14 *Graphical Models*

Graphical models represent the interaction between variables visually and have the advantage that inference over a large number of variables can be decomposed into a set of local calculations involving a small number of variables making use of conditional independencies. After some examples of inference by hand, we discuss the concept of d-separation and the belief propagation algorithm on a variety of graphs.

14.1 Introduction

GRAPHICAL MODELS
BAYESIAN NETWORKS
BELIEF NETWORKS
PROBABILISTIC NETWORKS

Graphical models, also called *Bayesian networks*, *belief networks*, or *probabilistic networks*, are composed of nodes and arcs between the nodes. Each node corresponds to a random variable, X, and has a value corresponding to the probability of the random variable, $P(X)$. If there is a directed arc from node X to node Y, this indicates that X has a *direct influence* on Y. This influence is specified by the conditional probability $P(Y|X)$. The network is a *directed acyclic graph* (DAG); namely, there are no cycles. The nodes and the arcs between the nodes define the *structure* of the network, and the conditional probabilities are the *parameters* given the structure.

DIRECTED ACYCLIC GRAPH

A simple example is given in figure 14.1, which models that rain causes the grass to get wet. It rains on 40 percent of the days and when it rains, there is a 90 percent chance that the grass gets wet; maybe 10 percent of the time it does not rain long enough for us to really consider the grass wet enough. The random variables in this example are binary; they are either true or false. There is a 20 percent probability that the grass gets wet without its actually raining, for example, when a sprinkler is used.

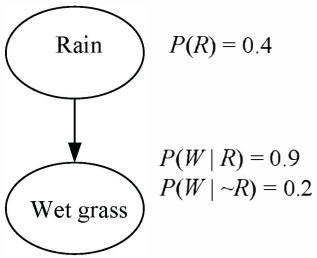

Figure 14.1 Bayesian network modeling that rain is the cause of wet grass.

We see that these three values completely specify the joint distribution of $P(R, W)$. If $P(R) = 0.4$, then $P(\sim R) = 0.6$, and similarly $P(\sim W|R) = 0.1$ and $P(\sim W|\sim R) = 0.8$. The joint is written as

$$P(R, W) = P(R)P(W|R)$$

We can calculate the individual (marginal) probability of wet grass by summing up over the possible values that its parent node can take:

$$\begin{aligned} P(W) &= \sum_R P(R, W) = P(W|R)P(R) + P(W|\sim R)P(\sim R) \\ &= 0.9 \cdot 0.4 + 0.2 \cdot 0.6 = 0.48 \end{aligned}$$

If we knew that it rained, the probability of wet grass would be 0.9; if we knew for sure that it did not, it would be as low as 0.2; not knowing whether it rained or not, the probability is 0.48.

CAUSAL GRAPH

Figure 14.1 shows a *causal graph* in that it explains that the cause of wet grass is rain. Bayes' rule allows us to invert the dependencies and have a *diagnosis*. For example, knowing that the grass is wet, the probability that it rained can be calculated as follows:

$$P(R|W) = \frac{P(W|R)P(R)}{P(W)} = 0.75$$

Knowing that the grass is wet increased the probability of rain from 0.4 to 0.75; this is because $P(W|R)$ is high and $P(W|\sim R)$ is low.

We form graphs by adding nodes and arcs and generate dependencies.

INDEPENDENCE

X and Y are *independent events* if

$$p(X, Y) = P(X)P(Y) \tag{14.1}$$

CONDITIONAL INDEPENDENCE

X and Y are *conditionally independent events* given a third event Z if

$$P(X, Y|Z) = P(X|Z)P(Y|Z) \tag{14.2}$$

which can also be rewritten as

$$P(X|Y, Z) = P(X|Z) \tag{14.3}$$

In a graphical model, not all nodes are connected; actually, in general, a node is connected to only a small number of other nodes. Certain subgraphs imply conditional independence statements, and these allow us to break down a complex graph into smaller subsets in which inferences can be done locally and whose results are later propagated over the graph. There are three canonical cases and larger graphs are constructed using these as subgraphs.

14.2 Canonical Cases for Conditional Independence

Case 1: Head-to-Tail Connection

Three events may be connected serially, as seen in figure 14.2a. We see here that X and Z are independent given Y: Knowing Y tells Z everything; knowing the state of X does not add any extra knowledge for Z; we write $P(Z|Y, X) = P(Z|Y)$. We say that Y *blocks* the path from X to Z, or in other words, it *separates* them in the sense that if Y is removed, there is no path between X to Z. In this case, the joint is written as

$$P(X, Y, Z) = P(X)P(Y|X)P(Z|Y) \tag{14.4}$$

Writing the joint this way implies independence:

$$P(Z|X, Y) = \frac{P(X, Y, Z)}{P(X, Y)} = \frac{P(X)P(Y|X)P(Z|Y)}{P(X)P(Y|X)} = P(Z|Y) \tag{14.5}$$

Typically, X is the cause of Y and Y is the cause of Z. For example, as seen in figure 14.2b, X can be cloudy sky, Y can be rain, and Z can be wet grass. We can propagate information along the chain. If we do not know the state of cloudy, we have

$$\begin{aligned} P(R) &= P(R|C)P(C) + P(R|\sim C)P(\sim C) = 0.38 \\ P(W) &= P(W|R)P(R) + P(W|\sim R)P(\sim R) = 0.48 \end{aligned}$$

Let us say in the morning we see that the weather is cloudy; what can we say about the probability that the grass will be wet? To do this, we

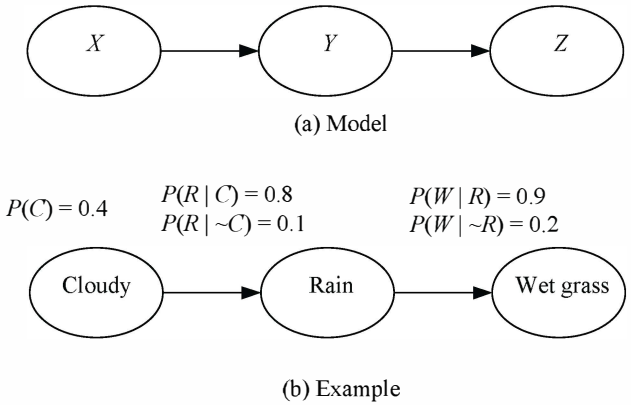

Figure 14.2 Head-to-tail connection. (a) Three nodes are connected serially. X and Z are independent given the intermediate node Y: $P(Z|Y,X) = P(Z|Y)$. (b) Example: Cloudy weather causes rain, which in turn causes wet grass.

need to propagate evidence first to the intermediate node R, and then to the query node W.

$$P(W|C) = P(W|R)P(R|C) + P(W|\sim R)P(\sim R|C) = 0.76$$

Knowing that the weather is cloudy increased the probability of wet grass. We can also propagate evidence back using Bayes' rule. Let us say that we were traveling and on our return, see that our grass is wet; what is the probability that the weather was cloudy that day? We use Bayes' rule to invert the direction:

$$P(C|W) = \frac{P(W|C)P(C)}{P(W)} = 0.65$$

Knowing that the grass is wet increased the probability of cloudy weather from its default (prior) value of 0.4 to 0.65.

Case 2: Tail-to-Tail Connection

X may be the parent of two nodes Y and Z, as shown in figure 14.3a. The joint density is written as

$$P(X,Y,Z) = P(X)P(Y|X)P(Z|X) \tag{14.6}$$

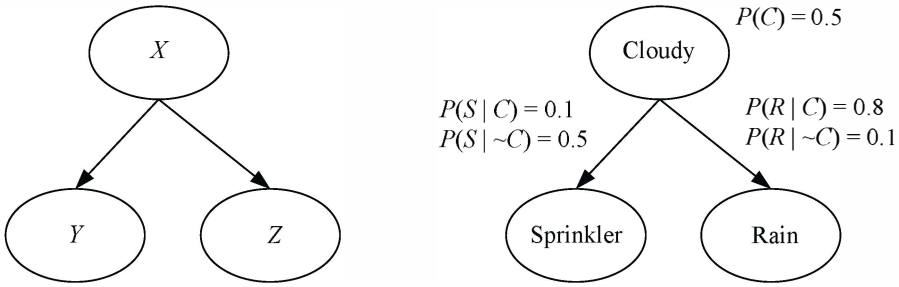

Figure 14.3 Tail-to-tail connection. X is the parent of two nodes Y and Z. The two child nodes are independent given the parent: $P(Y|X,Z) = P(Y|X)$. In the example, cloudy weather causes rain and also makes us less likely to turn the sprinkler on.

Normally Y and Z are dependent through X; given X, they become independent:

$$P(Y,Z|X) = \frac{P(X,Y,Z)}{P(X)} = \frac{P(X)P(Y|X)P(Z|X)}{P(X)} = P(Y|X)P(Z|X) \tag{14.7}$$

When its value is known, X blocks the path between Y and Z or, in other words, separates them.

In figure 14.3b, we see an example where cloudy weather influences both rain and the use of the sprinkler, one positively and the other negatively. Knowing that it rained, for example, we can invert the dependency using Bayes' rule and infer the cause:

$$\begin{aligned} P(C|R) &= \frac{P(R|C)P(C)}{P(R)} = \frac{P(R|C)P(C)}{\sum_C P(R,C)} \\ &= \frac{P(R|C)P(C)}{P(R|C)P(C) + P(R|\sim C)P(\sim C)} = 0.89 \end{aligned} \tag{14.8}$$

Note that this value is larger than $P(C)$; knowing that it rained increased the probability that the weather is cloudy.

In figure 14.3a, if X is not known, knowing Y, for example, we can infer X that we can then use to infer Z. In figure 14.3b, knowing the state of the sprinkler has an effect on the probability that it rained. If we know that the sprinkler is on,

$$P(R|S) = \sum_C P(R,C|S) = P(R|C)P(C|S) + P(R|\sim C)P(\sim C|S) \tag{14.9}$$

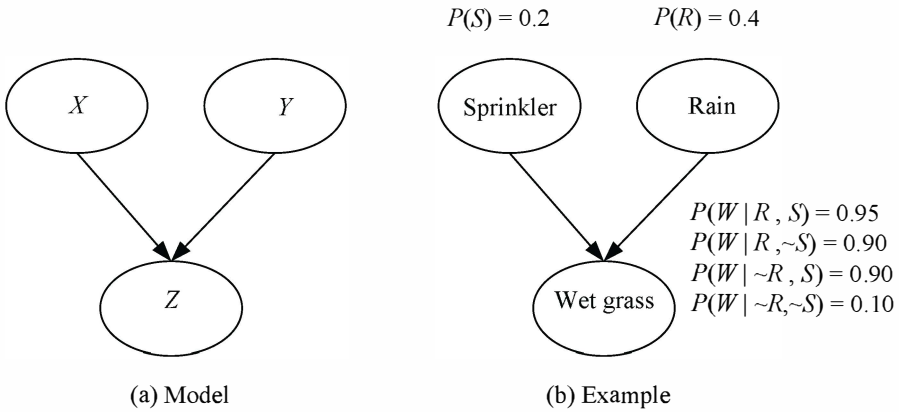

Figure 14.4 Head-to-head connection. A node has two parents that are independent unless the child is given. For example, an event may have two independent causes.

$$\begin{aligned} &= P(R|C)\frac{P(S|C)P(C)}{P(S)} + P(R|\sim C)\frac{P(S|\sim C)P(\sim C)}{P(S)} \\ &= 0.22 \end{aligned}$$

This is less than $P(R) = 0.45$; that is, knowing that the sprinkler is on decreases the probability that it rained because sprinkler and rain happens for different states of cloudy weather. If the sprinkler is known to be off, using the same approach, we find that $P(R|\sim S) = 0.55$; the probability of rain increases this time.

Case 3: Head-to-Head Connection

In a head-to-head node, there are two parents X and Y to a single node Z, as shown in figure 14.4a. The joint density is written as

$$P(X, Y, Z) = P(X)P(Y)P(Z|X, Y) \tag{14.10}$$

X and Y are independent: $P(X, Y) = P(X) \cdot P(Y)$ (exercise 2); they become dependent when Z is known. The concept of blocking or separation is different for this case: The path between X and Y is blocked, or they are separated, when Z is *not* observed; when Z (or any of its descendants) is observed, they are not blocked, separated, or independent.

We see, for example, in figure 14.4b that node W has two parents, R and S, and thus its probability is conditioned on the values of those two, $P(W|R,S)$.

Not knowing anything else, the probability that grass is wet is calculated by marginalizing over the joint:

$$\begin{aligned} P(W) &= \sum_{R,S} P(W,R,S) \\ &= P(W|R,S)P(R,S) + P(W|{\sim}R,S)P({\sim}R,S) \\ &\quad + P(W|R,{\sim}S)P(R,{\sim}S) + P(W|{\sim}R,{\sim}S)P({\sim}R,{\sim}S) \\ &= P(W|R,S)P(R)P(S) + P(W|{\sim}R,S)P({\sim}R)P(S) \\ &\quad + P(W|R,{\sim}S)P(R)P({\sim}S) + P(W|{\sim}R,{\sim}S)P({\sim}R)P({\sim}S) \\ &= 0.52 \end{aligned}$$

Now, let us say that we know that the sprinkler is on, and we check how this affects the probability. This is a causal (predictive) inference:

$$\begin{aligned} P(W|S) &= \sum_{R} P(W,R|S) \\ &= P(W|R,S)P(R|S) + P(W|{\sim}R,S)P({\sim}R|S) \\ &= P(W|R,S)P(R) + P(W|{\sim}R,S)P({\sim}R) \\ &= 0.92 \end{aligned}$$

We see that $P(W|S) > P(W)$; knowing that the sprinkler is on, the probability of wet grass increases.

We can also calculate the probability that the sprinkler is on, given that the grass is wet. This is a diagnostic inference.

$$P(S|W) = \frac{P(W|S)P(S)}{P(W)} = 0.35$$

$P(S|W) > P(S)$, that is, knowing that the grass is wet increased the probability of having the sprinkler on. Now let us assume that it rained. Then we have

$$\begin{aligned} P(S|R,W) &= \frac{P(W|R,S)P(S|R)}{P(W|R)} = \frac{P(W|R,S)P(S)}{P(W|R)} \\ &= 0.21 \end{aligned}$$

EXPLAINING AWAY

which is less than $P(S|W)$. This is called *explaining away*; given that we know it rained, the probability of the sprinkler causing the wet grass

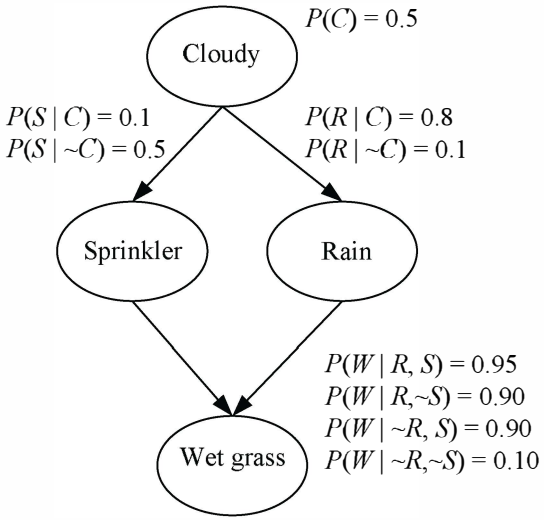

Figure 14.5 Larger graphs are formed by combining simpler subgraphs over which information is propagated using the implied conditional independencies.

decreases. Knowing that the grass is wet, rain and sprinkler become dependent. Similarly, $P(S|{\sim}R, W) > P(S|W)$. We see the same behavior when we compare $P(R|W)$ and $P(R|W, S)$ (exercise 3).

We can construct larger graphs by combining such subgraphs. For example, in figure 14.5 where we combine the two subgraphs, we can, for example, calculate the probability of having wet grass if it is cloudy:

$$\begin{aligned}
P(W|C) &= \sum_{R,S} P(W, R, S|C) \\
&= P(W, R, S|C) + P(W, {\sim}R, S|C) \\
&\quad + P(W, R, {\sim}S|C) + P(W, {\sim}R, {\sim}S|C) \\
&= P(W|R, S, C)P(R, S|C) \\
&\quad + P(W|{\sim}R, S, C)P({\sim}R, S|C) \\
&\quad + P(W|R, {\sim}S, C)P(R, {\sim}S|C) \\
&\quad + P(W|{\sim}R, {\sim}S, C)P({\sim}R, {\sim}S|C) \\
&= P(W|R, S)P(R|C)P(S|C) \\
&\quad + P(W|{\sim}R, S)P({\sim}R|C)P(S|C) \\
&\quad + P(W|R, {\sim}S)P(R|C)P({\sim}S|C)
\end{aligned}$$

$$+P(W|\sim R,\sim S)P(\sim R|C)P(\sim S|C)$$

where we have used that $P(W|R,S,C) = P(W|R,S)$; given R and S, W is independent of C: R and S between them block the path between W and C. Similarly, $P(R,S|C) = P(R|C)P(S|C)$; given C, R and S are independent. We see the advantage of Bayesian networks here, which explicitly encode independencies and allow breaking down inference into calculation over small groups of variables that are propagated from evidence nodes to query nodes.

We can calculate $P(C|W)$ and have a diagnostic inference:

$$P(C|W) = \frac{P(W|C)P(C)}{P(W)}$$

The graphical representation is visual and helps understanding. The network represents conditional independence statements and allows us to break down the problem of representing the joint distribution of many variables into *local* structures; this eases both analysis and computation. Figure 14.5 represents a joint density of four binary variables that would normally require fifteen values ($2^4 - 1$) to be stored, whereas here there are only nine. If each node has a small number of parents, the complexity decreases from exponential to linear (in the number of nodes). As we have seen earlier, inference is also easier as the joint density is broken down into conditional densities of smaller groups of variables:

$$P(C,S,R,W) = P(C)P(S|C)P(R|C)P(W|S,R) \tag{14.11}$$

In the general case, when we have variables $X_1,\ldots,X_d$, we write

$$P(X_1,\ldots,X_d) = \prod_{i=1}^{d} P(X_i|\text{parents}(X_i)) \tag{14.12}$$

Then given any subset of X_i, namely, setting them to certain values due to evidence, we can calculate the probability distribution of some other subset of X_i by marginalizing over the joint. This is costly because it requires calculating an exponential number of joint probability combinations, even though each of them can be simplified as in equation 14.11. Note, however, that given the same evidence, for different X_i, we may be using the same intermediate values (products of conditional probabilities and sums for marginalization), and in section 14.5, we will discuss the belief propagation algorithm to do inference cheaply by doing the local intermediate calculations once which we can use multiple times for different query nodes.

Though in this example we use binary variables, it is straightforward to generalize for cases where the variables are discrete with any number of possible values (with m possible values and k parents, a table of size m^k is needed for the conditional probabilities), or they can be continuous (parameterized, e.g., $p(Y|x) \sim \mathcal{N}(\mu(x|\theta), \sigma^2)$; see figure 14.7).

One major advantage to using a Bayesian network is that we do not need to designate explicitly certain variables as input and certain others as output. The value of any set of variables can be established through evidence and the probabilities of any other set of variables can be inferred, and the difference between unsupervised and supervised learning becomes blurry. From this perspective, a graphical model can be thought of as a "probabilistic database" (Jordan 2004), a machine that can answer queries regarding the values of random variables.

HIDDEN VARIABLES

In a problem, there may also be *hidden variables* whose values are never known through evidence. The advantage of using hidden variables is that the dependency structure can be more easily defined. For example, in basket analysis when we want to find the dependencies among items sold, let us say we know that there is a dependency among "baby food," "diapers," and "milk" in that a customer buying one of these is very much likely to buy the other two. Instead of putting (noncausal) arcs among these three, we may designate a hidden node "baby at home" as the hidden cause of the consumption of these three items. When there are hidden nodes, their values are estimated given the values of observed nodes and filled in.

CAUSALITY

It should be stressed at this point that a link from a node X does not, and need not, always imply a *causality*. It only implies a *direct influence* of X over Y in the sense that the probability of Y is conditioned on the value of X, and two nodes may have a link between them even if there is no direct cause. It is preferable to have the causal relations in constructing the network by providing an explanation of how the data is generated (Pearl 2000) but such causes may not always be accessible.

14.3 Generative Models

GENERATIVE MODEL

Still, graphical models are frequently used to visualize *generative models* for representing the process that we believe has created the data. For example, for the case of classification, the corresponding graphical model is shown in figure 14.6a, with $\boldsymbol{x}$ as the input and C a multinomial variable

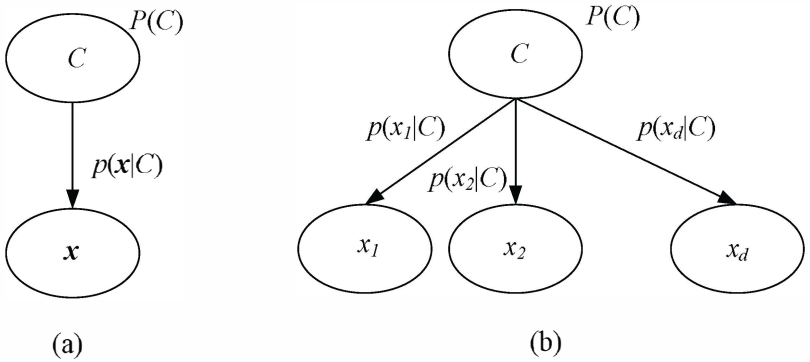

Figure 14.6 (a) Graphical model for classification. (b) Naive Bayes' classifier assumes independent inputs.

taking one of K states for the class code. It is as if we first pick a class C at random by sampling from $P(C)$, and then having fixed C, we pick an $\boldsymbol{x}$ by sampling from $p(\boldsymbol{x}|C)$. Bayes' rule inverts the generative direction and allows a diagnosis, as in the rain and wet grass case we saw in figure 14.1:

$$P(C|\boldsymbol{x}) = \frac{P(C)p(\boldsymbol{x}|C)}{P(\boldsymbol{x})}$$

Note that clustering is similar except that instead of the multinomial class indicator variable C we have the cluster indicator variable Z, and it is not observed during training. The E-step of the expectation-maximization algorithm (section 7.4) uses Bayes' rule to invert the arc and fills in the cluster indicator given the input.

NAIVE BAYES' CLASSIFIER

If the inputs are independent, we have the graph shown in figure 14.6b, which is called the *naive Bayes' classifier*, because it ignores possible dependencies, namely, correlations, among the inputs and reduces a multivariate problem to a group of univariate problems:

$$p(\boldsymbol{x}|C) = \prod_{j=1}^{d} p(x_j|C)$$

We have discussed classification for this case in sections 5.5 and 5.7 for numeric and discrete $\boldsymbol{x}$, respectively.

Linear regression can be visualized as a graphical model, as shown in figure 14.7. Input $\boldsymbol{x}^t$ is drawn from a prior $p(\boldsymbol{x})$, and the dependent variable r^t depends on the input $\boldsymbol{x}$ and the weights $\boldsymbol{w}$. Here, we define a

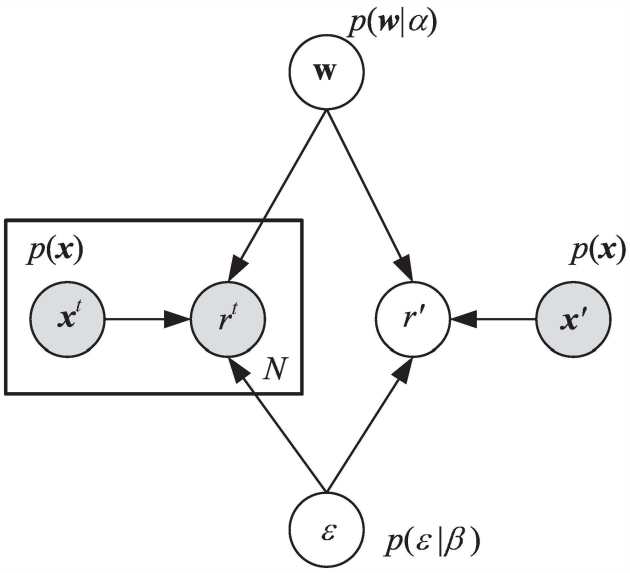

Figure 14.7 Graphical model for linear regression.

node for the weights $\boldsymbol{w}$ with a prior parameterized by α, namely, $p(\boldsymbol{w}) \sim \mathcal{N}(\mathbf{0}, \alpha^{-1}\mathbf{I})$. There is also a node for the noise ϵ variable, parameterized by β, namely, $p(\epsilon) \sim \mathcal{N}(0, \beta^{-1})$:

$$p(r^t|\boldsymbol{x}^t, \boldsymbol{w}) \sim \mathcal{N}(\boldsymbol{w}^T\boldsymbol{x}^t, \beta^{-1}) \tag{14.13}$$

There are N such pairs in the training set, which is shown by the rectangular *plate* in the figure—the plate corresponds to the training set $\mathcal{X}$. Given a new input $\boldsymbol{x}'$, the aim is to estimate r'. The weights $\boldsymbol{w}$ are not given but they can be estimated using the training set of $\mathcal{X}$ which we can divide as $[\mathbf{X}, \boldsymbol{r}]$.

In equation 14.9, where C is the cause of R and S, we write

$$P(R|S) = \sum_C P(R, C|S) = P(R|C)P(C|S) + P(R|{\sim}C)P({\sim}C|S)$$

filling in C using observed S and average over all possible values of C. Similarly here, we write

$$\begin{aligned} p(r'|\boldsymbol{x}', \boldsymbol{r}, \mathbf{X}) &= \int p(r'|\boldsymbol{x}', \boldsymbol{w})p(\boldsymbol{w}|\mathbf{X}, \boldsymbol{r})d\boldsymbol{w} \\ &= \int p(r'|\boldsymbol{x}', \boldsymbol{w})\frac{p(\boldsymbol{r}|\mathbf{X}, \boldsymbol{w})p(\boldsymbol{w})}{p(\boldsymbol{r})}d\boldsymbol{w} \end{aligned}$$

$$\propto \int p(r'|\mathbf{x}', \mathbf{w}) \prod_t p(r^t|\mathbf{x}^t, \mathbf{w}) p(\mathbf{w}) d\mathbf{w} \tag{14.14}$$

where the second line is due to Bayes' rule and the third line is due to the independence of instances in the training set.

Note that what we have in figure 14.7 is a Bayesian model where we designate parameter $\mathbf{w}$ as a random variable with a prior distribution. As we see in equation 14.14, what we are effectively doing is estimating the posterior $p(\mathbf{w}|\mathbf{X}, \mathbf{r})$ and then integrating over it. We began discussing this in section 4.4, and we discuss it in greater detail in chapter 16, for different generative models and different sets of parameters.

14.4 d-Separation

D-SEPARATION

We now generalize the concept of blocking and separation under the name of *d-separation*, and we define it in a way so that for arbitrary subsets of nodes A, B, and C, we can check if A and B are independent given C. Jordan (2004) visualizes this as a ball bouncing over the graph and calls this the *Bayes' ball*. We set the nodes in C to their values, place a ball at each node in A, let the balls move around according to a set of rules, and check whether a ball reaches any node in B. If this is the case, they are dependent; otherwise, they are independent.

BAYES' BALL

To check whether A and B are d-separated given C, we consider all possible paths between any node in A and any node in B. Any such path is *blocked* if

(a) the directions of the edges on the path either meet head-to-tail (case 1) or tail-to-tail (case 2) and the node is in C, or

(b) the directions of the edges on the path meet head-to-head (case 3) and neither that node nor any of its descendant is in C.

If all paths are blocked, we say that A and B are d-separated, that is, independent, given C; otherwise, they are dependent. Examples are given in figure 14.8.

14.5 Belief Propagation

Having discussed some inference examples by hand, we now are interested in an algorithm that can answer queries such as $P(X|E)$ where X

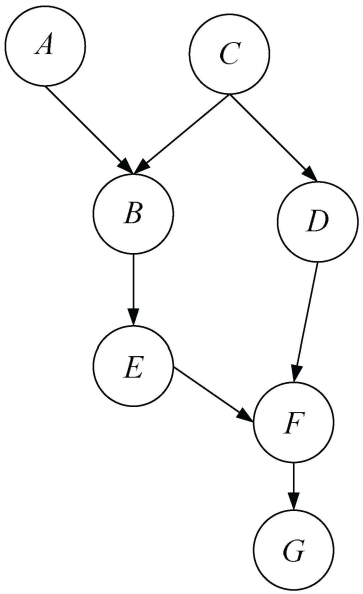

Figure 14.8 Examples of d-separation. The path $BCDF$ is blocked given C because C is a tail-to-tail node. $BEFG$ is blocked by F because F is a head-to-tail node. $BEFD$ is blocked unless F (or G) is given.

is any *query node* in the graph and E is any subset of *evidence nodes* whose values are set to certain value. Following Pearl (1988), we start with the simplest case of chains and gradually move on to more complex graphs. Our aim is to find the graph operation counterparts of probabilistic procedures such as Bayes' rule or marginalization, so that the task of inference can be mapped to general-purpose graph algorithms.

14.5.1 Chains

A *chain* is a sequence of head-to-tail nodes with one *root* node without any parent; all other nodes have exactly one parent node, and all nodes except the very last, *leaf*, have a single child. If evidence is in the ancestors of X, we can just do a diagnostic inference and propagate evidence down the chain; if evidence is in the descendants of X, we can do a causal inference and propagate upward using Bayes' rule. Let us see the general case where we have evidence in both directions, up the chain E^+ and

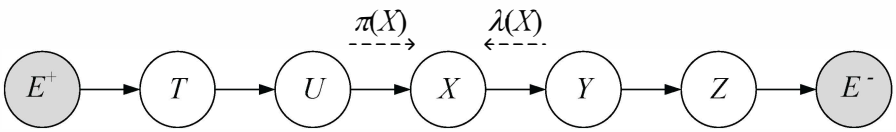

Figure 14.9 Inference along a chain.

down the chain E^- (see figure 14.9). Note that any evidence node separates X from the nodes on the chain on the other side of the evidence and their values do not affect $p(X)$; this is true in both directions.

We consider each node as a processor that receives messages from its neighbors and pass it along after some local calculation. Each node X locally calculates and stores two values: $\lambda(X) \equiv P(E^-|X)$ is the propagated E^- that X receives from its child and forwards to its parent, and $\pi(X) \equiv P(X|E^+)$ is the propagated E^+ that X receives from its parent and passes on to its child.

$$\begin{aligned} P(X|E) &= \frac{P(E|X)P(X)}{P(E)} = \frac{P(E^+,E^-|X)P(X)}{P(E)} \\ &= \frac{P(E^+|X)P(E^-|X)P(X)}{P(E)} \\ &= \frac{P(X|E^+)P(E^+)P(E^-|X)P(X)}{P(X)P(E)} \\ &= \alpha P(X|E^+)P(E^-|X) = \alpha\pi(X)\lambda(X) \end{aligned} \tag{14.15}$$

for some normalizing constant α, not dependent on the value of X. The second line is there because E^+ and E^- are independent given X, and the third line is due to Bayes' rule.

If a node E is instantiated to a certain value $\tilde{e}$, $\lambda(\tilde{e}) \equiv 1$ and $\lambda(e) \equiv 0$, for $e \neq \tilde{e}$. The leaf node X that is not instantiated has its $\lambda(x) \equiv 1$, for all x values. The root node X that is not instantiated takes the prior probabilities as π values: $\pi(x) \equiv P(x), \forall x$.

Given these initial conditions, we can devise recursive formulas to propagate evidence along the chain.

For the π-messages, we have

$$\begin{aligned} \pi(X) &\equiv P(X|E^+) = \sum_U P(X|U,E^+)P(U|E^+) \\ &= \sum_U P(X|U)P(U|E^+) = \sum_U P(X|U)\pi(U) \end{aligned} \tag{14.16}$$

where the second line follows from the fact that U blocks the path between X and E^+.

For the λ-messages, we have

$$\begin{aligned}\lambda(X) &\equiv P(E^-|X) = \sum_Y P(E^-|X,Y)P(Y|X) \\ &= \sum_Y P(E^-|Y)P(Y|X) = \sum_Y P(Y|X)\lambda(Y)\end{aligned} \tag{14.17}$$

where the second line follows from the fact that Y blocks the path between X and E^-.

When the evidence nodes are set to a value, they initiate traffic and nodes continue updating until there is convergence. Pearl (1988) views this as a parallel machine where each node is implemented by a processor that works in parallel with others and exchanges information through λ- and π-messages with its parent and child.

14.5.2 Trees

Chains are restrictive because each node can have only a single parent and a single child, that is, a single cause and a single symptom. In a *tree*, each node may have several children but all nodes, except the single root, have exactly one parent. The same belief propagation also applies here with the difference from chains being that a node receives different λ-messages from its children, $\lambda_Y(X)$ denoting the message X receives from its child Y, and sends different π-messages to its children, $\pi_Y(X)$ denoting the message X sends to its child Y.

Again, we divide possible evidence to two parts, E^- are nodes that are in the subtree rooted at the query node X, and E^+ are evidence nodes elsewhere (see figure 14.10). Note that this second need not be an ancestor of X but may also be in a subtree rooted at a sibling of X. The important point is that again X separates E^+ and E^- so that we can write $P(E^+, E^-|X) = P(E^+|X)P(E^-|X)$, and hence have

$$P(X|E) = \alpha\pi(X)\lambda(X)$$

where again α is a normalizing constant.

$\lambda(X)$ is the evidence in the subtree rooted at X, and if X has two children Y and Z, as shown in figure 14.10, it can be calculated as

$$\begin{aligned}\lambda(X) &\equiv P(E_X^-|X) = P(E_Y^-, E_Z^-|X) \\ &= P(E_Y^-|X)P(E_Z^-|X) = \lambda_Y(X)\lambda_Z(X)\end{aligned} \tag{14.18}$$

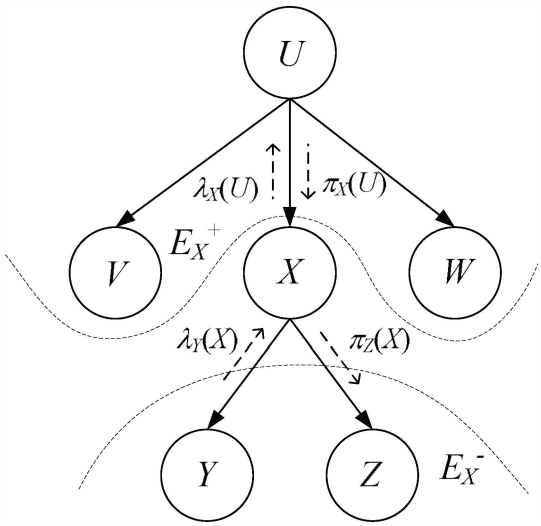

Figure 14.10 In a tree, a node may have several children but a single parent.

In the general case, if X has m children, $Y_j, j = 1, \ldots, m$, then we multiply all their λ values:

$$\lambda(X) = \prod_{j=1}^{m} \lambda_{Y_j}(X) \tag{14.19}$$

Once X accumulates λ evidence from its children's λ-messages, it propagates it up to its parent:

$$\lambda_X(U) = \sum_X \lambda(X) P(X|U) \tag{14.20}$$

Similarly and in the other direction, $\pi(X)$ is the evidence elsewhere that is accumulated in $P(U|E^+)$ and passed on to X as a π-message:

$$\pi(X) \equiv P(X|E_X^+) = \sum_U P(X|U) P(U|E_X^+) = \sum_U P(X|U) \pi_X(U) \tag{14.21}$$

This calculated π value is then propagated down to X's children. Note that what Y receives from X is what X receives from its parent U and also from its other child Z; together they make up E_Y^+ (see figure 14.10):

$$\pi_Y(X) \equiv P(X|E_Y^+) = P(X|E_X^+, E_Z^-)$$

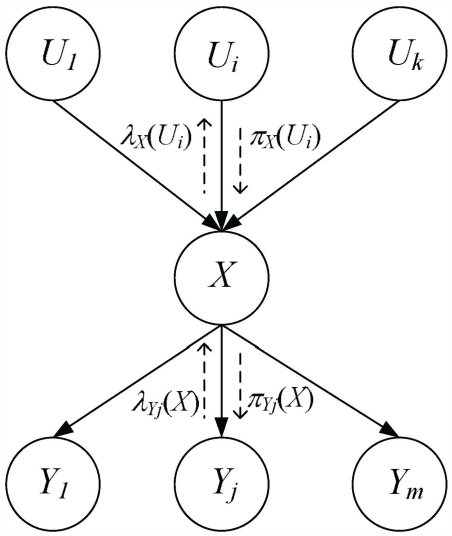

Figure 14.11 In a polytree, a node may have several children and several parents, but the graph is singly connected; that is, there is a single chain between U_i and Y_j passing through X.

$$= \frac{P(E_Z^-|X, E_X^+)P(X|E_X^+)}{P(E_Z^-)} = \frac{P(E_Z^-|X)P(X|E_X^+)}{P(E_Z^-)}$$

$$= \alpha\lambda_Z(X)\pi(X) \tag{14.22}$$

Again, if Y has not one sibling Z but multiple, we need to take a product over all their λ values:

$$\pi_{Y_j}(X) = \alpha \prod_{s \neq j} \lambda_{Y_s}(X)\pi(X) \tag{14.23}$$

14.5.3 Polytrees

POLYTREE

In a tree, a node has a single parent, that is, a single cause. In a *polytree*, a node may have multiple parents, but we require that the graph be singly connected, which means that there is a single chain between any two nodes. If we remove X, the graph will split into two components. This is necessary so that we can continue splitting E_X into E_X^+ and E_X^-, which are independent given X (see figure 14.11).

If X has multiple parents $U_i, i = 1, \ldots, k$, it receives π-messages from

all of them, $\pi_X(U_i)$, which it combines as follows:

$$\begin{aligned}
\pi(X) &\equiv P(X|E_X^+) = P(X, E_{U_1X}^+, E_{U_2X}^+, \ldots, E_{U_kX}^+) \\
&= \sum_{U_1}\sum_{U_2}\cdots\sum_{U_k} P(X|U_1, U_2, \ldots, U_k) P(U_1|E_{U_1X}^+)\cdots P(U_k|E_{U_kX}^+) \\
&= \sum_{U_1}\sum_{U_2}\cdots\sum_{U_k} P(X|U_1, U_2, \ldots, U_k) \prod_{i=1}^{k} \pi_X(U_i) \qquad (14.24)
\end{aligned}$$

and passes it on to its several children $Y_j, j = 1, \ldots, m$:

$$\pi_{Y_j}(X) = \alpha \prod_{s\neq j} \lambda_{Y_s}(X)\pi(X) \qquad (14.25)$$

In this case when X has multiple parents, a λ-message X passes on to one of its parents U_i combines not only the evidence X receives from its children but also the π-messages X receives from its other parents $U_r, r \neq i$; they together make up $E_{U_iX}^-$:

$$\begin{aligned}
\lambda_X(U_i) &\equiv P(E_{U_iX}^-|X) \\
&= \sum_X \sum_{U_{r\neq i}} P(E_X^-, E_{U_{r\neq i}X}^+, X, U_{r\neq i}|U_i) \\
&= \sum_X \sum_{U_{r\neq i}} P(E_X^-, E_{U_{r\neq i}X}^+|X, U_{r\neq i}, U_i) P(X, U_{r\neq i}|U_i) \\
&= \sum_X \sum_{U_{r\neq i}} P(E_X^-|X) P(E_{U_{r\neq i}X}^+|U_{r\neq i}) P(X|U_{r\neq i}, U_i) P(U_{r\neq i}|U_i) \\
&= \sum_X \sum_{U_{r\neq i}} P(E_X^-|X) \frac{P(U_{r\neq i}|E_{U_{r\neq i}X}^+) P(E_{U_{r\neq i}X}^+)}{P(U_{r\neq i})} P(X|U_{r\neq i}, U_i) P(U_{r\neq i}|U_i) \\
&= \beta \sum_X \sum_{U_{r\neq i}} P(E_X^-|X) P(U_{r\neq i}|E_{U_{r\neq i}X}^+) P(X|U_{r\neq i}, U_i) \\
&= \beta \sum_X \sum_{U_{r\neq i}} \lambda(X) \prod_{r\neq i} \pi_X(U_r) P(X|U_1, \ldots, U_k) \\
&= \beta \sum_X \lambda(X) \sum_{U_{r\neq i}} P(X|U_1, \ldots, U_k) \prod_{r\neq i} \pi_X(U_r) \qquad (14.26)
\end{aligned}$$

As in a tree, to find its overall λ, the parent multiplies the λ-messages it receives from its children:

$$\lambda(X) = \prod_{j=1}^{m} \lambda_{Y_j}(X) \qquad (14.27)$$

In this case of multiple parents, we need to store and manipulate the conditional probability given all the parents, $p(X|U_1,\ldots,U_k)$, which is costly for large k. Approaches have been proposed to decrease the complexity from exponential in k to linear. For example, in a *noisy OR gate*, any of the parents is sufficient to cause the event and the likelihood does not decrease when multiple parent events occur. If the probability that X happens when only cause U_i happens is $1-q_i$

NOISY OR

$$P(X|U_i,\sim U_{p\neq j}) = 1-q_i \tag{14.28}$$

the probability that X happens when a subset T of them occur is calculated as

$$P(X|T) = 1-\prod_{u_i\in T} q_i \tag{14.29}$$

For example, let us say wet grass has two causes, rain and a sprinkler, with $q_R = q_S = 0.1$; that is, both singly have a 90 percent probability of causing wet grass. Then, $P(W|R,\sim S) = 0.9$ and $P(W|R,S) = 0.99$.

Another possibility is to write the conditional probability as some function given a set of parameters, for example, as a linear model

$$P(X|U_1,\ldots,U_k,w_0,w_1,\ldots,w_k) = \text{sigmoid}\left(\sum_{i=1}^{k} w_i U_i + w_0\right) \tag{14.30}$$

where sigmoid guarantees that the output is a probability between 0 and 1. During training, we can learn the parameters $w_i, i = 0,\ldots,d$, for example, to maximize the likelihood on a sample.

14.5.4 Junction Trees

If there is a loop, that is, if there is a cycle in the underlying undirected graph—for example, if the parents of X share a common ancestor—the algorithm we discussed earlier does not work. In such a case, there is more than one path on which to propagate evidence and, for example, while evaluating the probability at X, we cannot say that X separates E into E_X^+ and E_X^- as causal (upward) and diagnostic (downward) evidence; removing X does not split the graph into two. Conditioning them on X does not make them independent and the two can interact through some other path not involving X.

We can still use the same algorithm if we can convert the graph to a polytree. We define *clique nodes* that correspond to a set of original variables and connect them so that they form a tree (see figure 14.12). We

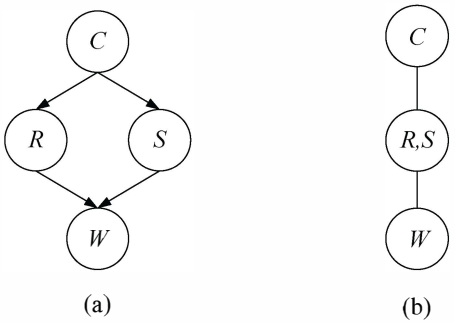

Figure 14.12 (a) A multiply connected graph, and (b) its corresponding junction tree with nodes clustered.

JUNCTION TREE

can then run the same belief propagation algorithm with some modifications. This is the basic idea behind the *junction tree algorithm* (Lauritzen and Spiegelhalter 1988; Jensen 1996; Jordan 2004).

14.6 Undirected Graphs: Markov Random Fields

MARKOV RANDOM FIELD

Up to now, we have discussed directed graphs where the influences are undirectional and have used Bayes' rule to invert the arcs. If the influences are symmetric, we represent them using an undirected graphical model, also known as a *Markov random field*. For example, neighboring pixels in an image tend to have the same color—that is, are correlated—and this correlation goes both ways.

Directed and undirected graphs define conditional independence differently, and, hence, there are probability distributions that are represented by a directed graph and not by an undirected graph, and vice versa (Pearl 1988).

Because there are no directions and hence no distinction between the head or the tail of an arc, the treatment of undirected graphs is simpler. For example, it is much easier to check if A and B are independent given C. We just check if after removing all nodes in C, we still have a path between a node in A and a node in B. If so, they are dependent, otherwise, if all paths between nodes in A and nodes in B pass through nodes in C such that removal of C leaves nodes of A and nodes of B in separate components, we have independence.

CLIQUE

In the case of an undirected graph, we do not talk about the parent or the child but about *cliques*, which are sets of nodes such that there exists a link between any two nodes in the set. A *maximal* clique has the maximum number of elements. Instead of conditional probabilities (implying a direction), in undirected graphs we have *potential functions* $\psi_C(X_C)$ where X_C is the set of variables in clique C, and we define the joint distribution as the product of the potential functions of the maximal cliques of the graph

POTENTIAL FUNCTION

$$p(X) = \frac{1}{Z}\prod_C \psi_C(X_C) \tag{14.31}$$

where Z is the normalization constant to make sure that $\sum_X p(X) = 1$:

$$Z = \sum_X \prod_C \psi_C(X) \tag{14.32}$$

It can be shown that a directed graph is already normalized (exercise 5).

Unlike in directed graphs, the potential functions in an undirected graph do not need to have a probabilistic interpretation, and one has more freedom in defining them. In general, we can view potential functions as expressing local constraints, that is, favoring some local configurations over others. For example, in an image, we can define a pairwise potential function between neighboring pixels, which takes a higher value if their colors are similar than the case when they are different (Bishop 2006). Then, setting some of the pixels to their values given as evidence, we can estimate the values of other pixels that are not known, for example, due to occlusion.

If we have the directed graph, it is easy to redraw it as an undirected graph, simply by dropping all the directions, and if a node has a single parent, we can set the pairwise potential function simply to the conditional probability. If the node has more than one parent, however, the "explaining away" phenomenon due to the head-to-head node makes the parents dependent, and hence we should have the parents in the same clique so that the clique potential includes all the parents. This is done by connecting all the parents of a node by links so that they are completely connected among them and form a clique. This is called "marrying" the parents, and the process is called *moralization*. Incidentally, moralization is one of the steps in generating a junction tree, which is undirected.

MORALIZATION

It is straightforward to adapt the belief propagation algorithm to work on undirected graphs, and it is easier because the potential function is

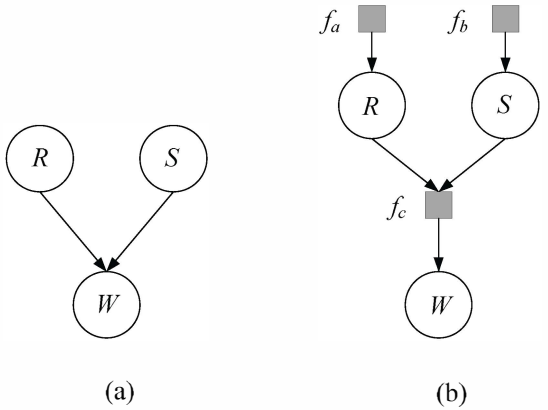

Figure 14.13 (a) A directed graph that would have a loop after moralization, and (b) its corresponding factor graph that is a tree. The three factors are $f_a(R) \equiv P(R)$, $f_b(S) \equiv P(S)$, and $f_c(R,S,W) \equiv P(W|R,S)$.

symmetric and we do not need to make a difference between causal and diagnostic evidence. Thus, we can do inference on undirected chains and trees. But in polytrees where a node has multiple parents and moralization necessarily creates loops, this would not work. One trick is to convert it to a *factor graph* that uses a second kind of *factor nodes* in addition to the variable nodes, and we write the joint distribution as a product of factors (Kschischang, Frey, and Loeliger 2001)

FACTOR GRAPH

$$p(X) = \frac{1}{Z}\prod_S f_S(X_S) \tag{14.33}$$

where X_s denotes a subset of the variable nodes used by factor S. Directed graphs are a special case where factors correspond to local conditional distributions, and undirected graphs are another special case where factors are potential functions over maximal cliques. The advantage is that, as we can see in figure 14.13, the tree structure can be kept even after moralization.

It is possible to generalize the belief propagation algorithm to work on factor graphs; this is called the *sum-product algorithm* (Bishop 2006; Jordan 2004) where there is the same idea of doing local computations once and propagating them through the graph as messages. The difference now is that there are two types of messages because there are two kinds of nodes, factors and variables, and we make a distinction between their

SUM-PRODUCT ALGORITHM

messages. Note, however, that a factor graph is bipartite, and one kind of node can have a close encounter only with the second kind.

In belief propagation, or the sum-product algorithm, the aim is to find the probability of a set of nodes X given that another set of evidence nodes E are clamped to a certain value, that is, $P(X|E)$. In some applications, we may be interested in finding the setting of all X that maximizes the full joint probability distribution $p(X)$. For example, in the undirected case where potential functions code locally consistent configurations, such an approach would propagate local constraints over the whole graph and find a solution that maximizes global consistency. In a graph where nodes correspond to pixels and pairwise potential functions favor correlation, this approach would implement noise removal (Bishop 2006). The algorithm for this, named the *max-product algorithm* (Bishop 2006; Jordan 2004) is the same as the sum-product algorithm where we take the maximum (choose the most likely value) rather than the sum (marginalize). This is analogous to the difference between the forward-backward procedure and the Viterbi algorithm in hidden Markov models that we discussed in chapter 15.

MAX-PRODUCT ALGORITHM

Note that the nodes need not correspond to low-level concepts like pixels; in a vision application, for instance, we may have nodes for corners of different types or lines of different orientations with potential functions checking for compatibility, so as to see if they can be part of the same interpretation—remember the Necker cube, for example—so that overall consistent solutions emerge after the consolidation of local evidences.

The complexity of the inference algorithms on polytrees or junction trees is determined by the maximum number of parents or the size of the largest clique, and when this is large, exact inference may be infeasible. In such a case, one needs to use an approximation or a sampling algorithm (Jordan 1999; Bishop 2006).

14.7 Learning the Structure of a Graphical Model

As in any approach, learning a graphical model has two parts. The first is the learning of parameters given a structure; this is relatively easier (Buntine 1996), and, in graphical models, conditional probability tables or their parameterizations (as in equation 14.30) can be trained to maximize the likelihood, or by using a Bayesian approach if suitable priors are known (chapter 16).

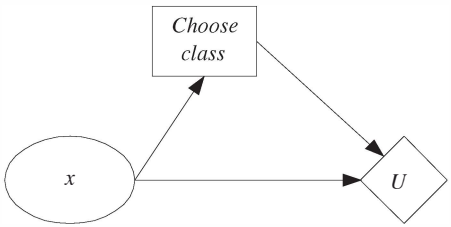

Figure 14.14 Influence diagram corresponding to classification. Depending on input x, a class is chosen that incurs a certain utility (risk).

The second, more difficult, and interesting part is to learn the graph structure (Cowell et al. 1999). This is basically a model selection problem, and just like the incremental approaches for learning the structure of a multilayer perceptron (section 11.9), we can see this as a search in the space of all possible graphs. One can, for example, consider operators that can add/remove arcs and/or hidden nodes and then do a search evaluating the improvement at each step (using parameter learning at each intermediate iteration). Note, however, that to check for overfitting, one should regularize properly, corresponding to a Bayesian approach with a prior that favors simpler graphs (Neapolitan 2004). However, because the state space is large, it is most helpful if there is a human expert who can manually define causal relationships among variables and creates subgraphs of small groups of variables.

In chapter 16, we discuss the Bayesian approach and in section 16.8, we discuss the nonparametric Bayesian methods where model structure can be made more complex in time as more data arrives.

14.8 Influence Diagrams

INFLUENCE DIAGRAMS

Just as in chapter 3, we generalized from probabilities to actions with risks, *influence diagrams* are graphical models that allow the generalization of graphical models to include decisions and utilities. An influence diagram contains *chance nodes* representing random variables that we use in graphical models (see figure 14.14). It also has decision nodes and a utility node. A *decision node* represents a choice of actions. A *utility node* is where the utility is calculated. Decisions may be based on chance nodes and may affect other chance nodes and the utility node.

Inference on an influence diagram is an extension to belief propagation on a graphical model. Given evidence on some of the chance nodes, this evidence is propagated, and for each possible decision, the utility is calculated and the decision having the highest utility is chosen. The influence diagram for classification of a given input is shown in figure 14.14. Given the input, the decision node decides on a class, and for each choice we incur a certain utility (risk).

14.9 Notes

Graphical models have two advantages. One is that we can visualize the interaction of variables and have a better understanding of the process, for example, by using a causal generative model. The second is that by finding graph operations that correspond to basic probabilistic procedures such as Bayes' rule or marginalization, the task of inference can be mapped to general-purpose graph algorithms that can be efficiently represented and implemented.

The idea of visual representation of variables and dependencies between them as a graph, and the related factorization of a complicated global function of many variables as a product of local functions involving a small subset of the variables for each, seems to be used in different domains in decision making, coding, and signal processing; Kschischang, Frey, and Loeliger (2001) give a review.

The complexity of the inference algorithms on polytrees or junction trees is determined by the maximum number of parents or the size of the largest clique, and when this is large exact inference may be infeasible. In such a case, one needs to use an approximation or a sampling algorithm. Variational approximations and Markov chain Monte Carlo (MCMC) algorithms are discussed in Jordan et al. 1999, MacKay 2003, Andrieu et al. 2003, Bishop 2006, and Murphy 2012.

Graphical models are especially suited to represent Bayesian approaches where in addition to nodes for observed variables, we also have nodes for hidden variables as well as the model parameters. We may also introduce a hierarchy where we have nodes for hyperparameters—that is, second-level parameters for the priors of the first-level parameters.

Thinking of data as sampled from a causal generative model that can be visualized as a graph can ease understanding and also inference in many domains. For example, in text categorization, generating a text may be

thought of as the process whereby an author decides to write a document on a number of topics and then chooses a set of words for each topic. In bioinformatics, one area among many where a graphical approach used is the modeling of a *phylogenetic tree*; namely, it is a directed graph whose leaves are the current species, whose nonterminal nodes are past ancestors that split into multiple species during a speciation event, and whose conditional probabilities depend on the evolutionary distance between a species and its ancestor (Jordan 2004).

PHYLOGENETIC TREE

The hidden Markov model we discuss in chapter 15 is one type of graphical model where inputs are dependent sequentially, as in speech recognition, where a word is a particular sequence of basic speech sounds called phonemes (Ghahramani 2001). Such *dynamic graphical models* find applications in many areas where there is a temporal dimension, such as speech, music, and so on (Zweig 2003; Bilmes and Bartels 2005).

Graphical models are also used in computer vision—for example, in information retrieval (Barnard et al. 2003) and scene analysis (Sudderth et al. 2008). A review of the use of graphical models in bioinformatics (and related software) is given in Donkers and Tuyls 2008.

14.10 Exercises

1. With two independent inputs in a classification problem, that is, $p(x_1, x_2|C) = p(x_1|C)p(x_2|C)$, how can we calculate $p(x_1|x_2, C)$? Derive the formula for $p(x_j|C_i) \sim \mathcal{N}(\mu_{ij}, \sigma_{ij}^2)$.

2. For a head-to-head node, show that equation 14.10 implies $P(X, Y) = P(X) \cdot P(Y)$.

   SOLUTION: We know that $P(X, Y, Z) = P(Z|X, Y)P(X, Y)$, and if we also know that $P(X, Y, Z) = P(X)P(Y)P(Z|X, Y)$, we see that $P(X, Y) = P(X)P(Y)$.

3. In figure 14.4, calculate $P(R|W)$, $P(R|W, S)$, and $P(R|W, \sim S)$.

   SOLUTION:

$$
\begin{aligned}
P(R|W) &= \frac{P(R,W)}{P(W)} = \frac{\sum_S P(R,W,S)}{\sum_R \sum_S P(R,W,S)} \\
&= \frac{\sum_S P(R)P(S)P(W|R,S)}{\sum_R \sum_S P(R)P(S)P(W|R,S)} \\
P(R|W,S) &= \frac{P(R,W,S)}{P(W,S)} = \frac{P(R)P(S)P(W|R,S)}{\sum_R P(R)P(S)P(W|R,S)} \\
P(R|W,\sim S) &= \frac{P(R,W,\sim S)}{P(W,\sim S)} = \frac{P(R)P(\sim S)P(W|R,\sim S)}{\sum_R P(R)P(\sim S)P(W|R,\sim S)}
\end{aligned}
$$

4. In equation 14.30, X is binary. How do we need to modify it if X can take one of K discrete values?

   SOLUTION: Let us say there are $j = 1, \ldots, K$ states. Then, keeping the model linear, we need to parameterize each by a separate $\mathbf{w}_j$ and use softmax to map to probabilities:

$$P(X = j | U_1, \ldots, U_k, \{w_{ji}\}) = \frac{\exp \sum_{i=1}^{k} w_{ji} U_i + w_{j0}}{\sum_{l=1}^{K} \exp \sum_{i=1}^{k} w_{li} U_i + w_{l0}}$$

5. Show that in a directed graph where the joint distribution is written as equation 14.12, $\sum_{\mathbf{x}} p(\mathbf{x}) = 1$.

   SOLUTION: The terms cancel when we sum up over all possible values because these are probabilities. Let us, for example, take figure 14.3:

$$\begin{aligned}
P(X,Y,Z) &= P(X)P(Y|X)P(Z|X) \\
\sum_X \sum_Y \sum_Z P(X,Y,Z) &= \sum_X \sum_Y \sum_Z P(X)P(Y|X)P(Z|X) \\
&= \sum_X \sum_Y P(X)P(Y|X) \sum_Z P(Z|X) \\
&= \sum_X \sum_Y P(X)P(Y|X) \sum_Z \frac{P(Z,X)}{P(X)} \\
&= \sum_X \sum_Y P(X)P(Y|X) \frac{P(X)}{P(X)} \\
&= \sum_X \sum_Y P(X)P(Y|X) \\
&= \sum_X P(X) \sum_Y P(Y|X) = \sum_X P(X) = 1
\end{aligned}$$

6. Draw the Necker cube as a graphical model defining links to indicate mutually reinforcing or inhibiting relations between different corner interpretations.

   SOLUTION: We are going to have nodes corresponding to corners, and they take values depending on the interpretation; there will be positive, enforcing, excitatory connections between corners that are part of the same interpretation, and negative, inhibitory connections between corners that are part of different interpretations (see figure 14.15).

7. Write down the graphical model for linear logistic regression for two classes in the manner of figure 14.7.

8. Propose a suitable goodness measure that can be used in learning graph structure as a state-space search. What are suitable operators?

   SOLUTION: We need a *score function* that is the sum of two parts, one quantifying a goodness of fit, that is, how likely is the data given the model, and one quantifying the complexity of the graph, to alleviate overfitting. In measuring complexity, we must take into account the total number of nodes and

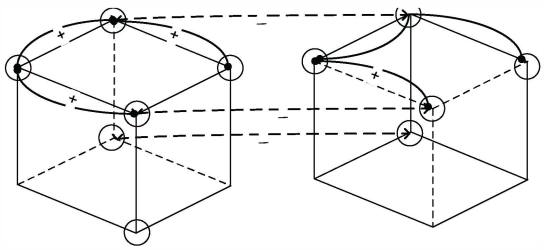

Figure 14.15 Two different interpretations of the Necker cube. Solid lines, marked by '+,' are excitatory and dashed lines, marked by '–,' are inhibitory.

the number of parameters needed to represent the conditional probability distributions. For example, we should try to have nodes with as few parents as possible. Possible operators are there to add/remove an edge and add/remove a hidden node.

9. Generally, in a newspaper, a reporter writes a series of articles on successive days related to the same topics as the story develops. How can we model this using a graphical model?

14.11 References

Andrieu, C., N. de Freitas, A. Doucet, and M. I. Jordan. 2003. "An Introduction to MCMC for Machine Learning." *Machine Learning* 50:5–43.

Barnard, K., P. Duygulu, D. Forsyth, N. de Freitas, D. M. Blei, and M. I. Jordan. 2003. "Matching Words and Pictures." *Journal of Machine Learning Research* 3:1107–1135.

Bilmes, J., and C. Bartels. 2005. "Graphical Model Architectures for Speech Recognition." *IEEE Signal Processing Magazine* 22:89–100.

Bishop, C. M. 2006. *Pattern Recognition and Machine Learning*. New York: Springer.

Buntine, W. 1996. "A Guide to the Literature on Learning Probabilistic Networks from Data." *IEEE Transactions on Knowledge and Data Engineering* 8:195–210.

Cowell, R. G., A. P. Dawid, S. L. Lauritzen, and D. J. Spiegelhalter. 1999. *Probabilistic Networks and Expert Systems*. New York: Springer.

Donkers, J., and K. Tuyls. 2008. "Belief Networks in Bioinformatics." In *Computational Intelligence in Bioinformatics*, ed. A. Kelemen, A. Abraham, and Y. Chen, 75–111. Berlin: Springer.

Ghahramani, Z. 2001. "An Introduction to Hidden Markov Models and Bayesian Networks." *International Journal of Pattern Recognition and Artificial Intelligence* 15:9–42.

Jensen, F. 1996. *An Introduction to Bayesian Networks*. New York: Springer.

Jordan, M. I., ed. 1999. *Learning in Graphical Models*. Cambridge, MA: MIT Press.

Jordan, M. I. 2004. "Graphical Models." *Statistical Science* 19:140–155.

Jordan, M. I., Z. Ghahramani, T. S. Jaakkola, and L. K. Saul. 1999. "An Introduction to Variational Methods for Graphical Models." In *Learning in Graphical Models*, ed. M. I. Jordan, 105–161. Cambridge, MA: MIT Press.

Kschischang, F. R., B. J. Frey, and H.-A. Loeliger. 2001. "Factor Graphs and the Sum-Product Algorithm." *IEEE Transactions on Information Theory* 47:498–519.

Lauritzen, S. L., and D. J. Spiegelhalter. 1988. "Local Computations with Probabilities on Graphical Structures and their Application to Expert Systems." *Journal of Royal Statistical Society B* 50:157–224.

MacKay, D. J. C. 2003. *Information Theory, Inference, and Learning Algorithms*. Cambridge, UK: Cambridge University Press.

Murphy, K. P. 2012. *Machine Learning: A Probabilistic Perspective*. Cambridge, MA: MIT Press.

Neapolitan, R. E. 2004. *Learning Bayesian Networks*. Upper Saddle River, NJ: Pearson.

Pearl, J. 1988. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. San Francisco, CA: Morgan Kaufmann.

Pearl, J. 2000. *Causality: Models, Reasoning, and Inference*. Cambridge, UK: Cambridge University Press.

Sudderth, E. B., A. Torralba, W. T. Freeman, and A. S. Willsky. 2008. "Describing Visual Scenes Using Transformed Objects and Parts." *International Journal of Computer Vision* 77:291–330.

Zweig, G. 2003. "Bayesian Network Structures and Inference Techniques for Automatic Speech Recognition." *Computer Speech and Language* 17:173–193.