CHAPTER

PROBABILISTIC GRAPHICAL MODELS: PART I

15

CHAPTER OUTLINE

15.1 INTRODUCTION

In Figure 13.2, we used a graphical description to indicate conditional dependencies among various parameters that control the "fusion" of the prior and conditional pdfs in a hierarchical manner. Our purpose there was more of a pedagogical nature; we could live without it. In this chapter, graphical

Machine Learning. http://dx.doi.org/10.1016/B978-0-12-801522-3.00016-1

models emerge out of necessity. In many everyday machine learning applications involving multivariate statistical modeling, even simple inference tasks can easily become computationally intractable. Typical applications involve bioinformatics, speech recognition, machine vision, and text mining, to name but a few.

Graph theory has proved a powerful and elegant tool that has extensively been used in optimization and computational theory. A graph encodes dependencies among interacting variables and can be used to formalize the probabilistic structure that underlies our modeling assumptions. This can then be used to facilitate computation in a number of inference tasks, such as the calculation of marginals, modes, and conditional probabilities. Moreover, graphical models can be used as a vehicle to impose approximations onto the models when computational needs go beyond the available resources.

Early celebrated examples of the use of such models in learning tasks are the hidden Markov models, Kalman filtering, and error correcting coding, which have been popular since the early sixties.

This is the first of two chapters dedicated to probabilistic graphical models. This chapter focuses on the basic definitions and concepts, and most of its material is a must for a first reading on the topic. A number of basic graphical models are discussed, such as Bayesian networks (BNs) and Markov random fields. Exact inference is presented, and the elegant message-passing algorithm for inference on chains and trees is introduced.

15.2 THE NEED FOR GRAPHICAL MODELS

Let us consider a simplified example of a learning system in the context of a medical application. Such a system comprises a set of m diseases that correspond to hidden variables and a set of n symptoms (findings). The diseases are treated as random variables, $\mathrm{d}_1, \mathrm{d}_2, \ldots, \mathrm{d}_m$, and each of them can be absent or present and thus can be encoded by a zero or one, that is, $\mathrm{d}_j \in \{0, 1\}$, $j = 1, 2, \ldots, m$. The same applies to the symptoms, f_i, which can either be absent or present; hence, $\mathrm{f}_i \in \{0, 1\}$, $i = 1, 2, \ldots, n$. The symptoms comprise the observed variables.[1]

The goal of the system is to predict a disease hypothesis, that is, the presence of a number of diseases, given the presence of a set of symptoms, which have been observed. During the training, which is based on experts' assessments, the system learns the prior probabilities $P(d_j)$, and the conditional probabilities $P_{ij} = P(f_i = 1|d_j = 1)$, $i = 1, 2, \ldots, n$, $j = 1, 2, \ldots, m$. The latter comprise a table of nm entries. For a realistic system, these numbers can be very large. For example, in [41], m is of the order of 500-600 and n of the order of 4000. Let $\boldsymbol{f}$ be the vector that corresponds to a specific set of observations for the findings, indicating the presence or absence of the respective symptoms. Assuming that symptoms are *conditionally independent*, given any disease hypothesis, $\boldsymbol{d}$, then we can write

$$P(\boldsymbol{f}|\boldsymbol{d}) = \prod_{i=1}^{n} P(f_i|\boldsymbol{d}). \tag{15.1}$$

Ideally, one should be able to obtain the conditional probabilities $P(f_i|\boldsymbol{d})$ for each disease hypothesis. However, for all possible 2^m combinations of $\boldsymbol{d}$, this should require a huge amount of training data, which is impossible to collect for any practical system. This is bypassed by adopting the following model,

[1] In a more realistic system, some of the findings may not be available, that is, they may be unobservable.

$$P(f_i = 0|\boldsymbol{d}) = \prod_{j=1}^{m}(1 - P_{ij})^{d_j}, \tag{15.2}$$

where the exponent is set to zero, $d_j = 0$, when the disease is not related to the symptom. This is known as the *noisy-OR* model. That is, it is assumed that for a negative finding the individual causes are independent [37]. Obviously,

$$P(f_i = 1|\boldsymbol{d}) = 1 - P(f_i = 0|\boldsymbol{d}).$$

Let us now assume that we observe a set of findings, $\boldsymbol{f}$, and we want to infer $P(d_j|\boldsymbol{f})$ for some j. Then,

$$\begin{aligned} P(d_j = 1|\boldsymbol{f}) &= \frac{P(\boldsymbol{f}|d_j = 1)P(d_j = 1)}{P(\boldsymbol{f})} \\ &= \frac{\sum_{\boldsymbol{d}:d_j=1} P(\boldsymbol{f}|\boldsymbol{d})P(\boldsymbol{d})}{\sum_{\boldsymbol{d}} P(\boldsymbol{f}|\boldsymbol{d})P(\boldsymbol{d})}. \end{aligned} \tag{15.3}$$

The summation in the denominator involves 2^m terms. For $m \sim 500$, this is a formidable task that simply cannot be carried out in a realistic time.

The previous example indicates that once one gets involved with complex systems, even innocent-looking tasks turn out to be computationally intractable. Thus, one has either to be more clever in exploiting possible independencies in the data, which can reduce the required number of computations, or make certain assumptions/approximations. In this chapter, we will study both alternatives.

Before we proceed further, it is interesting to point out another source of computational obstacles besides the calculation of Eq. (15.3). In practice, it may be more convenient to perform addition instead of implementing multiplication; multiplying a large number of variables of small values such as probabilities may cause arithmetic accuracy problems. One way to bypass products is either via logarithmic or exponential operations, which transform products into summations. For example, Eq. (15.2) can be rewritten as

$$P(f_i = 0|\boldsymbol{d}) = \exp\left(-\sum_{j=1}^{m}\theta_{ij}d_j\right), \tag{15.4}$$

where $\theta_{ij} := -\ln(1 - P_{ij})$ and

$$P(f_i = 1|\boldsymbol{d}) = 1 - \exp\left(-\sum_{j=1}^{m}\theta_{ij}d_j\right). \tag{15.5}$$

Observe that the presence in Eq. (15.1) of terms corresponding to negative findings contributes linearly to the complexity (product of exponentials correspond to summations). However, this is not the case with the terms associated with positive findings. Take, for example, the extreme case where all findings are negative. Then,

$$\begin{aligned} P(\boldsymbol{f} = \boldsymbol{0}|\boldsymbol{d}) &= \prod_{i=1}^{n}\exp\left(-\sum_{j=1}^{m}\theta_{ij}d_j\right) \\ &= \exp\left(-\sum_{i=1}^{n}\left(\sum_{j=1}^{m}\theta_{ij}d_j\right)\right). \end{aligned} \tag{15.6}$$

Consider now $f_1 = 1$ and the rest to be $f_i = 0$, $i = 2, \ldots, n$. Then,

$$P(\boldsymbol{f}|\boldsymbol{d}) = \left(1 - \exp\left(-\sum_{j=1}^{m} \theta_{1j} d_j\right)\right) \exp\left(-\sum_{i=2}^{n}\left(\sum_{j=1}^{m} \theta_{ij} d_j\right)\right), \tag{15.7}$$

where now the number of exponents to be computed is two. It can easily be shown that the cross-product terms lead to an exponential computational growth [20] (Problem 15.1).

The path we will follow in order to derive efficient exact inference algorithms as well as to derive efficient approximation rules, when exact inference is not possible, will be via the use of graphical models.

15.3 BAYESIAN NETWORKS AND THE MARKOV CONDITION

Before we move on to definitions, let us first see how the existence of some structure in a joint distribution can simplify the task of marginalization. We will demonstrate it using discrete probabilities, where the use of counting can make things simpler.

Let us consider l discrete jointly distributed random variables. Applying the product rule of probability, we obtain

$$P(x_1, x_2, \ldots, x_l) = P(x_l|x_{l-1}, x_{l-2}, \ldots, x_1) P(x_{l-1}|x_{l-2}, \ldots, x_1) \ldots P(x_1). \tag{15.8}$$

Assume that each one of these variables takes values in the discrete set $\{1, 2, \ldots, k\}$. In the general case, if we want to marginalize with respect to one of the variables, say x_1, we must sum over the others, that is,

$$P(x_1) = \sum_{x_2} \cdots \sum_{x_l} P(x_1, x_2, \ldots, x_l),$$

where each one of the summations is over k possible values, which is equivalent to $\mathcal{O}\left(k^l\right)$ summations; for large values of k and/or l, this is a formidable and sometimes impossible task. Let us consider now one extreme case, where all the involved variables are mutually independent. Then the product rule becomes

$$P(x_1, x_2, \ldots, x_l) = \prod_{i=1}^{l} P(x_i),$$

and marginalization turns out to be the trivial identity

$$P(x_1) = \Big(\sum_{x_l} P(x_l) \sum_{x_{l-1}} P(x_{l-1}) \cdots \sum_{x_2} P(x_2)\Big) P(x_1), \tag{15.9}$$

because each summation is carried out independently, and of course results to one. In other words, exploiting the product rule and the statistical independence can bypass the obstacle of the exponential growth of the computational load. As a matter of fact, the previous full-independence assumptions give birth to the naive Bayes classifier, Chapter 7.

In this chapter, we are going to study cases that lie between the previous two extremes. The general idea is to be able to express the joint probability distribution (probability density/mass function) in terms of *products* of factors, where each one of them depends on a *subset* of the involved variables. This can be expressed by writing the joint distribution as

$$\boxed{p(x_1, x_2, \ldots, x_l) = \prod_{i=1}^{l} p(x_i|\mathrm{Pa}_i),} \tag{15.10}$$

where Pa_i denotes the subset of variables associated with the random variable x_i. Take the following example,

$$p(x_1, x_2, x_3, x_4, x_5, x_6) = p(x_6|x_4)\, p(x_5|x_3, x_4)\, p(x_4|x_1, x_2)\, p(x_3|x_1)\, p(x_2)\, p(x_1). \tag{15.11}$$

Then $\mathrm{Pa}_6 = \{\mathrm{x}_4\}$, $\mathrm{Pa}_5 = \{\mathrm{x}_3, \mathrm{x}_4\}$, $\mathrm{Pa}_4 = \{\mathrm{x}_1, \mathrm{x}_2\}$, $\mathrm{Pa}_3 = \{\mathrm{x}_1\}$, $\mathrm{Pa}_2 = \emptyset$, $\mathrm{Pa}_1 = \emptyset$. The variables in the set, Pa_i, are defined as the *parents* of the respective, x_i, and from a statistical point of view this means that x_i is statistically independent of *all* the variables *given* the values of its parents. Every $p(x_i|\mathrm{Pa}_i)$ expresses a *conditional independence* relationship and it imposes a *probabilistic structure* that underlies our multivariate set. It is such types of independencies that we will exploit in order to perform inference tasks at a lower computational cost.

15.3.1 GRAPHS: BASIC DEFINITIONS

A graph $G = \{V, E\}$ is a collection of nodes/vertices $V = \{x_1, \ldots, x_l\}$ and a collection of edges (arcs) $E \subset V \times V$. Each edge connects two vertices and it is denoted as a pair, $(x_i, x_j) \in E$. An edge can be either *directed*; we write $(x_i \rightarrow x_j)$ to indicate the direction or *undirected*, and in this case we simply write (x_i, x_j). Suppose we have a set of nodes $x_1, x_2, \ldots, x_k$, $k \geq 2$ and a corresponding set of edges $(x_{i-1}, x_i) \in E$ or $(x_{i-1} \rightarrow x_i) \in E$, $2 \leq i \leq k$; that is, the edges connect pairs of nodes in *sequence* and they can be either directed or not. This sequence of edges is called a *path* from x_1 to x_k. If there is at least one directed edge, the path is called *directed*. A *cycle* is a path from a node to itself. A *chain* or a *trail* is a path that can be "run" either from x_1 to x_k or from x_k to x_1; that is, all directed edges are replaced by undirected ones.

A *directed* graph comprises directed edges only and it is called a *directed acyclic graph* (DAG) if it contains *no* cycles. Given a DAG, a node in it, x_i, is called *parent* of x_j if there is a (directed) edge from x_i to x_j, and we call x_j the *child* of x_i. A node x_j is called a *descendant* of x_i and x_i an *ancestor* of x_j if there is a path from x_i to x_j. A node x_j is called *nondescendant* of x_i if it is not a descendant of x_i. A graph is said to be *fully connected* or *complete* if there is an edge between every pair of nodes. Figure 15.1 illustrates the previous definitions.

Definition 15.1. A *Bayesian network structure* is a DAG whose nodes represent random variables, $\mathrm{x}_1, \ldots, \mathrm{x}_l$, and every variable (node), x_i, is *conditionally independent* of the set of *all* its nondescendants, given the set of all its *parents*. Sometimes this is also known as the *Markov condition*.

If we denote the set of the nondescendants of a node x_i as ND_i, the Markov condition can be written as [12] $\mathrm{x}_i \perp ND_i | \mathrm{Pa}_i, \forall\, i = 1, 2, \ldots, l$. Sometimes, the conditional independencies are also known as *local independencies*. Stated differently, a BN graphical structure is a convenient way to encode conditional independencies. Figure 15.2 shows the DAG that expresses the conditional independencies

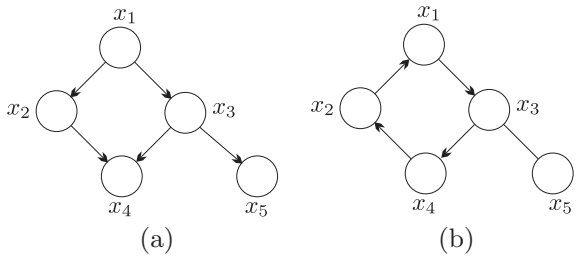

FIGURE 15.1

(a) This is a DAG because there are no cycles. x_1 is a parent of both x_2 and x_3. x_4 and x_5 are children of x_3. x_1, x_2, and x_3 are ancestors of x_4, while x_4 and x_5 are descendants of x_1. x_5 is a nondescendant of x_2 and x_4. (b) This is not a DAG and the sequence (x_2, x_1, x_3, x_4, x_2) comprises a cycle. The edge (x_3, x_5) is undirected. The sequence of nodes (x_1, x_3, x_5) forms a directed path, and the sequence (x_1, x_2, x_4, x_3) forms a chain, once directed edges are replaced by undirected ones.

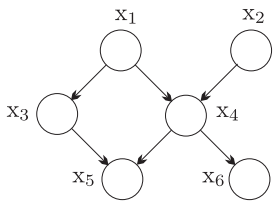

FIGURE 15.2

A Bayesian network corresponding to the pdf in Eq. (15.11). Observe that x_5 is conditionally independent of x_1 and x_2 given the values of x_3 and x_4.

used in Eq. (15.11), in order to express the joint distribution as a product of factors. Conditional independence among random variables, that is, $x \perp y|z$ or, equivalently, $p(x|y,z) = p(x|z)$, means that once we know the value of z, observing the value of y gives no additional information about x (note that the previous makes sense only if $p(y,z) > 0$). For example, the probability of children having a good education varies whether they grow up in a poor or a rich (low and high Gross National Product (GNP)) country. The probability of someone getting a high-paying job depends on her/his level of education. The probability of someone getting a high-paying job is independent of the country in which he or she was born and raised, given the level of her/his education.

Theorem 15.1. *Let G be a Bayesian network structure and p be the joint probability distribution of the random variables associated with the graph. Then p is equal to the product of the conditional distributions of all the nodes given the values of their parents, and we say that p factorizes over G.*

The proof of the theorem is done by induction, Problem 15.2. Moreover, the reverse of this theorem is also true. The previous theorem assumed a distribution and built the BN based on the underlying conditional independencies. The next theorem deals with the reverse procedure. One builds a graph based on a set of conditional distributions—one for each node of the network.

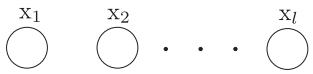

FIGURE 15.3

BN for independent variables. No edges are present because every variable is independent of all the others and no parents can be identified.

Theorem 15.2. *Let G be a DAG and associate a conditional probability for each node, given the values of its parents. Then the product of these conditional probabilities yields a joint probability of the variables. Moreover, the Markov condition is satisfied.*

The proof of this theorem is given in Problem 15.4. Note that in this theorem, we used the term probability and not distribution. The reason is that the theorem is not true for every form of conditional densities (pdfs) [14]. However, it holds true for a number of widely used pdfs, such as the Gaussians. This theorem is very useful because, often in practice, this is the way we construct a probabilistic graphical model—building it hierarchically, using reasoning on the corresponding physical process that we want to model, and encoding conditional independencies in the graph.

Figure 15.3 shows the BN structure describing a set of mutually independent variables (naive Bayes assumption).

Definition 15.2. A *Bayesian network* (BN) is a pair (G, p), where the distribution, p, factorizes over the DAG, G, in terms of a set of conditional probability distributions, associated with the nodes of G.

In other words, a Bayesian network is associated with a specific distribution. In contrast, a Bayesian network structure refers to any distribution that satisfies the Markov condition as expressed by the network structure.

Example 15.1. Consider the following simplified study relating the GNP of a country to the level of education and the type of a job an adult gets later in her/his professional life. Variable x_1 is binary with two values HGP and LGP, corresponding to countries with high and low GNP, respectively. Variable x_2 gets three values, NE, LE, and HE, corresponding to no education, low-level, and high-level education, respectively. Finally, variable x_3 gets also three possible values, UN, LP, HP corresponding to unemployed, low-paying and high-paying jobs, respectively. Using a large enough sample of data, the following probabilities are learned:

1. Marginal Probabilities:

$$P(x_1 = LGP) = 0.8,\ P(x_1 = HGP) = 0.2.$$

2. Conditional Probabilities:

$$\begin{aligned}
&P(x_2 = NE|x_1 = LGP) = 0.1,\ P(x_2 = LE|x_1 = LGP) = 0.7,\\
&P(x_2 = HE|x_1 = LGP) = 0.2,\\
&P(x_2 = NE|x_1 = HGP) = 0.05,\ P(x_2 = LE|x_1 = HGP) = 0.2,\\
&P(x_2 = HE|x_1 = HGP) = 0.75.\\
&P(x_3 = UN|x_2 = NE) = 0.15,\ P(x_3 = LP|x_2 = NE) = 0.8,\\
&P(x_3 = HP|x_2 = NE) = 0.05.
\end{aligned}$$

$$P(x_3 = UN|x_2 = LE) = 0.10,\ P(x_3 = LP|x_2 = LE) = 0.85,$$

$$P(x_3 = HP|x_2 = LE) = 0.05.$$

$$P(x_3 = UN|x_2 = HE) = 0.05,\ P(x_3 = LP|x_2 = HE) = 0.15,$$

$$P(x_3 = HP|x_2 = HE) = 0.8.$$

Note that although these values are not the result of a specific experiment, they are in line with the general trend provided by more professional studies, which involve many more random variables. However, for pedagogical reasons we must keep the example simple.

The first observation is that even for this simplistic example involving only three variables, one has to obtain seventeen probability values. This verifies the high computational load that may be required with such tasks.

Figure 15.4 shows the BN that captures the previously stated conditional probabilities. Note that the Markov condition renders x_3 independent of x_1, given the value of x_2. Indeed, the job that one finds is independent of the GNP of the country, given her/his education level. We will verify that by playing with the laws of probability for the previously defined values.

According to Theorem 15.2, the joint probability of an event is given by the product

$$P(x_1, x_2, x_3) = P(x_3|x_2)P(x_2|x_1)P(x_1). \tag{15.12}$$

In other words, the probability of someone coming from a rich country, having a good education, and getting a high-paying job will be equal to $(0.8)(0.75)(0.2) = 0.12$; similarly, the probability of somebody coming from a poor country with low-level education to get a low-paying job is 0.476.

As a next step, we will verify the Markov condition, implied by the Bayesian network structure, using the probability values given before. That is, we will verify that using conditional probabilities to build the network, these probabilities basically encode *conditional independencies*, as Theorem 15.2 suggests. Let us consider,

$$P(x_3 = HP|x_2 = HE, x_1 = HGP) = \frac{P(x_3 = HP, x_2 = HE, x_1 = HGP)}{P(x_2 = HE, x_1 = HGP)}$$

$$= \frac{0.12}{P(x_2 = HE, x_1 = HGP)}.$$

Also,

$$P(x_2 = HE, x_1 = HGP) = P(x_2 = HE|x_1 = HGP)P(x_1 = HGP)$$

$$= 0.75 \times 0.2 = 0.15,$$

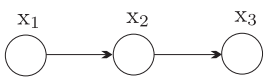

FIGURE 15.4

BN for Example 15.1. Note that $x_3 \perp x_1 | x_2$.

which finally results to

$$P(x_3 = HP|x_2 = HE, x_1 = HGP) = 0.8 = P(x_3 = HP|x_2 = HE),$$

which verifies the claim. The reader can check that this is true for all possible combinations of values.

15.3.2 SOME HINTS ON CAUSALITY

The existence of directed links in a Bayesian network *does not* necessarily reflect a cause-effect relationship from a parent to a child node.[2] It is a well-known fact in statistics that correlation between two variables does not always establish a causal relationship between them. For example, their correlation may be due to the fact that they both relate to a latent (unknown) variable. A typical example is the discussion related to whether smoking causes cancer or they are both due to an unobserved genotype that causes cancer and at the same time a craving for nicotine; this has been the defense line of the tobacco companies.

Let us return to Example 15.1. Although GNP and quality of education are correlated, one cannot say that GNP is a cause of the educational system. No doubt there is a multiplicity of reasons, such as the political system, the social structure, the economic system, historical reasons, and tradition, all of which need to be taken into consideration. As a matter of fact, the structure of the graph relating the three variables in the example could be reversed. We could collect data in the other way around; obtain the probabilities $P(\mathrm{x}_3 = UN)$, $P(\mathrm{x}_3 = LP)$, $P(\mathrm{x}_3 = HP)$, and then the conditional probabilities $P(x_2|x_3)$ (e.g., $P(\mathrm{x}_2 = HE|\mathrm{x}_3 = UN)$) and finally $P(x_1|x_2)$ (e.g., $P(\mathrm{x}_1 = HGP|\mathrm{x}_2 = HE)$). In principle, such data can also be collected from a sample of people. In such a case, the resulting Bayesian network would comprise again three nodes as in Figure 15.4, but with the direction of the arrows reversed. This is also reasonable because the probability of someone coming from a rich or a poor country is independent of her/his job, given the level of education. Moreover, both models should result in the same joint-probability distribution for any joint event. Thus, if the direction of the arrows were to indicate causality, then this time, it would be that the educational system has a cause-effect relationship on the GNP. This, for the same reasons stated before, cannot be justified. Having said all that, it does not necessarily mean that cause-effect relationships are either absent in a BN or it is not important to know them. On the contrary, in many cases, there is good reason to strive to unveil the underlying cause-effect relationships while building a BN.

Let us elaborate a bit more on this and see why exploiting any underlying cause-effect relationships can be to our benefit. Take, for example, the BN in Figure 15.5 relating the presence or absence of a disease with the findings from two medical tests. Let x_1 indicate the presence or absence of a disease and x_2, x_3 the discrete outcomes that can result from the two tests, respectively.

The BN in Figure 15.5a complies with our common sense reasoning that x_1 (disease) causes x_2 and x_3 (tests). However, this is not possible to deduct by *simply* looking at the available probabilities. This is because the probability laws are *symmetric*. Even if x_1 is the cause, we can still compute $P(x_1|x_2)$ once $P(x_1, x_2, x_3)$ and $P(x_2)$ are available, that is,

$$P(x_1|x_2) = \frac{P(x_1, x_2)}{P(x_2)} = \frac{\sum_{x_3} P(x_1, x_2, x_3)}{P(x_2)}.$$

[2] This topic will not be pursued any further; its purpose is to make the reader aware of the issue. It can be bypassed in a first reading.

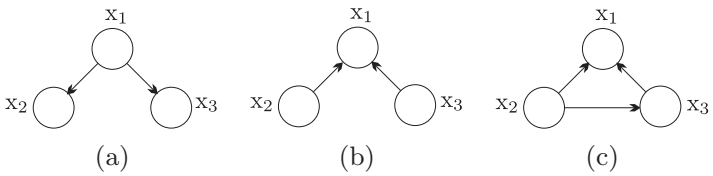

FIGURE 15.5

Three possible graphs relating a disease x_1, to the results of two tests, x_2, x_3. (a) The dependencies in this graph comply with common sense. (b) This graph renders x_2, x_3 statistically independent, which is not reasonable. (c) Training this graph needs an extra probability value compared to that in (a).

Previously, in order to say that x_1 causes x_2 and x_3, we used some *extra* information/knowledge, which we called common sense reasoning. Note that in this case, training requires the knowledge of the values of three probabilities, namely, $P(x_1)$, $P(x_2|x_1)$, $P(x_3|x_1)$. Let us now assume that we choose the graph model in Figure 15.5b. This time, ignoring the cause-effect relationship has resulted in the wrong model. This model renders x_2 and x_3 independent, which, obviously, cannot be the case. These should only be *conditionally* independent given x_1. The only sensible way to keep x_2 and x_3 as parents of x_1 is to add an extra link, as shown in Figure 15.5c, which establishes a relation between the two. However, to train such a network, besides the values of the three probabilities, $P(x_2)$, $P(x_3)$, and $P(x_1|x_2, x_3)$, one needs to know the values for an extra one, $P(x_3|x_2)$. Thus, when building a BN, it is always good to know any underlying cause-effect directions. Moreover, there are other reasons, too. For example, this may be related to the *interventions*, which are actions that change the state of a variable in order to study the respective impact on other variables; because a change propagates in the causal direction, such a study is only possible if the network has been structured in a cause-effect hierarchy. For example, in biology, there is a strong interest in understanding which genes affect activation levels of other genes, and in predicting the effects of turning certain genes on or off.

The notion of causality is not an easy one, and philosophers have been arguing about it for centuries. Although our intention here is by no means to touch this issue, it is interesting to quote two well-known philosophers.

According to David Hume, causality is not a property of the real world but a concept of the mind that helps us explain our perception of the world. Hume (1711–1776) was a Scottish philosopher best known for his philosophical empiricism and skepticism. His most well-known work is the "Treatise of Human Nature," and in contrast to the rationalistic philosophy school, he advocated that human nature is mainly governed by desire and not reason.

According to Bertrand Russell, the law of causality has nothing to do with the laws of physics, which are symmetrical (recall our statement before concerning conditional probabilities) and indicate no cause-effect relationship. For example, Newton's gravity law can be expressed in any of the following forms,

$$B = mg \text{ or } g = \frac{B}{m} \text{ or } m = \frac{B}{g},$$

and looking only at them, no cause-effect relationship can be deduced. Bertrand Russell (1872–1970) was a British philosopher, mathematician, and logician. He is considered one of the founders of analytic

philosophy. In *Principia Mathematica*, co-authored with A.N. Whitehead, they made an attempt to ground mathematics on mathematical logic. He was also an antiwar activist and a liberal.

The previously stated provocative arguments have been inspired by Judea Pearl's book [38], and we provided them in order to persuade the reader to read this book; he or she can only become wiser. Pearl has made a number of significant contributions to the field and was the recipient of the Turing award in 2011.

Although one cannot deduce causality by looking only at the laws of physics or probabilities, ways of identifying it have been developed. One way is to carry out *controlled* experiments; one can change the values of the variable and study the effect of the change on another. However, this has to be done in a controlled way in order to guarantee that the caused effects are not due to other related factors.

Besides experimentation, there has been a major effort to discover causal relationships from nonexperimental evidence. In modern applications, such as microarray measurements for gene expressions or fMRI brain imaging, the number of the involved variables can easily reach the order of a few thousand. Performing experiments for such tasks is out of the question. In [38], the notion of causality is related to that of the minimality in the structure of the obtained possible DAGs. Such a view ties causality with Occam's razor. More recently, inferring causality was attempted by comparing the conditional distributions of variables given their direct causes, for all hypothetical causal directions, and choosing the most plausible. The method builds upon some smoothness arguments that underlie the conditional distributions of the effect given the causes, compared to the marginal distributions of the effect/cause [43]. In [24], an interesting alternative for inferring causality is built upon arguments from Kolmogorov's complexity theory; causality is verified by comparing shortest description lengths of strings associated with the involved distributions. For further information, the interested reader may consult, for example, [42] and the references therein.

15.3.3 *D*-SEPARATION

Dependencies and independencies among a set of random variables play a key role in understanding their statistical behavior. Moreover, as we have already commented, they can be exploited to substantially reduce the computational load for solving inference tasks.

By the definition and the properties of a Bayesian network structure, G, we know that certain independencies hold and are readily observed via the parent-child links. The question that now is raised is whether there are additional independencies that the structure of the graph imposes on any joint probability distribution that factorizes over G. Unveiling extra independencies offers the designer more freedom to deal with computational complexity issues more aggressively.

We will attack the task of searching for conditional independencies across a network by observing whether probabilistic evidence, that becomes available at a node, x, can propagate and influence our certainty about another node, y.

Serial or head-to-tail connection. This type of node connection is shown in Figure 15.6a. Evidence on x will influence the certainty about y, which in turn will influence that of z. This is also true for the reverse direction, starting from z and propagating to x. However, if the state of y is known, then x and z become (conditionally) independent. In this case, we say that y *blocks* the path from x to z and vice versa. When the state at a node is fixed/known, we say that the node is *instantiated*.

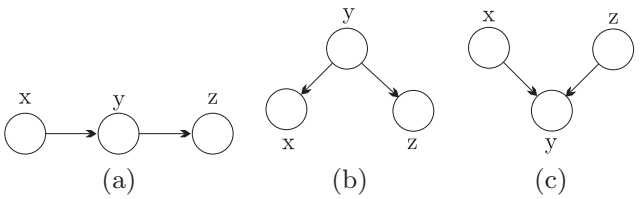

FIGURE 15.6

Three different types of connections: (a) serial, (b) diverging, and (c) converging.

Diverging or tail-to-tail connection. In this type of connection, shown in Figure 15.6b, evidence can propagate from y to x and from y to z, and also from x to z and from z to x via y, unless y is instantiated. In the latter case, y *blocks* the path from x to z and vice versa. That is, x and z become independent given the value of y. For example, if y represents "flu," x "runny nose," and z "sneezing," then if we do not know whether someone has the flu, then a runny nose is evidence that can change our certainty about her/him having the flu; this in turn changes our belief about sneezing. However, if we know that someone has the flu, seeing the nose running gives no extra information about sneezing.

Converging or head-to-head connection or v-structure. This type of connection is slightly more subtle than the previous two cases, and it is shown in Figure 15.6c. Evidence from x does not propagate to z and thus cannot change our certainty about it. Knowing something about x tells us nothing about z. For example, let z denote either of two countries (e.g., England and Greece), x "season," and y "cloudy weather." Obviously, knowing the season says nothing about a country. However, having some evidence about cloudy weather y, then knowing that it is summer provides information that can change our certainty about the country. This is in accordance with our intuition. Knowing that it is summer and that the weather is cloudy *explains away* that the country is Greece. This is the reason we sometimes refer to this type of reasoning as *explaining away*. Explaining away is an instance of a general reasoning pattern called *intercausal reasoning*, where different causes of the same effect can interact; this is a very common pattern of reasoning in humans.

For this particular type of connection, explaining away is also achieved by evidence that is provided by *any one* of the descendants of y. Figure 15.7 illustrates the case via an example. Having evidence about the rain will also establish a path so that evidence about the season (country), x (z), changes our certainty about the country (season), z (x).

To recapitulate, let us stress the delicate point here. For the first two cases, head-to-tail and tail-to-tail, the path is blocked if node y is instantiated, that is, when its state is disclosed to us. However, in the head-to-head connection, the path between x and z "*opens*" when probabilistic evidence becomes available, either at y or *at any one* of its descendants.

Definition 15.3. Let G be a BN structure, and let $x_1, \ldots, x_k$ comprise a chain of nodes. Let Z be a subset of *observed* variables. The chain, $x_1, \ldots, x_k$, is said to be *active given* the set, Z, if

- whenever a converging connection, $x_{i-1} \rightarrow x_i \leftarrow x_{i+1}$, is present in the chain, then either x_i or one of its descendants is in Z.
- no other node in the chain is in Z.

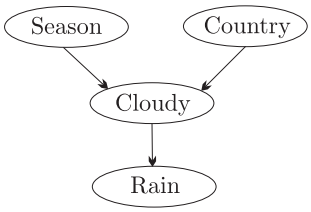

FIGURE 15.7

Having some evidence about either the weather being cloudy or rainy establishes the path for information flow between the nodes "season" and "country."

In other words, in an active chain, probabilistic evidence can flow from x_1 to x_k and vice versa because no nodes (links), which can block this information flow, are present.

Definition 15.4. Let G be a BN structure and let X, Y, Z be three mutually disjoint sets of nodes in G. We say that X and Y are *d-separated* given Z if there is *no* active chain between *any* node $x \in X$ and $y \in Y$ given Z. If these are not d-separated, we say that they are *d-connected*.

In other words, if two variables x and y are d-separated by a third one z then observing the state of z, blocks any evidence propagation from x to y and vice versa. That is, d-separation implies *conditional independence*. Moreover, the following very important theorem holds.

Theorem 15.3. *Let the pair (G, p) be a Bayesian network. For every three mutually disjoint subsets of nodes X, Y, Z, whenever X and Y are d-separated, given Z, then for every pair* $(x, y) \in X \times Y$, x *and* y *are conditionally independent in p given Z.*

The proof of the theorem was given in [45]. In other words, this theorem guarantees that d-separation implies conditional independence on any probability distribution that factorizes over G. Note that, unfortunately, the opposite is not true. There may be conditional independencies that cannot be identified by d-separation (e.g., Problem 15.5). However, for most practical applications, the reverse is also true. The number of distributions that do not comply with the reverse statement of the theorem is infinitesimally small (see, e.g., [25]). Identification of all d-separations in a graph can be carried out via a number of efficient algorithms (e.g., [25, 32]).

Example 15.2. Consider the DAG G of Figure 15.8, connecting two nodes x, y. It is obvious that these nodes are not d-separated and comprise an active chain. Consider the following probability distribution, which factorizes over G, with

$$P(y = 0|x = 0) = 0.2, \quad P(y = 1|x = 0) = 0.8,$$
$$P(y = 0|x = 1) = 0.2, \quad P(y = 1|x = 1) = 0.8.$$

It can easily be checked out that $P(y|x) = P(y)$ (independent of the values of $P(x = 1),\ P(x = 0)$) and the variables x and y are independent; this cannot be predicted by observing the d-separations. Note, however, that if we slightly perturb the values of the conditional probabilities, then the resulting distribution has as many independencies as those predicted by the d-separations; that is, in this case, none. As a matter of fact, this is a more general result. If we have a distribution that has independencies that are not predicted by d-separations, a small perturbation will almost always eliminate them (e.g., [25]).

FIGURE 15.8

This DAG involves no nodes that are *d*-separated.

Example 15.3. Consider the DAG shown in Figure 15.9. The red nodes indicate that the respective random variables have been observed; that is, these nodes have been instantiated. Node x_5 is *d*-connected to x_1, x_2, x_6. In contrast, node x_9 is *d*-separated from all the rest. Indeed, evidence starting from x_1 is blocked by x_3. However, it propagates via x_4 (instantiated and converging connection) to x_2, x_6 and then to x_5 (x_7 is instantiated and converging connection). In contrast, any flow of evidence toward x_9 is blocked by the instantiation of x_7. It is interesting to note that, although all neighbors of x_5 have been instantiated, still it remains *d*-connected with other nodes.

Definition 15.5. The *Markov blanket* of a node is the set of nodes comprising (a) its parents, (b) its children, and (c) the nodes sharing a child with this node. Once all the nodes in the blanket of a node are instantiated, then the node becomes *d*-separated from the rest of the network (Problem 15.7).

For example, in Figure 15.9, the Markov blanket of x_5 comprises the nodes x_3, x_4, x_8, x_7, and x_6. Note that if all these nodes are instantiated, then x_5 becomes *d*-separated from the rest of the nodes.

In the sequel, we give some examples of machine learning tasks, which can be cast in terms of a Bayesian graphical representation. As we will discuss, for many practical cases, the involved conditional probability distributions are expressed in terms of a set of parameters.

15.3.4 SIGMOIDAL BAYESIAN NETWORKS

We have already seen that when the involved random variables are discrete, the conditional probabilities $P(x_i|\text{Pa}_i)$, $i = 1, \ldots, l$, associated with the nodes of a Bayesian graph structure have to be learned from the training data. If the number of possible states and/or the number of the variables in Pa_i is large

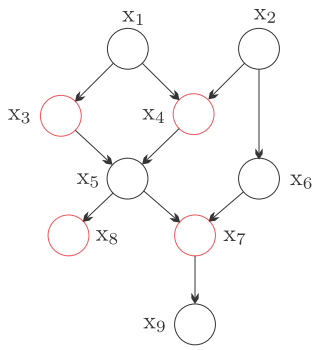

FIGURE 15.9

Red nodes are instantiated. Node x_5 is *d*-connected to x_1, x_2, x_6 and node x_9 is *d*-separated from all the nonobserved variables.

enough, this amounts to a large number of probabilities that have to be learned; thus, a large number of training points is required in order to obtain good estimates. This can be alleviated by expressing the conditional probabilities in a parametric form, that is,

$$P(x_i|\text{Pa}_i) = P(x_i|\text{Pa}_i; \boldsymbol{\theta}_i), \quad i = 1, 2, \ldots, l. \tag{15.13}$$

In case of binary valued variables, a common functional form is to view P as a logistic regression model; we used this model in the context of relevance vector machines in Chapter 13. Adopting this model, we have

$$P(x_i = 1|\text{Pa}_i; \boldsymbol{\theta}_i) = \sigma(t_i) = \frac{1}{1 + \exp(-t_i)}, \tag{15.14}$$

$$t_i := \theta_{i0} + \sum_{k: x_k \in \text{Pa}_i} \theta_{ik} x_k. \tag{15.15}$$

This reduces the number of parameter vectors to be learned to $O(l)$. The exact number of parameters depends on the size of the parent sets. Assuming the maximum number of parents for a node to be K, then the unknown number of parameters to be learned from the training data is less than or equal to lK. Taking into account the binary nature of the variables, we can write that

$$P(x_i|\text{Pa}_i; \boldsymbol{\theta}_i) = x_i \sigma(t_i) + (1 - x_i)(1 - \sigma(t_i)), \tag{15.16}$$

where t_i is given in Eq. (15.15).

Such models are also known as *sigmoidal Bayesian networks*, and they have been proposed as one type of neural network (Chapter 18) (e.g., [33]). Figure 15.10 presents the graphical structure of such a network. The network can be treated as a BN structure by associating a binary variable at each node and interpreting nodes' activations as probabilities, as dictated by Eq. (15.16). Performing inference and training of the parameters in such networks is not an easy task. We have to resort to approximations. We will come to this in Section 16.3.

15.3.5 LINEAR GAUSSIAN MODELS

The computational advantages of the Gaussian pdf have recurrently been exploited in this book. We will now see the advantage gains in the framework of graphical models when the conditional pdf at every node, given the values of its parents, is expressed in a Gaussian form. Let

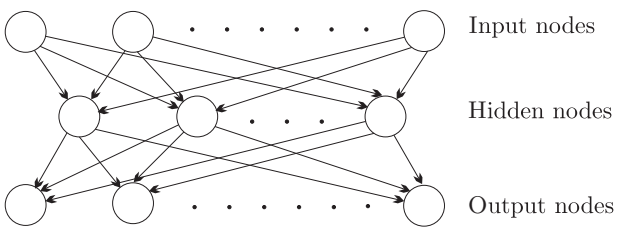

FIGURE 15.10

A sigmoidal Bayesian network.

$$p(x_i|\text{Pa}_i) = \mathcal{N}\left(x_i \left| \sum_{k:x_k \in \text{Pa}_i} \theta_{ik} x_k + \theta_{i0}, \sigma_i^2 \right.\right), \tag{15.17}$$

where σ_i^2 is the respective variance and θ_{i0} is the bias term. From the properties of a Bayesian network, the joint pdf will be given by the product of the conditional probabilities (Theorem 15.2, which is valid for Gaussians), and the respective logarithm is given by

$$\ln p(\boldsymbol{x}) = \sum_{i=1}^{l} \ln p(x_i|\text{Pa}_i) = -\sum_{i=1}^{l} \frac{1}{2\sigma_i^2} \left(x_i - \sum_{k:x_k \in \text{Pa}_i} \theta_{ik} x_k - \theta_{i0} \right)^2 + \text{constant}. \tag{15.18}$$

This is of a quadratic form, hence it is also of a Gaussian nature. The mean values and the covariance matrices for each one of the variables can be computed *recursively* in a straightforward way (Problem 15.8).

Note the computational elegance of such a Bayesian network. In order to obtain the joint pdf, one has only to sum up all the exponents, that is, an operation of linear complexity. Moreover, concerning training, one could readily think of a way to learn the unknown parameters; adopting the maximum likelihood method (although it may not be, necessarily, the best method), optimization with respect to the unknown parameters is a straightforward task. In contrast, one cannot make similar comments for the training of the sigmoidal Bayesian network. Unfortunately, products of sigmoid functions do not lead to an easy computational procedure. In such cases, one has to resort to approximations. For example, one way is to employ the variational bound approximation, as discussed in Chapter 13, in order to enforce, locally, a Gaussian functional form. We will discuss this technique in Section 16.3.1.

15.3.6 MULTIPLE-CAUSE NETWORKS

In the beginning of this chapter, we started with an example from the field of medical informatics. We were given a set of diseases and a set of symptoms/findings. The conditional probabilities for each symptom being absent, given the presence of a disease (Eq. (15.2)) were assumed known. We can consider the diseases as hidden causes (h) and the symptoms as observed variables (y), in a learning task. This can be represented in terms of a Bayesian network structure as in Figure 15.11. For the previous medical example, the variables h correspond to d (diseases) and the observed variables, y, to the findings, f.

However, the Bayesian structure given in Figure 15.11 can serve the needs of a number of inference and pattern recognition tasks, and sometimes it is referred to as a *multiple-cause network*, for obvious reasons. For example, in a machine vision application, the hidden causes, $\text{h}_1, \text{h}_2, \ldots, \text{h}_k$, may refer to the presence or absence of an object, and y_n, $n = 1, \ldots, N$, may correspond to the values of the observed pixels in an image [15]. The hidden variables can be binary (presence or absence of the respective object) and the conditional pdf can be formulated into a parameterized form, that is, $p(y_n|\boldsymbol{h}; \boldsymbol{\theta})$. The specific form of the pdf captures the way objects interact as well as the effects of the noise. Note that in this case, the Bayesian network has a mixed set of variables, the observations are continuous, and the hidden causes binary. We will return to this type of Bayesian structure when discussing approximate inference methods in Section 16.3.

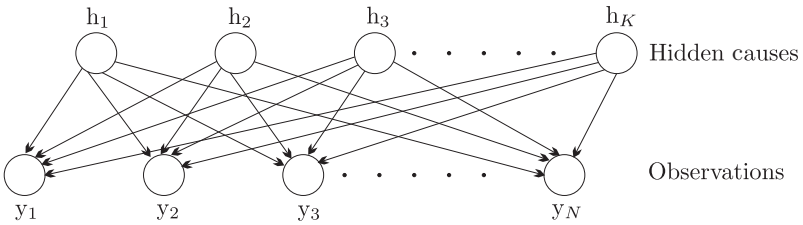

FIGURE 15.11

The general structure of a multiple-cause Bayesian network. The top-level nodes correspond to the hidden causes and the bottom ones to the observations.

15.3.7 I-MAPS, SOUNDNESS, FAITHFULNESS, AND COMPLETENESS

We have seen a number of definitions and theorems referring to the notion of conditional independence in graphs and probability distributions. Before we proceed further, it will be instructive to summarize what has been said and provide some definitions that will dress up our findings in a more formal language. This will prove useful for subsequent generalizations.

We have seen that a Bayesian network is a DAG that encodes a number of conditional independencies. Some of them are local ones, defined by the parent-child links, and some of them are of a more global nature and are the result of d-separations. Given a DAG, G, we denote as $I(G)$ the set of all independencies that correspond to d-separations. Also, let p be a probability distribution over a set of random variables, $\mathrm{x}_1, \ldots, \mathrm{x}_l$. We denote as $I(p)$ the set of all independence assertions of the type $\mathrm{x}_i \perp \mathrm{x}_j | Z$ that hold true for the distribution p.

Let G be a DAG and p a distribution that factorizes over G; in other words, it satisfies the local independencies as suggested by G. Then, we have seen (Theorem 15.3) that

$$I(G) \subseteq I(p). \tag{15.19}$$

We say that G is an *I-map* (independence map) for p. This property is sometimes referred to as *soundness*.

Definition 15.6. A distribution p is *faithful* to a graph G, if *any* independence in p is reflected in the d-separation properties of the graph.

In other words, the graph can represent all (and only) the conditional independence properties of the distribution. In such a case we write $I(p) = I(G)$. If equality is valid, we say that the graph, G, is a *perfect* map for p. Unfortunately, this is *not* valid for any distribution, p, that factorizes over G. However, for most practical purposes, $I(G) = I(P)$, which is true for *almost all* distributions that factorize over G.

Although $I(p) = I(G)$ is not valid for all distributions that factorize over G, the following two properties are always valid for any Bayesian network structure (e.g., [25]).

- If $\mathrm{x} \perp \mathrm{y} | Z$ for *all* distributions p that factorize over G, then x and y are d-separated given Z.
- If x and y are d-connected given Z, then there will be some distribution that factorizes over G where x and y are *dependent*.

A final definition concerns minimality.

Definition 15.7. A graph, G, is said to be *minimal* I-map for a set of independencies if the removal of any of its edges renders it *not* to be an I-map.

Note that a minimal I-map is not necessarily a perfect map. In the same way that there exist algorithms to find the set of d-separations, there exist algorithms to find perfect and minimal I-maps for a distribution (e.g., [25]).

15.4 UNDIRECTED GRAPHICAL MODELS

Bayesian structures and networks are not the only way to encode independencies in distributions. As a matter of fact, the directionality assigned to the edges of a DAG, while being advantageous and useful in some cases, becomes a disadvantage in others. A typical example is that of four variables, x_1, x_2, x_3, x_4. There is no directed graph that can encode the following conditional independencies simultaneously: $x_1 \perp x_4|\{x_2, x_3\}$ *and* $x_2 \perp x_3|\{x_1, x_4\}$. Figure 15.12, shows the possible DAGs; notice that both fail to capture the desired independencies.

In 15.12a, $x_1 \perp x_4|\{x_2, x_3\}$ because both paths, which connect x_1 and x_4, are blocked. However, x_2 and x_3 are d-connected given x_1 and x_4 (Why?). In 15.12b, $x_2 \perp x_3|\{x_1, x_4\}$ because the diverging links are blocked. However, we have violation of the other independence. (Why?)

Such situations can be overcome by resorting to undirected graphs. We will also see that this type of graphical modeling leads to a simplification concerning our search for conditional independencies.

Undirected graphical models or *Markov networks* have their roots in *Markov random fields* (MRF) in statistical physics. As was the case with the Bayesian models, each node of the graph is associated with a random variable. Edges connecting nodes are undirected, giving no preference to either of the two directions. Local interactions among connected nodes are expressed via functions of the involved variables, but they do not necessarily express probabilities. One can view these local functional interactions as a way to encode information related to the affinity/similarity among the involved variables. These local functions are known as *potential functions* or *compatibility functions* or *factors*, and they are *nonnegative*, usually positive, functions of their arguments. Moreover, as we will soon see, the global description of such a model is the result of the product of these local potential functions; this is in analogy to what holds true for the Bayesian networks.

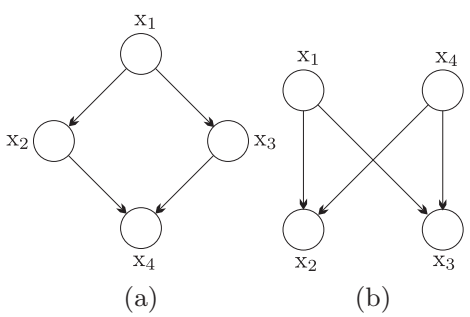

FIGURE 15.12

None of these DAGs can capture the two independencies: $x_1 \perp x_4|\{x_2, x_3\}$ and $x_2 \perp x_3|\{x_1, x_4\}$.

Following a path similar to that used for the directed graphs, we will begin with the factorization properties of a distribution over an MRF and then move on to study conditional independencies. Let $\mathrm{x}_1, \ldots, \mathrm{x}_l$, be a set of random variables that are grouped in K groups, $\mathbf{x}_1, \ldots, \mathbf{x}_K$; each random vector, $\mathbf{x}_k$, $k = 1, 2, \ldots, K$, involves a subset of the random variables, x_i, $i = 1, 2, \ldots, l$.

Definition 15.8. A distribution is called a *Gibbs* distribution if it can be factorized in terms of a set of potential functions, $\psi_1, \ldots, \psi_K$, such as

$$p(x_1, \ldots, x_l) = \frac{1}{Z} \prod_{k=1}^{K} \psi_k(\boldsymbol{x}_k). \tag{15.20}$$

The constant Z is known as the *partition* function and it is the normalizing constant to guarantee that $p(x_1, \ldots, x_l)$ is a probability distribution. Hence,

$$Z = \int \cdots \int \prod_{k=1}^{K} \psi_k(\boldsymbol{x}_k)\, \mathrm{d}x_1, \ldots, \mathrm{d}x_l, \tag{15.21}$$

which becomes a summation for the case of probabilities.

Note that nobody can prohibit us from assigning conditional probability distributions as potential functions and making (15.20) identical to Eq. (15.10); in this case, normalization is not explicitly required because each one of the conditional distributions is normalized. However, MRFs can deal with more general cases.

Definition 15.9. We say that a Gibbs distribution, p, factorizes over an MRF, H, if *each* group of the variables, $\mathbf{x}_k$, $k = 1, 2, \ldots, K$, involved in the K factors of the distribution p, forms a *complete subgraph* of H. Every complete subgraph of an MRF is known as a *clique*, and the corresponding factors of the Gibbs distribution are known as *clique potentials*.

Figure 15.13a shows an MRF and two cliques. Note that the set of nodes $\{\mathrm{x}_1, \mathrm{x}_3, \mathrm{x}_4\}$ does not comprise a clique because the respective subgraph is not fully connected. The same applies to the set $\{\mathrm{x}_1, \mathrm{x}_2, \mathrm{x}_3, \mathrm{x}_4\}$. In contrast, the sets $\{\mathrm{x}_1, \mathrm{x}_2, \mathrm{x}_3\}$ and $\{\mathrm{x}_3, \mathrm{x}_4\}$ form cliques. The fact that all variables in a group, $\mathbf{x}_k$, that are involved in the respective factor $\psi_k(\boldsymbol{x}_k)$ form a clique means that *all* these variables mutually interact, and the factor is a measure of such an interaction/dependence.

A clique is called *maximal* if we cannot include any other node from the graph in the set without its ceasing to be a clique. For example, both cliques in Figure 15.13a are maximal cliques. On the other hand, the clique in Figure 15.13b formed by $\{\mathrm{x}_1, \mathrm{x}_2, \mathrm{x}_3\}$ is not maximum because bringing x_4 into the new set $\{\mathrm{x}_1, \mathrm{x}_2, \mathrm{x}_3, \mathrm{x}_4\}$ is also a clique. The same holds true for the clique formed by $\{\mathrm{x}_2, \mathrm{x}_3, \mathrm{x}_4\}$.

15.4.1 INDEPENDENCIES AND I-MAPS IN MARKOV RANDOM FIELDS

We will now state the equivalent theorem of d-separation, which was established for Bayesian network structures; recall the respective definition in Section 15.3.3, via the notion of an active chain.

Definition 15.10. Let H be an MRF and let $\mathrm{x}_1, \mathrm{x}_2, \ldots, \mathrm{x}_k$, comprise a path.[3] If Z is a set of observed variables/nodes, the path is said to be *active given* Z if none of $\mathrm{x}_1, \mathrm{x}_2, \ldots, \mathrm{x}_k$, is in Z.

Given three disjoint sets, X, Y, Z, we say that the nodes of X are *separated* by the nodes of Y, given Z, if there is no active path between X and Y, given Z. Note that the previous definition is much

[3] Because edges are undirected, the notions of "chain" and "path" become identical.

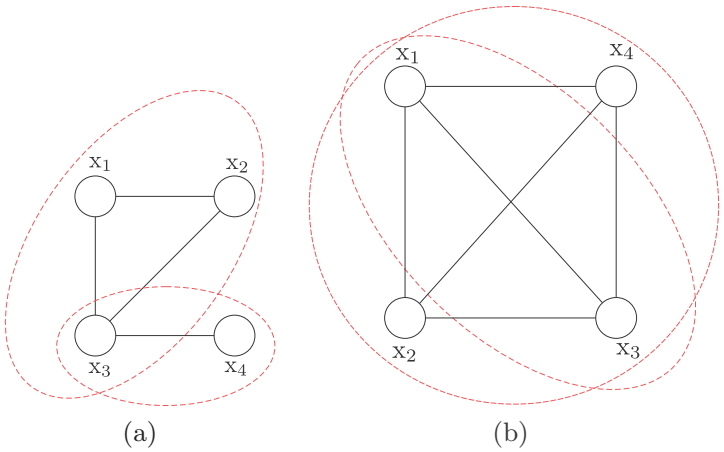

FIGURE 15.13

(a) There are two cliques encircled by the red lines. (b) There are as many possible cliques as the combinations of the points in pairs, in triples, and so on. Considering all points together also forms a clique, and this is a maximal clique.

simpler compared to the respective definition given for the Bayesian network structures. According to the current definition, for a set X to be separated from a set Y given a third set Z, it suffices that *all* possible paths from X to Y pass via Z. Figure 15.14 illustrates the geometry. In 15.14a, there is no active path connecting the nodes in X from the nodes in Y given the nodes in Z. In 15.14b, there exist active paths connecting X and Y given Z.

Let us now denote by $I(H)$ the set of all possible statements of the type "X separated by Y given Z." This is in analogy to the set of all possible d-separations associated with a Bayesian network structure. The following theorem (soundness) holds true (Problem 15.10).

Theorem 15.4. *Let p be a Gibbs distribution that factorizes over an MRF H. Then, this is an I-map for p, that is,*

$$I(H) \subseteq I(p). \tag{15.22}$$

This is the counterpart of Theorem 15.3 in its "I-map formulation" as introduced in Section 15.3.7. Moreover, given that an MRF, H, is an I-map for a distribution, p, then p factorizes over H. Note that this holds true for Bayesian network structures; indeed, if $I(G) \subseteq I(p)$, then p factorizes over G (Problem 15.12). However, for MRFs, it is only true for strictly positive Gibbs distributions and it is given by the following *Hammersley-Clifford theorem*.

Theorem 15.5. *Let H be an MRF over a set of random variables, $\mathrm{x}_1, \ldots, \mathrm{x}_l$, described by a probability distribution, $p > 0$. If H is an I-map for p, then p is a Gibbs distribution that factorizes over H.*

For a proof of this theorem, the interested reader is referred to the original paper [22] and also [5].

Our final touch on independencies in the context of MRFs concerns the notion of completeness. As was the case with the Bayesian networks, if p factorizes over an MRF, this does not necessarily establish completeness, although it is true for almost all practical cases. However, the weaker version

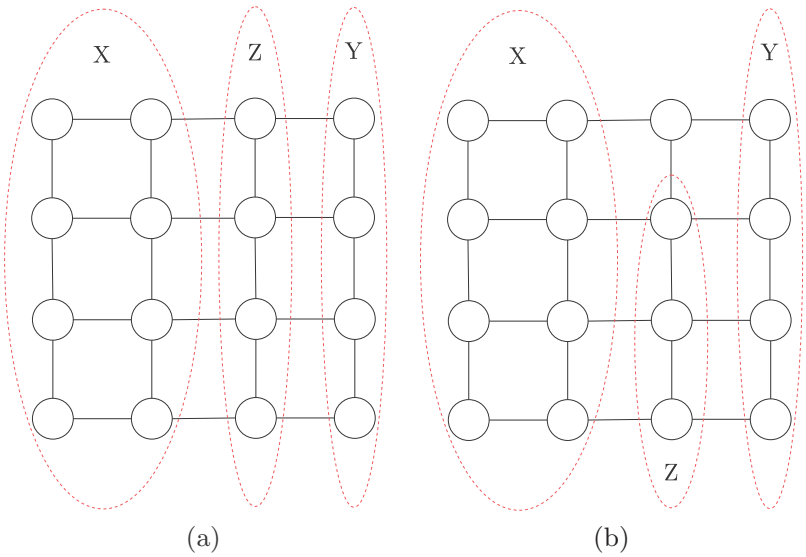

FIGURE 15.14

(a) The nodes of X and Y are separated by the nodes of Z. (b) There exist active paths that connect the nodes of X with the nodes of Y, given Z.

holds. That is, if x and y are two nodes in an MRF, which are *not* separated given a set Z, then there exists a Gibbs distribution, p, which factorizes over H and according to which x and y are dependent, given the variables in Z (see, e.g., [25]).

15.4.2 THE ISING MODEL AND ITS VARIANTS

The origin of the theory on Markov random fields is traced back to the discipline of statistical physics, and since then it has extensively been used in a number of different disciplines, including machine learning. In particular, in image processing and machine vision, MRFs have been established as a major tool in tasks such as de-noising, image segmentation, and stereo reconstruction (see, e.g., [29]). The goal of this section is to state a basic and rather primitive model, which, however, demonstrates the way information is captured and subsequently processed by such models.

Assume that each random variable takes binary values in $\{-1, 1\}$ and that the joint probability distribution is given by the following model,

$$p(x_1, \ldots, x_l) := p(\boldsymbol{x}) = \frac{1}{Z} \exp\left(-\sum_i \left(\sum_{j>i} \theta_{ij} x_i x_j + \theta_{i0} x_i\right)\right), \tag{15.23}$$

where $\theta_{ij} = 0$ if the respective nodes are not connected. It is readily seen that this model is the result of the product of potential functions (factors), each one of an exponential form, defined on cliques of size two. Also, $\theta_{ij} = \theta_{ji}$ and we sum such as $i < j$ in order to avoid duplication. This model was

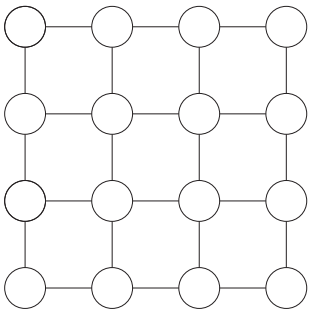

FIGURE 15.15

The graph of an MRF with pairwise dependencies among the nodes.

originally used by Ising in 1924 in his doctoral thesis to model phase transition phenomena in magnetic materials. The ± 1 of each node in the lattice models the two possible spin directions of the respective atoms. If $\theta_{ij} > 0$, interacting atoms tend to align spins in the same direction in order to decrease energy (ferromagnetism). The opposite is true if $\theta_{ij} < 0$. The corresponding graph is given in Figure 15.15.

This basic model has been exploited in computer vision and image processing for tasks such as image de-noising, image segmentation, and scene analysis. Let us take, as an example, a binarized image and let x_i denote the noiseless pixel values (± 1). Let y_i be the observed noisy pixels whose values have been corrupted by noise and have changed polarity; see Figure 15.16 for the respective graph. The task is to obtain the noiseless pixel values. One can rephrase the model in Eq. (15.23) to the needs of this task and rewrite it as [5, 21],

$$P(\boldsymbol{x}|\boldsymbol{y}) = \frac{1}{Z}\exp\left(\sum_i\left(\alpha\sum_{j>i}x_ix_j + \beta\, x_iy_i\right)\right), \tag{15.24}$$

where we have used only two parameters, α, β. Moreover, the summation $\sum_{j>i}$ involves only *neighboring* pixels. The goal now becomes that of estimating the pixel values, x_i, by maximizing the conditional (on the observations) probability. The adopted model is justified by the following two facts: (a) for low enough noise levels, most of the pixels will have the same polarity as the respective observations; this is encouraged by the presence of the product x_iy_i, where similar signs contribute to higher probability values; and (b) neighboring pixels are encouraged to have the same polarity because we know that real-world images tend to be smooth, except at the points that lie close to the edges in the image. Sometimes, a term cx_i, for an appropriately chosen value of c, is also present if we want to penalize either of the two polarities. The max-product or max-sum algorithms, to be discussed later in this chapter, are possible algorithmic alternatives for the maximization of the joint probability given in Eq. (15.24). However, these are not the only algorithmic possibilities to perform the optimization task. A number of alternative schemes that deal with inference in MRFs have been developed and studied. Some of them are suboptimal, yet they enjoy computational efficiency. Some classical references on the use of MRFs in image processing are [7–9, 47]. A number of variants of the basic Ising model result if one writes it as

$$P(\boldsymbol{x}) = \frac{1}{Z}\exp\left(-\sum_i\left(\sum_{j>i}f_{ij}(x_ix_j) + f_i(x_i)\right)\right) \tag{15.25}$$

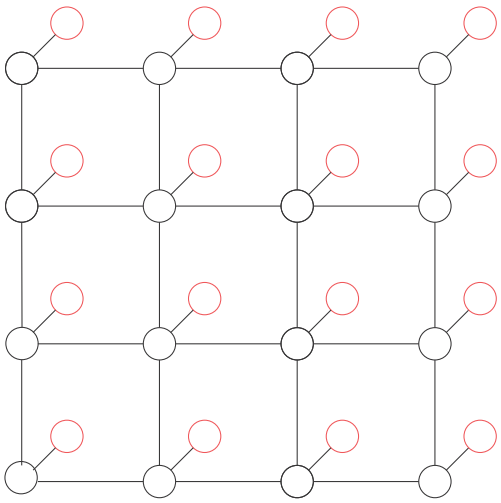

FIGURE 15.16

A pairwise MRF as in Figure 15.15, but now the observed values associated with each node are separately denoted as red nodes. For the image de-noising task, black nodes correspond to the noiseless pixel values (hidden variables) and red nodes to the observed pixel values.

and uses different functional forms for $f_{ij}(\cdot,\cdot)$ and $f_i(\cdot)$, and also by allowing the variables to take more than two values. This is sometimes known as the *Potts* model. In general, MRF models of the general form of Eq. (15.23) are also known as *pairwise MRFs undirected graphs* because the dependence among nodes is expressed in terms of products of pairs of variables. Further information on the applications of MRFs in image processing can be found in, for example, [29, 40].

Another name for Eq. (15.23) is Boltzmann distribution, where, usually, the variables take values in $\{0, 1\}$. Such a distribution has been used in *Boltzmann machines*, [23]. Boltzmann machines can be seen as the stochastic counterpart of Hopfield networks; the latter have been proposed to act as *associative memories* as well as a way to attack combinatoric optimization problems (e.g., [31]). The interest in Boltzmann machines has been revived in the context of deep learning, and we will discuss them in more detail in Chapter 18.

15.4.3 CONDITIONAL RANDOM FIELDS (CRFs)

All the graphical models (directed and undirected) that have been discussed so far evolve around the joint distribution of the involved random variables and its factorization on a corresponding graph. More recently, there is a trend to focus on the conditional distribution of some of the variables given the rest. The focus on the joint pdf originates from our interest in developing generative learning models.

However, this may not always be the most efficient way to deal with learning tasks, and we have already talked in Chapters 3 and 7 about the discriminative learning alternative. Let us assume that from the set of the jointly distributed variables, some correspond to output target variables, whose variables are to be inferred when the rest are observed. For example, the target variables may correspond to the labels in a classification task and the rest to the (input) features.

Let us denote the former set by the vector $\mathbf{y}$ and the latter by $\mathbf{x}$. Instead of focusing on the joint distribution $p(\boldsymbol{x},\boldsymbol{y})$, it may be more sensible to focus on $p(\boldsymbol{y}|\boldsymbol{x})$. In [27], graphical models were adopted to encode the conditional distribution, $p(\boldsymbol{y}|\boldsymbol{x})$.

A *conditional random Markov field* is an undirected graph, H, whose nodes correspond to the joint set of random variables, $(\mathbf{x},\mathbf{y})$, but we now assume that the conditional distribution is factorized, that is,

$$p(\boldsymbol{y}|\boldsymbol{x}) = \frac{1}{Z(\boldsymbol{x})}\prod_{k=1}^{K}\psi_k(\boldsymbol{x}_k,\boldsymbol{y}_k), \tag{15.26}$$

where $\{\boldsymbol{x}_k,\boldsymbol{y}_k\} \subseteq \{\boldsymbol{x},\boldsymbol{y}\},\ k = 1,2,\ldots,K$, and

$$Z(\boldsymbol{x}) = \int p(\boldsymbol{y}|\boldsymbol{x})d\boldsymbol{y}, \tag{15.27}$$

where for discrete distributions the integral becomes summation. To stress the difference with Eq. (15.20), note that there it is the joint distribution of all the involved variables that is factorized. As a result, and observing Eqs. (15.20) and (15.26), it turns out that the normalization constant is now a function of $\boldsymbol{x}$. This seemingly minor difference can offer a number of advantages in practice. Avoiding the explicit modeling of $p(\boldsymbol{x})$, we have the benefit of using as inputs variables with complex dependencies, because we do not care to model them. This has led CRFs to be applied in a number of applications such as text mining, bioinformatics, and computer vision. Although we are not going to get involved with CRFs from now on, it suffices to say that the efficient inference techniques, which will be discussed in subsequent sections, can also be adapted, with only minor modifications, to the case of CRFs. For a tutorial on CRFs, including a number of variants and techniques concerning inference and learning, the interested reader is referred to [44].

15.5 FACTOR GRAPHS

In contrast to a Bayesian network, an MRF does not necessarily indicate the specific form of factorization of the corresponding Gibbs distribution. Looking at a Bayesian network, the factorization evolves along the conditional distributions allocated in each node. Let us look at the MRF of Figure 15.17. The corresponding Gibbs distribution could be written as

$$p(x_1,x_2,x_3,x_4) = \frac{1}{Z}\psi_1(x_1,x_2)\psi_2(x_1,x_3)\psi_3(x_3,x_2)\psi_4(x_3,x_4)\psi_5(x_1,x_4), \tag{15.28}$$

or

$$p(x_1,x_2,x_3,x_4) = \frac{1}{Z}\psi_1(x_1,x_2,x_3)\psi_2(x_1,x_3,x_4). \tag{15.29}$$

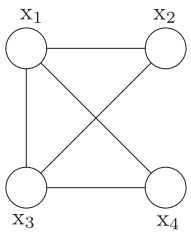

FIGURE 15.17

The Gibbs distribution can be written as a product of factors involving the cliques (x_1, x_2), (x_1, x_3), (x_3, x_4), (x_3, x_2), (x_1, x_4) or of (x_1, x_2, x_3), (x_1, x_3, x_4).

As an extreme case, if all the points of an MRF form a maximal clique, as is the case in Figure 15.13b, we could include only a single product term. Note that aiming at maximal cliques reduces the number of factors, but at the same time the complexity is increased; for example, this can amount to an exponential explosion in the number of terms that have to be learned in the case of discrete variables. At the same time, using large cliques hides modeling details. On the other hand, smaller cliques allow us to be more explicit and detailed in our description.

Factor graphs provide us with the means of making the decomposition of a probability distribution into a product of factors more explicit. A factor graph is an undirected *bipartite* graph involving two types of nodes (thus the term bipartite): one that corresponds to the random variables, denoted by circles; and one to the potential functions, denoted by squares. Edges exist only between two different types of nodes, that is, between "potential function" nodes and "variable" nodes [15, 16, 26].

Figure 15.18a is an MRF for four variables. The respective factor graph in Figure 15.18b corresponds to the product

$$p(x_1, x_2, x_3, x_4) = \frac{1}{Z}\psi_{c_1}(x_1, x_2, x_3)\psi_{c_2}(x_3, x_4), \tag{15.30}$$

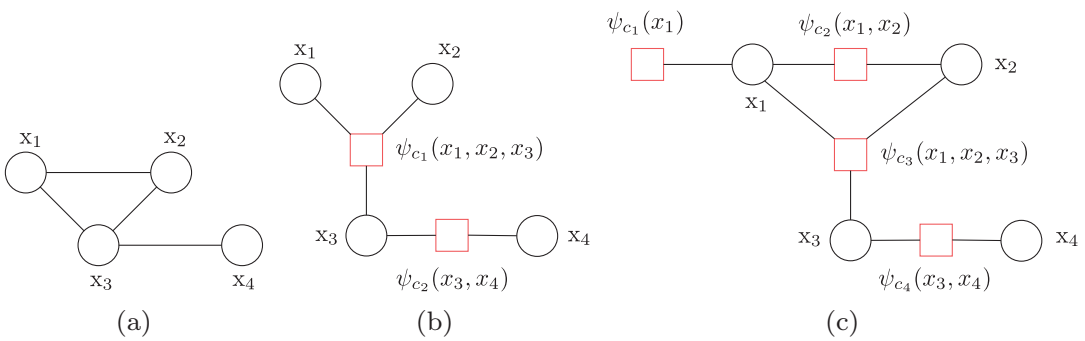

FIGURE 15.18

(a) An MRF and (b), (c) possible equivalent factor graphs at different fine-grained factorization in terms of product factors.

and the one in Figure 15.18c to

$$p(x_1, x_2, x_3, x_4) = \frac{1}{Z}\psi_{c_1}(x_1)\psi_{c_2}(x_1, x_2)\psi_{c_3}(x_1, x_2, x_3)\psi_{c_4}(x_3, x_4). \tag{15.31}$$

As an example, if the potential functions were chosen to express "interactions" among variables using probabilistic information, the involved functions in Figure 15.18 may be chosen as

$$\psi_{c_1}(x_1, x_2, x_3) = p(x_3|x_1, x_2)p(x_2|x_1)p(x_1), \tag{15.32}$$

and

$$\psi_{c_2}(x_3, x_4) = p(x_4|x_3). \tag{15.33}$$

For the case of Figure 15.18c,

$$\psi_{c_1}(x_1) = p(x_1),\ \psi_{c_2}(x_1, x_2) = p(x_2|x_1)$$

$$\psi_{c_3}(x_1, x_2, x_3) = p(x_3|x_1, x_2),\ \psi_{c_4}(x_3, x_4) = p(x_4|x_3).$$

For such an example, in both cases, it is readily seen that $Z = 1$. We will soon see that factor graphs turn out to be very useful for inference computations.

Remarks 15.1.

- A variant of the factor graphs has been more recently introduced, known as *normal factor graphs* (NFG). In an NFG, edges represent variables and vertices represent factors. Moreover, latent and observable variables (internal and external) are distinguished by being represented by edges of degree 2 and degree 1, respectively. Such models can lead to simplified learning algorithms and can nicely unify a number of previously proposed models (see, e.g., [2, 3, 18, 19, 30, 34, 35]).

15.5.1 GRAPHICAL MODELS FOR ERROR-CORRECTING CODES

Graphical models are extensively used for representing a class of error-correcting codes. In the *block parity check* codes (e.g., [31]), one sends k information bits (0, 1 for a binary code) in a block of N bits, $N > k$; thus, redundancy is introduced into the system to cope with the effects of noise in the transmission channel. The extra bits are known as parity-check bits. For each code, a parity-check matrix, H, is defined; in order to be a valid one, for each code word, $\boldsymbol{x}$, it must satisfy the parity check constraint (modulo-2 operations), $H\boldsymbol{x} = \mathbf{0}$. Take as an example the case of $k = 3$ and $N = 6$. The code comprises 2^3 (2^k in general) code words, each of them of length $N = 6$ bits. For the parity-check matrix,

$$H = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 \end{bmatrix},$$

the eight code words that satisfy the parity check constraint are 000000, 001011, 010101, 011110, 100110, 101101, 110011, and 111000. In each one of the eight words, the first three are the information bits and the remaining ones the parity-check bits, which are uniquely determined in order to satisfy the parity-check constraint. Each one of the three parity-check constraints can be expressed via a function, that is,

$$\psi_1(x_1, x_2, x_4) = \delta(x_1 \oplus x_2 \oplus x_4),$$
$$\psi_2(x_1, x_3, x_5) = \delta(x_1 \oplus x_3 \oplus x_5),$$
$$\psi_3(x_2, x_3, x_6) = \delta(x_2 \oplus x_3 \oplus x_6),$$

where $\delta(\cdot)$ is equal to one or zero, depending on whether its argument is one or zero, respectively, and $\oplus$ denotes the modulo-2 addition. The code words are transmitted to a noisy memoryless binary symmetric channel, where each transmitted bit, x_i, may be flipped over and be received as, y_i, according to the following rule:

$$P(y = 0|x = 1) = p, \quad P(y = 1|x = 1) = 1 - p,$$
$$P(y = 1|x = 0) = p, \quad P(y = 0|x = 0) = 1 - p.$$

Upon reception of the observation sequence, y_i, $i = 1, 2, \ldots, N$, one has to decide the value, x_i, that was transmitted. Because the channel has been assumed memoryless, every bit is affected by the noise independently of the other bits, and the overall posterior probability of each codeword is proportional to

$$\prod_{i=1}^{N} P(x_i|y_i).$$

In order to guarantee that only valid codewords are considered, and assuming *equiprobable* information bits, we write the joint probability as

$$P(\boldsymbol{x}, \boldsymbol{y}) = \frac{1}{Z}\psi_1(x_1, x_2, x_4)\psi_2(x_1, x_3, x_5)\psi_3(x_2, x_3, x_6)\prod_{i=1}^{N} P(y_i|x_i),$$

where the parity-check constraints have been taken into account. The respective factor model is shown in Figure 15.19, where

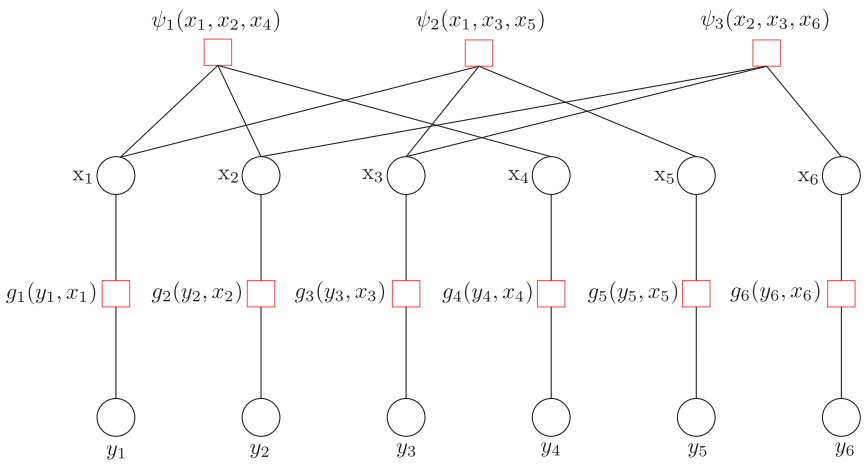

FIGURE 15.19

Factor graph for a (3,3) parity-check code.

$$g_i(y_i, x_i) = P(y_i|x_i).$$

The task of decoding is to derive an efficient inference scheme to compute the posteriors, and based on that to decide in favor of 1 or 0.

15.6 MORALIZATION OF DIRECTED GRAPHS

At a number of points we have already made bridges between Bayesian networks and MRFs. In this section, we will formalize the bridge and see how one can convert a Bayesian network to an MRF and discuss the subsequent effects of such a conversion on the implied conditional independencies.

We can trust common sense to drive us to construct such a conversion. Because the conditional distributions will play the role of the potential functions (factors), one has to make sure that edges do exist among all the involved variables in each one of these factors. Because edges from the parents to children exist, we have to (a) retain these edges and make them undirected and (b) add edges between nodes that are parents of a common child. This is shown in Figure 15.20. In Figure 15.20a, a DAG is shown, which is converted to the MRF of Figure 15.20b by adding undirected edges between x_1, x_2 (parents of x_3) and x_3, x_6 (parents of x_5). The procedure is known as *moralization* and the resulting undirected graph as a *moral graph*. The terminology stems from the fact that "parents are forced to be married." This conversion will be very useful soon, when an inference algorithm will be stated that covers both Bayesian networks and MRFs in a unifying framework.

The obvious question that is now raised is how moralization affects independencies. It turns out that if H is the resulting moral graph, then $I(H) \subseteq I(G)$ (Problem 15.11). In other words, the moral graph can guarantee a smaller number of independencies compared to the original BN via its set of d-separations. This is natural because one adds extra links. For example, in Figure 15.20a, x_1, x_2 in the converging node $x_1 \rightarrow x_3 \leftarrow x_2$ are marginally independent, not given x_3. However, in the resulting moral graph in 15.20b, this independence is lost. It can be shown that moralization adds the fewest extra links and hence retains the maximum number of independence, see for example, [25].

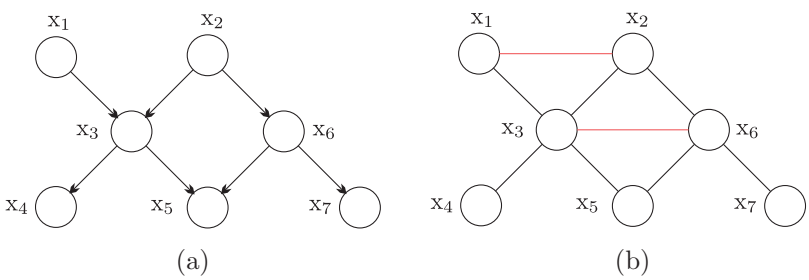

FIGURE 15.20

(a) A DAG and (b) the resulting MRF after applying moralization on the DAG. Directed edges become undirected and new edges, shown in red, are added to "marry" parents with common child-nodes.

15.7 EXACT INFERENCE METHODS: MESSAGE-PASSING ALGORITHMS

This section deals with efficient techniques for inference on undirected graphical models. So, even if our starting point was a BN, we assume that it was converted to an undirected one prior to the inference task. We will begin with the simplest case of graphical models—graphs comprising a chain of nodes. This will help the reader to grasp the basic notions behind exact inference schemes. It is interesting to note that, in general, the inference task in graphical models is an NP-hard one [10]. Moreover, it has been shown that for general Bayesian networks, approximate inference to a desired number of digits of precision is also an NP-hard task, [11]; that is, the time required has an exponential dependence on the number of digits of accuracy. However, as we will see in a number of cases that are commonly encountered in practice, the exponential growth in computational time can be bypassed by exploiting the underlying independencies and factorization properties of the associated distributions.

The inference tasks of interest are (a) computing the likelihood, (b) computing the marginals of the involved variables, (c) computing conditional probability distributions, and (d) finding modes of distributions.

15.7.1 EXACT INFERENCE IN CHAINS

Let us consider the chain graph of Figure 15.21 and focus our interest on computing marginals. The naive approach, which overlooks factorization and independencies, would be to compute first the joint distribution. Let us concentrate on discrete variables and assume that each one of them, l in total, has K states. Then, in order to compute the marginal of, say, x_j, we have to obtain the sum

$$\begin{aligned} P(x_j) :&= \sum_{x_1} \cdots \sum_{x_{j-1}} \sum_{x_{j+1}} \cdots \sum_{x_l} P(x_1, \ldots, x_l) \\ &= \sum_{x_i : i \neq j} P(x_1, \ldots, x_l). \end{aligned} \tag{15.34}$$

Each summation is over K values, hence, the number of the required computations amounts to $\mathcal{O}(K^l)$. Let us now bring factorization into the game and concentrate on computing $P(x_1)$. Assume that the joint probability factorizes over the graph, hence, we can write

$$\boxed{P(\boldsymbol{x}) := P(x_1, x_2, \ldots, x_l) = \frac{1}{Z} \prod_{i=1}^{l-1} \psi_{i,i+1}(x_i, x_{i+1}),} \tag{15.35}$$

and

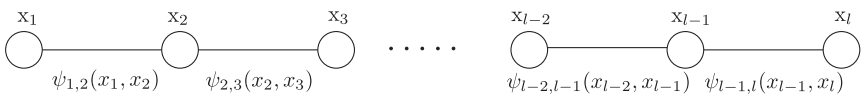

FIGURE 15.21

An undirected chain graph with l nodes. There are $l - 1$ cliques consisting of pairs of nodes.

$$P(x_1) = \frac{1}{Z} \sum_{x_i: i \neq 1} \prod_{i=1}^{l-1} \psi_{i,i+1}(x_i, x_{i+1}). \tag{15.36}$$

Note that the only term that depends on x_l is $\psi_{l-1,l}(x_{l-1}, x_l)$. Let us start by summing with respect to this last term, which leaves unaffected all the preceding factors in the sequence of products in Eq. (15.36),

$$P(x_1) = \frac{1}{Z} \sum_{x_i: i \neq 1, l} \prod_{i=1}^{l-2} \psi_{i,i+1}(x_i, x_{i+1}) \sum_{x_l} \psi_{l-1,l}(x_{l-1}, x_l), \tag{15.37}$$

where we exploited the basic property of arithmetic

$$\sum_i \alpha \beta_i = \alpha \sum_i \beta_i. \tag{15.38}$$

- Define:

$$\sum_{x_l} \psi_{l-1,l}(x_{l-1}, x_l) := \mu_b(x_{l-1}).$$

  Because the possible values of the pair (x_{l-1}, x_l) comprise a table with K^2 elements, the summation involves K^2 terms and $\mu_b(x_{l-1})$ consists of K possible values.
- After marginalizing out x_l, the only factor in the product that depends on x_{l-1} is

$$\psi_{l-2,l-1}(x_{l-2}, x_{l-1}) \mu_b(x_{l-1}).$$

  Then, in a similar way as before we obtain

$$P(x_1) = \frac{1}{Z} \sum_{x_i: i \neq 1, l-1, l} \prod_{i=1}^{l-3} \psi_{i,i+1}(x_i, x_{i+1}) \sum_{x_{l-1}} \psi_{l-2,l-1}(x_{l-2}, x_{l-1}) \mu_b(x_{l-1}),$$

  where this summation also involves K^2 terms.

We are now ready to define the general recursion as

$$\boxed{\begin{aligned} \mu_b(x_i) &:= \sum_{x_{i+1}} \psi(x_i, x_{i+1}) \mu_b(x_{i+1}), \\ \mu_b(x_l) &= 1, \end{aligned}} \tag{15.39}$$

whose repeated application leads to

$$\mu_b(x_1) = \sum_{x_1} \psi_{1,2}(x_1, x_2) \mu_b(x_2),$$

and finally

$$P(x_1) = \frac{1}{Z} \mu_b(x_1). \tag{15.40}$$

The series of recursions is illustrated in Figure 15.22.

We can think that every node, x_i, (a) receives a message from its right, $\mu_b(x_i)$, which for our case comprises K values, (b) performs locally sum-multiply operations and a new message $\mu_b(x_{i-1})$ is

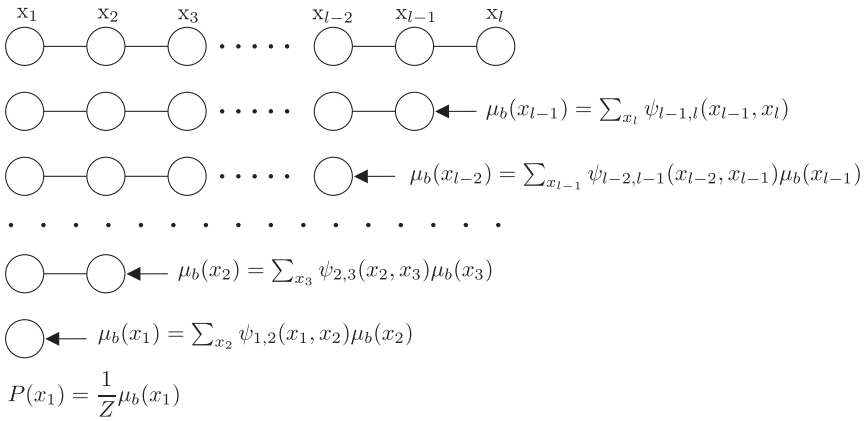

FIGURE 15.22

To compute $P(x_1)$, starting from the last node, x_l, each node (a) receives a message; (b) processes it locally via sum and product operations, which produces a new message; and (c) the latter is passed backward, to the node on its left. We have assumed that $\mu_b(x_l) = 1$.

computed, which (c) is passed to its left, to node x_{i-1}. The subscript "b," in μ_b, denotes "backward" to remind us of the flow of the message-passing activity from right to left.

If we wanted to compute $P(x_l)$, we would adopt the same reasoning but start summation from x_1. In this case, message-passing takes place forward (from left to right) and messages are defined as

$$\begin{aligned} &\mu_f(x_{i+1}) := \sum_{x_i} \psi_{i,i+1}(x_i, x_{i+1})\mu_f(x_i), \quad i = 1, \ldots, l-1, \\ &\mu_f(x_1) = 1, \end{aligned} \tag{15.41}$$

where "f" has been used to denote "forward" flow. The procedure is shown in Figure 15.23.

The term $\mu_b(x_j)$ is the result of summing the products over $x_{j+1}, x_{j+2}, \ldots, x_l$, and the term $\mu_f(x_j)$ over $x_1, x_2, \ldots, x_{j-1}$. At each iteration step, one variable is *eliminated* by summing up over all its possible values. It can be easily shown, following similar arguments, that the marginal at any point, x_j, $2 \leq j \leq l-1$ is obtained by (Problem 15.13)

$$P(x_j) = \frac{1}{Z}\mu_f(x_j)\mu_b(x_j), \quad j = 2, 3, \ldots, l-1. \tag{15.42}$$

The idea is to perform one forward and one backward message-passing operation, store the values, and then compute any one of the marginals of interest. The total cost will be $\mathcal{O}(2K^2 l)$, instead of K^l of the naive approach.

We still have to compute the normalizing constant Z. This is readily obtained by summing up both sides of Eq. (15.42), which requires $\mathcal{O}(K)$ operations,

$$Z = \sum_{x_j=1}^{K} \mu_f(x_j)\mu_b(x_j). \tag{15.43}$$

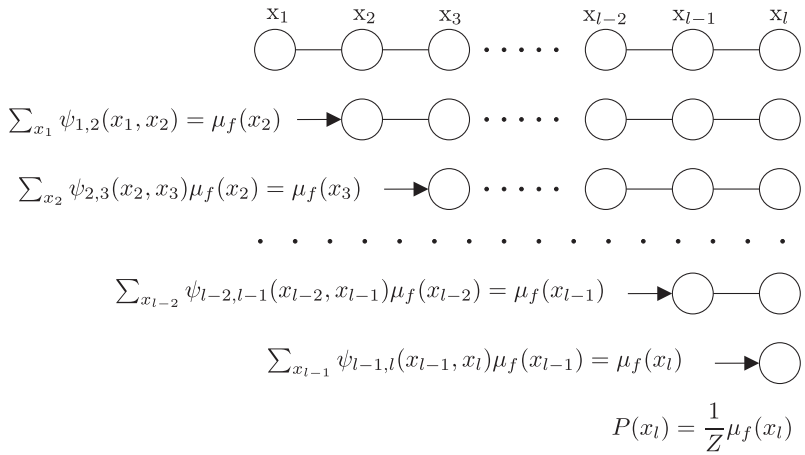

FIGURE 15.23

To compute $P(x_l)$, message-passing takes place in the forward direction, from left to right. As opposed to Figure 15.22, messages are denoted as μ_f, to remind us of the forward flow.

So far, we have considered the computation of marginal probabilities. Let us now turn our attention to their conditional counterparts. We start with the simplest case, for example, to compute $P(x_j|\mathrm{x}_k = \hat{x}_k)$, $k \neq j$. That is, we assume that variable x_k has been observed and its value is $\hat{x}_k$. The first step in computing the conditional is to recover the joint $P(x_j, \mathrm{x}_k = \hat{x}_k)$. This is an normalized version of the respective conditional, which can then be obtained as

$$P(x_j|\mathrm{x}_k = \hat{x}_k) = \frac{P(x_j, \mathrm{x}_k = \hat{x}_k)}{P(\hat{x}_k)}. \tag{15.44}$$

The only difference in computing $P(x_j, \mathrm{x}_k = \hat{x}_k)$ compared to the previous computations of the marginals is that now, in order to obtain the messages, we *do not sum* with respect to x_k. We just clump the respected potential function to its value $\hat{x}_k$. That is, the computations

$$\mu_b(x_{k-1}) = \sum_{x_k} \psi_{k-1,k}(x_{k-1}, x_k)\mu_b(x_k),$$
$$\mu_f(x_{k+1}) = \sum_{x_k} \psi_{k,k+1}(x_k, x_{k+1})\mu_f(x_k),$$

are replaced by

$$\mu_b(x_{k-1}) = \psi_{k-1,k}(x_{k-1}, \hat{x}_k)\mu_b(\hat{x}_k),$$
$$\mu_f(x_{k+1}) = \psi_{k,k+1}(\hat{x}_k, x_{k+1})\mu_f(\hat{x}_k).$$

In other words, x_k is considered a delta function at the instantiated value. Once $P(x_j, \mathrm{x}_k = \hat{x}_k)$ has been obtained, normalization is straightforward and is locally performed at the jth node. The procedure can be generalized when more than one variable is observed.

15.7.2 EXACT INFERENCE IN TREES

Having gained experience and learned the basic secrets in developing efficient inference algorithms for chains, we turn our attention to the more general case involving *tree-structured* undirected graphical models.

A tree is a graph in which there is a single path between any two nodes of the graph; thus, there are no cycles in the graph. Figures 15.24a and b are two examples of trees. Note that in a directed tree, any node has only a single parent. A tree can be directed or undirected. Furthermore, because in a directed one there are no children with two parents, the moralization step, which converts a directed graph to an undirected one, adds no extra links. The only change consists of making the edges undirected.

There is an important property of the trees, which will prove very important for our current needs. Let us denote a tree graph as T, which is a collection of vertices/nodes V and edges E, which link the nodes, that is, $T = \{V, E\}$. Consider any node $\mathrm{x} \in V$ and consider the set of all its neighbors, that is, all nodes that share an edge with x. Denote this set as

$$\mathcal{N}(\mathrm{x}) = \{\mathrm{y} \in V : (\mathrm{x}, \mathrm{y}) \in E\}.$$

Looking at Figure 15.24b, we have that $\mathcal{N}(\mathrm{x}) = \{\mathrm{y}, \mathrm{u}, \mathrm{z}, \mathrm{v}\}$. Then, for *each* element $\mathrm{r} \in \mathcal{N}(\mathrm{x})$, define the subgraph $T_r = \{V_r, E_r\}$ such that *any* node in this subgraph can be reached from r via paths that *do not pass* through x. In Figure 15.24b, the respective subgraphs, each associated with one element in (y, u, z, v), are encircled by dotted lines. By the definition of a tree, it can easily be deduced that each one of these subgraphs is also a tree. Moreover, these subgraphs are *disjoint*. In other words, *each one* of the neighboring nodes of a node, for example, x, can be viewed as a *root* of a subtree, and these subtrees are mutually disjoint, having no common nodes. This property will allow us to break a large problem into a number of smaller ones. Moreover, each one of the smaller problems can be further divided in

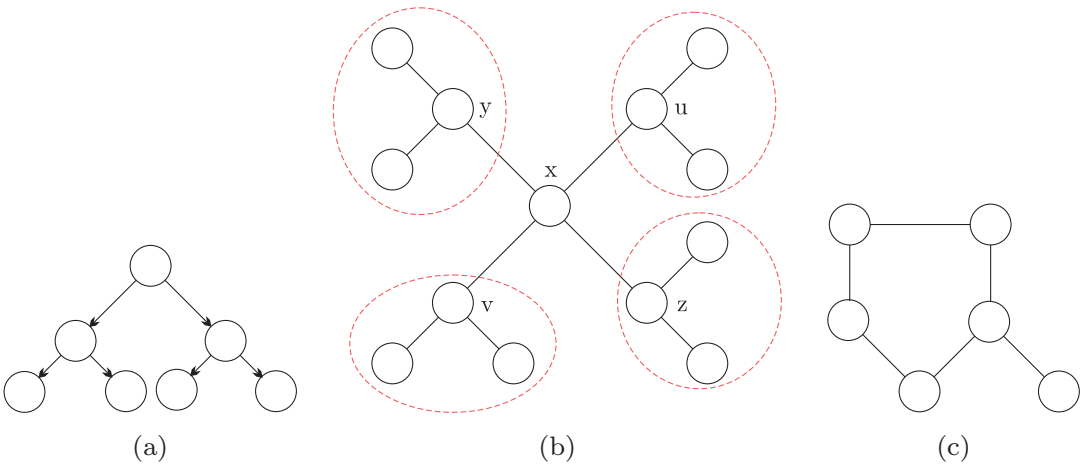

FIGURE 15.24

Examples of (a) directed and (b) undirected trees. Note that in the directed one, any node has a single parent. In both cases, there is only a single chain that connects any two nodes. (c) The graph is not a tree because there is a cycle.

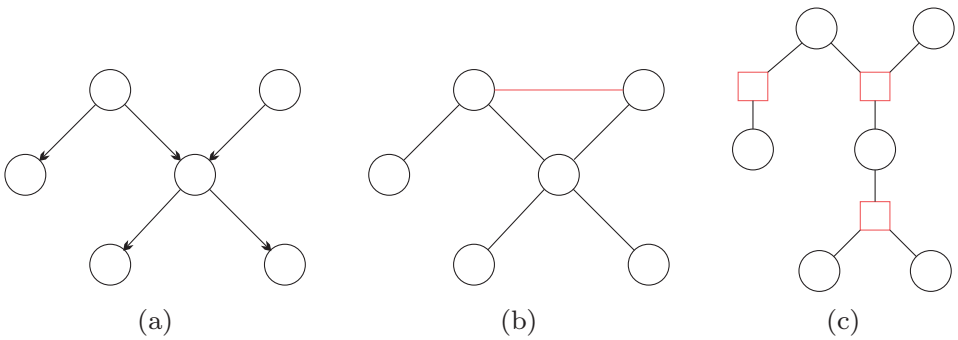

FIGURE 15.25

(a) Although there are no cycles, one of the nodes has two parents, hence the graph is a polytree. (b) The resulting structure after moralization has a cycle. (c) A factor graph for the polytree in (a).

the same way, being itself a tree. We have now all the basic ingredients to derive an efficient scheme for inference on trees (recall that such a breaking of a large problem into a sequence of smaller ones was at the heart of the message-passing algorithm for chains). However, let us first bring the notion of factor graphs into the scene. The reason is that using factor graphs allows us to deal with some more general graph structures, such as *polytrees*.

A directed polytree is a graph in which, although there are no cycles, a child may have more than one parent. Figure 15.25a shows an example of a polytree. The unpleasant situation results after the moralization step because marrying the parents results in cycles, and we *cannot* derive *exact* inference algorithms in graphs with cycles. However, if one converts the original directed polytree into a factor graph, the resulting bipartite entity has a tree structure, with no cycles involved. Thus, everything we said before about tree structures applies to these factor graphs.

15.7.3 THE SUM-PRODUCT ALGORITHM

We will develop the algorithm in a "bottom-up" approach, via the use of an example. Once the rationale is understood, the generalization can readily be obtained. Let us consider the factor tree of Figure 15.26. The factor nodes are denoted by capital letters and squares, and each one is associated with a potential function. The rest are variable nodes, denoted by circles. Assume that we want to compute the marginal $P(x_1)$. Node x_1, being a variable node, is connected to factor nodes only. We split the graph in as many (tree) subgraphs as the factor nodes connected to x_1 (three in our case). In the figure, each one of these subgraphs is encircled, having as roots the nodes A, H, and G, respectively. Recall that the joint $P(\boldsymbol{x})$ is given as the product of all the potential functions, each one associated with one factor node, divided by the normalizing constant, Z. Focusing on the node of interest, x_1, this product can be written as

$$P(\boldsymbol{x}) = \frac{1}{Z}\psi_A(x_1, \boldsymbol{x}_A)\psi_H(x_1, \boldsymbol{x}_H)\psi_G(x_1, \boldsymbol{x}_G), \tag{15.45}$$

where $\boldsymbol{x}_A$ denotes the vector corresponding to all the variables in T_A; the vectors $\boldsymbol{x}_H$ and $\boldsymbol{x}_G$ are similarly defined. The function $\psi_A(x_1, \boldsymbol{x}_A)$ is the product of all the potential functions associated with the factor

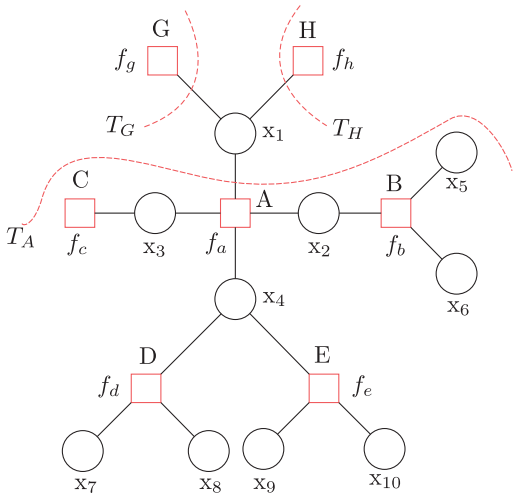

FIGURE 15.26

The tree is subdivided into three subtrees, each one having as its root one of the factor nodes connected to x_1. This is the node whose marginal is computed in the text. Messages are initiated from the leaf nodes toward x_1. Once messages arrive at x_1, a new propagation of messages starts, this time from x_1 to the leaves.

nodes in T_A, and $\psi_H(x_1, \boldsymbol{x}_H)$, $\psi_G(x_1, \boldsymbol{x}_G)$ are defined in an analogous way. Then, the marginal of interest is given by

$$P(x_1) = \frac{1}{Z} \sum_{\boldsymbol{x}_A \in V_A} \sum_{\boldsymbol{x}_H \in V_H} \sum_{\boldsymbol{x}_G \in V_G} \psi_A(x_1, \boldsymbol{x}_A) \psi_H(x_1, \boldsymbol{x}_H) \psi_G(x_1, \boldsymbol{x}_G). \tag{15.46}$$

We will concentrate on the subtree with root A, denoted as $T_A := \{V_A, E_A\}$, where V_A stands for the nodes in T_A and E_A for the respective set of edges. Because the three subtrees are disjoint, we can split the previous expression in Eq. (15.46) into

$$P(x_1) = \frac{1}{Z} \sum_{\boldsymbol{x}_A \in V_A} \psi_A(x_1, \boldsymbol{x}_A) \sum_{\boldsymbol{x}_H \in V_H} \psi_H(x_1, \boldsymbol{x}_H) \sum_{\boldsymbol{x}_G \in V_G} \psi_G(x_1, \boldsymbol{x}_G). \tag{15.47}$$

Note that $x_1 \notin V_A \cup V_H \cup V_G$. Having reserved the symbol $\psi_A(\cdot, \cdot)$ to denote the product of all the potentials in the subtree T_A (and similarly for T_H, T_G), let us denote the individual potential functions, for each one of the factor nodes, via the symbol f, as shown in Figure 15.26. Thus, we can now write

$$\begin{aligned} \sum_{\boldsymbol{x}_A \in V_A} \psi_A(x_1, \boldsymbol{x}_A) &= \sum_{\boldsymbol{x}_A \in V_A} f_a(x_1, x_2, x_3, x_4) f_c(x_3) f_b(x_2, x_5, x_6) \\ &\quad \times f_d(x_4, x_7, x_8) f_e(x_4, x_9, x_{10}) \\ &= \sum_{x_2} \sum_{x_3} \sum_{x_4} f_a(x_1, x_2, x_3, x_4) f_c(x_3) \sum_{x_7} \sum_{x_8} f_d(x_4, x_7, x_8) \\ &\quad \times \sum_{x_9} \sum_{x_{10}} f_e(x_4, x_9, x_{10}) \sum_{x_6} \sum_{x_5} f_b(x_2, x_5, x_6). \end{aligned} \tag{15.48}$$

Recall from our treatment of the chain graph that messages were nothing but locally computed summations over products. Having this experience, let us define as

$$\mu_{f_b \to x_2}(x_2) = \sum_{x_6} \sum_{x_5} f_b(x_2, x_5, x_6),$$

$$\mu_{f_e \to x_4}(x_4) = \sum_{x_9} \sum_{x_{10}} f_e(x_4, x_9, x_{10}),$$

$$\mu_{f_d \to x_4}(x_4) = \sum_{x_7} \sum_{x_8} f_d(x_4, x_7, x_8),$$

$$\mu_{f_c \to x_3}(x_3) = f_c(x_3),$$

$$\mu_{x_4 \to f_a}(x_4) = \mu_{f_d \to x_4}(x_4)\mu_{f_e \to x_4}(x_4),$$

$$\mu_{x_2 \to f_a}(x_2) = \mu_{f_b \to x_2}(x_2),$$

$$\mu_{x_3 \to f_a}(x_3) = \mu_{f_c \to x_3}(x_3),$$

and

$$\mu_{f_a \to x_1}(x_1) = \sum_{x_2} \sum_{x_3} \sum_{x_4} f_a(x_1, x_2, x_3, x_4)\mu_{x_2 \to f_a}(x_2)\mu_{x_3 \to f_a}(x_3)\mu_{x_4 \to f_a}(x_4).$$

Observe that we were led to define two types of messages; one type is passed from variable nodes to factor nodes and the other one for messages passed from factor to variable nodes.

- Variable node to factor node messages (Figure 15.27a):

$$\boxed{\mu_{x \to f}(x) = \prod_{s: f_s \in \mathcal{N}(x) \setminus f} \mu_{f_s \to x}(x).} \tag{15.49}$$

We use $\mathcal{N}(\mathrm{x})$ to denote the set of the nodes with which a variable node, x, is connected. $\mathcal{N}(\mathrm{x}) \setminus f$ refers to all nodes *excluding* the factor node, f; note that all these nodes are factor nodes. In other

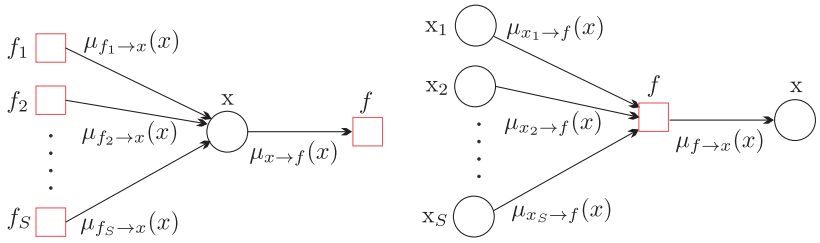

FIGURE 15.27

(a) Variable x is connected to S factor nodes, besides f; that is, $\mathcal{N}(x) \setminus f = \{f_1, f_2, \ldots, f_S\}$. The output message from x to f is the product of the incoming messages. The arrows indicate directions of flow of the message propagation. (b) The factor node f is connected to S node variables, besides x; that is, $\mathcal{N}(f) \setminus \mathrm{x} = \{\mathrm{x}_1, \mathrm{x}_2, \ldots, \mathrm{x}_S\}$.

words, the action of a variable node, as far as message-passing is concerned, is to multiply incoming messages. Obviously, if it is connected only to one factor node (except f), then such a variable node passes what it receives, without any computation. This is, for example, the case of $\mu_{x_2\to f_a}$, as previously defined.

- Factor node to variable node messages (Figure 15.27b):

$$\boxed{\mu_{f\to x}(x)=\sum_{x_i\in\mathcal{N}(f)\backslash \mathrm{x}} f(\boldsymbol{x}^f)\prod_{i:x_i\in\mathcal{N}(f)\backslash \mathrm{x}}\mu_{x_i\to f}(x_i),} \tag{15.50}$$

where $\mathcal{N}(f)$ denotes the set of the (variable) nodes connected to f and $\mathcal{N}(f)\setminus \mathrm{x}$ the corresponding set if we exclude node x. The vector $\boldsymbol{x}^f$ comprises all the variables involved as arguments in f, that is, all the variables/nodes in $\mathcal{N}(f)$.

If a node is a leaf, we adopt the following convention. If it is a variable node, x, connected to a factor node, f, then

$$\mu_{x\to f}(x)=1. \tag{15.51}$$

If it is a factor node, f, connected to variable node, x, then

$$\mu_{f\to x}(x)=f(x). \tag{15.52}$$

Adopting the previously stated definitions, Eq. (15.48) is now written as

$$\begin{aligned}&\sum_{\boldsymbol{x}_A\in V_A}\psi_A(x_1,\boldsymbol{x}_A)\\&=\sum_{x_2}\sum_{x_3}\sum_{x_4}f_a(x_1,x_2,x_3,x_4)\mu_{x_2\to f_a}(x_2)\mu_{x_3\to f_a}(x_3)\mu_{x_4\to f_a}(x_4)\\&=\mu_{f_a\to x_1}(x_1).\end{aligned} \tag{15.53}$$

Working similarly for the other two subtrees, T_G and T_H, we finally obtain

$$P(x_1)=\frac{1}{Z}\mu_{f_a\to x_1}(x_1)\mu_{f_g\to x_1}(x_1)\mu_{f_h\to x_1}(x_1). \tag{15.54}$$

Note that each summation can be viewed as a step that "removes" a variable and produces a message. This is the reason that sometimes this procedure is called *variable elimination*. We are now ready to summarize the steps of the algorithm.

The Algorithmic Steps

1. Pick the variable, x, whose marginal, $P(x)$, will be computed.
2. Divide the tree into as many subtrees as the number of factor nodes to which the variable node, x, is connected.
3. For each one of these subtrees, identify the leaf nodes.
4. Start message-passing, toward x, by initializing the leaf nodes according to Eqs. (15.51) and (15.52), by utilizing Eqs. (15.49) and (15.50).
5. Compute the marginal according to Eq. (15.54), or in general

$$P(x)=\frac{1}{Z}\prod_{s:f_s\in\mathcal{N}(\mathrm{x})}\mu_{f_s\to x}(x). \tag{15.55}$$

The normalizing constant, Z, can be obtained by adding both sides of Eq. (15.55) over all possible values of x. As was the case with the chain graphs, if a variable is observed, then we replace the summation over this variable, in the places where this is required, by a single term evaluated on the observed value.

Although so far we have considered discrete variables, everything that has been said also applies to continuous variables by substituting summations by integrals. For such integrations, Gaussian models turn out to be very convenient.

Remarks 15.2.

- Thus far, we have concentrated on the computation of the marginal for a single variable, x. If the marginal of another variable is required, the obvious way would be for the whole process to be repeated. However, as we already commented in subsection 15.7.1, such an approach is computationally wasteful because many of the computations are common and can be shared for the evaluation of the marginals of the various nodes. Note that in order to compute the marginal at any variable node, one needs all the messages from the factor nodes, to which the specific variable node is connected, to be available (15.55). Assume now that we pick one node, say x_1, and compute the marginal, once all the required messages have "arrived." Then, this node initiates a new message-propagation phase, this time toward the leaves. It is not difficult to see (Problem 15.15) that this process, once completed, will make available to every node all the required messages for the computation of the respective marginals. In other words, this two stage message-passing procedure, in two opposite-flow directions, suffices to provide all the necessary information for the computation of the marginals at every node. The total number of messages passed is just twice the number of edges in the graph. Similar to the case of chain graphs, in order to compute conditional probabilities, say $P(x_i|\mathrm{x}_k = \hat{x}_k)$, node x_k has to be instantiated. Running the sum-product algorithm will provide the joint probability $P(x_i, \mathrm{x}_k = \hat{x}_k)$, from which the respective conditional is obtained after normalization; this is performed locally at the respective variable node.
- The (joint) marginal probability of all the variables, $\mathrm{x}_1, \mathrm{x}_2, \ldots, \mathrm{x}_S$, associated with a factor node, f, is given by (Problem 15.16),

$$P(x_1, \ldots, x_S) = \frac{1}{Z} f(x_1, \ldots, x_S) \prod_{s=1}^{S} \mu_{x_s \to f}(x_s). \tag{15.56}$$

- Earlier versions of the sum-product algorithm, known as *belief propagation*, were first developed in the context of singly connected graphs independently in [28, 36, 37]. However, the problem of variable elimination has an older history and has been discovered in different communities (e.g., [4, 6, 39]). Sometimes, the general sum-product algorithm, as described before, is also called *the generalized forward-backward algorithm*.

15.7.4 THE MAX-PRODUCT AND MAX-SUM ALGORITHMS

Let us now turn our attention from marginals to modes of distributions. That is, given a distribution, $P(\boldsymbol{x})$, that factorizes over a tree (factor) graph, the task is to compute efficiently the quantity

$$\max_{\boldsymbol{x}} P(\boldsymbol{x}).$$

We will focus on discrete variables. Following similar arguments as before, one can readily write the counterpart of Eq. (15.46), for the case of Figure 15.26, as

$$\max_{\boldsymbol{x}} P(\boldsymbol{x}) = \frac{1}{Z} \max_{x_1} \max_{\boldsymbol{x}_A \in V_A} \max_{\boldsymbol{x}_H \in V_H} \max_{\boldsymbol{x}_G \in V_G} \psi_A(x_1, \boldsymbol{x}_A)\psi_H(x_1, \boldsymbol{x}_H)\psi_G(x_1, \boldsymbol{x}_G). \tag{15.57}$$

Exploiting the property of the max operator, that is,

$$\max_{b,c}(ab, ac) = a \max_{b,c}(b, c) \ , \quad a \geq 0,$$

we can rewrite Eq. (15.57) as

$$\max_{\boldsymbol{x}} P(\boldsymbol{x}) = \frac{1}{Z} \max_{x_1} \max_{\boldsymbol{x}_A \in V_A} \psi_A(x_1, \boldsymbol{x}_A) \max_{\boldsymbol{x}_H \in V_H} \psi_H(x_1, \boldsymbol{x}_H) \max_{\boldsymbol{x}_G \in V_G} \psi_G(x_1, \boldsymbol{x}_G).$$

Following similar arguments as for the sum-product rule, we arrive at the counterpart of Eq. (15.48), that is,

$$\begin{aligned} \max_{x_1} \max_{\boldsymbol{x}_A \in V_A} \psi_A(x_1, \boldsymbol{x}_A) = \max_{x_1} \max_{x_2, x_3, x_4} f_a(x_1, x_2, x_3, x_4) f_c(x_3) \max_{x_7, x_8} f_d(x_4, x_7, x_8) \\ \times \max_{x_9, x_{10}} f_e(x_4, x_9, x_{10}) \max_{x_5, x_6} f_b(x_2, x_6, x_5). \end{aligned} \tag{15.58}$$

Equation (15.58) suggests that everything that was said before for the sum-product message-passing algorithm holds true here, provided we replace summations with the max operations, the definitions of the messages passed between nodes change to

$$\mu_{x \to f}(x) = \prod_{s: f_s \in \mathcal{N}(x) \backslash f} \mu_{f_s \to x}(x), \tag{15.59}$$

and

$$\boxed{\mu_{f \to x}(x) = \max_{x_i : x_i \in \mathcal{N}(f) \backslash x} f(\boldsymbol{x}^f) \prod_{i: x_i \in \mathcal{N}(f) \backslash x} \mu_{x_i \to f}(x_i),} \tag{15.60}$$

with the same definition of symbols as for Eq. (15.50). Then, the mode of $P(\boldsymbol{x})$ is given by

$$\max_{\boldsymbol{x}} P(\boldsymbol{x}) = \frac{1}{Z} \max_{x_1} \mu_{f_a \to x_1}(x_1) \mu_{f_g \to x_1}(x_1) \mu_{f_h \to x_1}(x_1), \tag{15.61}$$

or in general

$$\max_{\boldsymbol{x}} P(\boldsymbol{x}) = \frac{1}{Z} \max_{x} \prod_{s: f_s \in \mathcal{N}(x)} \mu_{f_s \to x}(x), \tag{15.62}$$

where x is the node chosen to play the role of the root, toward which the flow of the messages is directed, starting from the leaves. The resulting scheme is known as the *max-product algorithm*.

In practice, an alternative formulation of the previously stated max-product algorithm is usually adopted. Often, the involved potential functions are probabilities (by absorbing the normalization constant) and their magnitude is less than one; however, if a large number of product terms is involved it may lead to arithmetic inaccuracies. A way to bypass this is to involve the logarithmic function, which transforms products into summations. This is justified by the fact that the logarithmic function is monotonic and increasing, hence it does not affect the point $\boldsymbol{x}$ at which a maximum occurs, that is,

$$\boldsymbol{x}_* := \arg\max_{\boldsymbol{x}} P(\boldsymbol{x}) = \arg\max_{\boldsymbol{x}} \ \ln P(\boldsymbol{x}). \tag{15.63}$$

Under this formulation, the following *max-sum* version of the algorithm results. It is straightforward to see that Eqs. (15.59) and (15.60) now take the form of

$$\mu_{x\to f}(x) = \sum_{s:f_s\in\mathcal{N}(\mathrm{x})\setminus f} \mu_{f_s\to x}(x), \tag{15.64}$$

$$\mu_{f\to x}(x) = \max_{x_i:x_i\in\mathcal{N}(f)\setminus\mathrm{x}} \left\{ \ln f(\boldsymbol{x}^f) + \sum_{i:x_i\in\mathcal{N}(f)\setminus\mathrm{x}} \mu_{x_i\to f}(x_i) \right\}. \tag{15.65}$$

In place of Eqs. (15.51) and (15.52) for the initial messages, sent by the leaf nodes, we now define

$$\mu_{x\to f}(x) = 0, \text{ and } \mu_{f\to x}(x) = \ln f(x). \tag{15.66}$$

Note that after one pass of the message flow, the maximum value of $P(\boldsymbol{x})$ has been obtained. However, one is also interested in knowing the corresponding value $\boldsymbol{x}_*$, for which the maximum occurs, that is,

$$\boldsymbol{x}_* = \arg\max_{\boldsymbol{x}} P(\boldsymbol{x}).$$

This is achieved by a reverse message-passing process, which is slightly different than what we have discussed so far, and it is known as back-tracking.

Back-tracking: Assume that x_1 is the chosen node to play the role of the root, where the flow of messages "converge." From Eq. (15.61), we get

$$x_{1*} = \arg\max_{x_1} \mu_{f_a\to x_1}(x_1)\mu_{f_g\to x_1}(x_1)\mu_{f_h\to x_1}(x_1). \tag{15.67}$$

A new message-passing flow now starts, and the root node, x_1, passes the obtained optimal value to the factor nodes, to which it is connected. Let us follow this message-passing flow within the nodes of the subtree T_A.

- Node A: It receives x_{1*} from node x_1.
  - Selection of the optimal values: Recall that

$$\mu_{f_a\to x_1}(x_1) = \max_{x_2,x_3,x_4} f_a(x_1,x_2,x_3,x_4)\mu_{x_4\to f_a}(x_4) \times \mu_{x_3\to f_a}(x_3)\mu_{x_2\to f_a}(x_2).$$

    Thus, for different values of x_1, different optimal values for (x_2, x_3, x_4) will result. For example, assume that in our discrete variable setting, each variable can take one out of four possible values, that is, $\mathrm{x} \in \{1, 2, 3, 4\}$. Then, if $x_{1*} = 2$, say that the resulting optimal values are $(x_{2*}, x_{3*}, x_{4*}) = (1, 1, 3)$. On the other hand, if $x_{1*} = 4$, then maximization may result to, let us say, $(x_{2*}, x_{3*}, x_{4*}) = (2, 3, 4)$. However, having obtained a *specific* value for x_{1*} via the maximization at node x_1, we choose the triplet (x_{2*}, x_{3*}, x_{4*}) such as

$$(x_{2*}, x_{3*}, x_{4*}) = \arg\max_{x_2,x_3,x_4} f_a(x_{1*},x_2,x_3,x_4)\mu_{x_4\to f_a}(x_4) \times \mu_{x_3\to f_a}(x_3)\mu_{x_2\to f_a}(x_2). \tag{15.68}$$

    Hence, during the first pass, the obtained optimal values have to be stored to be used during the second (backward) pass.
  - Message-passing: Node A passes x_{4*} to node x_4, x_{2*} to node x_2 and x_{3*} to node x_3.

- Node x_4 passes x_{4*} to nodes D and E.
- Node D
  - Selection of the optimal values: Select (x_{7*}, x_{8*}) such as

$$(x_{7*}, x_{8*}) = \arg\max_{x_7, x_8} f_d(x_{4*}, x_7, x_8)\mu_{x_7 \to f_d}(x_7)\mu_{x_8 \to f_d}(x_8).$$

  - Message Passing: Node D passes (x_{7*}, x_{8*}) to nodes x_7, x_8, respectively.

This type of flow spreads toward all the leaves and finally,

$$\boldsymbol{x}_* = \arg\max_{\boldsymbol{x}} P(\boldsymbol{x}) \tag{15.69}$$

is obtained. One may wonder why not use a similar two-stage message passing as we did with the sum-product rule, and recover x_{i*}, for each node i. This would be possible if there were a guarantee for a unique optimum, $\boldsymbol{x}_*$. If this is not the case and we have two optimal values, say, $\boldsymbol{x}_*^1$ and $\boldsymbol{x}_*^2$, which result from Eq. (15.69), then we run the danger of failing to obtain them. To see this, let us take an example of four variables, x_1, x_2, x_3, x_4, each taking values in the discrete set $\{1, 2, 3, 4\}$. Assume that $P(\boldsymbol{x})$ does not have a unique maximum and the two combinations for optimality are

$$(x_{1*}, x_{2*}, x_{3*}, x_{4*}) = (1, 1, 2, 3), \tag{15.70}$$

and

$$(x_{1*}, x_{2*}, x_{3*}, x_{4*}) = (1, 2, 2, 4). \tag{15.71}$$

Both of them are acceptable because they correspond to max $P(x_1, x_1, x_3, x_4)$. The back-tracking procedure guarantees to give either of the two. In contrast, using two-stage message passing may result to a combination of values, for example,

$$(x_{1*}, x_{2*}, x_{3*}, x_{4*}) = (1, 1, 2, 4), \tag{15.72}$$

which does not correspond to the maximum. Note that this result is correct in its own rationale. It provides, for every node, a value for which an optimum may result. Indeed, searching for a maximum of $P(\boldsymbol{x})$, node x_2, can take either the value of 1 or 2. However, what we want to find is the correct *combination* for all nodes. This is guaranteed by the back-tracking procedure.

Remarks 15.3.

The max-product (max-sum) algorithm is a generalization of the celebrated Viterbi algorithm [46], which has extensively been used in communications [17] and speech recognition [39]. The algorithm has been generalized to arbitrary commutative semirings on tree-structured graphs (e.g., [1, 13]).

Example 15.4. Consider the Bayesian network of Figure 15.28a. The involved variables are binary, $(0, 1)$, and the respective probabilities are

$$P(\mathrm{x} = 1) = 0.7,\ P(\mathrm{x} = 0) = 0.3,$$
$$P(\mathrm{w} = 1) = 0.8,\ P(\mathrm{w} = 0) = 0.2,$$
$$P(\mathrm{y} = 1|\mathrm{x} = 0) = 0.8,\ P(\mathrm{y} = 0|\mathrm{x} = 0) = 0.2,$$
$$P(\mathrm{y} = 1|\mathrm{x} = 1) = 0.6,\ P(\mathrm{y} = 0|\mathrm{x} = 1) = 0.4,$$
$$P(\mathrm{z} = 1|\mathrm{y} = 0) = 0.7,\ P(\mathrm{z} = 0|\mathrm{y} = 0) = 0.3,$$

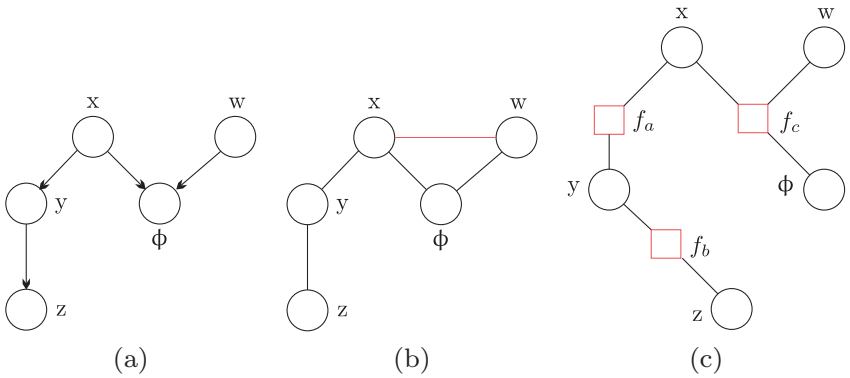

FIGURE 15.28

(a) The Bayesian network of Example 15.4; (b) its moralized version, where the two parents of ϕ have been connected; and (c) a possible factor graph.

$$
\begin{aligned}
P(\mathrm{z}=1|\mathrm{y}=1)=0.9,\ P(\mathrm{z}=0|\mathrm{y}=1)=0.1,\\
P(\phi=1|\mathrm{x}=0,\mathrm{w}=0)=0.25,\ P(\phi=0|\mathrm{x}=0,\mathrm{w}=0)=0.75,\\
P(\phi=1|\mathrm{x}=1,\mathrm{w}=0)=0.3,\ P(\phi=0|\mathrm{x}=1,\mathrm{w}=0)=0.7,\\
P(\phi=1|\mathrm{x}=0,\mathrm{w}=1)=0.2,\ P(\phi=0|\mathrm{x}=0,\mathrm{w}=1)=0.8,\\
P(\phi=1|\mathrm{x}=1,\mathrm{w}=1)=0.4,\ P(\phi=0|\mathrm{x}=1,\mathrm{w}=1)=0.6.
\end{aligned}
$$

Compute the combination, $x_*, y_*, z_*, \phi_*, w_*$, which results in the maximum of the joint probability,

$$
\begin{aligned}
P(x,y,z,\phi,w) &= P(z|y,x,\phi,w)P(y|x,\phi,w)P(\phi|x,w)P(x|w)P(w)\\
&= P(z|y)P(y|x)P(\phi|x,w)P(x)P(w),
\end{aligned}
$$

which is the factorization imposed by the Bayesian network.

In order to apply the max-product rule, we first moralize the graph and then form a factor graph version, as shown in Figures 15.28b, c, respectively. The factor nodes realize the following potential (factor) functions,

$$
\begin{aligned}
f_a(x,y) &= P(y|x)P(x),\\
f_b(y,z) &= P(z|y),\\
f_c(\phi,x,w) &= P(\phi|x,w)P(w),
\end{aligned}
$$

and obviously

$$
P(x,y,z,\phi,w) = f_a(x,y)f_b(y,z)f_c(\phi,x,w).
$$

Note that in this case, the normalizing constant, $Z=1$. Thus, the values these factor functions take, according to their previous definitions, are

$$f_a(x,y): \begin{cases} f_a(1,1)=0.42 \\ f_a(1,0)=0.28 \\ f_a(0,1)=0.24 \\ f_a(0,0)=0.06 \end{cases}, \quad f_b(y,z): \begin{cases} f_b(1,1)=0.9 \\ f_b(1,0)=0.1 \\ f_b(0,1)=0.7 \\ f_b(0,0)=0.3 \end{cases}$$

$$f_c(\phi,x,w): \begin{cases} f_c(1,1,1)=0.32 \\ f_c(1,1,0)=0.06 \\ f_c(1,0,1)=0.48 \\ f_c(1,0,0)=0.14 \\ f_c(0,1,1)=0.16 \\ f_c(0,1,0)=0.05 \\ f_c(0,0,1)=0.64 \\ f_c(0,0,0)=0.15 \end{cases}$$

Note that the number of possible values of a factor explodes by increasing the number of the involved variables.

Application of the max-product algorithm: Choose as root the node x. Then the nodes z, ϕ, and w become the leaves.

- Initialization:

$$\mu_{z\to f_b}(z)=1, \quad \mu_{\phi\to f_c}(\phi)=1, \; \mu_{w\to f_c}(w)=1.$$

- Begin message-passing:
  - $f_b \to$ y:

$$\mu_{f_b\to y}(y)=\max_z f_b(y,z)\mu_{z\to f_b}(z),$$

or

$$\mu_{f_b\to y}(1)=0.9, \quad \mu_{f_b\to y}(0)=0.7,$$

where the first one occurs at $z=1$ and the second one at $z=1$.

  - y $\to f_a$:

$$\mu_{y\to f_a}(y)=\mu_{f_b\to y}(y),$$

or

$$\mu_{y\to f_a}(1)=0.9, \quad \mu_{y\to f_a}(0)=0.7.$$

  - $f_a \to$ x :

$$\mu_{f_a\to x}(x)=\max_y f_a(x,y)\mu_{y\to f_a}(y),$$

or

$$\mu_{f_a\to x}(1)=0.42\cdot 0.9=0.378,$$

which occurs for $y = 1$. Note that for $y = 0$, the value for $\mu_{f_a \to x}(1)$ would be $0.7 \cdot 0.28 = 0.196$, which is smaller than 0.378. Also,

$$\mu_{f_a \to x}(0) = 0.24 \cdot 0.9 = 0.216,$$

which also occurs for $y = 1$.

- $f_c \to$ x :

$$\mu_{f_c \to x}(x) = \max_{w,\phi} f_c(\phi, x, w)\mu_{w \to f_c}(w)\mu_{\phi \to f_c}(\phi),$$

or

$$\mu_{f_c \to x}(1) = 0.48,$$

which occurs for $\phi = 0$ and $w = 1$, and

$$\mu_{f_c \to x}(0) = 0.64,$$

which occurs for $\phi = 0$ and $w = 1$.

- Obtain the optimal value:

$$x_* = \arg\max \mu_{f_a \to x}(x)\mu_{f_c \to x}(x),$$

or

$$x_* = 1,$$

and the corresponding maximum value is

$$\max P(x, y, z, w, \phi) = 0.378 \cdot 0.48 = 0.1814.$$

- Back-tracking:
  - Node f_c:

$$\max_{w,\phi} f_c(1, \phi, w)\mu_{w \to f_c}(w)\mu_{\phi \to f_c}(\phi),$$

which has occurred for

$$\phi_* = 0 \text{ and } w_* = 1.$$

  - Node f_a:

$$\max_{y} f_a(1, y)\mu_{y \to f_a}(y),$$

which has occurred for

$$y_* = 1.$$

  - Node f_b:

$$\max_{z} f_b(1, z)\mu_{z \to f_b}(z),$$

which has occurred for

$$z_* = 1.$$

Thus, the optimizing combination is

$$(x_*, y_*, z_*, \phi_*, w_*) = (1, 1, 1, 0, 1).$$

PROBLEMS

15.1 Show that in the product

$$\prod_{i=1}^{n}(1 - x_i),$$

the number of cross-product terms, $x_1, x_2, \ldots, x_k,\ 1 \leq k \leq n$, for all possible combinations of $x_1, \ldots, x_n$ is equal to $2^n - n - 1$.

15.2 Prove that if a probability distribution p satisfies the Markov condition, as implied by a BN, then p is given as the product of the conditional distributions given the values of the parents.

15.3 Show that if a probability distribution factorizes according to a Bayesian network structure, then it satisfies the Markov condition.

15.4 Consider a DAG and associate each node with a random variable. Define for each node the conditional probability of the respective variable given the values of its parents. Show that the product of the conditional probabilities yields a valid joint probability and that the Markov condition is satisfied.

15.5 Consider the graph in Figure 15.29. Random variable x has two possible outcomes, with probabilities $P(x_1) = 0.3$ and $P(x_2) = 0.7$. Variable y has three possible outcomes, with conditional probabilities

$$P(y_1|x_1) = 0.3,\ P(y_2|x_1) = 0.2,\ P(y_3|x_1) = 0.5,$$
$$P(y_1|x_2) = 0.1,\ P(y_2|x_2) = 0.4,\ P(y_3|x_2) = 0.5.$$

Finally, the conditional probabilities for z are

$$P(z_1|y_1) = 0.2,\ P(z_2|y_1) = 0.8,$$
$$P(z_1|y_2) = 0.2,\ P(z_2|y_2) = 0.8,$$
$$P(z_1|y_3) = 0.4,\ P(z_2|y_3) = 0.6.$$

Show that this probability distribution, which factorizes over the graph, renders x and z independent. However, x and z in the graph are not d-separated because y is not instantiated.

15.6 Consider the DAG in Figure 15.30. Detect the d-separations and d-connections in the graph.

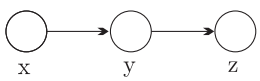

FIGURE 15.29

Graphical Model for Problem 15.5.

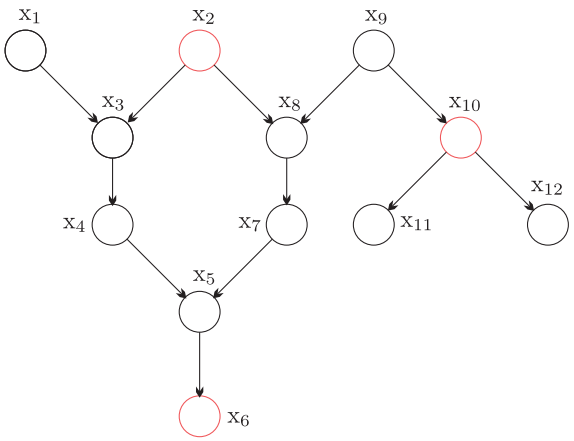

FIGURE 15.30

DAG for Problem 15.6. Nodes in red have been instantiated.

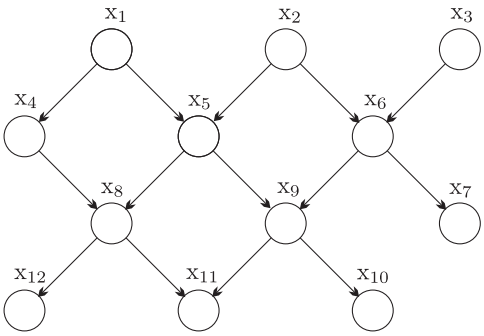

FIGURE 15.31

The graph structure for Problem 15.7.

15.7 Consider the DAG of Figure 15.31. Detect the blanket of node x_5 and verify that if all the nodes in the blanket are instantiated, then the node becomes d-separated from the rest of the nodes in the graph.

15.8 In a linear Gaussian Bayesian network model, derive the mean values and the respective covariance matrices for each one of the variables in a recursive manner.

15.9 Assuming the variables associated with the nodes of the Bayesian structure of Figure 15.32 to be Gaussian, find the respective mean values and covariances.

15.10 Prove that if p is a Gibbs distribution that factorizes over an MRF H, then H is an I-map for p.

15.11 Show that if H is the moral graph that results from moralization of a BN structure, then

$$I(H) \subseteq I(G).$$

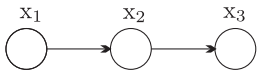

FIGURE 15.32

Network for Problem 15.9.

15.12 Consider a Bayesian network structure and a probability distribution p. Then show that if $I(G) \subseteq I(p)$, then p factorizes over G.

15.13 Show that in an undirected chain graphical model, the marginal probability $P(x_j)$ of a node, x_j, is given by

$$P(x_j) = \frac{1}{Z}\mu_f(x_j)\mu_b(x_j),$$

where $\mu_f(x_j)$ and $\mu_b(x_j)$ are the received by the node forward and backward messages.

15.14 Show that the joint distribution of two neighboring nodes in an undirected chain graphical model is given by

$$P(x_j, x_{j+1}) = \frac{1}{Z}\mu_f(x_j)\psi_{j,j+1}(x_j, x_{j+1})\mu_b(x_{j+1}).$$

15.15 Using Figure 15.26, prove that if there is a second message passing, starting from x, toward the leaves, then any node will have the available information for the computation of the respective marginals.

15.16 Consider the tree graph of Figure 15.26. Compute the marginal probability $P(x_1, x_2, x_3, x_4)$.

15.17 Repeat the message-passing procedure to find the optimal combination of variables for Example 15.4 using the logarithmic version and the max-sum algorithm.

REFERENCES

[1] S.M. Aji, R.J. McEliece, The generalized distributive law, IEEE Trans. Inform. Theory 46 (2000) 325-343.
[2] A. Al-Bashabsheh, Y. Mao, Normal factor graphs and holographic transformations, IEEE Trans. Inform. Theory 57 (February (2)) (2011) 752-763.
[3] A. Al-Bashabsheh, Y. Mao, Normal Factor Graphs as Probabilistic Models, 2012, arXiv:1209.3300v1 [cs.IT] 14 September 2012.
[4] U. Bertele, F. Brioschi, Nonserial Dynamic Programming, Academic Press, Boston, 1972.
[5] J. Besag, Spatial interaction and the statistical analysis of lattice systems, J. R. Stat. Soc. B 36 (2) (1974) 192-236.
[6] C.E. Cannings, A. Thompson, M.H. Skolnick, The recursive derivation of likelihoods on complex pedigrees, Adv. Appl. Probab. 8 (4) (1976) 622-625.
[7] R. Chellappa, R.L. Kashyap, Digital image restoration using spatial interaction models, IEEE Trans. Acoust. Speech Signal Process. 30 (1982) 461-472.
[8] R. Chellappa, S. Chatterjee, Classification of textures using Gaussian Markov random field models, IEEE Trans. Acoust. Speech Signal Process. 33 (1985) 959-963.
[9] R. Chellappa, A.K. Jain (Eds.), Markov Random Fields: Theory and Applications, Academic Press, Boston, 1993.

[10] G.F. Cooper, The computational complexity of probabilistic inference using Bayesian belief networks, Artif. Intell. 42 (1990) 393-405.
[11] P. Dagum, M. Luby, Approximating probabilistic inference in Bayesian belief networks is NP-hard, Artif. Intell. 60 (1993) 141-153.
[12] A.P. Dawid, Conditional independence in statistical theory, J. R. Stat. Soc. B 41 (1978) 1-31.
[13] A.P. Dawid, Applications of a general propagation algorithm for probabilistic expert systems, Stat. Comput. 2 (1992) 25-36.
[14] A.P. Dawid, M. Studeny, Conditional products: an alternative approach to conditional independence, in: D. Heckerman, J. Whittaker (Eds.), Artificial Intelligence and Statistics, Morgan-Kaufmann, San Mateo, 1999.
[15] B.J. Frey, Graphical Models for Machine Learning and Digital Communications, MIT Press, Cambridge, MA, 1998.
[16] B.J. Frey, F.R. Kschishany, H.A. Loeliger, N. Wiberg, Factor graphs and algorithms, Proceedings of the 35th Alerton Conference on Communication, Control and Computing, 1999.
[17] G.D. Forney Jr., The Viterbi algorithm, Proc. IEEE 61 (1973) 268-277.
[18] G.D. Forney Jr., Codes on graphs: normal realizations, IEEE Trans. Inform. Theory 47 (2001) 520-548.
[19] G.D. Forney Jr., Codes on graphs: duality and MacWilliams identities, IEEE Trans. Inform. Theory 57 (3) (2011) 1382-1397.
[20] D. Geiger, T. Verma, J. Pearl, d-Separation: From theorems to algorithms, in: M. Henrion, R.D. Shachter, L.N. Kanal, J.F. Lemmer (Eds.), Proceedings 5th Annual Conference on Uncertainty in Artificial Intelligence, 1990.
[21] S. Geman, D. Geman, Stochastic relaxation, Gibbs distributions and the Bayesian restoration of images, IEEE Trans. Pattern Anal. Mach. Intell. 6 (1) (1984) 721-741.
[22] J.M. Hammersley, P. Clifford, Markov fields on finite graphs and lattices, Unpublished manuscript available the web, 1971.
[23] G.E. Hinton, T. Sejnowski, Learning and relearning in Boltzmann machines, in: D.E. Rumelhart, J.L. McClelland (Eds.), Parallel Distributed Processing, vol. 1, MIT Press, Cambridge, MA, 1986.
[24] D. Janzing, B. Schölkopf, Causal inference using the algorithmic Markov condition, IEEE Trans. Inform. Theory 56 (2010) 5168-5194.
[25] D. Koller, N. Friedman, Probabilistic Graphical Models: Principles and Techniques, MIT Press, Cambridge, MA, 2009.
[26] F.R. Kschischang, B.J. Frey, H.A. Loeliger, Factor graphs and the sum-product algorithm, IEEE Trans. Inform. Theory 47(2) (2001) 498-519.
[27] J. Lafferty, A. McCallum, F. Pereira, Conditional random fields: probabilistic models for segmenting and labeling sequence data, in: International Conference on Machine Learning, 2001, pp. 282-289.
[28] 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.
[29] S.Z. Li, Markov Random Field Modeling in Image Analysis, Springer-Verlag, New York, 2009.
[30] H.A. Loeliger, J. Dauwels, J. Hu, S. Korl, L. Ping, F.R. Kschischang, The factor graph approach to model-based signal processing, Proc. IEEE, 95 (6) (2007) 1295-1322.
[31] D.J.C. MacKay, Information Inference and Learning Algorithms, Cambridge University Press, Cambridge, 2003.
[32] R.E. Neapolitan, Learning Bayesian Networks, Prentice Hall, Upper Saddle River, NJ, 2004.
[33] R.M. Neal, Connectionist learning of belief networks, Artif. Intell. 56 (1992) 71-113.
[34] F.A.N. Palmieri, Learning nonlinear functions with factor graphs, IEEE Trans. Signal Process. 61 (12) (2013) 4360-4371.
[35] F.A.N. Palmieri, A Comparison of algorithms for learning hidden variables in normal graphs, 2013, arXiv: 1308.5576v1 [stat.ML] 26 August 2013.

[36] J. Pearl, Fusion, propagation, and structuring in belief networks, Artif. Intell. 29 (1986) 241-288.
[37] J. Pearl, Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference, Morgan-Kaufmann, San Mateo, 1988.
[38] J. Pearl, Causality, Reasoning and Inference, second ed., Cambridge University Press, Cambridge, 2012.
[39] L. Rabiner, A tutorial on hidden Markov models and selected applications in speech processing, Proc. IEEE 77 (1989) 257-286.
[40] U. Schmidt, Learning and Evaluating Markov Random Fields for Natural Images, Master's Thesis, Department of Computer Science, Technishe Universitat Darmstadt, Germany, 2010.
[41] M.A. Shwe, G.F. Cooper, An empirical analysis of likelihood-weighting simulation on a large, multiply connected medical belief network, Comput. Biomed. Res. 24 (1991) 453-475.
[42] P. Spirtes, Introduction to causal inference, J. Mach. Learn. Res. 11 (2010) 1643-1662.
[43] X. Sun, D. Janzing, B. Schölkopf, Causal inference by choosing graphs with most plausible Markov kernels, in: Proceedings, 9th International Symposium on Artificial Intelligence and Mathematics, Fort Lauderdale, 2006, pp. 1-11.
[44] C. Sutton, A. McCallum, An introduction to conditional random fields, 2010, arXiv:1011.4088v1 [stat.ML] 17 November 2010.
[45] T. Verma, J. Pearl, Causal networks: semantics and expressiveness, in: R.D. Schachter, T.S. Levitt, L.N. Kanal, J.F. Lemmer (Eds.), Proceedings of the 4th Conference on Uncertainty in Artificial Intelligence, North-Holland, 1990.
[46] A.J. Viterbi, Error bounds for convolutional codes and an asymptotically optimum decoding algorithm, IEEE Trans. Inform. Theory IT-13 (1967) 260-269.
[47] J.W. Woods, Two-dimensional discrete Markovian fields, IEEE Trans. Inform. Theory, 18 (2) (1972) 232-240.