymbol{v}_n, \tag{16.29}$$

where $\boldsymbol{\eta}_n$ and $\boldsymbol{v}_n$ are zero-mean, mutually independent noise disturbances modeled by Gaussian distributions. This is the celebrated Kalman filter, which we have already discussed in Chapter 4 and it will also be considered, from a probabilistic perspective, in Chapter 17. The probabilistic counterparts of (16.28)-(16.29) are

$$p(\boldsymbol{x}_n|\boldsymbol{x}_{n-1}) = \mathcal{N}(\boldsymbol{x}_n|F_n\boldsymbol{x}_{n-1}, Q_n), \tag{16.30}$$

$$p(\boldsymbol{y}_n|\boldsymbol{x}_n) = \mathcal{N}(\boldsymbol{y}_n|H_n\boldsymbol{x}_n, R_n), \tag{16.31}$$

where Q_n and R_n are the covariance matrices of $\boldsymbol{\eta}_n$ and $\boldsymbol{v}_n$, respectively.

16.5 HIDDEN MARKOV MODELS

Hidden Markov models are represented by the graphical model in Figures 16.15 and Eqs. (16.26), (16.27). The latent variables are discrete; hence, we write the *transition* probability as $P(\boldsymbol{x}_n|\boldsymbol{x}_{n-1})$ and this corresponds to a table of probabilities. Observation variables can either be discrete or continuous. Basically, an HMM is used to model a *quasi-stationary* process that undergoes *sudden* changes among a number of, say K, subprocesses. Each one of these subprocesses is described by different statistical properties. One could alternatively view it as a combined system comprising a number of subsystems; each one of these subsystems generates data/observations according to a different statistical model; for example, one may follow a Gaussian and the other one a student's t-distribution. Observations are emitted by these subsystems; however, once an observation is received, we do not know which subsystem this was emitted from. This reminds us of the mixture modeling task of a pdf; however, in mixture modeling, we did not care about the sequence in which observations occur.

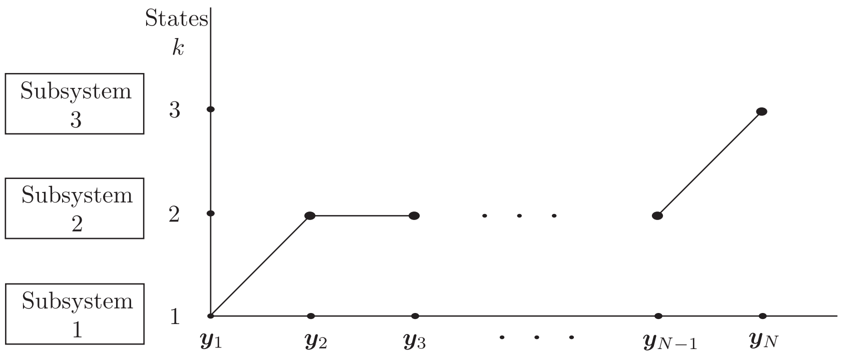


FIGURE 16.16

The unfolding in time of a trajectory that associates observations with states.

For modeling purposes, we associate with each observation, $\boldsymbol{y}_n$, a hidden variable, $\mathrm{k}_n = 1, 2, \dots, K$, which is the (random) index indicating the subsystem/subprocess that generated the respective observation vector. We will call it the *state*. Each k_n corresponds to $\mathbf{x}_n$ of the general model. The sequence of the complete observation set, $(\boldsymbol{y}_n, k_n)$, $n = 1, 2, \dots, N$, forms a trajectory in a two-dimensional grid, having the states on one axis and the observations on the other. This is shown in Figure 16.16 for $K = 3$. Such a path reveals the origin of each observation; $\boldsymbol{y}_1$ was emitted from state $k_1 = 1$, $\boldsymbol{y}_2$ from $k_2 = 2$, $\boldsymbol{y}_3$ from $k_3 = 2$, and $\boldsymbol{y}_N$ from $k_N = 3$. Note that each trajectory is associated with a probability distribution; that is, the joint distribution of the complete set. Indeed, the probability that the trajectory of Figure 16.16 will occur depends on the value of $P\big((\boldsymbol{y}_1, k_1 = 1), (\boldsymbol{y}_2, k_2 = 2), (\boldsymbol{y}_3, k_3 = 2), \dots, (\boldsymbol{y}_N, k_N = 3)\big)$. We will soon see that some of the possible trajectories that can be drawn in the grid are not allowed in practice; this may be due to physical constraints concerning the data generation mechanism that underlies the corresponding system/process.

Transition Probabilities. As already said, the dynamics of a latent Markov model are described in terms of the distribution $p(\boldsymbol{x}_n|\boldsymbol{x}_{n-1})$, which for an HMM becomes the set of probabilities

$$P(k_n|k_{n-1}), \quad k_n, k_{n-1} = 1, 2, \dots, K,$$

indicating the probability of the system to "jump" at time n to state k_n from state k_{n-1}, where it was at time $n-1$. In general, this table of probabilities may be time-varying. In the standard form of HMM, this is considered to be independent of time and we say that our model is *homogeneous*. Thus, we can write

$$P(k_n|k_{n-1}) = P(i|j) := P_{ij}, \quad i, j = 1, 2, \dots, K.$$

Note that some of these transition probabilities can be zero depending on the modeling assumptions. Figure 16.17a shows an example of a three-state system. The model is of the so called *left-to-right* type, where two types of transitions are allowed: (a) self transitions and (b) transitions from a state of a lower index to a state of a higher index. The system, once it jumps into a state k, emits data according to a probability distribution $p(\boldsymbol{y}|k)$, as illustrated in Figure 16.17b. Besides the left-to-right models, other alternatives have also been proposed [8, 63]. The states correspond to certain physical characteristics of the corresponding system. For example in speech recognition, the number of states, that are chosen

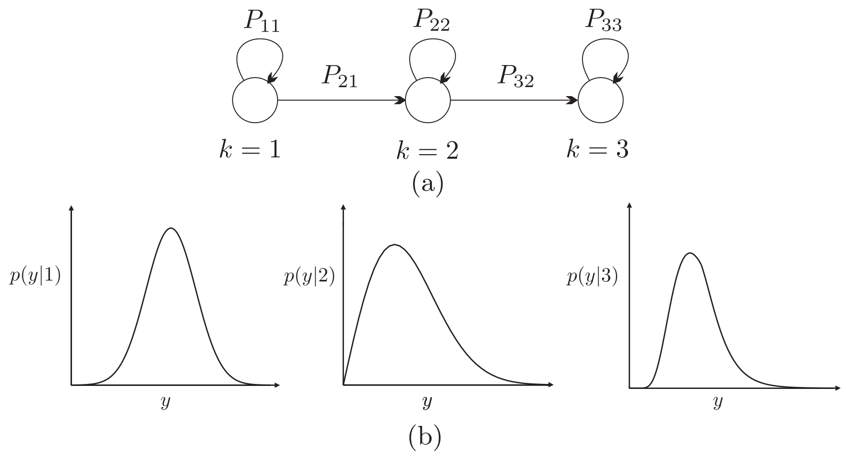


FIGURE 16.17

(a) A three-state left-to-right HMM model. (b) Each state is characterized by different statistical properties.

to model a spoken word, depends on the expected number of sound phenomena (phonemes) within the word. Typically, three to four states are used per phoneme. Another modeling path uses the average number of observations resulting from various versions of a spoken word as an indication of the number of states. Seen from the transition probabilities perspective, an HMM is basically a stochastic finite state automaton that generates an observation string. Note that the semantics of Figure 16.17 is different and must not be confused with the graphical structure given in Figure 16.15. Figure 16.17 is a graphical interpretation of the transition probabilities among the states; it says nothing about independencies among the involved random variables. Once a state transition model has been adopted, some trajectories in the trellis diagram of Figure 16.16 will not be allowed. In Figure 16.18, the red trajectory is not in line with the model of Figure 16.17.

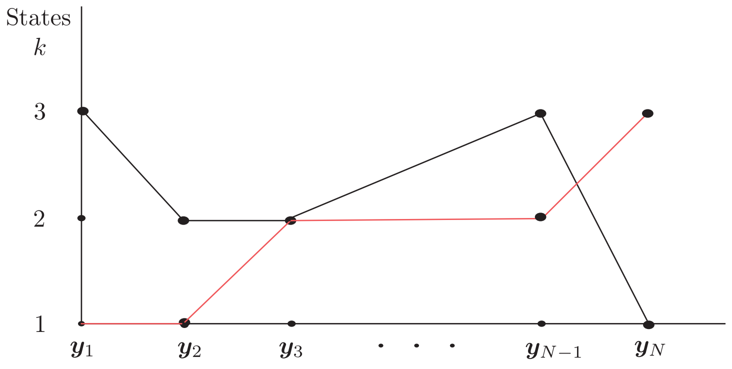


FIGURE 16.18

The black trajectory is not allowed to occur, under the HMM model of Figure 16.17. Transitions from state $k = 3$ to state $k = 2$ and from $k = 3$ to $k = 1$ are not permitted. In contrast, the state unfolding in the red curve is a valid one.

16.5.1 INFERENCE

As in any graphical modeling task, the ultimate goal is inference. Two types of inference are of particular interest in the context of classification/recognition. Let us discuss it in the framework of speech recognition; similar arguments hold true for other applications. We are given a set of (output variables) observations, $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_N$, and we have to decide to which spoken word these correspond. In the database, each spoken word is represented by an HMM model, which is the result of extensive training. An HMM model is fully described by the following set of parameters:

HMM Model Parameters

1. Number of states K.
2. The probabilities for the initial state at $n = 1$ to be at state k, that is, $P_k, \ k = 1, 2, \ldots, K$.
3. The set of transition probabilities $P_{ij}, \ i, j = 1, 2, \ldots, K$.
4. The state emission distributions $p(\boldsymbol{y}|k)$, $k = 1, 2, \ldots, K$, which can either be discrete or continuous. Often, these probability distributions may be parameterized, $p(\boldsymbol{y}|k; \boldsymbol{\theta}_k)$, $k = 1, 2, \ldots, K$.

Prior to inference, all the involved parameters are assumed to be known. Learning of the HMM parameters takes place in the training phase; we will come to it shortly.

For the recognition, a number of scores can be used. Here we will discuss two alternatives that come as a direct consequence of our graphical modeling approach. For a more detailed discussion, see, for example, [72].

In the first one, the joint distribution for the observed sequence is computed, after marginalizing out all hidden variables; this is done for each one of the models/words. Then the word that scores the larger value is selected. This method corresponds to the sum-product rule. The other path is to compute, for each model/word, the optimal trajectory in the trellis diagram; that is, the trajectory that scores the highest joint probability. In the sequel, we decide in favor of the model/word that corresponds to the largest optimal value. This method is an implementation of the max-sum rule.

The Sum-Product Algorithm: The HMM Case.

The first step is to transform the directed graph of Figure 16.15 to an undirected one; a factor graph or a junction tree graph. Note that this is trivial for this case, as the graph is already a tree. Let us work with the junction tree formulation. Also, in order to use the message-passing formulas of (16.4) and (16.5) as well as (16.7) and (16.6) for computing the distribution values, we will first adopt a more compact way of representing the conditional probabilities. We will employ the technique that was used in Section 13.4 for the mixture modeling case. Let us denote each latent variable as a K-dimensional vector, $\boldsymbol{x}_n \in \mathbb{R}^K$, $n = 1, 2, \ldots, N$, whose elements are all zero except at the kth location, where k is the index of the (unknown) state from which $\boldsymbol{y}_n$ has been emitted, that is,

$$\boldsymbol{x}_n^T = [x_{n,1}, x_{n,2}, \ldots, x_{n,K}] : \begin{cases} x_{n,i} = 0, & i \neq k \\ x_{n,k} = 1. \end{cases}$$

Then, we can compactly write

$$P(\boldsymbol{x}_1) = \prod_{k=1}^{K} P_k^{x_{1,k}}, \tag{16.32}$$

and

$$P(\boldsymbol{x}_n|\boldsymbol{x}_{n-1}) = \prod_{i=1}^{K}\prod_{j=1}^{K} P_{ij}^{x_{n-1,j}x_{n,i}}. \tag{16.33}$$

Indeed, if the jump is from a specific state j at time $n-1$, to a specific state i at time n, then the only term that survives in the previous product is the corresponding factor, P_{ij}. The joint probability distribution of the complete set, as a direct consequence of the Bayesian network model of Figure 16.15, is written as

$$p(Y,X) = P(\boldsymbol{x}_1)p(\boldsymbol{y}_1|\boldsymbol{x}_1)\prod_{n=2}^{N} P(\boldsymbol{x}_n|\boldsymbol{x}_{n-1})p(\boldsymbol{y}_n|\boldsymbol{x}_n), \tag{16.34}$$

where

$$p(\boldsymbol{y}_n|\boldsymbol{x}_n) = \prod_{k=1}^{K} \left(p(\boldsymbol{y}_n|k;\boldsymbol{\theta}_k)\right)^{x_{n,k}}. \tag{16.35}$$

The corresponding junction tree is trivially obtained from the graph in 16.15. Replacing directed links with undirected ones and considering cliques of size two, the graph in Figure 16.19a results. However, as all the $\boldsymbol{y}_n$ variables are observed (*instantiated*) and no marginalization is required, their multiplicative contribution can be absorbed by the respective conditional probabilities, which leads to the graph of 16.19b. Alternatively, this junction tree can be obtained if one considers the nodes $(\mathbf{x}_1, \boldsymbol{y}_1)$ and $(\mathbf{x}_{n-1}, \mathbf{x}_n, \boldsymbol{y}_n)$, $n = 2, 3, \ldots, N$, to form cliques associated with the potential functions

$$\psi_1(\boldsymbol{x}_1, \boldsymbol{y}_1) = P(\boldsymbol{x}_1)p(\boldsymbol{y}_1|\boldsymbol{x}_1), \tag{16.36}$$

and

$$\psi_n(\boldsymbol{x}_{n-1}, \boldsymbol{y}_n, \boldsymbol{x}_n) = P(\boldsymbol{x}_n|\boldsymbol{x}_{n-1})p(\boldsymbol{y}_n|\boldsymbol{x}_n), \quad n = 2, \ldots, N. \tag{16.37}$$

The junction tree of Figure 16.19b results by eliminating nodes from the cliques starting from $\mathbf{x}_1$. Note that the normalizing constant is equal to one, $Z = 1$.

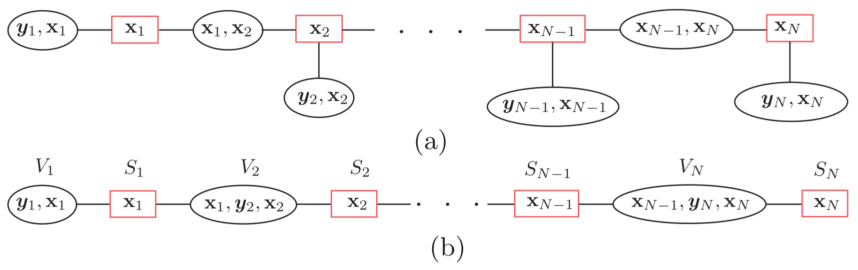


FIGURE 16.19

(a) The junction tree that results from the graph of Figure 16.15. (b) Because $\boldsymbol{y}_n$ are observed, their effect is only of a multiplicative nature (no marginalization is involved) and its contribution can be trivially absorbed by the potential functions (distributions) associated with the latent variables.

To apply the sum-product rule for junction trees, Eq. (16.5) now becomes

$$\begin{aligned}\mu_{V_n\to S_n}(\boldsymbol{x}_n) &= \sum_{\boldsymbol{x}_{n-1}} \psi_n(\boldsymbol{x}_{n-1},\boldsymbol{y}_n,\boldsymbol{x}_n)\mu_{S_{n-1}\to V_n}(\boldsymbol{x}_{n-1})\\ &= \sum_{\boldsymbol{x}_{n-1}} \mu_{S_{n-1}\to V_n}(\boldsymbol{x}_{n-1})P(\boldsymbol{x}_n|\boldsymbol{x}_{n-1})p(\boldsymbol{y}_n|\boldsymbol{x}_n).\end{aligned}$$

Also,

$$\mu_{S_{n-1}\to V_n}(\boldsymbol{x}_{n-1}) = \mu_{V_{n-1}\to S_{n-1}}(\boldsymbol{x}_{n-1}). \tag{16.38}$$

Thus,

$$\mu_{V_n\to S_n}(\boldsymbol{x}_n) = \sum_{\boldsymbol{x}_{n-1}} \mu_{V_{n-1}\to S_{n-1}}(\boldsymbol{x}_{n-1})P(\boldsymbol{x}_n|\boldsymbol{x}_{n-1})p(\boldsymbol{y}_n|\boldsymbol{x}_n), \tag{16.39}$$

with

$$\mu_{V_1\to S_1}(\boldsymbol{x}_1) = P(\boldsymbol{x}_1)p(\boldsymbol{y}_1|\boldsymbol{x}_1). \tag{16.40}$$

In the HMM literature, it is common to use the "alpha" symbol for the exchanged messages, that is,

$$\alpha(\boldsymbol{x}_n) := \mu_{V_n\to S_n}(\boldsymbol{x}_n). \tag{16.41}$$

If one considers that the message-passing terminates at a node V_n, then based on (16.7), and taking into account that the variables $\boldsymbol{y}_1,\ldots,\boldsymbol{y}_n$, are clumped to the observed values (recall related comment following Eq. (15.44)), it is readily seen that

$$\alpha(\boldsymbol{x}_n) = p(\boldsymbol{y}_1,\boldsymbol{y}_2,\ldots,\boldsymbol{y}_n,\boldsymbol{x}_n), \tag{16.42}$$

which can also be deduced by the respective definitions in (16.39, 16.40); all hidden variables, except $\mathbf{x}_n$, have been marginalized out. This is a set of K probability values (one for each value of $\boldsymbol{x}_n$). For example, for $\boldsymbol{x}_n : \boldsymbol{x}_{n,k} = 1$, $\alpha(\boldsymbol{x}_n)$ is the probability of the trajectory to be at time n at state k, *and* having obtained the specific observations up to and including time n. From (16.42), one can readily obtain the joint probability distribution (evidence) over the observation sequence, comprising N time instants, that is,

$$\boxed{p(Y) = \sum_{\boldsymbol{x}_N} p(\boldsymbol{y}_1,\boldsymbol{y}_2,\ldots,\boldsymbol{y}_N,\boldsymbol{x}_N) = \sum_{\boldsymbol{x}_N}\alpha(\boldsymbol{x}_N) : \text{ Evidence of Observations,}}$$

which, as said in the beginning of the section, is a quantity used for classification/recognition.

In the signal processing "jargon," the computation of $\alpha(\boldsymbol{x}_n)$ is referred to as the *filtering* recursion. By the definition of $\alpha(\boldsymbol{x}_n)$, we have that [2]

$$\boxed{\alpha(\boldsymbol{x}_n) = \underbrace{p(\boldsymbol{y}_n|\boldsymbol{x}_n)}_{\text{corrector}} \cdot \underbrace{\sum_{\boldsymbol{x}_{n-1}} \alpha(\boldsymbol{x}_{n-1})P(\boldsymbol{x}_n|\boldsymbol{x}_{n-1})}_{\text{predictor}} : \quad \text{Filtering Recursion.}} \tag{16.43}$$

As it is the case with the Kalman filter to be treated in Chapter 17, the only difference there will be that the summation is replaced by integration. Having adopted Gaussian distributions, these integrations translate into updates of the respective mean values and covariance matrices. The physical meaning of (16.43) is that the predictor provides a prediction on the state using all the past information prior to n.

Then this information is corrected based on the observation $\boldsymbol{y}_n$, which is received at time n. Thus, the updated information, based on the entire observation sequence up to and including the current time n, is readily available by

$$P(\boldsymbol{x}_n|Y_{[1:n]}) = \frac{\alpha(\boldsymbol{x}_n)}{p(Y_{[1:n]})},$$

where the denominator is given by $\sum_{\boldsymbol{x}_n} \alpha(\boldsymbol{x}_n)$, and $Y_{[1:n]} := (\boldsymbol{y}_1, \dots, \boldsymbol{y}_n)$.

Let us now carry on with the second message-passing phase, in the opposite direction than before, in order to obtain

$$\mu_{V_{n+1}\to S_n}(\boldsymbol{x}_n) = \sum_{\boldsymbol{x}_{n+1}} \mu_{S_{n+1}\to V_{n+1}}(\boldsymbol{x}_{n+1}) P(\boldsymbol{x}_{n+1}|\boldsymbol{x}_n) p(\boldsymbol{y}_{n+1}|\boldsymbol{x}_{n+1}),$$

with

$$\mu_{S_{n+1}\to V_{n+1}}(\boldsymbol{x}_{n+1}) = \mu_{V_{n+2}\to S_{n+1}}(\boldsymbol{x}_{n+1}).$$

Hence,

$$\mu_{V_{n+1}\to S_n}(\boldsymbol{x}_n) = \sum_{\boldsymbol{x}_{n+1}} \mu_{V_{n+2}\to S_{n+1}}(\boldsymbol{x}_{n+1}) P(\boldsymbol{x}_{n+1}|\boldsymbol{x}_n) p(\boldsymbol{y}_{n+1}|\boldsymbol{x}_{n+1}), \tag{16.44}$$

with

$$\mu_{V_{N+1}\to S_N}(\boldsymbol{x}_N) = 1. \tag{16.45}$$

Note that $\mu_{V_{n+1}\to S_n}(\boldsymbol{x}_n)$ involves K values and for the computation of each one of them K summations are performed. So, the complexity scales as $\mathcal{O}(K^2)$ per time instant. In HMM literature, the symbol "beta" is used,

$$\beta(\boldsymbol{x}_n) = \mu_{V_{n+1}\to S_n}(\boldsymbol{x}_n). \tag{16.46}$$

From the recursive definition in (16.44, 16.45), where $\boldsymbol{x}_{n+1}, \boldsymbol{x}_{n+2}, \dots, \boldsymbol{x}_N$ have been marginalized out, we can equivalently write

$$\beta(\boldsymbol{x}_n) = p(\boldsymbol{y}_{n+1}, \boldsymbol{y}_{n+2}, \dots, \boldsymbol{y}_N|\boldsymbol{x}_n). \tag{16.47}$$

That is, conditioned on the values of $\boldsymbol{x}_n$, for example, $\boldsymbol{x}_n : x_{nk} = 1$, $\beta(\boldsymbol{x}_n)$ is the value of the joint distribution for the observed values, $\boldsymbol{y}_{n+1}, \dots, \boldsymbol{y}_N$, to be emitted when the system is at state k at time n.

We have now all the "ingredients" in order to compute marginals. From (16.7), we obtain (justify it based on the independence properties that underlie an HMM)

$$\begin{aligned} p(\boldsymbol{x}_n, \boldsymbol{y}_1, \boldsymbol{y}_2, \dots, \boldsymbol{y}_N) &= \mu_{V_{n-1}\to S_n}(\boldsymbol{x}_n)\mu_{V_{n+1}\to S_n}(\boldsymbol{x}_n) \\ &= \alpha(\boldsymbol{x}_n)\beta(\boldsymbol{x}_n), \end{aligned} \tag{16.48}$$

which in turns leads to

$$\boxed{\gamma(\boldsymbol{x}_n) := P(\boldsymbol{x}_n|Y) = \frac{\alpha(\boldsymbol{x}_n)\beta(\boldsymbol{x}_n)}{p(Y)} : \quad \text{Smoothing Recursion.}} \tag{16.49}$$

This part of the recursion is known as the *smoothing* recursion. Note that in this computation, both past (via $\alpha(\boldsymbol{x}_n)$) and future (via $\beta(\boldsymbol{x}_n)$) data are involved.

An alternative way to obtain $\gamma(\boldsymbol{x}_n)$ is via its own recursion together with $\alpha(\boldsymbol{x}_n)$, by avoiding $\beta(\boldsymbol{x}_n)$ (Problem 16.16). In such a scenario, both passing messages are related to densities with regard to $\boldsymbol{x}_n$, which has certain advantages for the case of linear dynamic systems.

Finally, from (16.6) and recalling (16.38), (16.41), and (16.46), we obtain

$$p(\boldsymbol{x}_{n-1}, \boldsymbol{x}_n, Y) = P(\boldsymbol{x}_n|\boldsymbol{x}_{n-1})p(\boldsymbol{y}_n|\boldsymbol{x}_n)\mu_{S_n \to V_n}(\boldsymbol{x}_n)\mu_{S_{n-1}\to V_n}(\boldsymbol{x}_{n-1})$$

$$= \alpha(\boldsymbol{x}_{n-1})P(\boldsymbol{x}_n|\boldsymbol{x}_{n-1})p(\boldsymbol{y}_n|\boldsymbol{x}_n)\beta(\boldsymbol{x}_n), \tag{16.50}$$

or

$$p(\boldsymbol{x}_{n-1}, \boldsymbol{x}_n|Y) = \frac{\alpha(\boldsymbol{x}_{n-1})P(\boldsymbol{x}_n|\boldsymbol{x}_{n-1})p(\boldsymbol{y}_n|\boldsymbol{x}_n)\beta(\boldsymbol{x}_n)}{p(Y)}$$

$$:= \xi(\boldsymbol{x}_{n-1}, \boldsymbol{x}_n). \tag{16.51}$$

Thus, $\xi(\cdot,\cdot)$ is a table of K^2 probability values. Let $\xi(x_{n-1,j}, x_{n,i})$ correspond to $x_{n-1,j} = x_{n,i} = 1$. Then $\xi(x_{n-1,j}, x_{n,i})$ is the probability of the system being at states j and i at times $n-1$ and n, respectively, conditioned on the transmitted sequence of observations.

In Section 15.7.4 a message-passing scheme was proposed for the efficient computation of the maximum of the joint distribution. This can also be applied in the junction tree associated with an HMM. The resulting algorithm is known as the *Viterbi* algorithm. The Viterbi algorithm results in a straightforward way from the general max-sum algorithm. The algorithm is similar to the one derived before; all one has to do is to replace summations with the maximum operations. As we have already commented, while discussing the max-product rule, computing the sequence of the complete set $(\boldsymbol{y}_n, \boldsymbol{x}_n)$, $n = 1, 2, \ldots, N$, that maximizes the joint probability, using back-tracking, equivalently defines the optimal trajectory in the two-dimensional grid.

Another inference task that is of interest in practice, besides recognition, is prediction. That is, given an HMM and the observation sequence $\boldsymbol{y}_n$, $n = 1, 2, \ldots, N$, to optimally predict the value $\boldsymbol{y}_{n+1}$. This can also be performed efficiently by appropriate marginalization (Problem 16.17).

16.5.2 LEARNING THE PARAMETERS IN AN HMM

This is the second time we refer to the learning of graphical models. The first time was at the end of Section 16.3.2. The most natural way to obtain the unknown parameters is to maximize the likelihood/evidence of the joint probability distribution. Because our task involves both observed as well as latent variables, the EM algorithm is the first one that comes to mind. However, the underlying independencies in an HMM will be employed in order to come up with an efficient learning scheme. The set of the unknown parameters, $\boldsymbol{\Theta}$, involves (a) the initial state probabilities, P_k, $k = 1, \ldots, K$, (b) the transition probabilities, P_{ij}, $i, j = 1, 2, \ldots, K$, and (c) the parameters in the probability distributions associated with the observations, $\boldsymbol{\theta}_k$, $k = 1, 2, \ldots, K$.

Expectation Step: From the general scheme presented in Section 12.5.1 (with Y in place of $\mathcal{X}$, X in place of $\mathcal{X}^l$, and Θ in place of $\boldsymbol{\xi}$) at the $(t+1)$th iteration, we have to compute

$$\mathcal{Q}(\Theta, \Theta^{(t)}) = \mathbb{E}\left[\ln p(Y, X; \Theta)\right]$$

where $\mathbb{E}[\cdot]$ is the expectation with respect to $P(X|Y;\Theta^{(t)})$. From (16.32), (16.33), (16.34), and (16.35) we obtain

$$\begin{aligned}\ln p(Y,X;\Theta) &= \sum_{k=1}^{K}\big(x_{1,k}\ln P_k + \ln p(\boldsymbol{y}_1|k;\boldsymbol{\theta}_k)\big)\\ &+\sum_{n=2}^{N}\sum_{i=1}^{K}\sum_{j=1}^{K}(x_{n-1,j}x_{n,i})\ln P_{ij}\\ &+\sum_{n=2}^{N}\sum_{k=1}^{K}x_{n,k}\ln p(\boldsymbol{y}_n|k;\boldsymbol{\theta}_k),\end{aligned}$$

thus,

$$\begin{aligned}Q(\Theta,\Theta^{(t)}) = \sum_{k=1}^{K}\mathbb{E}[\mathrm{x}_{1,k}]\ln P_k + \sum_{n=2}^{N}\sum_{i=1}^{K}\sum_{j=1}^{K}\mathbb{E}[\mathrm{x}_{n-1,j}\mathrm{x}_{n,i}]\ln P_{ij}\\ +\sum_{n=1}^{N}\sum_{k=1}^{K}\mathbb{E}[\mathrm{x}_{n,k}]\ln p(\boldsymbol{y}_n|k;\boldsymbol{\theta}_k).\end{aligned}\tag{16.52}$$

Let us now recall (16.49) to obtain

$$\mathbb{E}[\mathrm{x}_{n,k}] = \sum_{\boldsymbol{x}_n}P(\boldsymbol{x}_n|Y;\Theta^{(t)})x_{n,k} = \sum_{\boldsymbol{x}_n}\gamma(\boldsymbol{x}_n;\Theta^{(t)})x_{n,k}.$$

Note that $x_{n,k}$ can either be zero or one; hence, its mean value will be equal to the probability that $\boldsymbol{x}_n$ has the kth element $x_{n,k}=1$ and we denote it as

$$\mathbb{E}[\mathrm{x}_{n,k}] = \gamma(x_{n,k}=1;\Theta^{(t)}),\tag{16.53}$$

Recall that given $\Theta^{(t)}$, $\gamma(\cdot;\Theta^{(t)})$ can be efficiently computed via the sum-product algorithm described before. In a similar spirit and mobilizing the definition in (16.51), we can write

$$\begin{aligned}\mathbb{E}[\mathrm{x}_{n-1,j}\mathrm{x}_{n,i}] &= \sum_{\boldsymbol{x}_n}\sum_{\boldsymbol{x}_{n-1}}P(\boldsymbol{x}_n,\boldsymbol{x}_{n-1}|Y;\Theta^{(t)})x_{n-1,j}x_{n,i}\\ &= \sum_{\boldsymbol{x}_n}\sum_{\boldsymbol{x}_{n-1}}\xi(\boldsymbol{x}_n,\boldsymbol{x}_{n-1};\Theta^{(t)})x_{n-1,j}x_{n,i}\\ &= \xi(x_{n-1,j}=1,x_{n,i}=1;\Theta^{(t)}).\end{aligned}\tag{16.54}$$

Note that $\xi(\cdot,\cdot;\Theta^{(t)})$ can also be efficiently computed as a by-product of the sum-product algorithm, given $\Theta^{(t)}$. Thus, we can summarize the E-step as

$$\begin{aligned}Q(\Theta,\Theta^{(t)}) &= \sum_{k=1}^{K}\gamma(x_{1,k}=1;\Theta^{(t)})\ln P_k\\ &+\sum_{n=2}^{N}\sum_{i=1}^{K}\sum_{j=1}^{K}\xi(x_{n-1,j}=1,x_{n,i}=1;\Theta^{(t)})\ln P_{ij}\\ &+\sum_{n=1}^{N}\sum_{k=1}^{K}\gamma(x_{n,k}=1;\Theta^{(t)})\ln p(\boldsymbol{y}_n|k;\boldsymbol{\theta}_k).\end{aligned}\tag{16.55}$$

Maximization Step: In this step, it suffices to obtain the derivatives/gradients with regard to P_k, P_{ij}, and $\boldsymbol{\theta}_k$ and equate them to zero in order to obtain the new estimates, which will comprise $\Theta^{(t+1)}$. Note that P_k and P_{ij} are probabilities; hence, their maximization should be constrained so that

$$\sum_{k=1}^{K} P_k = 1 \quad \text{and} \quad \sum_{i=1}^{K} P_{ij} = 1, \quad j = 1, 2, \ldots K.$$

The resulting reestimation formulas are (Problem 16.18)

$$P_k^{(t+1)} = \frac{\gamma(x_{1,k} = 1; \Theta^{(t)})}{\sum_{i=1}^{K} \gamma(x_{1,i} = 1; \Theta^{(t)})}, \tag{16.56}$$

$$P_{ij}^{(t+1)} = \frac{\sum_{n=2}^{N} \xi(x_{n-1,j} = 1, x_{n,i} = 1; \Theta^{(t)})}{\sum_{n=2}^{N} \sum_{k=1}^{K} \xi(x_{n-1,j} = 1, x_{n,k} = 1; \Theta^{(t)})}. \tag{16.57}$$

The reestimation of $\boldsymbol{\theta}_k$ depends on the form of the corresponding distribution $p(\boldsymbol{y}_n|k; \boldsymbol{\theta}_k)$. For example, in the Gaussian scenario, the parameters are the mean values and the elements of the covariance matrix. In this case, we obtain exactly the same iterations as those resulting for the problem of Gaussian mixtures (see Eqs. (12.87) and (12.88)), if in place of the posterior we use γ.

In summary, training an HMM comprises the following steps:

1. Initialize the parameters in Θ.
2. Run the sum-product algorithm to obtain $\gamma(\cdot)$ and $\xi(\cdot, \cdot)$, using the current set of parameter estimates.
3. Update the parameters as in (16.56) and (16.57).

Iterations in steps 2 and 3 continue until a convergence criterion is met, such as in EM. This iterative scheme is also known as the *Baum-Welch* or *forward-backward* algorithm. Besides the forward-backward algorithm for training HMMs, the literature is rich in a number of alternatives with the goal of either simplifying computations or improving performance. For example, a simpler training algorithm can be derived tailored to the Viterbi scheme for computing the optimum path (e.g., [63, 72]). Also, to further simplify the training algorithm, we can assume that our state observation variables, $\boldsymbol{y}_n$, are discretized (quantized) and can take values from a finite set of L possible ones, $\{1, 2, \ldots, L\}$. This is often the case in practice. Furthermore, assume that the first state is also known. This is, for example, the case for left-to-right models like the one shown in Figure 16.17. In such a case, we need not compute estimates of the initial probabilities. Thus, the unknown parameters to be estimated are the transition probabilities and the probabilities $P_{\boldsymbol{y}}(r|i)$, $r = 1, 2, \ldots, L$, $i = 1, 2, \ldots, K$; that is, the probability of emitting symbol r from state i.

Viterbi reestimation: In the speech literature, the algorithm is also known as the *segmental k-means training* algorithm [63]. It evolves around the concept of the best path.
Definitions:

- $n_{i|j}$:= number of transitions from state j to state i.
- $n_{\cdot|j}$:= number of transitions originated from state j.
- $n_{i|\cdot}$:= number of transitions terminated at state i.
- $n(r|i)$:= number of times observation $r \in \{1, 2, \ldots, L\}$ occurs jointly with state i.

Iterations:

- Initial conditions: Assume the initial estimates of the unknown parameters. Obtain the best path and compute the associated cost, say D, along the path.
- Step 1: From the available best path, reestimate the new model parameters as

$$P^{(\text{new})}(i|j) = \frac{n_{i|j}}{n_{\cdot|j}}$$

$$P_{\boldsymbol{x}}^{(\text{new})}(r|i) = \frac{n(r|i)}{n_{i|\cdot}}$$

- Step 2: For the new model parameters, obtain the best path and compute the corresponding overall cost $D^{(\text{new})}$. Compare it with the cost D of the previous iteration. If $D^{(\text{new})} - D > \epsilon$, set $D = D^{(\text{new})}$ and go to step 1. Otherwise stop.

The Viterbi reestimation algorithm can be shown to converge to a proper characterization of the underlying observations [14].

Remarks 16.4.

- *Scaling*: The probabilities, α and β, being less than one, as iterations progress can take very small values. In practice, the dynamic range of their computed values may exceed that of the computer. This phenomenon can be efficiently dealt within an appropriate scaling. If this is done properly on both α and β, then the effect of scaling cancels out [63].
- *Insufficient Training Data Set*: Generally, a large amount of training data is necessary to learn the HMM parameters. The observation sequence must be sufficiently long with respect to the number of states of the HMM model. This will guarantee that all state transitions will appear a sufficient number of times, so that the reestimation algorithm learns their respective parameters. If this is not the case, a number of techniques have been devised to cope with the issue. For a more detailed treatment, the reader may consult [8, 63] and the references therein.

16.5.3 DISCRIMINATIVE LEARNING

Discriminative learning is another path that has attracted a lot of attention. Note that the EM algorithm optimizes the likelihood with respect to the unknown parameters of a *single* HMM in “isolation”; that is, without considering the rest of the HMMs, which model the other words (in case of speech recognition) or other templates/prototypes that are stored in the database. Such an approach is in line with what we defined as generative learning in Chapter 3. In contrast, the essence of discriminative learning is to optimize the set of parameters so that the models become optimally discriminated over the training sets (e.g., in terms of the error probability criterion). In other words, the parameters describing the different statistical models (HMMs) are optimized in a *combined* way, not individually. The goal is to make the different HMM models as distinct as possible, according to a criterion. This has been an intense line of research and a number of techniques have been developed around criteria that lead to either convex or nonconvex optimization methods; see, for example, [33] and the references therein.

Remarks 16.5.

- Besides the basic HMM scheme, which was described in this section, a number of variants have been proposed in order to overcome some of its shortcomings. For example, alternative modeling paths concern the first-order Markov property and propose models to extend correlations to longer times.

In the *autoregressive HMM* [11], links are added among the observation nodes of the basic HMM scheme in Figure 16.15; for example, $\mathbf{y}_n$ is not only linked to $\mathbf{x}_n$ but it shares direct links with, for example, $\mathbf{y}_{n-2}$, $\mathbf{y}_{n-1}$, $\mathbf{y}_{n+1}$, and $\mathbf{y}_{n+2}$, if the model extends correlations up to two time instants away.
A different concept has been introduced in [56] in the context of *segment modeling*. According to this model, each state is allowed to emit, say, d, successive observations, that comprise a segment. The length of the segment, d, is itself a random variable and it is associated with a probability $P(d|k),\ k = 1, 2, \ldots, K$. In this way, correlation is introduced via the joint distribution of the samples comprising the segment.

- *Variable Duration HMM*: A serious shortcoming of the HMMs, which is often observed in practice, is associated with the self transition probabilities, $P(k|k)$, which are among the model parameters associated with an HMM. Note that the probability of the model being at state k for d successive instants (initial transition to the state and $d - 1$ self transitions) is given by

$$P_k(d) = (P(k|k))^{d-1}(1 - P(k|k)),$$

where $1 - P(k|k)$ is the probability of leaving the state. For many cases, this exponential state duration dependence is not realistic. In variable duration HMMs, $P_k(d)$ is explicitly modeled. Different models for $P_k(d)$ can be employed (see, e.g., [46, 68, 72]).

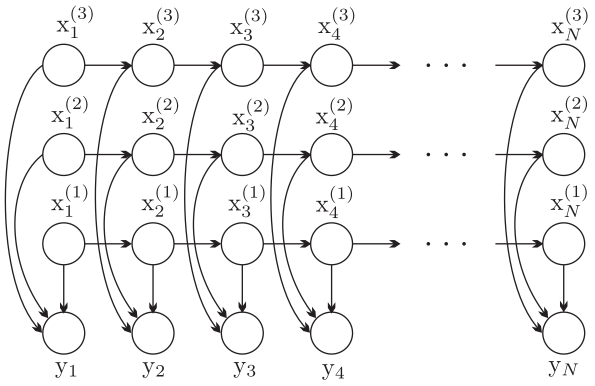
- Hidden Markov modeling is among the most powerful tools in machine learning and has been widely used in a large number of applications besides speech recognition. Some sampled references are [9] in bioinformatics, [16, 36] in communications, [4, 73] in optical character recognition (OCR), and [40, 61, 62] in music analysis/recognition, to name but a few.
For a further discussion on HMMs, see, for example, [8, 64, 72].

16.6 BEYOND HMMS: A DISCUSSION

In this section, some notable extensions of the hidden Markov models, which were previously discussed, are considered in order to meet requirements of applications where either the number of states is large or the homogeneity assumption is no more justified.

16.6.1 FACTORIAL HIDDEN MARKOV MODELS

In the HMMs considered before, the system dynamics is described via the hidden variables, whose graphical representation is a chain. However, such a model may turn out to be too simple for certain applications. A variant of the HMM involves M chains, instead of one chain, where each chain of hidden variables unfolds in time independently of the others. Thus at time n, M hidden variables are involved, denoted as $\mathbf{x}_n^{(m)}$, $m = 1, 2, \ldots, M$, [17, 34, 81]. The observations occur as a combined emission where all hidden variables are involved. The respective graphical structure is shown in Figure 16.20 for $M = 3$. Each one of the chains develops on its own, as the graphical model suggests. Such models are known as *factorial HMM* (FHMM). One obvious question is why not use a single chain of hidden variables by increasing the number of possible states? It turns out that such a naive approach would blow up complexity. Take as an example the case of $M = 3$, where for each one of the hidden variables, the number of states is equal to 10. The table of transition probabilities for



$x_1^{(3)}$ $x_2^{(3)}$ $x_3^{(3)}$ $x_4^{(3)}$ $x_N^{(3)}$
$x_1^{(2)}$ $x_2^{(2)}$ $x_3^{(2)}$ $x_4^{(2)}$ $x_N^{(2)}$
$x_1^{(1)}$ $x_2^{(1)}$ $x_3^{(1)}$ $x_4^{(1)}$ $x_N^{(1)}$
y_1 y_2 y_3 y_4 y_N

FIGURE 16.20

A factorial HMM with three chains of hidden variables.

each chain requires 10^2 entries which amounts to a total number of 300; i.e., $P_{ij}^{(m)}$, $i, j = 1, 2, \ldots, 10$, $m = 1, 2, 3$. Moreover, the total number of state combinations, which can be realized is $10^3 = 1000$. To implement the same number of states via a single chain one would need a table of transition probabilities equal to $(10^3)^2 = 10^6$!

Let $\mathbb{X}_n$ be the M-tuple $(\boldsymbol{x}_n^{(1)}, \ldots, \boldsymbol{x}_n^{(M)})$, where each $\boldsymbol{x}_n^{(m)}$ has only one of its elements equal to 1 (indicating a state) and the rest are zero. Then,

$$P(\mathbb{X}_n|\mathbb{X}_{n-1}) = \prod_{m=1}^{M} P^{(m)}\left(\boldsymbol{x}_n^{(m)}|\boldsymbol{x}_{n-1}^{(m)}\right).$$

In [17], the Gaussian distribution was employed for the observations, that is,

$$p(\boldsymbol{y}_n|\mathbb{X}_n) = \mathcal{N}\left(\boldsymbol{y}_n \middle| \sum_{m=1}^{M} \mathcal{M}^{(m)}\boldsymbol{x}_n^{(m)}, \Sigma\right), \tag{16.58}$$

where

$$\mathcal{M}^{(m)} = \left[\boldsymbol{\mu}_1^{(m)}, \ldots, \boldsymbol{\mu}_K^{(m)}\right], \quad m = 1, 2, \ldots, M \tag{16.59}$$

are the matrices comprising the mean vectors associated with each state and the covariance matrix is assumed to be known and the same for all. The joint probability distribution is given by

$$p(\mathbb{X}_1 \ldots \mathbb{X}_N, Y) = \prod_{m=1}^{M}\left(P^{(m)}\left(\boldsymbol{x}_1^{(m)}\right)\prod_{n=2}^{N} P^{(m)}\left(\boldsymbol{x}_n^{(m)}|\boldsymbol{x}_{n-1}^{(m)}\right)\right) \times \prod_{n=1}^{N} p(\boldsymbol{y}_n|\mathbb{X}_n) \tag{16.60}$$

The challenging task in factional HMMs is complexity. This is illustrated in Figure 16.21, where the explosion in the size of cliques after performing the moralization and triangulation steps is readily deduced.

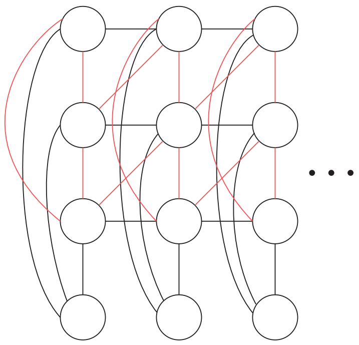


FIGURE 16.21

The graph resulting from a factorial HMM with three chains of hidden variables, after the moralization (it links variables in the same time instant) and triangulation (it links variables between neighboring time instants) steps.

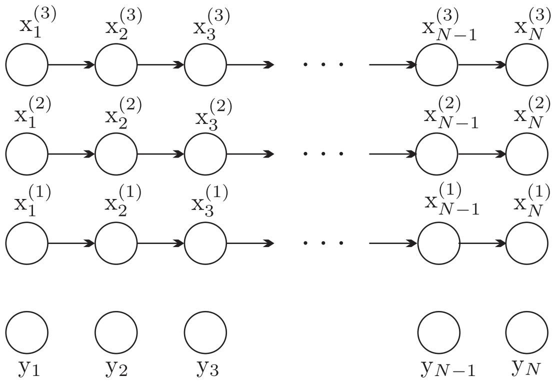


FIGURE 16.22

The simplified graphical structure of a FHMM comprising three chains used in the framework of variational approximation. The nodes associated with the observed variables are delinked.

In [17], the variational approximation method is adopted to simplify the structure. However, in contrast to the complete factorization scheme, which was adopted for the approximating distribution, Q, in Eq. (16.17) for the Boltzmann machine (corresponding to the removal of all edges in the graph), here the approximating graph will have a more complex structure. Only the edges connected to the output nodes are removed; this results in the graphical structure of Figure 16.22, for $M = 3$. Because this structure is tractable, there is no need for further simplifications. The approximate conditional distribution, Q, of the simplified structure is parameterized in terms of a set of variational parameters, $\lambda_n^{(m)}$ (one for each delinked node), and it is written as

$$Q(\mathbb{X}_1 \dots \mathbb{X}_N|Y;\boldsymbol{\lambda}) = \prod_{m=1}^{M} \left(\tilde{P}^{(m)} \left(\boldsymbol{x}_1^{(m)} \right) \prod_{n=2}^{N} \tilde{P}^{(m)} \left(\boldsymbol{x}_n^{(m)} | \boldsymbol{x}_{n-1}^{(m)} \right) \right), \tag{16.61}$$

where,

$$\tilde{P}^{(m)}\left(\boldsymbol{x}_n^{(m)}|\boldsymbol{x}_{n-1}^{(m)}\right) = P^{(m)}\left(\boldsymbol{x}_n^{(m)}|(\boldsymbol{x}_{n-1}^{(m)}\right)\lambda_n^{(m)}, \quad m = 2, \dots, M, \; n = 1, 2, \dots, N,$$

and

$$\tilde{P}^{(1)}(\boldsymbol{x}_1) = P^{(1)}(\boldsymbol{x}_1)\,\lambda_1^{(m)}.$$

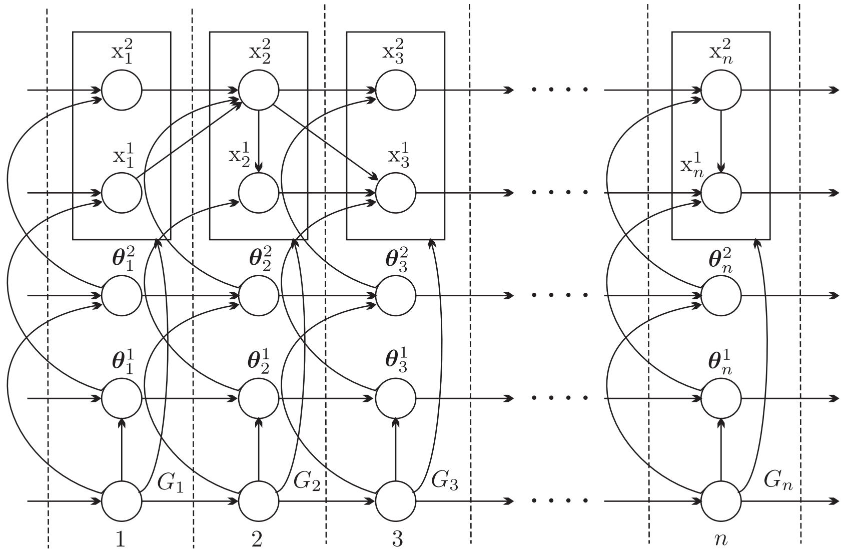
The variational parameters are estimated by minimizing the Kullback-Leibler distance between Q and the conditional distribution associated with (16.60). This compensates for some of the information loss caused by the removal of the observation nodes. The optimization process renders the variational parameters interdependent; this (deterministic) interdependence can be viewed as an approximation to the probabilistic dependence imposed by the exact structure prior to the approximation.

16.6.2 TIME-VARYING DYNAMIC BAYESIAN NETWORKS

Hidden Markov as well as factorial hidden Markov models are homogeneous; hence, both the structure and the parameters are fixed throughout time. However, such an assumption is not satisfying for a number of applications where the underlying relationships as well as the structural pattern of a system undergoes changes as time evolves. For example, the gene interactions do not remain the same throughout life; the appearance of an object across multiple cameras is continuously changing. For systems that are described by parameters whose values are varying slowly in an interval, we have already discussed a number of alternatives in previous chapters. The theory of graphical models provides the tools to study systems with a mixed set of parameters (discrete and continuous); also, graphical models lend themselves to modeling of nonstationary environments, where step changes are also involved.

One path toward time-varying modeling is to consider graphical models of fixed structure but with time varying parameters, known as *switching linear dynamic systems* (SLDS). Such models serve the needs of systems in which a linear dynamic model jumps from one parameter setting to another; hence, the latent variables are both of discrete as well as of continuous nature. At time instant n, a switch discrete variable, $\mathrm{s}_n \in \{1, 2, \dots, M\}$, selects a single LDS from an available set of M (sub)systems. The dynamics of s_n is also modeled to comply with the Markovian philosophy, and transitions from one LDS to another are governed by $P(\mathrm{s}_n|\mathrm{s}_{n-1})$. This problem has a long history and its origins can be traced back to the time just after the publication of the seminal paper by Kalman [35]; see, for example, [18] and [2, 3] for a more recent review of related techniques concerning the approximate inference task in such networks.

Another path is to consider that both the structure as well as the parameters change over time. One route is to adopt a quasi-stationary rationale, and assume that the data sequence is piece-wise stationary in time, for example, [12, 54, 66]. Nonstationarity is conceived as a cascade of stationary models, which have previously been learned by presegmented subintervals. The other route assumes that the structure and parameters are continuously changing, for example, [42, 79]. An example for the latter case is a Bayesian network where the parents of each node and the parameters, which define the conditional distributions, are time varying. A separate variable is employed that defines the structure at each time



x_1^2
x_2^2
x_3^2
x_n^2
x_1^1
x_2^1
x_3^1
x_n^1
$\boldsymbol{\theta}_1^2$
$\boldsymbol{\theta}_2^2$
$\boldsymbol{\theta}_3^2$
$\boldsymbol{\theta}_n^2$
$\boldsymbol{\theta}_1^1$
$\boldsymbol{\theta}_2^1$
$\boldsymbol{\theta}_3^1$
$\boldsymbol{\theta}_n^1$
G_1
G_2
G_3
G_n
1
2
3
n

FIGURE 16.23

The figure corresponds to a time varying dynamic Bayesian network with two variables x_n^1 and x_n^2, $n = 1, 2, \ldots$. The parameters controlling the conditional distributions are considered as separate nodes, $\boldsymbol{\theta}_n^1$ and $\boldsymbol{\theta}_n^2$, respectively, for the two variables. The structure variable, G_n, controls the values of the parameters as well as the structure of the network, which is continuously changing.

instant; that is, the set of linking directed edges. The concept is illustrated in Figure 16.23. The method has been applied to the task of active camera tracking [79].

16.7 LEARNING GRAPHICAL MODELS

Learning a graphical model consists of two parts. Given a number of observations, one has to specify both the graphical structure as well as the associated parameters.

16.7.1 PARAMETER ESTIMATION

Once a graphical model has been adopted, one has to estimate the unknown parameters. For example, in a Bayesian network involving discrete variables one has to estimate the values of the conditional probabilities. In Section 16.5, the case of learning the unknown parameters in the context of an HMM was presented. The key point was to maximize the joint pdf over the observed output variables. This is among the most popular criteria used for parameter estimation in different graphical structures. In the HMM case, some of the variables were latent, hence the EM algorithm was mobilized. If all the variables of the graph can be observed, then the task of parameter learning becomes a typical maximum likelihood one. More specifically, let a network with l nodes representing the variables, $\mathrm{x}_1, \dots, \mathrm{x}_l$, which are compactly written as a random vector $\mathbf{x}$. Let also, $\boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_N$, be a set of observations; then

$$\hat{\boldsymbol{\theta}} = \arg\max_{\Theta} p(\boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_N; \boldsymbol{\theta}),$$

where $\boldsymbol{\theta}$ comprises all the parameters in the graph. If latent variables are involved, then one has to marginalize them out. Any of the parameter estimation techniques that were discussed in Chapters 12 and 13 can be used. Moreover, one can take advantage of the special structure of the graph (i.e., the underlying independencies) to simplify computations. In the HMM case, its Bayesian network structure was exploited by bringing the sum-product algorithm into the game. Besides maximum likelihood, one can adopt any other method related to parameter estimation/inference. For example, one can impose a prior $p(\boldsymbol{\theta})$ on the unknown parameters and resort to a MAP estimation. Moreover, the full Bayesian scenario can also be employed and assume the parameters to be random variables. Such a line presupposes that the unknown parameters have been included as extra nodes to the network, linked appropriately to those of the variables that they affect. As a matter of fact, this is what we did in Figure 13.2, although there, we had not talked about graphical models yet (see also Figure 16.24). Note that in this case, in order to perform any inference on the variables of the network one should marginalize out the parameters. For example, assume that our l variables correspond to the nodes of a Bayesian network, where the local conditional distributions,

$$p(x_i|\mathrm{Pa}_i; \boldsymbol{\theta}_i), \quad i = 1, 2, \dots, l,$$

depend on the parameters $\boldsymbol{\theta}_i$. Also, assume that the (random) parameters Θ_i, $i = 1, 2, \dots, l$, are mutually independent. Then, the joint distribution over the variables is given by

$$p(x_1, x_2, \dots, x_l) = \prod_{i=1}^{l} \int_{\boldsymbol{\theta}_i} p(x_i|\mathrm{Pa}_i; \boldsymbol{\theta}_i) p(\boldsymbol{\theta}_i)\, \mathrm{d}\boldsymbol{\theta}_i.$$

Using convenient priors, that is, conjugate priors, computations can be significantly facilitated; we have demonstrated such examples in Chapters 12 and 13.

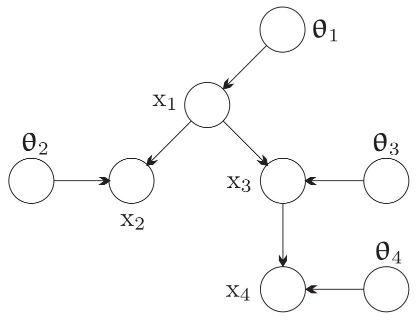


FIGURE 16.24

An example of a Bayesian network, where new nodes associated with the parameters have been included in order to treat parameters as random variables, as it is required by the Bayesian parameter learning approach.

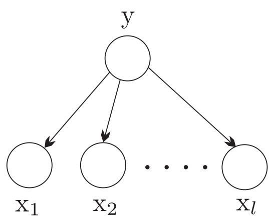


FIGURE 16.25

The Bayesian network associated with the naive Bayes classifier. The joint pdf factorizes as $p(y, x_1, \ldots, x_l) = p(y) \prod_{i=1}^{l} p(x_i|y)$.

Besides the previous techniques, which are off-springs of the generative modeling of the underlying processes, discriminative techniques have also been developed.

In a general setting, let us consider a pattern recognition task where the (output) label variable y, and the (input) feature variables, $\mathrm{x}_1, \ldots, \mathrm{x}_l$, are jointly distributed according to a distribution that can be factorized over a graph, which is parameterized in terms of a vector parameter $\boldsymbol{\theta}$; that is, $p(y, x_1, x_2, \ldots, x_l; \boldsymbol{\theta})$ [13]. A typical example of such modeling is the naive Bayes classifier, which was discussed in Chapter 7, whose graphical representation is given in Figure 16.25. For a given set of training data, $(\boldsymbol{y}_n, \boldsymbol{x}_n)$, $n = 1, 2, \ldots, N$, the log-likelihood function becomes

$$L(Y, X; \boldsymbol{\theta}) = \sum_{n=1}^{N} \ln p(y_n, x_n; \boldsymbol{\theta}). \tag{16.62}$$

Estimating $\boldsymbol{\theta}$ by maximizing $L(\cdot, \cdot; \boldsymbol{\theta})$, one would obtain an estimate that guarantees the best (according to the ML criterion) fit of the corresponding distribution to the available training set. However, our ultimate goal is not to model the generation "mechanism" of the data. Our ultimate goal is to classify them correctly. Let us rewrite (16.62) as

$$L(Y, X; \boldsymbol{\theta}) = \sum_{n=1}^{N} \ln P(y_n|x_n; \boldsymbol{\theta}) + \sum_{n=1}^{N} \ln p(x_n; \boldsymbol{\theta}).$$

Getting biased toward the classification task, it is more sensible to obtain $\boldsymbol{\theta}$ by maximizing the first of the two terms only, that is,

$$\begin{aligned} L_c(Y, X; \boldsymbol{\theta}) &= \sum_{n=1}^{N} \ln P(y_n|x_n; \boldsymbol{\theta}) \\ &= \sum_{n=1}^{N} \left(\ln p(y_n, x_n; \boldsymbol{\theta}) - \ln \sum_{y_n} p(y_n, x_n; \boldsymbol{\theta}) \right), \end{aligned} \tag{16.63}$$

where the summation over y_n is over all possible values of y_n (classes). This is known as the *conditional log-likelihood*, see, for example [19, 20, 67]. The resulting estimate, $\hat{\boldsymbol{\theta}}$, guarantees that overall the posterior class probabilities, given the feature values, are maximized over the training data set; after all, Bayesian classification is based on selecting the class of $\boldsymbol{x}$ according to the maximum of the posterior probability. However, one has to be careful. The price one pays for such approaches is that the conditional log-likelihood is not decomposable, and more sophisticated optimization schemes have to be mobilized. Maximizing the conditional log-likelihood does not guarantee that the error probability is also minimized. This can only be guaranteed if one estimates $\boldsymbol{\theta}$ so as to minimize the empirical error probability. However, such a criterion is hard to deal with, as it is not differentiable; attempts to deal with it by using approximate smoothing functions or hill-climbing greedy techniques have been proposed, for example, [58] and the references therein. Note that the rationale behind the conditional log-likelihood is closely related to that behind conditional random fields, discussed in Section 15.4.3.

Another route in discriminative learning is to obtain the estimate of $\boldsymbol{\theta}$ by *maximizing the margin*. The probabilistic class margin, for example, [21, 59], is defined as

$$\begin{aligned} d_n &= \min_{y \neq y_n} \frac{P(y_n|x_n;\boldsymbol{\theta})}{P(y|x_n;\boldsymbol{\theta})} = \frac{P(y_n|x_n;\boldsymbol{\theta})}{\max_{y\neq y_n} P(y|x_n;\boldsymbol{\theta})} \\ &= \frac{p(y_n,x_n;\boldsymbol{\theta})}{\max_{y\neq y_n} p(y,x_n;\boldsymbol{\theta})}. \end{aligned}$$

The idea is to estimate $\boldsymbol{\theta}$ so as to maximize the minimum margin over all training data, that is,

$$\hat{\boldsymbol{\theta}} = \arg\max_{\boldsymbol{\theta}} \min(d_1, d_2, \ldots, d_N).$$

The interested reader may also consult [10, 55, 60] for related reviews and methodologies.

Example 16.4. The goal of this example is to obtain the values in the conditional probability table in a general Bayesian network, which consists of l discrete random nodes/variables, $\mathrm{x}_1, \mathrm{x}_2, \ldots, \mathrm{x}_l$. We assume that all the involved variables can be observed and we have a training set of N observations. The maximum likelihood method will be employed. Let $x_i(n)$, $n = 1, 2, \ldots, N$, denote the nth observation of the ith variable.

The joint pdf under the Bayesian network assumption is given by

$$P(x_1, \ldots, x_l) = \prod_{i=1}^{l} P(x_i|\mathrm{Pa}_i;\boldsymbol{\theta}_i),$$

and the respective log-likelihood is

$$L(X;\boldsymbol{\theta}) = \sum_{n=1}^{N}\sum_{i=1}^{l} \ln P\big(x_i(n)|\mathrm{Pa}_i(n);\boldsymbol{\theta}_i\big).$$

Assuming $\boldsymbol{\theta}_i$ to be disjoint with $\boldsymbol{\theta}_j$, $i \neq j$, then optimization over each $\boldsymbol{\theta}_i$, $i = 1, 2, \ldots, l$, can take place separately. This property is referred to as the *global decomposition* of the likelihood function. Thus, it suffices to perform the optimization locally on each node, that is,

$$l(\boldsymbol{\theta}_i) = \sum_{n=1}^{N} \ln P\big(x_i(n)|\mathrm{Pa}_i(n);\boldsymbol{\theta}_i\big), \quad i = 1, 2, \ldots, l. \tag{16.64}$$

Let us now focus on the case where all the involved variables are discrete, and the unknown quantities at any node, i, are the values of the conditional probabilities in the respective conditional probability table. For notational convenience, denote as $\boldsymbol{h}_i$ the vector comprising the state indices of the parent variables of x_i. Then, the respective (unknown) probabilities are denoted as $P_{x_i|\boldsymbol{h}_i}(x_i, \boldsymbol{h}_i)$, for all possible combinations of values of x_i and $\boldsymbol{h}_i$. For example, if all the involved variables are binary and x_i has two parent nodes, then $P_{x_i|\boldsymbol{h}_i}(x_i, \boldsymbol{h}_i)$ can take a total of eight values that have to be estimated. Equation (16.64) can now be rewritten as

$$l(\boldsymbol{\theta}_i) = \sum_{\boldsymbol{h}_i} \sum_{x_i} s(x_i, \boldsymbol{h}_i) \ln P_{x_i|\boldsymbol{h}_i}(x_i, \boldsymbol{h}_i), \tag{16.65}$$

where $s(x_i, \boldsymbol{h}_i)$ is the number of times the specific combination of $(x_i, \boldsymbol{h}_i)$ appeared in the N samples of the training set. We assume that N is large enough so that all possible combinations occurred at least once, that is, $s(x_i, \boldsymbol{h}_i) \neq 0$, $\forall (x_i, \boldsymbol{h}_i)$. All one has to do now is to maximize (16.65) with respect to $P_{x_i|\boldsymbol{h}_i}(\cdot, \cdot)$, taking into account that

$$\sum_{x_i} P_{x_i|\boldsymbol{h}_i}(x_i, \boldsymbol{h}_i) = 1.$$

Note that $P_{x_i|\boldsymbol{h}_i}$ are independent for different values of $\boldsymbol{h}_i$. Thus, maximization of (16.65) can take place separately for each $\boldsymbol{h}_i$, and it is straightforward to see that

$$\hat{P}_{x_i|\boldsymbol{h}_i} = \frac{s(x_i, \boldsymbol{h}_i)}{\sum_{x_i} s(x_i, \boldsymbol{h}_i)}. \tag{16.66}$$

In words, the maximum likelihood estimate of the unknown conditional probabilities complies with our common sense; given a specific combination of the parent values, $\boldsymbol{h}_i$, $P_{x_i|\boldsymbol{h}_i}$ is approximated by the fraction of times the specific combination $(x_i, \boldsymbol{h}_i)$ appeared in the data set, over the total number of times $\boldsymbol{h}_i$ occurred (relate (16.66) to the Viterbi algorithm in Section 16.5.2). One can now see that in order to obtain good estimates, the number of training points, N, should be large enough so that each combination occurs a sufficiently large number of times. If the average number of parent nodes is large and/or the number of states is large, this poses heavy demands on the size of the training set. This is where parametrization of the conditional probabilities can prove to be very helpful.

16.7.2 LEARNING THE STRUCTURE

In the previous subsection, we considered the structure of the graph to be known and our task was to estimate the unknown parameters. We now turn our attention to learning the structure. In general, this is a much harder task. We only intend to provide a sketch of some general directions.

One path, known as *constrained-based*, is to try to build up a network that satisfies the data independencies, which are "measured" using different statistical tests on the training data set. The method relies a lot on intuition and such methods are not particularly popular in practice.

The other path comes under the name of *s-based* methods. This path treats the task as a typical model selection problem. The score that is chosen to be maximized provides a tradeoff between model complexity and accuracy of the fit to the data. Classical model fitting criteria such as Bayesian information criterion (BIC) and minimum description length (MDL) have been used among others. The main difficulty with all these criteria is that their optimization is an NP-hard task and the issue is to find appropriate approximate optimization schemes.

The third main path draws its existence from the Bayesian philosophy. Instead of a single structure, an ensemble of structures is employed by embedding appropriate priors into the problem. The readers who are interested in a further and deeper study are referred to more specialized books and papers, for example, [22, 41, 53].

PROBLEMS

16.1 Prove that an undirected graph is triangulated if and only if its cliques can be organized into a join tree.

16.2 For the graph of 16.3a give all possible perfect elimination sequences and draw the resulting sequence of graphs.

16.3 Derive the formulas for the marginal probabilities of the variables in (a) a clique node and (b) in a separator node in a junction tree.

16.4 Prove that in a junction tree the joint pdf of the variables is given by Eq. (16.8).

16.5 Show that obtaining the marginal over a single variable is independent of which one from the clique/separator nodes, which contain the variable, the marginalization is performed.
Hint. Prove it for the case of two neighboring clique nodes in the junction tree.

16.6 Consider the graph in Figure 16.26. Obtain a triangulated version of it.

16.7 Consider the Bayesian network structure given in Figure 16.27. Obtain an equivalent join tree.

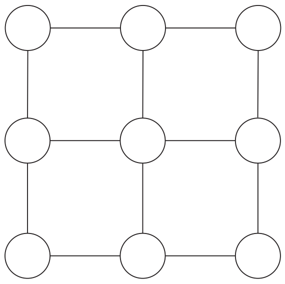


FIGURE 16.26

The graph for the Problem 16.6.

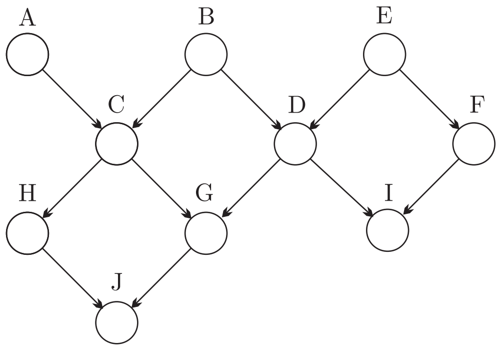


FIGURE 16.27

The Bayesian network structure for Problem 16.7.

16.8 Consider the random variables A, B, C, D, E, F, G, H, I, J and assume that the joint distribution is given by the product of the following potential functions

$$p = \frac{1}{Z}\psi_1(A,B,C,D)\psi_2(B,E,D)\psi_3(E,D,F,I)\psi_4(C,D,G)\psi_5(C,H,G,I).$$

Construct an undirected graphical model on which the previous joint probability factorizes and in the sequence derive an equivalent junction tree.

16.9 Prove that the function

$$g(x) = 1 - \exp(-x), \quad x > 0,$$

is log-concave.

16.10 Derive the conjugate function of

$$f(x) = \ln(1 - \exp(-x)).$$

16.11 Show that minimizing the bound in (16.12) is a convex optimization task.

16.12 Show that the function $1 + \exp(-x)$, $x \in \mathbb{R}$ is log-convex and derive the respective conjugate one.

16.13 Derive the KL divergence between $P(\mathcal{X}^l|\mathcal{X})$ and $Q(\mathcal{X}^l)$ for the mean field Boltzmann machine and obtain the respective l variational parameters.

16.14 Given a distribution in the exponential family,

$$p(\boldsymbol{x}) = \exp\left(\boldsymbol{\theta}^T\boldsymbol{u}(\boldsymbol{x}) - A(\boldsymbol{\theta})\right),$$

show that $A(\boldsymbol{\theta})$ generates the respective mean parameters that define the exponential family

$$\frac{\partial A(\boldsymbol{\theta})}{\partial \theta_i} = \mathbb{E}[u_i(\boldsymbol{x})] = \mu_i.$$

Also, show that $A(\boldsymbol{\theta})$ is a convex function.

16.15 Show that the conjugate function of $A(\boldsymbol{\theta})$, associated with an exponential distribution such as that in Problem 16.14, is the corresponding negative entropy function. Moreover, if $\boldsymbol{\mu}$ and $\boldsymbol{\theta}(\boldsymbol{\mu})$ are doubly coupled, then

$$\boldsymbol{\mu} = \mathbb{E}[\boldsymbol{u}(\boldsymbol{x})],$$

where $\mathbb{E}[\cdot]$ is with respect to $p(\boldsymbol{x};\boldsymbol{\theta}(\boldsymbol{\mu}))$.

16.16 Derive a recursion for updating $\gamma(\boldsymbol{x}_n)$ in HMMs independent on $\beta(\boldsymbol{x}_n)$.

16.17 Derive an efficient scheme for prediction in HMM models; that is, to obtain $p(\boldsymbol{y}_{N+1}|Y)$, where $Y = \{\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_N\}$.

16.18 Prove the estimation formulas for the probabilities P_k, $k = 1, 2, \ldots, K$, and P_{ij}, $i, j = 1, 2, \ldots, K$, in the context of the forward-backward algorithm for training HMM.

16.19 Consider the Gaussian Bayesian network of Section 15.3.5 defined by the local conditional pdfs

$$p(x_i|\text{Pa}_i) = \mathcal{N}\left(x_i \mid \sum_{k:x_k\in\text{Pa}_i} \theta_{ik}x_k + \theta_{i0}, \sigma^2\right), \quad i = 1, 2, \ldots, l.$$

Assume a set of N observations, $x_i(n)$, $n = 1, 2, \ldots, N$, $i = 1, 2, \ldots, l$, and derive a maximum likelihood estimate of the parameters $\boldsymbol{\theta}$; assume the common variance σ^2 to be known.

REFERENCES

[1] S. Arnborg, D. Cornell, A. Proskurowski, Complexity of finding embeddings in a k-tree, SIAM J. Algebr. Discrete Meth 8 (2) (1987) 277-284.

[2] D. Barber, A.T. Cemgil, Graphical models for time series, IEEE Signal Process. Mag. 27 (2010) 18-28.

[3] D. Barber, Bayesian Reasoning and Machine Learning, Cambridge University Press, Cambridge, 2013.

[4] R. Bartolami, H. Bunke, Hidden Markov model-based ensemble methods for off-line handwritten text line recognition, Pattern Recognit. 41 (11) (2008) 3452-3460.

[5] C.A. Berrou, A. Glavieux, P. Thitimajshima, Near Shannon limit error-correcting coding and decoding: turbo-codes, in: Proceedings IEEE International Conference on Communications, Geneva, Switzerland, 1993.

[6] L. Christensen, J. Zarsen, On data and parameter estimation using the variational Bayesian EM algorithm for block-fading frequency-selective MIMO channels, in: International Conference on Acoustics Speech and Signal Processing (ICASSP), vol. 4, 2006, pp. 465-468.

[7] R.G. Cowell, A.P. Dawid, S.L. Lauritzen, D.J. Spiegehalter, Probabilistic Networks and Expert Systems, Springer-Verlag, New York, 1999.

[8] J. Deller, J. Proakis, J.H.L. Hansen, Discrete-Time Processing of Speech Signals, Macmillan, New York, 1993.

[9] R. Durbin, S. Eddy, A. Krogh, G. Mitchison, Biological Sequence Analysis: Probabilistic Models of Proteins and Nuclear Acids, Cambridge University Press, Cambridge, 1998.

[10] J. Domke, Learning graphical parameters with approximate marginal inference, 2013, arXiv:1301.3193v1 [cs,LG] 15 January 2013.

[11] Y. Ephraim, D. Malah, B.H. Juang, On the application of hidden Markov models for enhancing noisy speech, IEEE Trans. Acoust. Speech Signal Process. 37 (12) (1989) 1846-1856.

[12] P. Fearhead, Exact and efficient Bayesian inference for multiple problems, Stat. Comput. 16 (2) (2006) 203-213.

[13] N. Friedman, D. Geiger, M. Goldszmidt, Bayesian network classifiers, Mach. Learn. 29 (1997) 131-163.

[14] K.S. Fu, Syntactic Pattern Recognition and Applications, Prentice Hall, Upper Saddle River, NJ, 1982.

[15] R.G. Gallager, Low density parity-check codes, IEEE Trans. Inform. Theory 2 (1968) 21-28.

[16] C. Georgoulakis, S. Theodoridis, Blind and Semi-blind equalization using hidden Markov models and clustering techniques, Signal Process. 80 (9) (2000) 1795-1805.

[17] Z. Ghahramani, M.I. Jordan, Factorial hidden Markov models, Mach. Learn. 29 (1997) 245-273.

[18] Z. Ghahramani, G.E. Hinton, Variational Learning for switching state space models, Neural Comput 12 (4) (1998) 963-996.

[19] R. Greiner, W. Zhou, Structural extension to logistic regression: Discriminative parameter learning of belief net classifiers, in: Proceedings 18th International Conference on Artificial Intelligence, 2002, pp. 167-173.

[20] D. Grossman, P. Domingos, Learning Bayesian network classifiers by maximizing conditional likelihood, in: Proceedings 21st International Conference on Machine Learning, Bauff, Canada, 2004.

[21] Y. Guo, D. Wilkinson, D. Schuurmans, Maximum margin Bayesian networks, in: Proceedings, International Conference on Uncertainty in Artificial Intelligence, 2005.

[22] D. Heckerman, D. Geiger, M. Chickering, Learning Bayesian networks: the combination of knowledge and statistical data, Mach. Learn. 20 (1995) 197-243.

[23] B. Hu, I. Land, L. Rasmussen, R. Piton, B. Fleury, A divergence minimization approach to joint multiuser decoding for coded CDMA, IEEE J. Select. Areas Commun. 26 (3) (2008) 432-445.

[24] A. Ihler, J.W. Fisher, P.L. Moses, A.S. Willsky, Nonparametric belief propagation for self-localization of sensor networks, J. Select. Areas Commun. 23 (4) (2005) 809-819.
[25] A. Ihler, D. McAllester, Particle belief propagation, in: International Conference on Artificial Intelligence and Statistics, 2009, pp. 256-263.
[26] S. Ikeda, T. Tanaka, S.I. Amari, Information geometry of turbo and low-density parity-check codes, IEEE Trans. Inform. Theory, 50 (6) (2004) 1097-1114.
[27] T.S. Jaakola, Variational Methods for Inference and Estimation in Graphical Models, PhD Thesis, Department of Brain and Cognitive Sciences, M.I.T., 1997.
[28] T.S. Jaakola, M.I. Jordan, Recursive algorithms for approximating probabilities in graphical models, in: M.C. Mozer, M.I. Jordan, T. Petsche (Eds.), Proceedings in Advances in Neural Information Processing Systems, NIPS, MIT Press, Cambridge, MA, 1997.
[29] T.S. Jaakola, M.I. Jordan, Improving the mean field approximation via the use of mixture distributions, in: M.I. Jordan (Ed.), Learning in Graphical Models, MIT Press, Cambridge, MA, 1999.
[30] T.S. Jaakola, M.I. Jordan, Variational methods and the QMR-DT database, J. Artif. Intell. Res. 10 (1999) 291-322.
[31] T.S. Jaakola, Tutorial on variational approximation methods, in: M. Opper, D. Saad (Eds.), Advanced Mean Field Methods: Theory and Practice, MIT Press, Cambridge, MA, 2001, pp. 129-160.
[32] F.V. Jensen, Bayesian Networks and Decision Graphs, Springer, New York, 2001.
[33] H. Jiang, X. Li, Parameter estimation of statistical models using convex optimization, IEEE Signal Process. Mag. 27 (3) (2010) 115-127.
[34] M.I. Jordan, Z. Ghahramani, T.S. Jaakola, L.K. Saul, An introduction to variational methods for graphical models, Mach. Learn. 37 (1999) 183-233.
[35] R.E. Kalman, A new approach to linear filtering and prediction problems, Trans. ASME J. Basic Eng. 82 (1960) 34-45.
[36] G.K. Kaleh, R. Vallet, Joint parameter estimation and symbol detection for linear and nonlinear channels, IEEE Trans. Commun. 42 (7) (1994) 2406-2414.
[37] R. Kikuchi, The theory of cooperative phenomena, Phys. Rev. 81 (1951) 988-1003.
[38] G.E. Kirkelund, C.N. Manchon, L.P.B. Christensen, E. Riegler, Variational message-passing for joint channel estimation and decoding in MIMO-OFDM, in: Proceedings, IEEE Globecom, 2010.
[39] U. Kjærulff, Triangulation of graphs: Algorithms giving small total state space, Technical Report, R90-09, Aalborg University, Denmark, 1990.
[40] A.P. Klapuri, A.J. Eronen, J.T. Astola, Analysis of the meter of acoustic musical signals, IEEE Trans. Audio Speech Lang. Process. 14 (1) (2006) 342-355.
[41] D. Koller, N. Friedman, Probabilistic Graphical Models: Principles and Techniques, MIT Press, Cambridge, MA, 2009.
[42] M. Kolar, L. Song, A. Ahmed, E.P. Xing, Estimating time-varying networks, Ann. Appl. Stat. 4 (2010) 94-123.
[43] J.B. Kruskal, On the shortest spanning subtree and the travelling salesman problem, Proc. Am. Math. Soc. 7 (1956) 48-50.
[44] S.L. Lauritzen, D.J. Spiegelhalter, Local computations with probabilities on graphical structures and their application to expert systems, J. R. Stat. Soc. B 50 (1988) 157-224.
[45] S.L. Lauritzen, Graphical Models, Oxford University Press, Oxford, 1996.
[46] S.E. Levinson, Continuously variable duration HMMs for automatic speech recognition, Comput. Speech Lang. 1 (1986) 29-45.
[47] D.J.C. MacKay, Good error-correcting codes based on very sparse matrices, IEEE Trans. Inform. Theory 45 (2) (1999) 399-431.
[48] Y. Mao, F.R. Kschischang, B. Li, S. Pasupathy, A factor graph approach to link loss monitoring in wireless sensor networks, J. Select. Areas Commun. 23 (4) (2005) 820-829.

[49] R.J. McEliece, D.J.C. MacKay, J.F. Cheng, Turbo decoding as an instance of Pearl's belief propagation algorithm, IEEE J. Select. Areas Commun. 16 (2) (1998) 140-152.
[50] T.P. Minka, Expectation propagation for approximate inference, in: Proceedings 17th Conference on Uncertainty in Artificial Intelligence, Morgan-Kaufmann, San Mateo, 2001, pp. 362-369.
[51] T.P. Minka, Divergence measures and message passing, Technical Report MSR-TR-2005-173, Microsoft Research Cambridge, 2005.
[52] K.P. Murphy, T. Weiss, M.J. Jordan, Loopy belief propagation for approximate inference: an empirical study, in: Proceedings 15th Conference on Uncertainties on Artificial Intelligence, 1999.
[53] K.P. Murphy, Machine Learning: A Probabilistic Perspective, MIT Press, Cambridge, MA, 2012.
[54] S.H. Nielsen, T.D. Nielsen, Adapting Bayesian network structures to non-stationary domains, Int. J. Approx. Reason. 49 (2) (2008) 379-397.
[55] S. Nowozin, C.H. Lampert, Structured learning and prediction in computer vision, Found. Trends Comput. Graph. Vis. 6 (2011) 185-365.
[56] M. Ostendorf, V. Digalakis, O. Kimball, From HMM's to segment models: a unified view of stochastic modeling for speech, IEEE Trans. Audio Speech Process. 4 (5) (1996) 360-378.
[57] J. Pearl, Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference, Morgan-Kaufmann, San Mateo, 1988.
[58] F. Pernkopf, J. Bilmes, Efficient heuristics for discriminative structure learning of Bayesian network classifiers, J. Mach. Learn. Res. 11 (2010) 2323-2360.
[59] F. Pernkopf, M. Wohlmayr, S. Tschiatschek, Maximum margin Bayesian network classifiers, IEEE Trans. Pattern Anal. Mach. Intell. 34 (3) (2012) 521-532.
[60] F. Pernkopf, R. Peharz, S. Tschiatschek, Introduction to probabilistic graphical models, in: R. Chellappa, S. Theodoridis (Eds.), E-Reference in Signal Processing, vol. 1, 2013.
[61] A. Pikrakis, S. Theodoridis, D. Kamarotos, Recognition of musical patterns using hidden Markov models, IEEE Trans. Audio Speech Lang. Process. 14 (5) (2006) 1795-1807.
[62] Y. Qi, J.W. Paisley, L. Carin, Music analysis using hidden Markov mixture models, IEEE Trans. Signal Process. 55 (11) (2007) 5209-5224.
[63] L. Rabiner, A tutorial on hidden Markov models and selected applications in speech processing, Proc. IEEE 77 (1989) 257-286.
[64] L. Rabiner, B.H. Juang, Fundamentals of Speech Recognition, Prentice Hall, Upper Saddle River, NJ, 1993.
[65] E. Riegler, G.E. Kirkeland, C.N. Manchon, M.A. Bodin, B.H. Fleury, Merging belief propagation and the mean field approximation: a free energy approach, 2012, arXiv:1112.0467v2 [cs.IT].
[66] J.W. Robinson, A.J. Hartemink, Learning nonstationary dynamic Bayesian networks, J. Mach. Learn. Res. 11 (2010) 3647-3680.
[67] T. Roos, H. Wertig, P. Grunvald, P. Myllmaki, H. Tirvi, On discriminative Bayesian network classifiers and logistic regression, Mach. Learn. 59 (2005) 267-296.
[68] M.J. Russell, R.K. Moore, Ecplicit modeling of state occupancy in HMMs for automatic speech recognition, in: Proceedings of the Intranational Conference on Acoustics, Speech and Signal processing, ICASSP, vol. 1, 1985, pp. 5-8.
[69] L.K. Saul, M.I. Jordan, A mean field learning algorithm for unsupervised neural networks, in: M.I. Jordan (Ed.), Learning in Graphical Models, MIT Press, Cambridge, MA, 1999.
[70] Z. Shi, C. Schlegel, Iterative multiuser detection and error control coding in random CDMA, IEEE Trans. Inform. Theory 54 (5) (2006) 1886-1895.
[71] R. Tarjan, M. Yanakakis, Simple linear-time algorithms to test chordality of graphs, test acyclicity of hypergraphs, and selectively reduce acyclic hypergraphs, SIAM J. Comput. 13 (3) (1984) 566-579.
[72] S. Theodoridis, K. Koutroumbas, Pattern Recognition, fourth ed., Academic Press, Boston, 2009.
[73] J.A. Vlontzos, S.Y. Kung, Hidden Markov models for character recognition, IEEE Trans. Image Process. 14 (4) (1992) 539-543.

[74] M.J. Wainwright, T.S. Jaakola, A.S. Willsky, A new class of upper bounds on the log partition function, IEEE Trans. Inform. Theory 51 (7) (2005) 2313-2335.

[75] M.J. Wainwright, M.I. Jordan, A variational principle for graphical models, in: S. Haykin, J. Principe, T. Sejnowski, J. Mcwhirter (Eds.), New Directions in Statistical Signal Processing, MIT Press, Cambridge, MA, 2005.

[76] M.J. Wainwright, Sparse graph codes for side information and binning, IEEE Signal Process. Mag. 24 (5) (2007) 47-57.

[77] M.J. Wainwright, M.I. Jordan, Graphical models, exponential families, and variational inference, Found. Trends Mach. Learn. 1 (1-2) (2008) 1-305.

[78] J.M. Walsh, P.A. Regalia, Belief propagation, Dykstra's algorithm, and iterated information projections, IEEE Trans. Inform. Theory 56 (8) (2010) 4114-4128.

[79] Z. Wang, E.E. Kuruoglu, X. Yang, T. Xu, T.S. Huang, Time varying dynamic Bayesian network for nonstationary events modeling and online inference, IEEE Trans. Signal Process. 59 (2011) 1553-1568.

[80] W. Wiegerinck, Variational approximations between mean field theory and the junction tree algorithm, in: Proceedings 16th Conference on Uncertainty in Artificial Intelligence, 2000.

[81] C.K.I. Williams, G.E. Hinton, Mean field networks that learn to discriminate temporally distorted strings, in: D.S. Touretzky, J.L. Elman, T.J. Sejnowski, G.E. Hinton (Eds.), Proceedings of 1990 Connectionist Models Summer School, Morgan-Kauffman, San Mateo, CA, 1991.

[82] J. Win, C.M. Bishop, Variational massage passing, J. Mach. Learn. Res. 6 (2005) 661-644.

[83] J. Yedidia, W.T. Freeman, T. Weiss, Generalized belief propagation, in: Advances on Neural Information Processing System, NIPS, MIT Press, Cambridge, MA, 2001, pp. 689-695.

[84] J. Yedidia, W.T. Freeman, T. Weiss, Understanding belief propagation and its generalization, Technical Report TR-2001-22, Mitsubishi Electric Research Laboratories, 2001.

[85] A.L. Yuille, CCCP algorithms to minimize the Bethe and Kikuchi free energies: convergent alternatives to belief propagation, Neural Comput. 14 (7) (2002) 1691-1722.