10 *Directed graphical models (Bayes nets)*

10.1 Introduction

> I basically know of two principles for treating complicated systems in simple ways: the first is the principle of modularity and the second is the principle of abstraction. I am an apologist for computational probability in machine learning because I believe that probability theory implements these two principles in deep and intriguing ways — namely through factorization and through averaging. Exploiting these two mechanisms as fully as possible seems to me to be the way forward in machine learning. — Michael Jordan, 1997 (quoted in (Frey 1998)).

Suppose we observe multiple correlated variables, such as words in a document, pixels in an image, or genes in a microarray. How can we compactly *represent* the **joint distribution** $p(\mathbf{x}|\boldsymbol{\theta})$? How can we use this distribution to *infer* one set of variables given another in a reasonable amount of computation time? And how can we *learn* the parameters of this distribution with a reasonable amount of data? These questions are at the core of probabilistic modeling, inference and learning, and form the topic of this chapter.

10.1.1 Chain rule

By the **chain rule** of probability, we can always represent a joint distribution as follows, using any ordering of the variables:

$$p(x_{1:V}) = p(x_1)p(x_2|x_1)p(x_3|x_2,x_1)p(x_4|x_1,x_2,x_3)\dots p(x_V|x_{1:V-1}) \tag{10.1}$$

where V is the number of variables, the Matlab-like notation $1:V$ denotes the set $\{1,2,\dots,V\}$, and where we have dropped the conditioning on the fixed parameters $\boldsymbol{\theta}$ for brevity. The problem with this expression is that it becomes more and more complicated to represent the conditional distributions $p(x_t|\mathbf{x}_{1:t-1})$ as t gets large.

For example, suppose all the variables have K states. We can represent $p(x_1)$ as a table of $O(K)$ numbers, representing a discrete distribution (there are actually only $K-1$ free parameters, due to the sum-to-one constraint, but we write $O(K)$ for simplicity). Similarly, we can represent $p(x_2|x_1)$ as a table of $O(K^2)$ numbers by writing $p(x_2=j|x_1=i)=T_{ij}$; we say that $\mathbf{T}$ is a **stochastic matrix**, since it satisfies the constraint $\sum_j T_{ij}=1$ for all rows i, and $0 \le T_{ij} \le 1$ for all entries. Similarly, we can represent $p(x_3|x_1,x_2)$ as a 3d table with

$O(K^3)$ numbers. These are called **conditional probability tables** or **CPTs**. We see that there are $O(K^V)$ parameters in the model. We would need an awful lot of data to learn so many parameters.

One solution is to replace each CPT with a more parsimonius **conditional probability distribution** or **CPD**, such as multinomial logistic regression, i.e., $p(x_t = k|\mathbf{x}_{1:t-1}) = \mathcal{S}(\mathbf{W}_t\mathbf{x}_{1:t-1})_k$. The total number of parameters is now only $O(K^2V^2)$, making this a compact density model (Neal 1992; Frey 1998). This is adequate if all we want to do is evaluate the probability of a fully observed vector $\mathbf{x}_{1:T}$. For example, we can use this model to define a class-conditional density, $p(\mathbf{x}|y = c)$, thus making a generative classifier (Bengio and Bengio 2000). However, this model is not useful for other kinds of prediction tasks, since each variable depends on all the previous variables. So we need another approach.

10.1.2 Conditional independence

The key to efficiently representing large joint distributions is to make some assumptions about conditional independence (**CI**). Recall from Section 2.2.4 that X and Y are conditionally independent given Z, denoted $X \perp Y|Z$, if and only if (iff) the conditional joint can be written as a product of conditional marginals, i.e.,

$$X \perp Y|Z \iff p(X, Y|Z) = p(X|Z)p(Y|Z) \tag{10.2}$$

Let us see why this might help. Suppose we assume that $x_{t+1} \perp \mathbf{x}_{1:t-1}|x_t$, or in words, "the future is independent of the past given the present". This is called the (first order) **Markov assumption**. Using this assumption, plus the chain rule, we can write the joint distribution as follows:

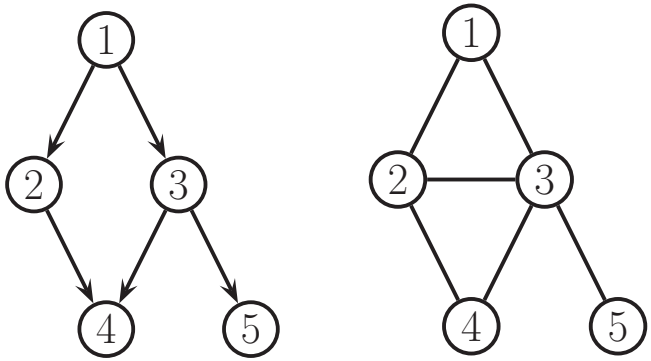
$$p(\mathbf{x}_{1:V}) = p(x_1)\prod_{t=1}^{V} p(x_t|x_{t-1}) \tag{10.3}$$

This is called a (first-order) **Markov chain**. They can be characterized by an initial distribution over states, $p(x_1 = i)$, plus a **state transition matrix** $p(x_t = j|x_{t-1} = i)$. See Section 17.2 for more information.

10.1.3 Graphical models

Although the first-order Markov assumption is useful for defining distributions on 1d sequences, how can we define distributions on 2d images, or 3d videos, or, in general, arbitrary collections of variables (such as genes belonging to some biological pathway)? This is where graphical models come in.

A **graphical model** (**GM**) is a way to represent a joint distribution by making CI assumptions. In particular, the nodes in the graph represent random variables, and the (lack of) edges represent CI assumptions. (A better name for these models would in fact be "independence diagrams", but the term "graphical models" is now entrenched.) There are several kinds of graphical model, depending on whether the graph is directed, undirected, or some combination of directed and undirected. In this chapter, we just study directed graphs. We consider undirected graphs in Chapter 19.



(a) (b)

Figure 10.1 (a) A simple DAG on 5 nodes, numbered in topological order. Node 1 is the root, nodes 4 and 5 are the leaves. (b) A simple undirected graph, with the following maximal cliques: $\{1, 2, 3\}$, $\{2, 3, 4\}$, $\{3, 5\}$.

10.1.4 Graph terminology

Before we continue, we must define a few basic terms, most of which are very intuitive.

A **graph** $G = (\mathcal{V}, \mathcal{E})$ consists of a set of **nodes** or **vertices**, $\mathcal{V} = \{1, \ldots, V\}$, and a set of **edges**, $\mathcal{E} = \{(s, t) : s, t \in \mathcal{V}\}$. We can represent the graph by its **adjacency matrix**, in which we write $G(s, t) = 1$ to denote $(s, t) \in \mathcal{E}$, that is, if $s \to t$ is an edge in the graph. If $G(s, t) = 1$ iff $G(t, s) = 1$, we say the graph is **undirected**, otherwise it is **directed**. We usually assume $G(s, s) = 0$, which means there are no **self loops**.

Here are some other terms we will commonly use:

- **Parent** For a directed graph, the **parents** of a node is the set of all nodes that feed into it: $\mathrm{pa}(s) \triangleq \{t : G(t, s) = 1\}$.
- **Child** For a directed graph, the **children** of a node is the set of all nodes that feed out of it: $\mathrm{ch}(s) \triangleq \{t : G(s, t) = 1\}$.
- **Family** For a directed graph, the **family** of a node is the node and its parents, $\mathrm{fam}(s) = \{s\} \cup \mathrm{pa}(s)$.
- **Root** For a directed graph, a **root** is a node with no parents.
- **Leaf** For a directed graph, a **leaf** is a node with no children.
- **Ancestors** For a directed graph, the **ancestors** are the parents, grand-parents, etc of a node. That is, the ancestors of t is the set of nodes that connect to t via a trail: $\mathrm{anc}(t) \triangleq \{s : s \rightsquigarrow t\}$.
- **Descendants** For a directed graph, the **descendants** are the children, grand-children, etc of a node. That is, the descendants of s is the set of nodes that can be reached via trails from s: $\mathrm{desc}(s) \triangleq \{t : s \rightsquigarrow t\}$.
- **Neighbors** For any graph, we define the **neighbors** of a node as the set of all immediately connected nodes, $\mathrm{nbr}(s) \triangleq \{t : G(s, t) = 1 \vee G(t, s) = 1\}$. For an undirected graph, we

write $s \sim t$ to indicate that s and t are neighbors (so $(s,t) \in \mathcal{E}$ is an edge in the graph).

- **Degree** The **degree** of a node is the number of neighbors. For directed graphs, we speak of the **in-degree** and **out-degree**, which count the number of parents and children.
- **Cycle or loop** For any graph, we define a **cycle** or **loop** to be a series of nodes such that we can get back to where we started by following edges, $s_1 - s_2 \cdots - s_n - s_1$, $n \geq 2$. If the graph is directed, we may speak of a directed cycle. For example, in Figure 10.1(a), there are no directed cycles, but $1 \rightarrow 2 \rightarrow 4 \rightarrow 3 \rightarrow 1$ is an undirected cycle.
- **DAG** A **directed acyclic graph** or **DAG** is a directed graph with no directed cycles. See Figure 10.1(a) for an example.
- **Topological ordering** For a DAG, a **topological ordering** or **total ordering** is a numbering of the nodes such that parents have lower numbers than their children. For example, in Figure 10.1(a), we can use $(1,2,3,4,5)$, or $(1,3,2,5,4)$, etc.
- **Path or trail** A **path** or **trail** $s \rightsquigarrow t$ is a series of directed edges leading from s to t.
- **Tree** An undirected **tree** is an undirectecd graph with no cycles. A directed tree is a DAG in which there are no directed cycles. If we allow a node to have multiple parents, we call it a **polytree**, otherwise we call it a moral directed tree.
- **Forest** A **forest** is a set of trees.
- **Subgraph** A (node-induced) **subgraph** G_A is the graph created by using the nodes in A and their corresponding edges, $G_A = (\mathcal{V}_A, \mathcal{E}_A)$.
- **Clique** For an undirected graph, a **clique** is a set of nodes that are all neighbors of each other. A **maximal clique** is a clique which cannot be made any larger without losing the clique property. For example, in Figure 10.1(b), $\{1,2\}$ is a clique but it is not maximal, since we can add 3 and still maintain the clique property. In fact, the maximal cliques are as follows: $\{1,2,3\}$, $\{2,3,4\}$, $\{3,5\}$.

10.1.5 Directed graphical models

A **directed graphical model** or **DGM** is a GM whose graph is a DAG. These are more commonly known as **Bayesian networks**. However, there is nothing inherently "Bayesian" about Bayesian networks: they are just a way of defining probability distributions. These models are also called **belief networks**. The term "belief" here refers to subjective probability. Once again, there is nothing inherently subjective about the kinds of probability distributions represented by DGMs. Finally, these models are sometimes called **causal networks**, because the directed arrows are sometimes interpreted as representing causal relations. However, there is nothing inherently causal about DGMs. (See Section 26.6.1 for a discussion of causal DGMs.) For these reasons, we use the more neutral (but less glamorous) term DGM.

The key property of DAGs is that the nodes can be ordered such that parents come before children. This is called a topological ordering, and it can be constructed from any DAG. Given such an order, we define the **ordered Markov property** to be the assumption that a node only depends on its immediate parents, not on all predecessors in the ordering, i.e.,

$$x_s \perp \mathbf{x}_{\mathrm{pred}(s) \setminus \mathrm{pa}(s)} | \mathbf{x}_{\mathrm{pa}(s)} \tag{10.4}$$

where $\mathrm{pa}(s)$ are the parents of node s, and $\mathrm{pred}(s)$ are the predecessors of node s in the ordering. This is a natural generalization of the first-order Markov property to from chains to general DAGs.

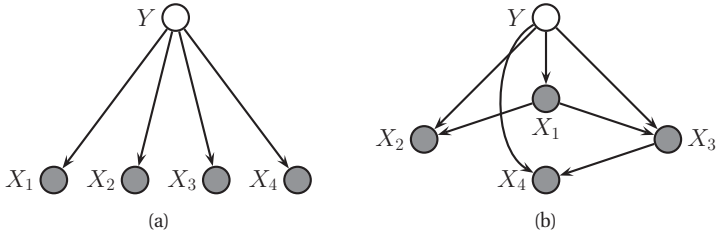


Figure 10.2 (a) A naive Bayes classifier represented as a DGM. We assume there are $D = 4$ features, for simplicity. Shaded nodes are observed, unshaded nodes are hidden. (b) Tree-augmented naive Bayes classifier for $D = 4$ features. In general, the tree topology can change depending on the value of y.

For example, the DAG in Figure 10.1(a) encodes the following joint distribution:

$$p(\mathbf{x}_{1:5}) = p(x_1)p(x_2|x_1)p(x_3|x_1,\cancel{x_2})p(x_4|\cancel{x_1},x_2,x_3)p(x_5|\cancel{x_1},\cancel{x_2},x_3,\cancel{x_4}) \quad (10.5)$$

$$= p(x_1)p(x_2|x_1)p(x_3|x_1)p(x_4|x_2,x_3)p(x_5|x_3) \quad (10.6)$$

In general, we have

$$p(\mathbf{x}_{1:V}|G) = \prod_{t=1}^{V} p(x_t|\mathbf{x}_{\mathrm{pa}(t)}) \quad (10.7)$$

where each term $p(x_t|\mathbf{x}_{\mathrm{pa}(t)})$ is a CPD. We have written the distribution as $p(\mathbf{x}|G)$ to emphasize that this equation only holds if the CI assumptions encoded in DAG G are correct. However, we will usual drop this explicit conditioning for brevity. If each node has $O(F)$ parents and K states, the number of parameters in the model is $O(VK^F)$, which is much less than the $O(K^V)$ needed by a model which makes no CI assumptions.

10.2 Examples

In this section, we show a wide variety of commonly used probabilistic models can be conveniently represented as DGMs.

10.2.1 Naive Bayes classifiers

In Section 3.5, we introduced the naive Bayes classifier. This assumes the features are conditionally independent given the class label. This assumption is illustrated in Figure 10.2(a). This allows us to write the joint distirbution as follows:

$$p(y,\mathbf{x}) = p(y)\prod_{j=1}^{D} p(x_j|y) \quad (10.8)$$

The naive Bayes assumption is rather naive, since it assumes the features are conditionally independent. One way to capture correlation between the features is to use a graphical model. In particular, if the model is a tree, the method is known as a **tree-augmented naive Bayes**

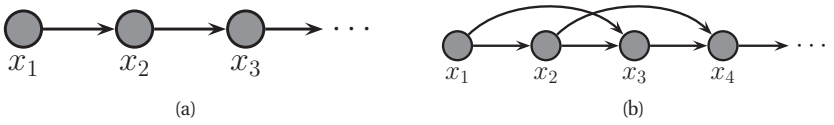


Figure 10.3 A first and second order Markov chain.

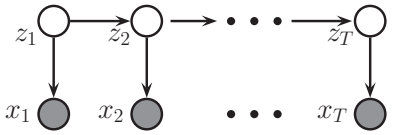


Figure 10.4 A first-order HMM.

classifier or **TAN** model (Friedman et al. 1997). This is illustrated in Figure 10.2(b). The reason to use a tree, as opposed to a generic graph, is two-fold. First, it is easy to find the optimal tree structure using the Chow-Liu algorithm, as explained in Section 26.3. Second, it is easy to handle missing features in a tree-structured model, as we explain in Section 20.2.

10.2.2 Markov and hidden Markov models

Figure 10.3(a) illustrates a first-order Markov chain as a DAG. Of course, the assumption that the immediate past, x_{t-1}, captures everything we need to know about the entire history, $\mathbf{x}_{1:t-2}$, is a bit strong. We can relax it a little by adding a dependence from x_{t-2} to x_t as well; this is called a **second order Markov chain**, and is illustrated in Figure 10.3(b). The corresponding joint has the following form:

$$p(\mathbf{x}_{1:T}) = p(x_1, x_2)p(x_3|x_1, x_2)p(x_4|x_2, x_3)\ldots = p(x_1, x_2)\prod_{t=3}^{T} p(x_t|x_{t-1}, x_{t-2}) \tag{10.9}$$

We can create higher-order Markov models in a similar way. See Section 17.2 for a more detailed discussion of Markov models.

Unfortunately, even the second-order Markov assumption may be inadequate if there are long-range correlations amongst the observations. We can't keep building ever higher order models, since the number of parameters will blow up. An alternative approach is to assume that there is an underlying hidden process, that can be modeled by a first-order Markov chain, but that the data is a noisy observation of this process. The result is known as a **hidden Markov model** or **HMM**, and is illustrated in Figure 10.4. Here z_t is known as a **hidden variable** at "time" t, and x_t is the observed variable. (We put "time" in quotation marks, since these models can be applied to any kind of sequence data, such as genomics or language, where t represents location rather than time.) The CPD $p(z_t|z_{t-1})$ is the **transition model**, and the CPD $p(\mathbf{x}_t|z_t)$ is the **observation model**.

h_0	h_1	h_2	$P(v=0\|h_1,h_2)$	$P(v=1\|h_1,h_2)$
1	0	0	θ_0	$1-\theta_0$
1	1	0	$\theta_0\theta_1$	$1-\theta_0\theta_1$
1	0	1	$\theta_0\theta_2$	$1-\theta_0\theta_2$
1	1	1	$\theta_0\theta_1\theta_2$	$1-\theta_0\theta_1\theta_2$

Table 10.1 Noisy-OR CPD for 2 parents augmented with leak node. We have omitted the t subscript for brevity.

The hidden variables often represent quantities of interest, such as the identity of the word that someone is currently speaking. The observed variables are what we measure, such as the acoustic waveform. What we would like to do is estimate the hidden state given the data, i.e., to compute $p(z_t|\mathbf{x}_{1:t}, \boldsymbol{\theta})$. This is called **state estimation**, and is just another form of probabilistic inference. See Chapter 17 for further details on HMMs.

10.2.3 Medical diagnosis

Consider modeling the relationship between various variables that are measured in an intensive care unit (ICU), such as the breathing rate of a patient, their blood pressure, etc. The **alarm network** in Figure 10.5(a) is one way to represent these (in)dependencies (Beinlich et al. 1989). This model has 37 variables and 504 parameters.

Since this model was created by hand, by a process called **knowledge engineering**, it is known as a **probabilistic expert system**. In Section 10.4, we discuss how to learn the parameters of DGMs from data, assuming the graph structure is known, and in Chapter 26, we discuss how to learn the graph structure itself.

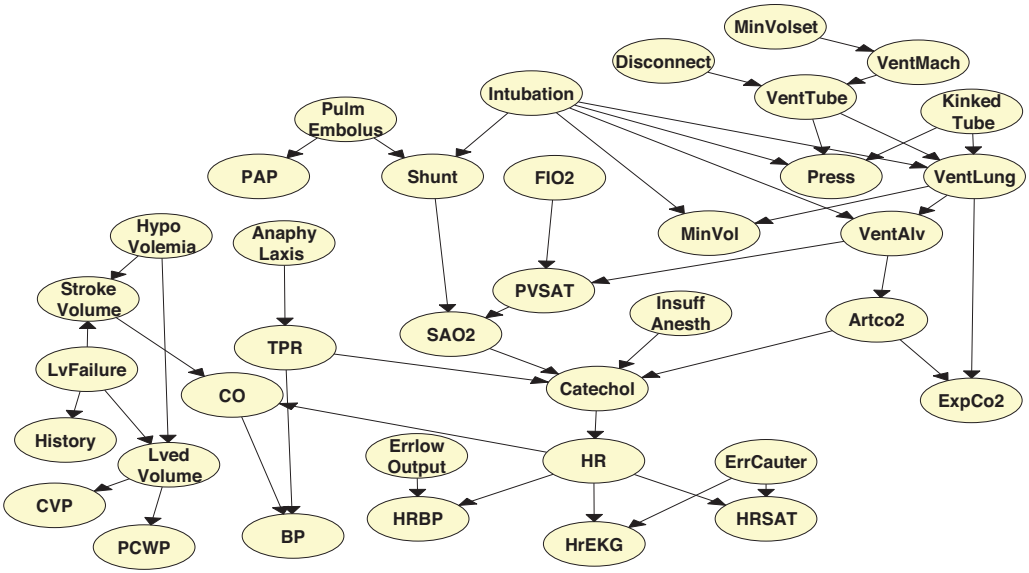
A different kind of medical diagnosis network, known as the **quick medical reference** or **QMR** network (Shwe et al. 1991), is shown in Figure 10.5(b). This was designed to model infectious diseases. The QMR model is a **bipartite graph** structure, with diseases (causes) at the top and symptoms or findings at the bottom. All nodes are binary. We can write the distribution as follows:

$$p(\mathbf{v}, \mathbf{h}) = \prod_s p(h_s) \prod_t p(v_t|\mathbf{h}_{\mathrm{pa}(t)}) \tag{10.10}$$

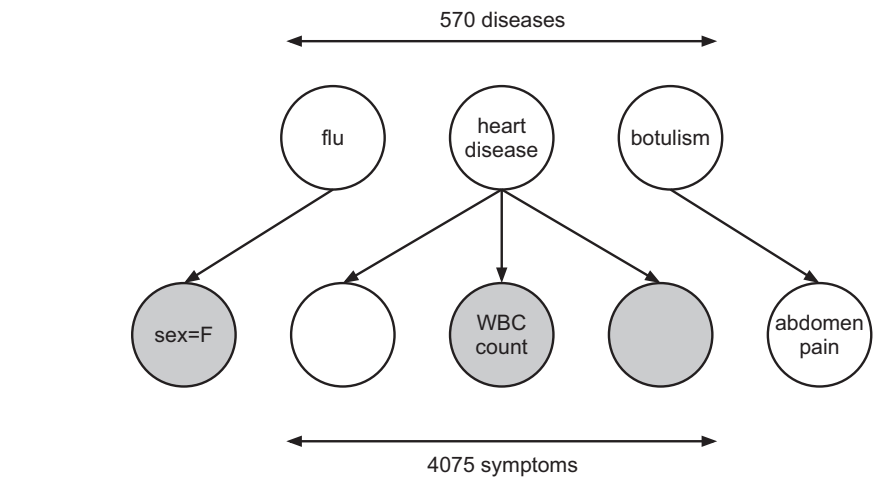
where h_s represent the **hidden nodes** (diseases), and v_t represent the **visible nodes** (symptoms).

The CPD for the root nodes are just Bernoulli distributions, representing the prior probability of that disease. Representing the CPDs for the leaves (symptoms) using CPTs would require too many parameters, because the **fan-in** (number of parents) of many leaf nodes is very high. A natural alternative is to use logistic regression to model the CPD, $p(v_t = 1|\mathbf{h}_{\mathrm{pa}(t)}) = \mathrm{sigm}(\mathbf{w}_t^T \mathbf{h}_{\mathrm{pa}(t)})$. (A DGM in which the CPDs are logistic regression distributions is known as a **sigmoid belief net** (Neal 1992).) However, since the parameters of this model were created by hand, an alternative CPD, known as the **noisy-OR** model, was used.

The noisy-OR model assumes that if a parent is on, then the child will usually also be on (since it is an or-gate), but occasionally the "links" from parents to child may fail, independently at random. In this case, even if the parent is on, the child may be off. To model this more precisely, let $\theta_{st} = 1 - q_{st}$ be the probability that the $s \to t$ link fails, so $q_{st} = 1 - \theta_{st} = p(v_t =$



(a)



(b)

Figure 10.5 (a) The alarm network. Figure generated by `visualizeAlarmNetwork`. (b) The QMR network.

G^p	G^m	$p(X=a)$	$p(X=b)$	$p(X=o)$	$p(X=ab)$
a	a	1	0	0	0
a	b	0	0	0	1
a	o	1	0	0	0
b	a	0	0	0	1
b	b	0	1	0	0
b	o	0	1	0	0
o	a	1	0	0	0
o	b	0	1	0	0
o	o	0	0	1	0

Table 10.2 CPT which encodes a mapping from genotype to phenotype (bloodtype). This is a deterministic, but many-to-one, mapping.

$1|h_s = 1, \mathbf{h}_{-s} = 0)$ is the probability that s can activate t on its own (its "causal power"). The only way for the child to be off is if all the links from all parents that are on fail independently at random. Thus

$$p(v_t = 0|\mathbf{h}) = \prod_{s \in \mathrm{pa}(t)} \theta_{st}^{\mathbb{I}(h_s=1)} \tag{10.11}$$

Obviously, $p(v_t = 1|\mathbf{h}) = 1 - p(v_t = 0|\mathbf{h})$.

If we observe that $v_t = 1$ but all its parents are off, then this contradicts the model. Such a data case would get probability zero under the model, which is problematic, because it is possible that someone exhibits a symptom but does not have any of the specified diseases. To handle this, we add a dummy **leak node** h_0, which is always on; this represents "all other causes". The parameter q_{0t} represents the probability that the background leak can cause the effect on its own. The modified CPD becomes $p(v_t = 0|\mathbf{h}) = \theta_{0t} \prod_{s \in \mathrm{pa}(t)} \theta_{st}^{h_s}$. See Table 10.1 for a numerical example.

If we define $w_{st} \triangleq \log(\theta_{st})$, we can rewrite the CPD as

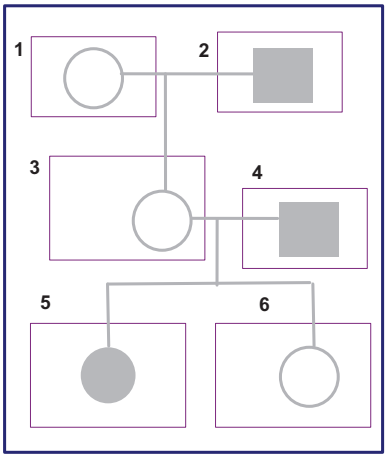
$$p(v_t = 1|\mathbf{h}) = 1 - \exp\left(w_{0t} + \sum_s h_s w_{st}\right) \tag{10.12}$$

We see that this is similar to a logistic regression model.

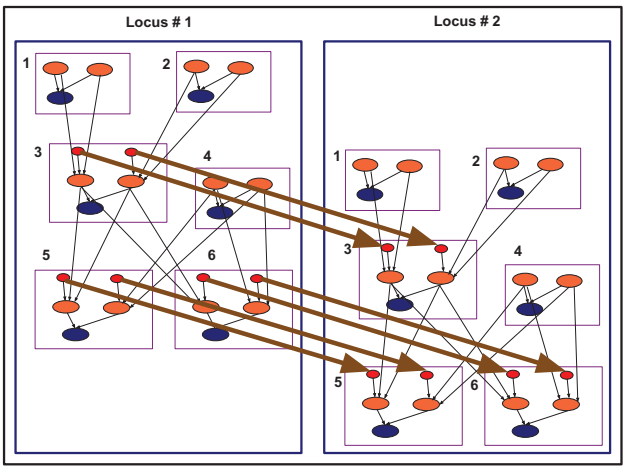
Bipartite models with noisy-OR CPDs are called **BN2O** models. It is relatively easy to set the θ_{st} parameters by hand, based on domain expertise. However, it is also possible to learn them from data (see e.g, (Neal 1992; Meek and Heckerman 1997)). Noisy-OR CPDs have also proved useful in modeling human causal learning (Griffiths and Tenenbaum 2005), as well as general binary classification settings (Yuille and Zheng 2009).

10.2.4 Genetic linkage analysis *

Another important (and historically very early) application of DGMs is to the problem of **genetic linkage analysis**. We start with a **pedigree graph**, which is a DAG that representing the relationship between parents and children, as shown in Figure 10.6(a). We then convert this to a DGM, as we explain below. Finally we perform probabilistic inference in the resulting model.



(a)



(b)

Figure 10.6 Left: family tree, circles are females, squares are males. Individuals with the disease of interest are highlighted. Right: DGM for two loci. Blue nodes X_{ij} is the observed phenotype for individual i at locus j. All other nodes are hidden. Orange nodes $G_{ij}^{p/m}$ is the paternal/ maternal allele. Small red nodes $z_{ijl}^{p/m}$ are the paternal/ maternal selection switching variables. These are linked across loci, $z_{ij}^{m} \rightarrow z_{i,j+1}^{m}$ and $z_{ij}^{p} \rightarrow z_{i,j+1}^{p}$. The founder (root) nodes do not have any parents, and hence do no need switching variables. Based on Figure 3 from (Friedman et al. 2000).

In more detail, for each person (or animal) i and location or locus j along the genome, we create three nodes: the observed **marker** X_{ij} (which can be a property such as blood type, or just a fragment of DNA that can be measured), and two hidden **alleles**, G^m_{ij} and G^p_{ij}, one inherited from i's mother (maternal allele) and the other from i's father (paternal allele). Together, the ordered pair $\mathbf{G}_{ij} = (G^m_{ij}, G^p_{ij})$ constitutes i's hidden **genotype** at locus j.

Obviously we must add $G^m_{ij} \to X_{ij}$ and $G^p_{ij} \to X_{ij}$ arcs representing the fact that genotypes cause phenotypes (observed manifestations of genotypes). The CPD $p(X_{ij}|G^m_{ij}, G^p_{ij})$ is called the **penetrance model**. As a very simple example, suppose $X_{ij} \in \{A, B, O, AB\}$ represents person i's observed bloodtype, and $G^m_{ij}, G^p_{ij} \in \{A, B, O\}$ is their genotype. We can represent the penetrance model using the deterministic CPD shown in Table 10.2. For example, A dominates O, so if a person has genotype AO or OA, their phenotype will be A.

In addition, we add arcs from i's mother and father into $\mathbf{G}_{ij}$, reflecting the **Mendelian inheritance** of genetic material from one's parents. More precisely, let $m_i = k$ be i's mother. Then G^m_{ij} could either be equal to G^m_{kj} or G^p_{kj}, that is, i's maternal allele is a copy of one of its mother's two alleles. Let Z^m_{ij} be a hidden variable than specifies the choice. We can model this using the following CPD, known as the **inheritance model**:

$$p(G^m_{ij}|G^m_{kj}, G^p_{kj}, Z^m_{ij}) = \begin{cases} \mathbb{I}(G^m_{ij} = G^m_{kj}) & \text{if } Z^m_{ij} = m \\ \mathbb{I}(G^m_{ij} = G^p_{kj}) & \text{if } Z^m_{ij} = p \end{cases} \tag{10.13}$$

We can define $p(G^p_{ij}|G^m_{kj}, G^p_{kj}, Z^p_{ij})$ similarly, where $k = p_i$ is i's father. The values of the Z_{ij} are said to specify the **phase** of the genotype. The values of $G^p_{i,j}$, $G^m_{i,j}$, $Z^p_{i,j}$ and $Z^m_{i,j}$ constitute the **haplotype** of person i at locus j.[1]

Next, we need to specify the prior for the root nodes, $p(G^m_{ij})$ and $p(G^p_{ij})$. This is called the **founder model**, and represents the overall prevalence of difference kinds of alleles in the population. We usually assume independence between the loci for these founder alleles.

Finally, we need to specify priors for the switch variables that control the inheritance process. These variables are spatially correlated, since adjacent sites on the genome are typically inherited together (recombination events are rare). We can model this by imposing a two-state Markov chain on the Z's, where the probability of switching state at locus j is given by $\theta_j = \frac{1}{2}(1 - e^{-2d_j})$, where d_j is the distance between loci j and $j + 1$. This is called the **recombination model**.

The resulting DGM is shown in Figure 10.6(b): it is a series of replicated pedigree DAGs, augmented with switching Z variables, which are linked using Markov chains. (There is a related model known as **phylogenetic HMM** (Siepel and Haussler 2003), which is used to model evolution amongst phylogenies.)

As a simplified example of how this model can be used, suppose we only have one locus, corresponding to blood type. For brevity, we will drop the j index. Suppose we observe $x_i = A$. Then there are 3 possible genotypes: $\mathbf{G}_i$ is (A, A), (A, O) or (O, A). There is ambiguity because the genotype to phenotype mapping is many-to-one. We want to reverse this mapping. This is known as an **inverse problem**. Fortunately, we can use the blood types of relatives to help disambiguate the evidence. Information will "flow" from the other $x_{i'}$'s up to their $\mathbf{G}_{i'}$'s, then across to i's $\mathbf{G}_i$ via the pedigree DAG. Thus we can combine our **local evidence** $p(x_i|\mathbf{G}_i)$

1. Sometimes the observed marker is equal to the unphased genotype, which is the unordered set $\{G^p_{ij}, G^m_{ij}\}$; however, the phased or hidden genotype is not directly measurable.

with an informative prior, $p(\mathbf{G}_i|\mathbf{x}_{-i})$, conditioned on the other data, to get a less entropic local posterior, $p(G_i|\mathbf{x}) \propto p(x_i|G_i)p(G_i|\mathbf{x}_{-i})$.

In practice, the model is used to try to determine where along the genome a given disease-causing gene is assumed to lie — this is the genetic linkage analysis task. The method works as follows. First, suppose all the parameters of the model, including the distance between all the marker loci, are known. The only unknown is the location of the disease-causing gene. If there are L marker loci, we construct $L+1$ models: in model ℓ, we postulate that the disease gene comes after marker ℓ, for $0 < \ell < L+1$. We can estimate the Markov switching parameter $\hat{\theta}_\ell$, and hence the distance d_ℓ between the disease gene and its nearest known locus. We measure the quality of that model using its likelihood, $p(\mathcal{D}|\hat{\theta}_\ell)$. We then can then pick the model with highest likelihood (which is equivalent to the MAP model under a uniform prior).

Note, however, that computing the likelihood requires marginalizing out all the hidden Z and G variables. See (Fishelson and Geiger 2002) and the references therein for some exact methods for this task; these are based on the variable elimination algorithm, which we discuss in Section 20.3. Unfortunately, for reasons we explain in Section 20.5, exact methods can be computationally intractable if the number of individuals and/or loci is large. See (Albers et al. 2006) for an approximate method for computing the likelihood; this is based on a form of variational inference, which we will discuss in Section 22.4.1.

10.2.5 Directed Gaussian graphical models *

Consider a DGM where all the variables are real-valued, and all the CPDs have the following form:

$$p(x_t|\mathbf{x}_{\mathrm{pa}(t)}) = \mathcal{N}(x_t|\mu_t + \mathbf{w}_t^T \mathbf{x}_{\mathrm{pa}(t)}, \sigma_t^2) \tag{10.14}$$

This is called a **linear Gaussian** CPD. As we show below, multiplying all these CPDs together results in a large joint Gaussian distribution of the form $p(\mathbf{x}) = \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma})$. This is called a directed GGM, or a **Gaussian Bayes net**.

We now explain how to derive $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ from the CPD parameters, following (Shachter and Kenley 1989, App. B). For convenience, we will rewrite the CPDs in the following form:

$$x_t = \mu_t + \sum_{s \in \mathrm{pa}(t)} w_{ts}(x_s - \mu_s) + \sigma_t z_t \tag{10.15}$$

where $z_t \sim \mathcal{N}(0,1)$, σ_t is the conditional standard deviation of x_t given its parents, w_{ts} is the strength of the $s \to t$ edge, and μ_t is the local mean.[2]

It is easy to see that the global mean is just the concatenation of the local means, $\boldsymbol{\mu} = (\mu_1, \ldots, \mu_D)$. We now derive the global covariance, $\boldsymbol{\Sigma}$. Let $\mathbf{S} \triangleq \mathrm{diag}(\boldsymbol{\sigma})$ be a diagonal matrix containing the standard deviations. We can rewrite Equation 10.15 in matrix-vector form as follows:

$$(\mathbf{x} - \boldsymbol{\mu}) = \mathbf{W}(\mathbf{x} - \boldsymbol{\mu}) + \mathbf{S}\mathbf{z} \tag{10.16}$$

2. If we do not subtract off the parent's mean (i.e., if we use $x_t = \mu_t + \sum_{s \in \mathrm{pa}(t)} w_{ts} x_s + \sigma_t z_t$), the derivation of $\boldsymbol{\Sigma}$ is much messier, as can be seen by looking at (Bishop 2006b, p370).

Now let $\mathbf{e}$ be a vector of noise terms:

$$\mathbf{e} \triangleq \mathbf{S}\mathbf{z} \tag{10.17}$$

We can rearrange this to get

$$\mathbf{e} = (\mathbf{I} - \mathbf{W})(\mathbf{x} - \boldsymbol{\mu}) \tag{10.18}$$

Since $\mathbf{W}$ is lower triangular (because $w_{ts} = 0$ if $t > s$ in the topological ordering), we have that $\mathbf{I} - \mathbf{W}$ is lower triangular with 1s on the diagonal. Hence

$$\begin{pmatrix} e_1 \\ e_2 \\ \vdots \\ e_d \end{pmatrix} = \begin{pmatrix} 1 & & & & \\ -w_{21} & 1 & & & \\ -w_{32} & -w_{31} & 1 & & \\ \vdots & & & \ddots & \\ -w_{d1} & -w_{d2} & \dots & -w_{d,d-1} & 1 \end{pmatrix} \begin{pmatrix} x_1 - \mu_1 \\ x_2 - \mu_2 \\ \vdots \\ x_d - \mu_d \end{pmatrix} \tag{10.19}$$

Since $\mathbf{I} - \mathbf{W}$ is always invertible, we can write

$$\mathbf{x} - \boldsymbol{\mu} = (\mathbf{I} - \mathbf{W})^{-1}\mathbf{e} \triangleq \mathbf{U}\mathbf{e} = \mathbf{U}\mathbf{S}\mathbf{z} \tag{10.20}$$

where we defined $\mathbf{U} = (\mathbf{I} - \mathbf{W})^{-1}$. Thus the regression weights correspond to a Cholesky decomposition of $\boldsymbol{\Sigma}$, as we now show:

$$\boldsymbol{\Sigma} = \text{cov}[\mathbf{x}] = \text{cov}[\mathbf{x} - \boldsymbol{\mu}] \tag{10.21}$$

$$= \text{cov}[\mathbf{U}\mathbf{S}\mathbf{z}] = \mathbf{U}\mathbf{S}\,\text{cov}[\mathbf{z}]\;\mathbf{S}\mathbf{U}^T = \mathbf{U}\mathbf{S}^2\mathbf{U}^T \tag{10.22}$$

10.3 Inference

We have seen that graphical models provide a compact way to define joint probability distributions. Given such a joint distribution, what can we do with it? The main use for such a joint distribution is to perform **probabilistic inference**. This refers to the task of estimating unknown quantities from known quantities. For example, in Section 10.2.2, we introduced HMMs, and said that one of the goals is to estimate the hidden states (e.g., words) from the observations (e.g., speech signal). And in Section 10.2.4, we discussed genetic linkage analysis, and said that one of the goals is to estimate the likelihood of the data under various DAGs, corresponding to different hypotheses about the location of the disease-causing gene.

In general, we can pose the inference problem as follows. Suppose we have a set of correlated random variables with joint distribution $p(\mathbf{x}_{1:V}|\boldsymbol{\theta})$. (In this section, we are assuming the parameters $\boldsymbol{\theta}$ of the model are known. We discuss how to learn the parameters in Section 10.4.) Let us partition this vector into the **visible variables** $\mathbf{x}_v$, which are observed, and the **hidden variables**, $\mathbf{x}_h$, which are unobserved. Inference refers to computing the posterior distribution of the unknowns given the knowns:

$$p(\mathbf{x}_h|\mathbf{x}_v, \boldsymbol{\theta}) = \frac{p(\mathbf{x}_h, \mathbf{x}_v|\boldsymbol{\theta})}{p(\mathbf{x}_v|\boldsymbol{\theta})} = \frac{p(\mathbf{x}_h, \mathbf{x}_v|\boldsymbol{\theta})}{\sum_{\mathbf{x}'_h} p(\mathbf{x}'_h, \mathbf{x}_v|\boldsymbol{\theta})} \tag{10.23}$$

Essentially we are **conditioning** on the data by **clamping** the visible variables to their observed values, $\mathbf{x}_v$, and then normalizing, to go from $p(\mathbf{x}_h, \mathbf{x}_v)$ to $p(\mathbf{x}_h|\mathbf{x}_v)$. The normalization constant $p(\mathbf{x}_v|\boldsymbol{\theta})$ is the likelihood of the data, also called the **probability of the evidence**.

Sometimes only some of the hidden variables are of interest to us. So let us partition the hidden variables into **query variables**, $\mathbf{x}_q$, whose value we wish to know, and the remaining **nuisance variables**, $\mathbf{x}_n$, which we are not interested in. We can compute what we are interested in by **marginalizing out** the nuisance variables:

$$p(\mathbf{x}_q|\mathbf{x}_v, \boldsymbol{\theta}) = \sum_{\mathbf{x}_n} p(\mathbf{x}_q, \mathbf{x}_n|\mathbf{x}_v, \boldsymbol{\theta}) \tag{10.24}$$

In Section 4.3.1, we saw how to perform all these operations for a multivariate Gaussian in $O(V^3)$ time, where V is the number of variables. What if we have discrete random variables, with say K states each? If the joint distribution is represented as a multi-dimensional table, we can always perform these operations exactly, but this will take $O(K^V)$ time. In Chapter 20, we explain how to exploit the factorization encoded by the GM to perform these operations in $O(VK^{w+1})$ time, where w is a quantity known as the treewidth of the graph. This measures how "tree-like" the graph is. If the graph is a tree (or a chain), we have $w = 1$, so for these models, inference takes time linear in the number of nodes. Unfortunately, for more general graphs, exact inference can take time exponential in the number of nodes, as we explain in Section 20.5. We will therefore examine various approximate inference schemes later in the book.

10.4 Learning

In the graphical models literature, it is common to distinguish between inference and learning. Inference means computing (functions of) $p(\mathbf{x}_h|\mathbf{x}_v, \boldsymbol{\theta})$, where v are the visible nodes, h are the hidden nodes, and $\boldsymbol{\theta}$ are the parameters of the model, assumed to be known. Learning usually means computing a MAP estimate of the parameters given data:

$$\hat{\boldsymbol{\theta}} = \operatorname*{argmax}_{\boldsymbol{\theta}} \sum_{i=1}^{N} \log p(\mathbf{x}_{i,v}|\boldsymbol{\theta}) + \log p(\boldsymbol{\theta}) \tag{10.25}$$

where $\mathbf{x}_{i,v}$ are the visible variables in case i. If we have a uniform prior, $p(\boldsymbol{\theta}) \propto 1$, this reduces to the MLE, as usual.

If we adopt a Bayesian view, the parameters are unknown variables and should also be inferred. Thus to a Bayesian, there is no distinction between inference and learning. In fact, we can just add the parameters as nodes to the graph, condition on $\mathcal{D}$, and then infer the values of all the nodes. (We discuss this in more detail below.)

In this view, the main difference between hidden variables and parameters is that the number of hidden variables grows with the amount of training data (since there is usually a set of hidden variables for each observed data case), whereas the number of parameters in usually fixed (at least in a parametric model). This means that we must integrate out the hidden variables to avoid overfitting, but we may be able to get away with point estimation techniques for parameters, which are fewer in number.

10.4.1 Plate notation

When inferring parameters from data, we often assume the data is iid. We can represent this assumption explicitly using a graphical model, as shown in Figure 10.7(a). This illustrates the

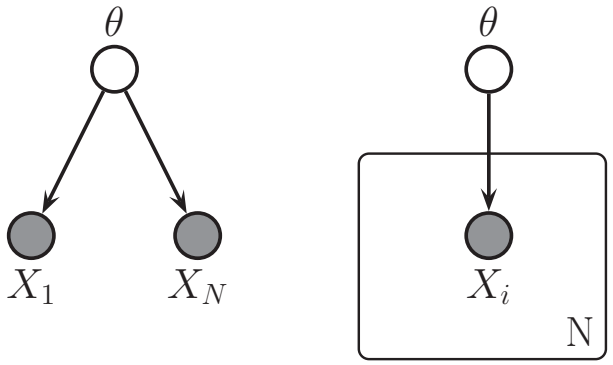


Figure 10.7 Left: data points x_i are conditionally independent given θ. Right: Plate notation. This represents the same model as the one on the left, except the repeated x_i nodes are inside a box, known as a plate; the number in the lower right hand corner, N, specifies the number of repetitions of the X_i node.

assumption that each data case was generated independently but from the same distribution. Notice that the data cases are only independent conditional on the parameters $\boldsymbol{\theta}$; marginally, the data cases are dependent. Nevertheless, we can see that, in this example, the order in which the data cases arrive makes no difference to our beliefs about $\boldsymbol{\theta}$, since all orderings will have the same sufficient statistics. Hence we say the data is **exchangeable**.

To avoid visual clutter, it is common to use a form of **syntactic sugar** called **plates**: we simply draw a little box around the repeated variables, with the convention that nodes within the box will get repeated when the model is **unrolled**. We often write the number of copies or repetitions in the bottom right corner of the box. See Figure 10.7(b) for a simple example. The corresponding joint distribution has the form

$$p(\boldsymbol{\theta}, \mathcal{D}) = p(\boldsymbol{\theta}) \left[\prod_{i=1}^{N} p(\mathbf{x}_i | \boldsymbol{\theta}) \right] \tag{10.26}$$

This DGM represents the CI assumptions behind the models we considered in Chapter 5.

A slightly more complex example is shown in Figure 10.8. On the left we show a naive Bayes classifier that has been "unrolled" for D features, but uses a plate to represent repetition over cases $i = 1 : N$. The version on the right shows the same model using **nested plate** notation. When a variable is inside two plates, it will have two sub-indices. For example, we write θ_{jc} to represent the parameter for feature j in class-conditional density c. Note that plates can be nested or crossing. Notational devices for modeling more complex parameter tying patterns can be devised (e.g., (Heckerman et al. 2004)), but these are not widely used. What is not clear from the figure is that θ_{jc} is used to generate x_{ij} iff $y_i = c$, otherwise it is ignored. This is an example of **context specific independence**, since the CI relationship $x_{ij} \perp \theta_{jc}$ only holds if $y_i \neq c$.

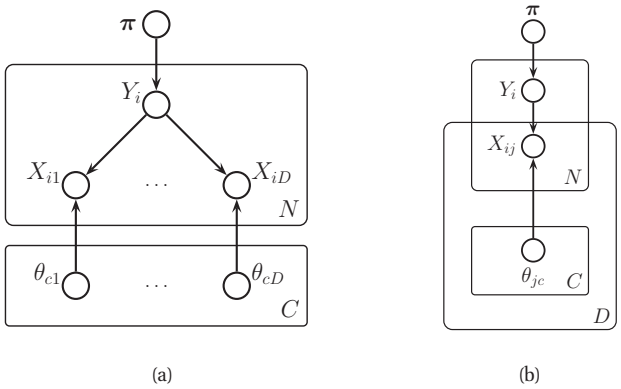


Figure 10.8 Naive Bayes classifier as a DGM. (a) With single plates. (b) WIth nested plates.

10.4.2 Learning from complete data

If all the variables are fully observed in each case, so there is no missing data and there are no hidden variables, we say the data is **complete**. For a DGM with complete data, the likelihood is given by

$$p(\mathcal{D}|\boldsymbol{\theta}) = \prod_{i=1}^{N} p(\mathbf{x}_i|\boldsymbol{\theta}) = \prod_{i=1}^{N}\prod_{t=1}^{V} p(x_{it}|\mathbf{x}_{i,\mathrm{pa}(t)}, \boldsymbol{\theta}_t) = \prod_{t=1}^{V} p(\mathcal{D}_t|\boldsymbol{\theta}_t) \tag{10.27}$$

where $\mathcal{D}_t$ is the data associated with node t and its parents, i.e., the t'th family. This is a product of terms, one per CPD. We say that the likelihood **decomposes** according to the graph structure.

Now suppose that the prior factorizes as well:

$$p(\boldsymbol{\theta}) = \prod_{t=1}^{V} p(\boldsymbol{\theta}_t) \tag{10.28}$$

Then clearly the posterior also factorizes:

$$p(\boldsymbol{\theta}|\mathcal{D}) \propto p(\mathcal{D}|\boldsymbol{\theta})p(\boldsymbol{\theta}) = \prod_{t=1}^{V} p(\mathcal{D}_t|\boldsymbol{\theta}_t)p(\boldsymbol{\theta}_t) \tag{10.29}$$

This means we can compute the posterior of each CPD independently. In other words,

$$\text{factored prior plus factored likelihood implies factored posterior} \tag{10.30}$$

Let us consider an example, where all CPDs are tabular, thus extending the earlier results of Secion 3.5.1.2, where discussed Bayesian naive Bayes. We have a separate row (i.e., a separate multinoulli distribution) for each **conditioning case**, i.e., for each combination of parent values, as in Table 10.2. Formally, we can write the t'th CPT as $x_t|\mathbf{x}_{\mathrm{pa}(t)} = c \sim \mathrm{Cat}(\boldsymbol{\theta}_{tc})$, where $\theta_{tck} \triangleq p(x_t = k|\mathbf{x}_{\mathrm{pa}(t)} = c)$, for $k = 1 : K_t$, $c = 1 : C_t$ and $t = 1 : T$. Here K_t is the number

of states for node t, $C_t \triangleq \prod_{s \in \text{pa}(t)} K_s$ is the number of parent combinations, and T is the number of nodes. Obviously $\sum_k \theta_{tck} = 1$ for each row of each CPT.

Let us put a separate Dirichlet prior on each row of each CPT, i.e., $\boldsymbol{\theta}_{tc} \sim \text{Dir}(\boldsymbol{\alpha}_{tc})$. Then we can compute the posterior by simply adding the pseudo counts to the empirical counts to get $\boldsymbol{\theta}_{tc}|\mathcal{D} \sim \text{Dir}(\mathbf{N}_{tc} + \boldsymbol{\alpha}_{tc})$, where N_{tck} is the number of times that node t is in state k while its parents are in state c:

$$N_{tck} \triangleq \sum_{i=1}^{N} \mathbb{I}(x_{i,t} = k, x_{i,\text{pa}(t)} = c) \tag{10.31}$$

From Equation 2.77, the mean of this distribution is given by the following:

$$\overline{\theta}_{tck} = \frac{N_{tck} + \alpha_{tck}}{\sum_{k'}(N_{tck'} + \alpha_{tck'})} \tag{10.32}$$

For example, consider the DGM in Figure 10.1(a). Suppose the training data consists of the following 5 cases:

x_1	x_2	x_3	x_4	x_5
0	0	1	0	0
0	1	1	1	1
1	1	0	1	0
0	1	1	0	0
0	1	1	1	0

Below we list all the sufficient statistics N_{tck}, and the posterior mean parameters $\overline{\theta}_{ick}$ under a Dirichlet prior with $\alpha_{ick} = 1$ (corresponding to add-one smoothing) for the $t = 4$ node:

x_2	x_3	$N_{tck=1}$	$N_{tck=0}$	$\overline{\theta}_{tck=1}$	$\overline{\theta}_{tck=0}$
0	0	0	0	1/2	1/2
1	0	1	0	2/3	1/3
0	1	0	1	1/3	2/3
1	1	2	1	3/5	2/5

It is easy to show that the MLE has the same form as Equation 10.32, except without the α_{tck} terms, i.e.,

$$\hat{\theta}_{tck} = \frac{N_{tck}}{\sum_{k'} N_{tck'}} \tag{10.33}$$

Of course, the MLE suffers from the zero-count problem discussed in Section 3.3.4.1, so it is important to use a prior to regularize the estimation problem.

10.4.3 Learning with missing and/or latent variables

If we have missing data and/or hidden variables, the likelihood no longer factorizes, and indeed it is no longer convex, as we explain in detail in Section 11.3. This means we will usually can only compute a locally optimal ML or MAP estimate. Bayesian inference of the parameters is even harder. We discuss suitable approximate inference techniques in later chapters.

10.5 Conditional independence properties of DGMs

At the heart of any graphical model is a set of conditional indepence (CI) assumptions. We write $\mathbf{x}_A \perp_G \mathbf{x}_B | \mathbf{x}_C$ if A is independent of B given C in the graph G, using the semantics to be defined below. Let $I(G)$ be the set of all such CI statements encoded by the graph.

We say that G is an **I-map** (independence map) for p, or that p is **Markov** wrt G, iff $I(G) \subseteq I(p)$, where $I(p)$ is the set of all CI statements that hold for distribution p. In other words, the graph is an I-map if it does not make any assertions of CI that are not true of the distribution. This allows us to use the graph as a safe proxy for p when reasoning about p's CI properties. This is helpful for designing algorithms that work for large classes of distributions, regardless of their specific numerical parameters $\boldsymbol{\theta}$.

Note that the fully connected graph is an I-map of all distributions, since it makes no CI assertions at all (since it is not missing any edges). We therefore say G is a **minimal I-map** of p if G is an I-map of p, and if there is no $G' \subseteq G$ which is an I-map of p.

It remains to specify how to determine if $\mathbf{x}_A \perp_G \mathbf{x}_B | \mathbf{x}_C$. Deriving these independencies for undirected graphs is easy (see Section 19.2), but the DAG situation is somewhat complicated, because of the need to respect the orientation of the directed edges. We give the details below.

10.5.1 d-separation and the Bayes Ball algorithm (global Markov properties)

First, we introduce some definitions. We say an *undirected path* P is **d-separated** by a set of nodes E (containing the evidence) iff at least one of the following conditions hold:

1. P contains a chain, $s \to m \to t$ or $s \leftarrow m \leftarrow t$, where $m \in E$
2. P contains a tent or fork, $s \swarrow^{m} \searrow t$, where $m \in E$
3. P contains a **collider** or **v-structure**, $s \searrow_{m} \swarrow t$, where m is not in E and nor is any descendant of m.

Next, we say that a *set of nodes* A is d-separated from a different set of nodes B given a third observed set E iff each undirected path from every node $a \in A$ to every node $b \in B$ is d-separated by E. Finally, we define the CI properties of a DAG as follows:

$$\mathbf{x}_A \perp_G \mathbf{x}_B | \mathbf{x}_E \iff \text{A is d-separated from B given E} \tag{10.34}$$

The **Bayes ball algorithm** (Shachter 1998) is a simple way to see if A is d-separated from B given E, based on the above definition. The idea is this. We "shade" all nodes in E, indicating that they are observed. We then place "balls" at each node in A, let them "bounce around" according to some rules, and then ask if any of the balls reach any of the nodes in B. The three main rules are shown in Figure 10.9. Notice that balls can travel opposite to edge directions. We see that a ball can pass through a chain, but not if it is shaded in the middle. Similarly, a ball can pass through a fork, but not if it is shaded in the middle. However, a ball cannot pass through a v-structure, unless it is shaded in the middle.

We can justify the 3 rules of Bayes ball as follows. First consider a chain structure $X \to Y \to Z$, which encodes

$$p(x, y, z) = p(x)p(y|x)p(z|y) \tag{10.35}$$

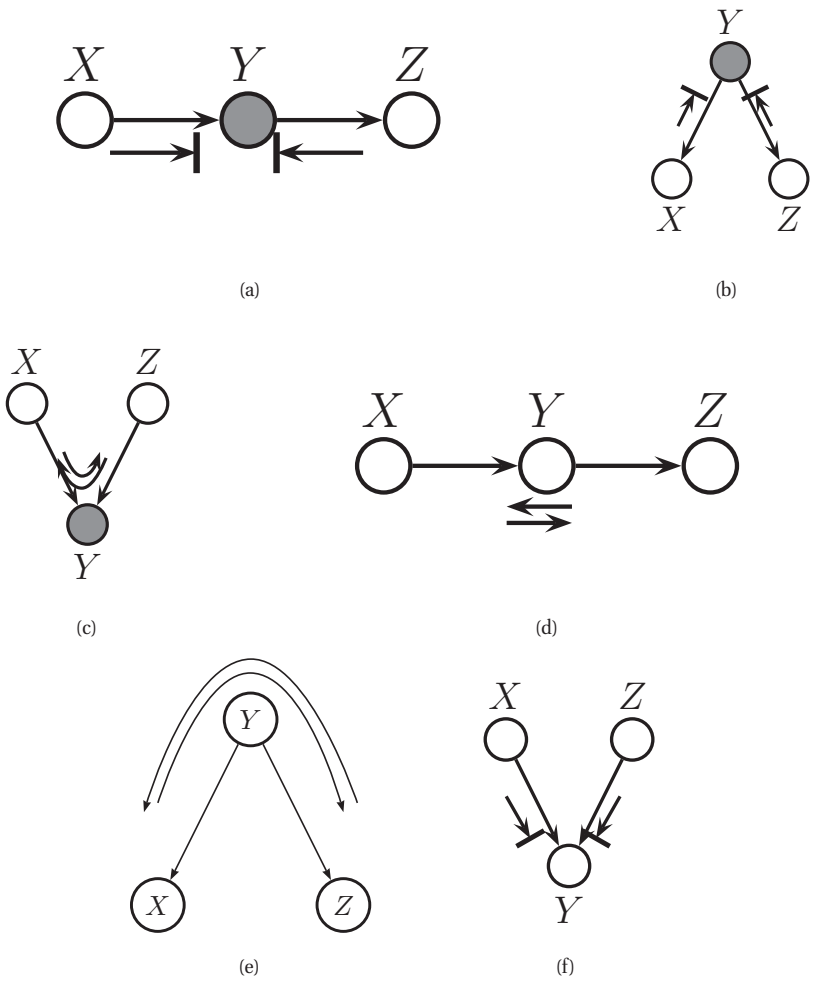


Figure 10.9 Bayes ball rules. A shaded node is one we condition on. If there is an arrow hitting a bar, it means the ball cannot pass through; otherwise the ball can pass through. Based on (Jordan 2007).

When we condition on y, are x and z independent? We have

$$p(x, z|y) = \frac{p(x)p(y|x)p(z|y)}{p(y)} = \frac{p(x, y)p(z|y)}{p(y)} = p(x|y)p(z|y) \tag{10.36}$$

and therefore $x \perp z|y$. So observing the middle node of chain breaks it in two (as in a Markov chain).

Now consider the tent structure $X \leftarrow Y \rightarrow Z$. The joint is

$$p(x, y, z) = p(y)p(x|y)p(z|y) \tag{10.37}$$

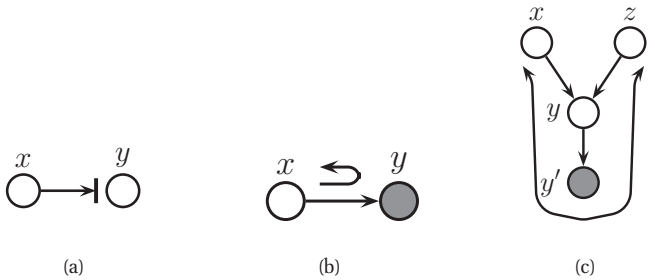


Figure 10.10 (a-b) Bayes ball boundary conditions. (c) Example of why we need boundary conditions. y' is an observed child of y, rendering y "effectively observed", so the ball bounces back up on its way from x to z.

When we condition on y, are x and z independent? We have

$$p(x, z|y) = \frac{p(x, y, z)}{p(y)} = \frac{p(y)p(x|y)p(z|y)}{p(y)} = p(x|y)p(z|y) \tag{10.38}$$

and therefore $x \perp z|y$. So observing a root node separates its children (as in a naive Bayes classifier: see Section 3.5).

Finally consider a v-structure $X \rightarrow Y \leftarrow Z$. The joint is

$$p(x, y, z) = p(x)p(z)p(y|x, z) \tag{10.39}$$

When we condition on y, are x and z independent? We have

$$p(x, z|y) = \frac{p(x)p(z)p(y|x, z)}{p(y)} \tag{10.40}$$

so $x \not\perp z|y$. However, in the unconditional distribution, we have

$$p(x, z) = p(x)p(z) \tag{10.41}$$

so we see that x and z are marginally independent. So we see that conditioning on a common child at the bottom of a v-structure makes its parents become dependent. This important effect is called **explaining away**, **inter-causal reasoning**, or **Berkson's paradox**. As an example of explaining away, suppose we toss two coins, representing the binary numbers 0 and 1, and we observe the "sum" of their values. A priori, the coins are independent, but once we observe their sum, they become coupled (e.g., if the sum is 1, and the first coin is 0, then we know the second coin is 1).

Finally, Bayes Ball also needs the "boundary conditions" shown in Figure 10.10(a-b). To understand where these rules come from, consider Figure 10.10(c). Suppose Y' is a noise-free copy of Y. Then if we observe Y', we effectively observe Y as well, so the parents X and Z have to compete to explain this. So if we send a ball down $X \rightarrow Y \rightarrow Y'$, it should "bounce back" up along $Y' \rightarrow Y \rightarrow Z$. However, if Y and all its children are hidden, the ball does not bounce back.

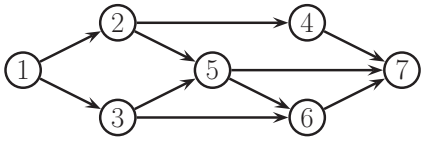


Figure 10.11 A DGM.

For example, in Figure 10.11, we see that $x_2 \perp x_6|x_5$, since the $2 \to 5 \to 6$ path is blocked by x_5 (which is observed), the $2 \to 4 \to 7 \to 6$ path is blocked by x_7 (which is hidden), and the $2 \to 1 \to 3 \to 6$ path is blocked by x_1 (which is hidden). However, we also see that $x_2 \not\perp x_6|x_5, x_7$, since now the $2 \to 4 \to 7 \to 6$ path is no longer blocked by x_7 (which is observed). Exercise 10.2 gives you some more practice in determining CI relationships for DGMs.

10.5.2 Other Markov properties of DGMs

From the d-separation criterion, one can conclude that

$$t \perp \mathrm{nd}(t) \setminus \mathrm{pa}(t)|\mathrm{pa}(t) \tag{10.42}$$

where the **non-descendants** of a node $\mathrm{nd}(t)$ are all the nodes except for its descendants, $\mathrm{nd}(t) = \mathcal{V} \setminus \{t \cup \mathrm{desc}(t)\}$. Equation 10.42 is called the **directed local Markov property**. For example, in Figure 10.11, we have $\mathrm{nd}(3) = \{2, 4\}$, and $\mathrm{pa}(3) = 1$, so $3 \perp 2, 4|1$.

A special case of this property is when we only look at predecessors of a node according to some topological ordering. We have

$$t \perp \mathrm{pred}(t) \setminus \mathrm{pa}(t)|\mathrm{pa}(t) \tag{10.43}$$

which follows since $\mathrm{pred}(t) \subseteq \mathrm{nd}(t)$. This is called the **ordered Markov property**, which justifies Equation 10.7. For example, in Figure 10.11, if we use the ordering $1, 2, \ldots, 7$. we find $\mathrm{pred}(3) = \{1, 2\}$ and $\mathrm{pa}(3) = 1$, so $3 \perp 2|1$.

We have now described three Markov properties for DAGs: the directed global Markov property G in Equation 10.34, the ordered Markov property O in Equation 10.43, and the directed local Markov property L in Equation 10.42. It is obvious that $G \implies L \implies O$. What is less obvious, but nevertheless true, is that $O \implies L \implies G$ (see e.g., (Koller and Friedman 2009) for the proof). Hence all these properties are equivalent.

Furthermore, any distribution p that is Markov wrt G can be factorized as in Equation 10.7; this is called the factorization property F. It is obvious that $O \implies F$, but one can show that the converse also holds (see e.g., (Koller and Friedman 2009) for the proof).

10.5.3 Markov blanket and full conditionals

The set of nodes that renders a node t conditionally independent of all the other nodes in the graph is called t's **Markov blanket**; we will denote this by $\mathrm{mb}(t)$. One can show that the Markov blanket of a node in a DGM is equal to the parents, the children, and the **co-parents**,

i.e., other nodes who are also parents of its children:

$$\mathrm{mb}(t) \triangleq \mathrm{ch}(t) \cup \mathrm{pa}(t) \cup \mathrm{copa}(t) \tag{10.44}$$

For example, in Figure 10.11, we have

$$\mathrm{mb}(5) = \{6,7\} \cup \{2,3\} \cup \{4\} = \{2,3,4,6,7\} \tag{10.45}$$

where 4 is a co-parent of 5 because they share a common child, namely 7.

To see why the co-parents are in the Markov blanket, note that when we derive $p(x_t|\mathbf{x}_{-t}) = p(x_t, \mathbf{x}_{-t})/p(\mathbf{x}_{-t})$, all the terms that do not involve x_t will cancel out between numerator and denominator, so we are left with a product of CPDs which contain x_t in their **scope**. Hence

$$p(x_t|\mathbf{x}_{-t}) \quad \propto \quad p(x_t|\mathbf{x}_{\mathrm{pa}(t)}) \prod_{s \in \mathrm{ch}(t)} p(x_s|\mathbf{x}_{\mathrm{pa}(s)}) \tag{10.46}$$

For example, in Figure 10.11 we have

$$p(x_5|\mathbf{x}_{-5}) \propto p(x_5|x_2,x_3)p(x_6|x_3,x_5)p(x_7|x_4,x_5,x_6) \tag{10.47}$$

The resulting expression is called t's **full conditional**, and will prove to be important when we study Gibbs sampling (Section 24.2).

10.6 Influence (decision) diagrams *

We can represent multi-stage (Bayesian) decision problems by using a graphical notation known as a **decision diagram** or an **influence diagram** (Howard and Matheson 1981; Kjaerulff and Madsen 2008). This extends directed graphical models by adding **decision nodes** (also called **action nodes**), represented by rectangles, and **utility nodes** (also called **value nodes**), represented by diamonds. The original random variables are called **chance nodes**, and are represented by ovals, as usual.

Figure 10.12(a) gives a simple example, illustrating the famous **oil wild-catter** problem.[3] In this problem, you have to decide whether to drill an oil well or not. You have two possible actions: $d = 1$ means drill, $d = 0$ means don't drill. You assume there are 3 states of nature: $o = 0$ means the well is dry, $o = 1$ means it is wet (has some oil), and $o = 2$ means it is soaking (has a lot of oil). Suppose your prior beliefs are $p(o) = [0.5, 0.3, 0.2]$. Finally, you must specify the utility function $U(d, o)$. Since the states and actions are discrete, we can represent it as a table (analogous to a CPT in a DGM). Suppose we use the following numbers, in dollars:

	$o=0$	$o=1$	$o=2$
$d=0$	0	0	0
$d=1$	-70	50	200

We see that if you don't drill, you incur no costs, but also make no money. If you drill a dry well, you lose \$70; if you drill a wet well, you gain \$50; and if you drill a soaking well, you gain \$200. Your prior expected utility if you drill is given by

$$EU(d=1) = \sum_{o=0}^{2} p(o)U(d,o) = 0.5 \cdot (-70) + 0.3 \cdot 50 + 0.2 \cdot 200 = 20 \tag{10.48}$$

3. This example is originally from (Raiffa 1968). Our presentation is based on some notes by Daphne Koller.

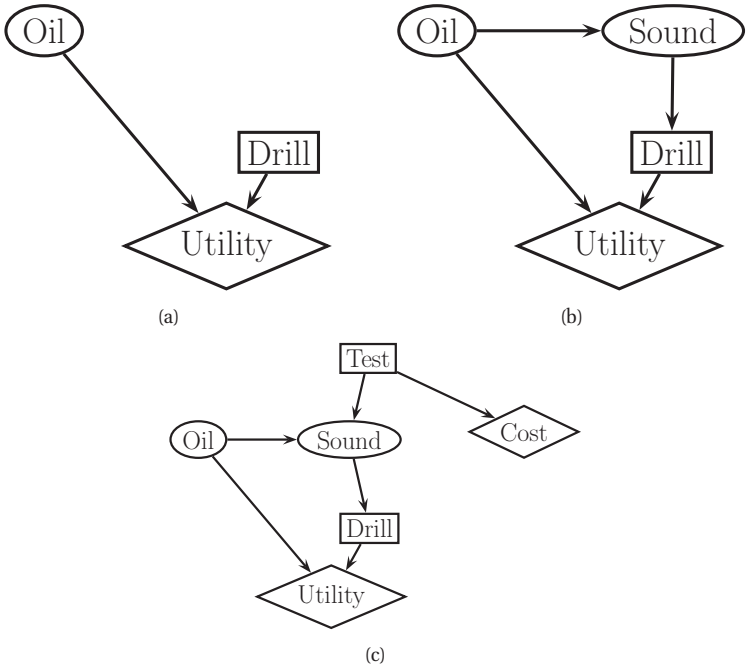


Figure 10.12 (a) Influence diagram for basic oil wild catter problem. (b) An extension in which we have an information arc from the Sound chance node to the Drill decision node. (c) An extension in which we get to decide whether to perform the test or not.

Your expected utility if you don't drill is 0. So your maximum expected utility is

$$MEU = \max\{EU(d=0), EU(d=1)\} = \max\{0, 20\} = 20 \tag{10.49}$$

and therefore the optimal action is to drill:

$$d^* = \arg\max\{EU(d=0), EU(d=1)\} = 1 \tag{10.50}$$

Now let us consider a slight extension to the model. Suppose you perform a sounding to estimate the state of the well. The sounding observation can be in one of 3 states: $s = 0$ is a diffuse reflection pattern, suggesting no oil; $s = 1$ is an open reflection pattern, suggesting some oil; and $s = 2$ is a closed reflection pattern, indicating lots of oil. Since S is caused by O, we add an $O \rightarrow S$ arc to our model. In addition, we assume that the outcome of the sounding test will be available before we decide whether to drill or not; hence we add an **information arc** from S to D. This is illustrated in Figure 10.12(b).

Let us model the reliability of our sensor using the following conditional distribution for $p(s|o)$:

	$s=0$	$s=1$	$s=2$
$o=0$	0.6	0.3	0.1
$o=1$	0.3	0.4	0.3
$o=2$	0.1	0.4	0.5

Suppose we do the sounding test and we observe $s = 0$. The posterior over the oil state is

$$p(o|s=0) = [0.732, 0.219, 0.049] \tag{10.51}$$

Now your posterior expected utility of performing action d is

$$EU(d|s=0) = \sum_{o=0}^{2} p(o|s=0)U(o,d) \tag{10.52}$$

If $d = 1$, this gives

$$EU(d=1|s=0) = 0.732 \times (-70) + 0.219 \times 50 + 0.049 \times 200 = -30.5 \tag{10.53}$$

However, if $d = 0$, then $EU(d = 0|s = 0) = 0$, since not drilling incurs no cost. So if we observe $s = 0$, we are better off not drilling, which makes sense.

Now suppose we do the sounding test and we observe $s = 1$. By similar reasoning, one can show that $EU(d = 1|s = 1) = 32.9$, which is higher than $EU(d = 0|s = 1) = 0$. Similarly, if we observe $s = 2$, we have $EU(d = 1|s = 2) = 87.5$ which is much higher than $EU(d = 0|s = 2) = 0$. Hence the optimal policy $d^*(s)$ is as follows: if $s = 0$, choose $d^*(0) = 0$ and get \$0; if $s = 1$, choose $d^*(1) = 1$ and get \$32.9; and if $s = 2$, choose $d^*(2) = 1$ and get \$87.5.

You can compute your **expected profit** or maximum expected utility as follows:

$$MEU = \sum_s p(s)EU(d^*(s)|s) \tag{10.54}$$

This is the expected utility given possible outcomes of the sounding test, assuming you act optimally given the outcome. The prior marginal on the outcome of the test is

$$p(s) = \sum_o p(o)p(s|o) = [0.41, 0.35, 0.24] \tag{10.55}$$

Hence your maximum expected utility is

$$MEU = 0.41 \times 0 + 0.35 \times 32.9 + 0.24 \times 87.5 = 32.2 \tag{10.56}$$

Now suppose you can choose whether to do the test or not. This can be modelled as shown in Figure 10.12(c), where we add a new test node T. If $T = 1$, we do the test, and S can enter 1 of 3 states, determined by O, exactly as above. If $T = 0$, we don't do the test, and S enters a special unknown state. There is also some cost associated with performing the test.

Is it worth doing the test? This depends on how much our MEU changes if we know the outcome of the test (namely the state of S). If you don't do the test, we have $MEU = 20$ from Equation 10.49. If you do the test, you have $MEU = 32.2$ from Equation 10.56. So the improvement in utility if you do the test (and act optimally on its outcome) is \$12.2. This is

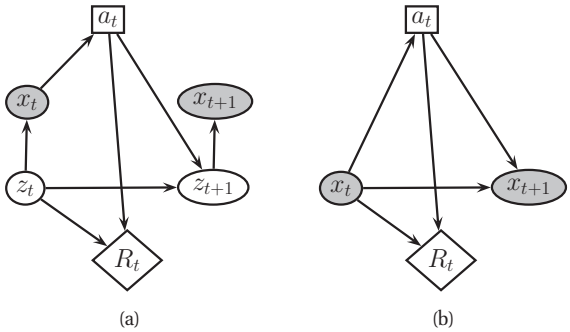


Figure 10.13 (a) A POMDP, shown as an influence diagram. z_t are hidden world states. We implicitly make the **no forgetting** assumption, which effectively means that a_t has arrows coming into it from all previous observations, $x_{1:t}$. (b) An MDP, shown as an influence diagram.

called the **value of perfect information** (VPI). So we should do the test as long as it costs less than $12.2.

In terms of graphical models, the VPI of a variable T can be determined by computing the MEU for the base influence diagram, I, and then computing the MEU for the same influence diagram where we add information arcs from T to the action nodes, and then computing the difference. In other words,

$$\text{VPI} = \text{MEU}(I + T \rightarrow D) - \text{MEU}(I) \tag{10.57}$$

where D is the decision node and T is the variable we are measuring.

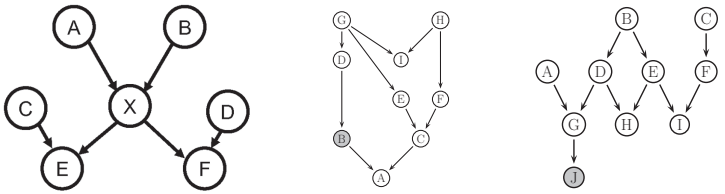
It is possible to modify the variable elimination algorithm (Section 20.3) so that it computes the optimal policy given an influence diagram. These methods essentially work backwards from the final time-step, computing the optimal decision at each step assuming all following actions are chosen optimally. See e.g., (Lauritzen and Nilsson 2001; Kjaerulff and Madsen 2008) for details.

We could continue to extend the model in various ways. For example, we could imagine a dynamical system in which we test, observe outcomes, perform actions, move on to the next oil well, and continue drilling (and polluting) in this way. In fact, many problems in robotics, business, medicine, public policy, etc. can be usefully formulated as influence diagrams unrolled over time (Raiffa 1968; Lauritzen and Nilsson 2001; Kjaerulff and Madsen 2008).

A generic model of this form is shown in Figure 10.13(a). This is known as a **partially observed Markov decision process** or **POMDP** (pronounced "pom-d-p"). This is basically a hidden Markov model (Section 17.3) augmented with action and reward nodes. This can be used to model the **perception-action** cycle that all intelligent agents use (see e.g., (Kaelbling et al. 1998) for details).

A special case of a POMDP, in which the states are fully observed, is called a **Markov decision process** or **MDP**, shown in Figure 10.13(b). This is much easier to solve, since we only have to compute a mapping from observed states to actions. This can be solved using dynamic programming (see e.g., (Sutton and Barto 1998) for details).

In the POMDP case, the information arc from x_t to a_t is not sufficient to uniquely determine



(a) (b) (c)

Figure 10.14 Some DGMs.

the best action, since the state is not fully observed. Instead, we need to choose actions based on our **belief state**, $p(z_t|\mathbf{x}_{1:t}, a_{1:t})$. Since the belief updating process is deterministic (see Section 17.4.2), we can compute a **belief state MDP**. For details on to compute the policies for such models, see e.g., (Kaelbling et al. 1998; Spaan and Vlassis 2005).

Exercises

Exercise 10.1 Marginalizing a node in a DGM

(Source: Koller.)

Consider the DAG G in Figure 10.14(a). Assume it is a minimal I-map for $p(A, B, C, D, E, F, X)$. Now consider marginalizing out X. Construct a new DAG G' which is a minimal I-map for $p(A, B, C, D, E, F)$. Specify (and justify) which extra edges need to be added.

Exercise 10.2 Bayes Ball

(Source: Jordan.)

Here we compute some global independence statements from some directed graphical models. You can use the "Bayes ball" algorithm, the d-separation criterion, or the method of converting to an undirected graph (all should give the same results).

a. Consider the DAG in Figure 10.14(b). List all variables that are independent of A given evidence on B.
b. Consider the DAG in Figure 10.14(c). List all variables that are independent of A given evidence on J.

Exercise 10.3 Markov blanket for a DGM

Prove that the full conditional for node i in a DGM is given by

$$p(X_i|X_{-i}) \quad \propto \quad p(X_i|Pa(X_i)) \prod_{Y_j \in ch(X_i)} p(Y_j|Pa(Y_j)) \tag{10.58}$$

where $ch(X_i)$ are the children of X_i and $Pa(Y_j)$ are the parents of Y_j.

Exercise 10.4 Hidden variables in DGMs

Consider the DGMs in Figure 11.1 which both define $p(X_{1:6})$, where we number empty nodes left to right, top to bottom. The graph on the left defines the joint as

$$p(X_{1:6}) = \sum_h p(X_1)p(X_2)p(X_3)p(H = h|X_{1:3})p(X_4|H = h)p(X_5|H = h)p(X_6|H = h) \tag{10.59}$$

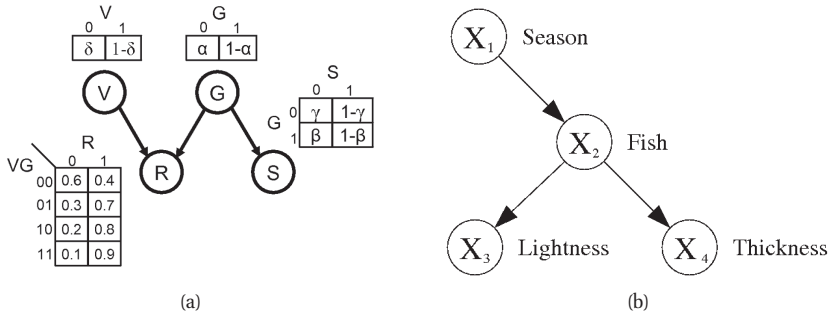


Figure 10.15 (a) Weather BN. (b) Fishing BN.

where we have marginalized over the hidden variable H. The graph on the right defines the joint as

$$p(X_{1:6}) = p(X_1)p(X_2)p(X_3)p(X_4|X_{1:3})p(X_5|X_{1:4})p(X_6|X_{1:5}) \tag{10.60}$$

a. (5 points) Assuming all nodes (including H) are binary and all CPDs are tabular, prove that the model on the left has 17 free parameters.
b. (5 points) Assuming all nodes are binary and all CPDs are tabular, prove that the model on the right has 59 free parameters.
c. (5 points) Suppose we have a data set $\mathcal{D} = X_{1:6}^n$ for $n = 1 : N$, where we observe the Xs but not H, and we want to estimate the parameters of the CPDs using maximum likelihood. For which model is this easier? Explain your answer.

Exercise 10.5 Bayes nets for a rainy day

(Source: Nando de Freitas.). In this question you must model a problem with 4 binary variables: $G =$"gray", $V =$"Vancouver", $R =$"rain" and $S =$"sad". Consider the directed graphical model describing the relationship between these variables shown in Figure 10.15(a).

a. Write down an expression for $P(S = 1|V = 1)$ in terms of $\alpha, \beta, \gamma, \delta$.
b. Write down an expression for $P(S = 1|V = 0)$. Is this the same or different to $P(S = 1|V = 1)$? Explain why.
c. Find maximum likelihood estimates of α, β, γ using the following data set, where each row is a training case. (You may state your answers without proof.)

V	G	R	S
1	1	1	1
1	1	0	1
1	0	0	0

(10.61)

Exercise 10.6 Fishing nets

(Source: (Duda et al. 2001)..) Consider the Bayes net shown in Figure 10.15(b). Here, the nodes represent the following variables

$$X_1 \in \{\text{winter, spring, summer, autumn}\},\ X_2 \in \{\text{salmon, sea bass}\} \tag{10.62}$$

$$X_3 \in \{\text{light, medium, dark}\},\ X_4 \in \{\text{wide, thin}\} \tag{10.63}$$

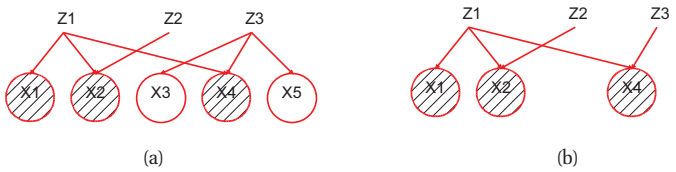


Figure 10.16 (a) A QMR-style network with some hidden leaves. (b) Removing the barren nodes.

The corresponding conditional probability tables are

$$p(x_1) = \begin{pmatrix} .25 & .25 & .25 & .25 \end{pmatrix}, \; p(x_2|x_1) = \begin{pmatrix} .9 & .1 \\ .3 & .7 \\ .4 & .6 \\ .8 & .2 \end{pmatrix} \tag{10.64}$$

$$p(x_3|x_2) = \begin{pmatrix} .33 & .33 & .34 \\ .8 & .1 & .1 \end{pmatrix}, \; p(x_4|x_2) = \begin{pmatrix} .4 & .6 \\ .95 & .05 \end{pmatrix} \tag{10.65}$$

Note that in $p(x_4|x_2)$, the rows represent x_2 and the columns x_4 (so each row sums to one and represents the child of the CPD). Thus $p(x_4 = \text{thin}|x_2 = \text{sea bass}) = 0.05$, $p(x_4 = \text{thin}|x_2 = \text{salmon}) = 0.6$, etc.

Answer the following queries. You may use matlab or do it by hand. In either case, show your work.

a. Suppose the fish was caught on December 20 — the end of autumn and the beginning of winter — and thus let $p(x_1) = (.5,\ 0,\ 0,\ .5)$ instead of the above prior. (This is called **soft evidence**, since we do not know the exact value of X_1, but we have a distribution over it.) Suppose the lightness has not been measured but it is known that the fish is thin. Classify the fish as salmon or sea bass.

b. Suppose all we know is that the fish is thin and medium lightness. What season is it now, most likely? Use $p(x_1) = \begin{pmatrix} .25 & .25 & .25 & .25 \end{pmatrix}$

Exercise 10.7 Removing leaves in BN20 networks

a. Consider the QMR network, where only some of the symtpoms are observed. For example, in Figure 10.16(a), X_4 and X_5 are hidden. Show that we can safely remove all the hidden leaf nodes without affecting the posterior over the disease nodes, i.e., prove that we can compute $p(\mathbf{z}_{1:3}|x_1, x_2, x_4)$ using the network in Figure 10.16(b). This is called barren node removal, and can be applied to any DGM.

b. Now suppose we partition the leaves into three groups: on, off and unknown. Clearly we can remove the unknown leaves, since they are hidden and do not affect their parents. Show that we can analytically remove the leaves that are in the "off state", by absorbing their effect into the prior of the parents. (This trick only works for noisy-OR CPDs.)

Exercise 10.8 Handling negative findings in the QMR network

Consider the QMR network. Let $\mathbf{d}$ be the hidden diseases, $\mathbf{f}^-$ be the negative findings (leaf nodes that are off), and $\mathbf{f}^-$ be the positive findings (leaf nodes that are on). We can compute the posterior $p(\mathbf{d}|\mathbf{f}^-,\mathbf{f}^+)$ in two steps: first absorb the negative findings, $p(\mathbf{d}|\mathbf{f}^-) \propto p(\mathbf{d})p(\mathbf{f}^-|\mathbf{d})$, then absorb the positive findings, $p(\mathbf{d}|\mathbf{f}^-,\mathbf{f}^+) \propto p(\mathbf{d}|\mathbf{f}^-)p(\mathbf{f}^+|\mathbf{d})$. Show that the first step can be done in $O(|\mathbf{d}||\mathbf{f}^-|)$ time, where $|\mathbf{d}|$ is the number of dieases and $|\mathbf{f}^-|$ is the number of negative findings. For simplicity, you can ignore leak nodes. (Intuitively, the reason for this is that there is no correlation induced amongst the parents when the finding is off, since there is no explaining away.)

Exercise 10.9 Moralization does not introduce new independence statements

Recall that the process of moralizing a DAG means connecting together all "unmarried" parents that share a common child, and then dropping all the arrows. Let M be the moralization of DAG G. Show that $CI(M) \subseteq CI(G)$, where CI are the set of conditional independence statements implied by the model.