Chapter 18
Directed Graphical Models

Graphs give a powerful way of representing independence relations and computing conditional probabilities among a set of random variables. In a directed graphical model, the probability of a set of random variables factors into a product of conditional probabilities, one for each node in the graph.

18.1 Introduction

A graphical model is a probabilistic model for which the conditional independence structure is encoded in a graph. In a graphical model, vertices (or nodes) represent random variables, and the edges encode conditional independence relations among the associated vertices. The graph characterizes the way in which the joint distribution factors into the product of many small components, each of which contains only a subset of variables. In this chapter, we introduce *directed graphical models*, in which the edges of the graph have directions (or arrows). An example of a directed graphical model is shown in Figure 18.1. In the next chapter we introduce *undirected graphical models*, in which the edges carry no directional information.

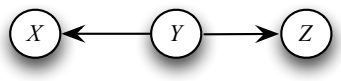


Figure 18.1. *A directed graph with vertices* $V = \{X, Y, Z\}$ *and edges* $E = \{(Y, X), (Y, Z)\}$.

Before getting into details, let us recall the definition of conditional independence.

Let X, Y and Z be random variables. X and Y are *conditionally independent* given Z, written $X \perp\!\!\!\perp Y \mid Z$, if

$$p(x, y|z) = p(x|z)p(y|z). \tag{18.1}$$

for all x, y and z.

Intuitively, this means that, once Z is known, Y provides no extra information about X. An equivalent definition is that $p(x|y, z) = p(x|z)$ for all x, y and z.

Directed graphs are useful for representing conditional independence relations among variables. They can also be used to represent causal relationships. Some people use the phrase *Bayesian network* to refer to a directed graph endowed with a probability distribution. This is a poor choice of terminology. Statistical inference for directed graphs can be performed using either frequentist or Bayesian methods, so it is misleading to call them Bayesian networks.

18.2 Directed Acyclic Graphs (DAGs)

A *directed graph* G consists of a set of vertices V and an edge set E of ordered pairs of vertices. For our purposes, each vertex corresponds to a random variable. If $(Y, X) \in E$ then there is an arrow pointing from Y to X. See Figure 18.1. One thing to note is that a node here can either represents a scalar random variable or a random vector.

If an arrow connects two variables X and Y (in either direction) we say that X and Y are *adjacent.* If there is an arrow from X to Y then X is a *parent* of Y and Y is a *child* of X. The set of all parents of X is denoted by π_X or $\pi(X)$. A *directed path* between two variables is a set of arrows all pointing in the same direction linking one variable to the other such as the chain shown in Figure 18.2:

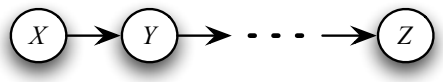


Figure 18.2. *A chain graph with a directed path.*

A sequence of adjacent vertices starting with X and ending with Y but ignoring the direction of the arrows is called an *undirected path.* The sequence $\{X, Y, Z\}$ in Figure 18.1 is an undirected path. X is an *ancestor* of Y if there is a directed path from X to Y (or $X = Y$). We also say that Y is a *descendant* of X.

There are three basic connection configurations of three-node subgraphs and larger graphs can be constructed using these three basic configurations. A configuration of the form as in Figure 18.3(a) is called a *collider* at Y (head-to-head connection). A configuration not of that form is called a *non-collider* (head-to-tail and tail-to-tail connections), for example, Figure 18.3(b) and Figure 18.3(c).

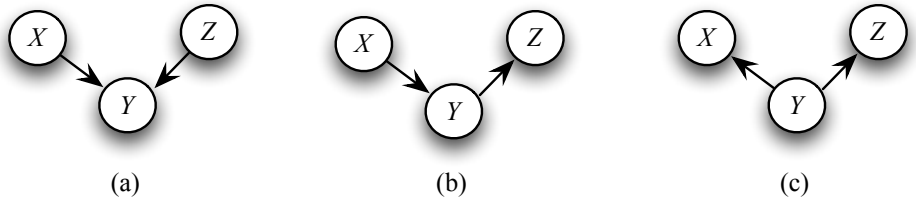


Figure 18.3. *(a) a collider at Y; (b), (c) non-colliders.*

The collider property is path dependent. In Figure 18.4, Y is a collider on the path $\{X, Y, Z\}$ but it is a non-collider on the path $\{X, Y, W\}$.

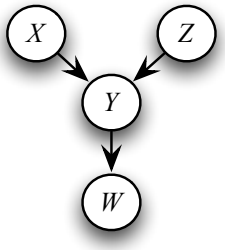


Figure 18.4. *A collider with a descendant.*

When the variables pointing into a collider are not adjacent, we say that the collider is *unshielded*. A directed path that starts and ends at the same variable is called a *cycle.* A directed graph is *acyclic* if it has no cycles. In this case we say that the graph is a *directed acyclic graph* or *DAG*. From now on, we only deal with directed acyclic graphs since it is very difficult to provide a coherent probability semantics over graphs with directed cycles.

18.3 Probability and DAGs

Let G be a DAG with vertices $V = (X_1, \ldots, X_d)$. For notational simplicity, we sometimes represent $V = \{1, \ldots, d\}$. If P is a distribution for V with probability function $p(x)$, we say that P is *Markov to* G, or that G *represents* P, if

$$p(x) = \prod_{j=1}^{d} p(x_j \mid \pi_{x_j}) \tag{18.2}$$

where π_{x_j} is the set of parent nodes of X_j. The set of distributions represented by G is denoted by $\mathcal{M}(G)$.

18.3 Example. Figure 18.5 shows a DAG with four variables. The probability function

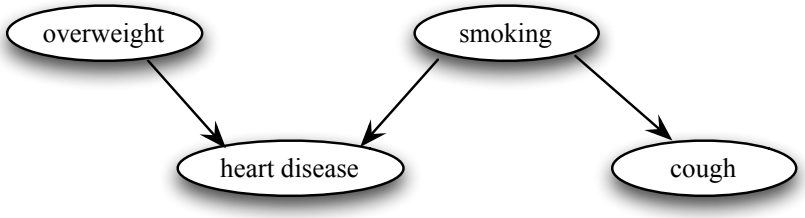


Figure 18.5. *DAG for Example 18.3.*

takes the following decomposition:

$$p(\text{overweight}, \text{smoking}, \text{heart disease}, \text{cough}) = p(\text{overweight}) \times f(\text{smoking}) \\ \times p(\text{heart disease} \,|\, \text{overweight}, \text{smoking}) \times p(\text{cough} \,|\, \text{smoking}).$$

□

18.4 Example. For the DAG in Figure 18.6, $P \in \mathcal{M}(G)$ if and only if its probability function $p(x)$ has the form $p(x, y, z, w) = p(x)p(y)p(z \mid x, y)p(w \mid z)$. □

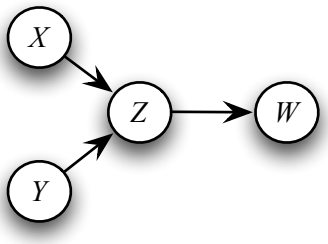


Figure 18.6. *Another DAG.*

The following theorem says that $P \in \mathcal{M}(G)$ if and only if the *Markov condition* holds. Roughly speaking, the Markov condition means that every variable W is independent of the "past" given its parents.

18.5 Theorem. *For a graph $G = (V, E)$, a distribution $P \in \mathcal{M}(G)$ if and only if the following Markov condition holds: for every variable W,*

$$W \perp\!\!\!\perp \widetilde{W} \mid \pi_W \tag{18.6}$$

where $\widetilde{W}$ denotes all the other variables except the parents and descendants of W.

Proof. First, we assume that $W \perp\!\!\!\perp \widetilde{W} \mid \pi_W$ and want to show that $p(x) = \prod_{j=1}^{d} p(x_j \mid \pi_{x_j})$, where $p(x)$ is the density function.

Without loss of generality, we let $X_1, X_2, \ldots, X_d$ be a topological ordering of the variables in the graph G. The following result follows from the chain rule:

$$p(x) = \prod_{j=1}^{d} p(x_j \,|\, x_1, \ldots, x_{j-1}). \tag{18.7}$$

From (18.6), it is easy to see that $p(x_j \,|\, x_1, \ldots, x_{j-1}) = p(x_j \,|\, \pi_{x_j})$. This proves one direction. The other direction is left as an exercise (see Exercise 1). □

Write $\mathcal{I}(G)$ to denote the set of conditional independence relations corresponding to the graph G. Also write $\mathcal{I}(P)$ to denote the set of conditional independence relations corresponding to the distribution P. Then we can restate Theorem 18.5 by saying that $P \in \mathcal{M}(G)$ if and only if $\mathcal{I}(G) \subset \mathcal{I}(P)$. Say that P is *faithful* to G if $\mathcal{I}(G) = \mathcal{I}(P)$.

18.8 Example. Let $G = (V, E)$ where $V = (X, Y)$ and $E = \{(X, Y)\}$. The graph has one edge from node X to node Y, denoted as $X \longrightarrow Y$. Then $\mathcal{I}(G) = \emptyset$. Suppose P has probability function $p(x)$ and that $p(x, y) = p(x)p(y)$. Then $\mathcal{I}(P) = \{X \perp\!\!\!\perp Y\}$. Hence $\mathcal{I}(G) \subset \mathcal{I}(P)$. Therefore P is Markov to G but is not faithful to G. □

18.9 Example. Consider the DAG in Figure 18.6, the Markov condition implies that $X \perp\!\!\!\perp Y$ and $W \perp\!\!\!\perp \{X, Y\} \mid Z$. □

18.10 Example. Consider the DAG in Figure 18.7. In this case the probability function

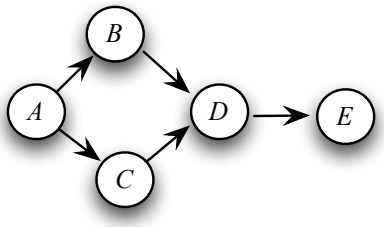


Figure 18.7. *Yet another DAG.*

must factor like $p(a, b, c, d, e) = p(a)p(b \,|\, a)p(c \,|\, a)p(d \,|\, b, c)p(e \,|\, d)$.

The Markov condition implies the following conditional independence relations:

$$D \perp\!\!\!\perp A \mid \{B, C\}, \quad E \perp\!\!\!\perp \{A, B, C\} \mid D \quad \text{and} \quad B \perp\!\!\!\perp C \mid A$$

□

18.11 Example. Let G be a chain graph denoted as in Figure 18.8. We then have $p(x) = p(x_0)p(x_1 \,|\, x_0)p(x_2 \,|\, x_1) \cdots$. The distribution of each variable depends only on its immediate predecessor. We say that P is a *Markov chain.* □

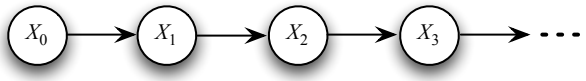


Figure 18.8. *A chain graph.*

18.12 Example. A *hidden Markov model (HMM)* involves two set of variables $X_1, X_2, \ldots$ and $Y_1, Y_2, \ldots$. The X_i's form a Markov chain but they are unobserved. The Y_i's are observed and the distribution of Y_i depends only on X_i. See Figure 18.9, in which we use gray variables to indicate the fact that Y_i's are observed. HMMs are used in genetics, speech recognition and many other applications. □

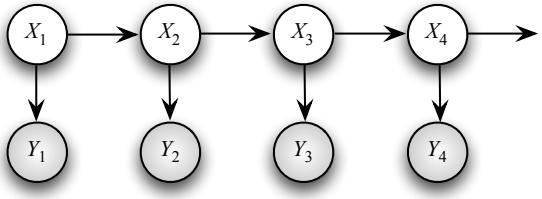


Figure 18.9. *A hidden Markov model. The gray variables indicate that they are observed.*

18.13 Example. Some statistical models can naturally be written in layers and are called *hierarchical models* or *random effects models.* For example, suppose we sample k counties and then we sample n_i people in the i-th county. We count the number Y_i that test positive for some disease. Then $Y_i \sim \text{Binomial}(n_i, \theta_i)$. The θ_i's can also be regarded as random variables sampled from some distribution $p(\theta; \psi)$. For example, we might have $\theta_i \sim \text{Beta}(\alpha, \beta)$ so that $\psi = (\alpha, \beta)$. The model can be written as

$$\begin{aligned} \theta_i &\sim \text{Beta}(\alpha, \beta) \\ Y_i|\theta_i &\sim \text{Binomial}(n_i, \theta_i). \end{aligned}$$

Figure 18.10 shows a DAG representing this model. □

18.4 More Independence Relations

The Markov condition allows us to list some independence relations implied by a DAG. These relations might imply other independence relations. Let's consider the DAG in Figure 18.11. The Markov condition implies:

$$X_1 \perp\!\!\!\perp X_2, \qquad X_2 \perp\!\!\!\perp \{X_1, X_4\}, \qquad X_3 \perp\!\!\!\perp X_4 \mid \{X_1, X_2\},$$

$$X_4 \perp\!\!\!\perp \{X_2, X_3\} \mid X_1, \qquad X_5 \perp\!\!\!\perp \{X_1, X_2\} \mid \{X_3, X_4\}$$

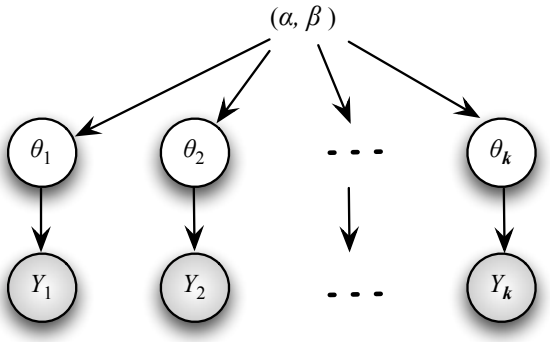


Figure 18.10. *A hierarchical model. All Y_i's are represented by gray nodes, indicating the fact that they are observed.*

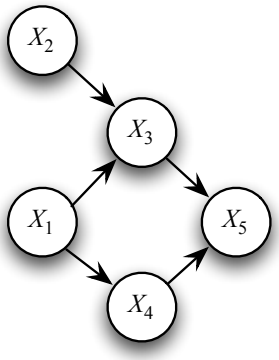


Figure 18.11. *And yet another DAG.*

It turns out (but it is not obvious) that these conditions imply that

$$\{X_4, X_5\} \perp\!\!\!\perp X_2 \mid \{X_1, X_3\}.$$

How do we find these extra conditional independence relations? The answer is "*d-separation*," which is short for "*directed separation*." The notion of d-separation can be summarized by three rules. Consider the four DAGs in Figure 18.12. The first three DAGs in Figure 18.12 have no colliders. The DAG in Figure 18.12 (d) has a collider. The DAG in Figure 18.4 has a collider with a descendant.

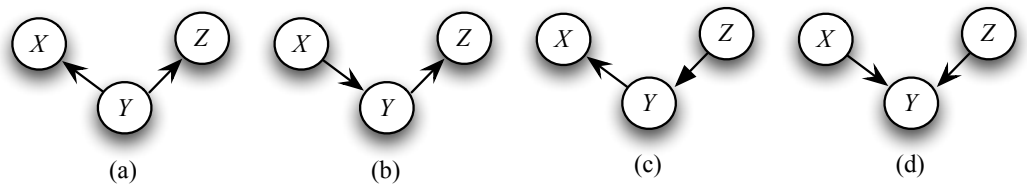


Figure 18.12. *The first three DAGs have no colliders. The fourth DAG has a collider at Y.*

The Rules of d-Separation

Consider the DAGs in Figures 18.12 and 18.4.

1. When Y is not a collider, X and Z are *d-connected*, but they are *d-separated* given Y.
2. If X and Z collide at Y, then X and Z are *d-separated*, but they are *d-connected* given Y.
3. Conditioning on the descendant of a collider has the same effect as conditioning on the collider. Thus in Figure 18.4, X and Z are *d-separated* but they are *d-connected* given W.

Here is a more formal definition of d-separation. Let X and Y be distinct vertices and let W be a set of vertices not containing X or Y. Then X and Y are *d-separated* given W if there exists no undirected path U between X and Y such that (i) every collider on U has a descendant in W, and (ii) no other vertex on U is in W. If A, B, and W are distinct sets of vertices and A and B are not empty, then A and B are d-separated given W if for every $X \in A$ and $Y \in B$, X and Y are d-separated given W. The sets of vertices that are not d-separated are said to be d-connected.

18.14 Example. Consider the DAG in Figure 18.13.

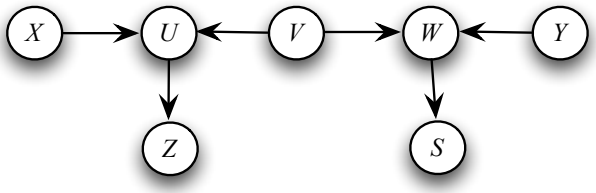


Figure 18.13. *Example for d-separation.*

From the d-separation rules we conclude that: (i) X and Y are d-separated (given the empty set); (ii) X and Y are d-connected given $\{Z, S\}$; (iii) X and Y are d-separated given $\{Z, S, V\}$. □

The following theorem, due to Verma and Pearl (1988), provides the probabilistic implications of d-separation.

18.15 Theorem. (Verma and Pearl (1988)) *If sets A and B are d-separated by C in a DAG G, then A is independent of B conditional on C in every distribution compatible with G. Conversely, if A and B are not d-separated by C in a DAG G, then A and B are dependent conditional on C in at least one distribution compatible with G.*

Proof Idea. Assuming A and B are d-separated by C, the proof uses the algebraic properties of conditional independence. For the converse, the proof constructs a distribution P such that A and B are correlated conditioned on C. Such a construction is possible since A and B are not d-separated by C. From the definition of d-separability, there are several cases to consider, depending on whether a collider appears between A and B. In each case, conditional probability distributions can be constructed so that the correlations can be propagated from A to B. □

18.16 Example. Consider the DAG in Figure 18.5. In this example, overweight and smoking are marginally independent but they are dependent given heart disease. □

Graphs that look different may actually imply the same independence relations. If G is a DAG, we let $\mathcal{I}(G)$ denote all the independence statements implied by G. Two DAGs G_1 and G_2 for the same variable set V are *Markov equivalent* if $\mathcal{I}(G_1) = \mathcal{I}(G_2)$. All DAGs can be partitioned into equivalence classes so that DAGs within each class are Markov equivalent. Given a DAG G, let $\text{skeleton}(G)$ denote the undirected graph obtained by replacing the arrows with undirected edges.

18.17 Theorem. (Verma and Pearl (1990)) *Two DAGs G_1 and G_2 are Markov equivalent if and only if (i) $\text{skeleton}(G_1) = \text{skeleton}(G_2)$ and (ii) G_1 and G_2 have the same unshielded colliders (i.e. any two nodes pointing to the same collider are not connected).*

Proof. See Exercise 3. □

18.18 Example. According to Theorem 18.17, the first three DAGs in Figure 18.12 are Markov equivalent. The DAG in Figure 18.12 (d) is not Markov equivalent to the other three. □

18.5 Gaussian DAGs

We now represent the multivariate Gaussian distribution using a directed graphical model. This representation is based on the *linear Gaussian model*, which is defined in the following

18.19 Definition. (Linear Gaussian Model) *Let Y be a continuous variable in a DAG with parents $X_1, \ldots, X_k$. We say that Y has a linear Gaussian model of its parents if there are parameters $\beta_0, \beta_1, \ldots, \beta_k$ and σ^2 such that*

$$Y = \beta_0 + \beta_1 X_1 + \ldots + \beta_k X_k + \epsilon, \quad \epsilon \sim N(0, \sigma^2). \tag{18.20}$$

18.21 Definition. (Gaussian Directed Graphical Models) *A directed graphical model is called a Gaussian directed graphical model (or Gaussian Bayesian network) if all the variables are continuous and every variable and its parents follow a linear Gaussian model.*

In a Gaussian Bayesian network, each variable X_j is modeled as a linear function of its parents plus normally distributed random noise. One important result is that there is one-to-one correspondence between a multivariate Gaussian distribution and a Gaussian Bayesian network. The next theorem shows that Gaussian Bayesian network defines a joint multivariate Gaussian distribution.

18.22 Theorem. *We assume that Y has a linear Gaussian model of its parents $X_1, \ldots, X_k$: $Y = \beta_0 + \sum_{j=1}^k \beta_j X_j + \epsilon$, where $\epsilon \sim N(0, \sigma^2)$. We also assume that $X = (X_1, \ldots, X_k)$ are jointly Gaussian with distribution $N(\mu, \mathbf{\Sigma})$. Then the joint distribution of $(Y, X_1, \ldots, X_k)$ is a Gaussian distribution with $\mathrm{Cov}(Y, X_i) = \sum_{j=1}^k \beta_j \mathbf{\Sigma}_{ij}$. Moreover, we have $\mathbb{E}(Y) = \beta_0 + \beta^T \mu$ and $\mathrm{Var}(Y) = \sigma^2 + \beta^T \mathbf{\Sigma} \beta$, where $\beta = (\beta_1, \ldots, \beta_k)$.*

Proof. Since the marginal distribution of X and the conditional distribution of Y given X are both Gaussians, the joint distribution of (Y, X) is Gaussian with the covariance:

$$\mathrm{Cov}(Y, X_i) = \mathrm{Cov}\Big(\beta_0 + \sum_{j=1}^k \beta_j X_j, X_i\Big) = \sum_{j=1}^k \beta_j \mathrm{Cov}(X_j, X_i) = \sum_{j=1}^k \beta_j \mathbf{\Sigma}_{ij}. \tag{18.23}$$

Also, since Y is the summation of $k+1$ Gaussian variables, the marginal distribution of Y is Gaussian with the desired mean and variance. □

The converse of this theorem is also true, which states that any multivariate Gaussian distribution can be converted to a Gaussian Bayesian network. We first state a simple lemma, the proof of which follows from straightforward algebraic manipulation.

18.24 Lemma. *Let (Y, X) be a multivariate Gaussian distribution:*

$$\begin{pmatrix} Y \\ X \end{pmatrix} \sim N\left(\begin{pmatrix} \mu_Y \\ \mu_X \end{pmatrix}, \begin{pmatrix} \mathbf{\Sigma}_{YY} & \mathbf{\Sigma}_{YX} \\ \mathbf{\Sigma}_{XY} & \mathbf{\Sigma}_{XX} \end{pmatrix} \right). \tag{18.25}$$

Then $Y \mid X \sim N(\beta_0 + \beta^T X, \sigma^2)$ where

$$\beta_0 = \mu_Y - \mathbf{\Sigma}_{YX} \mathbf{\Sigma}_{XX}^{-1} \mu_X, \ \ \beta = \mathbf{\Sigma}_{XX}^{-1} \mathbf{\Sigma}_{YX}, \ \ \text{and } \sigma^2 = \mathbf{\Sigma}_{YY} - \mathbf{\Sigma}_{YX} \mathbf{\Sigma}_{XX}^{-1} \mathbf{\Sigma}_{XY}.$$

18.26 Theorem. *Let P be the joint distribution of d-dimensional multivariate Gaussian random vector $X = (X_1, \ldots, X_d)$. For any ordering of the variables $X_{\tau(1)}, \ldots, X_{\tau(d)}$, we can construct a DAG G such that P is Markov to G, and where $X_{\tau(i)}$ is a linear Gaussian model of its parents $\pi(X_{\tau(i)}) \subset \{X_{\tau(1)}, \ldots, X_{\tau(i-1)}\}$ for all i.*

Proof. See Exercise 4. □

Next, we study the conditional independence properties of a Gaussian model. We start with the definition of *partial correlation*, which measures the degree of association between two random variables, with the effect of a set of controlling random variables removed.

18.27 Definition. *The partial correlation $\rho_{X,Y \mid Z}$ between two variables X and Y given another set of variables $Z = (Z_1, Z_2, \ldots, Z_k)$ is the correlation between the residuals resulting from regressing X on Z and Y on Z. More formally,*

$$\rho_{X,Y \mid Z} = \frac{\mathbb{E}(XY \mid Z) - \mathbb{E}(X \mid Z)\mathbb{E}(Y \mid Z)}{\sqrt{\mathrm{Var}(X \mid Z)}\sqrt{\mathrm{Var}(Y \mid Z)}}. \tag{18.28}$$

Conditional independencies can be inferred from partial correlations in a multivariate Gaussian distribution. This result is summarized in the next theorem, which follows from the elementary properties of the multivariate Gaussian (Lauritzen, 1996a).

18.29 Theorem. *Let X be a Gaussian random vector $X = (X_1, \ldots, X_d)$. For $i \neq j \in \{1, \ldots, d\}$, $K \subset \{1, \ldots, d\} \setminus \{i, j\}$, denote by $\rho_{i,j \mid K}$ the partial correlation between X_i and X_j given $\{X_k : k \in K\}$. Then $\rho_{i,j \mid K} = 0$ if and only if X_i and X_j are conditionally independent given $\{X_k : k \in K\}$.*

We can thus obtain estimates of conditional independencies for Gaussian DAGs by calculating sample partial correlations. These can be computed via regression, computing components of the inverse of the sample covariance, or recursively using the following proposition.

18.30 Proposition. *Let $X = (X_1, \ldots, X_d)$ be multivariate Gaussian. For $i \neq j \in \{1, \ldots, d\}$, $K \subset \{1, \ldots, d\} \setminus \{i, j\}$, let $\rho_{i,j \mid K}$ be the partial correlation between X_i and X_j given $\{X_k : k \in K\}$. Then for any $h \in K$,*

$$\rho_{i,j \mid K} = \frac{\rho_{i,j \mid K \setminus h} - \rho_{i,h \mid K \setminus h} \cdot \rho_{j,h \mid K \setminus h}}{\sqrt{1 - \rho^2_{i,h \mid K \setminus h}}\sqrt{1 - \rho^2_{j,h \mid K \setminus h}}}. \tag{18.31}$$

Proof. See Exercise 5. □

In the above proposition, when A is the empty set, the partial correlation $\rho_{i,j \mid A}$ reduces to the Pearson correlation between the random variables X_i and X_j. This enables calculation of higher order partial correlations in a recursive way.

18.6 Exact Inference

For *inference in directed graphical models*, some of the variables in a graph are set to certain values due to evidence, and we wish to compute the posterior distribution of some other variables. This requires calculating the probability distribution on a subset of variables by marginalizing over the joint. One thing to note is that inference in this section is just calculating marginal and conditional probabilities. which is different from the general term statistical inference used in this book.

Even a probability distribution is given, calculating marginal or conditional distributions can be costly because it requires summing over an exponential number of joint probability combinations. The first efficient algorithm proposed for probabilistic inference in DAGs used *message-passing* architecture and were limited to trees (Pearl, 1982). The main idea is to view each variable as a simple processor and reduce probabilistic inference to asynchronous local message passing among different nodes until equilibrium is achieved. In this section, we mainly focus on *exact inference* methods, and in the later computing chapters we will consider different *approximate inference* algorithms.

18.6.1 Belief Propagation (Sum-Product Algorithm) on Polytrees

We now introduce a local message-passing type algorithm named *belief propogation* (Pearl, 1988; Lauritzen and Spiegelhalter, 1988). For simplicity, we assume all the variables are discrete, therefore marginalization only requires summations. This assumption is not restrictive since for continuous variables we only need to replace summation with integration.

We focus on conducting exact inference on *polytrees*. In graph theory, a polytree is a directed graph with at most one undirected path between any two nodes. In other words, a polytree is a DAG for which there are no undirected cycles. One example of polytree is provided in Figure 18.14. To introduce belief propagation algorithm for polytrees, we consider an inference problem that takes the form $\mathbb{P}(X = x \mid E = e)$ where X is a *query node* and E is any subset of observed nodes (or *evidence nodes*) who have been set to certain values.

Without loss of generality, we assume $E = E_X^+ \cup E_X^-$ where E_X^+ is a subset of the ancestor nodes of X and E_X^- is a subset of the descendant nodes of X (Both E_X^+ and E_X^- may also be empty). For belief propagation, we treat each node as a processor that receives messages from its neighbors and pass them along after some local calculation. Let X has n children $Y_1, \ldots, Y_n$ and k parents $Z_1, \ldots, Z_k$. The node X receives and processes two types of messages from its neighbors. First, we use $m_{Y_j}^-(X)$ to represent the message sent from the child Y_j to X. Assuming X has n children, we have n such messages:

$$m_{Y_1}^-(X), \ldots, m_{Y_n}^-(X). \tag{18.32}$$

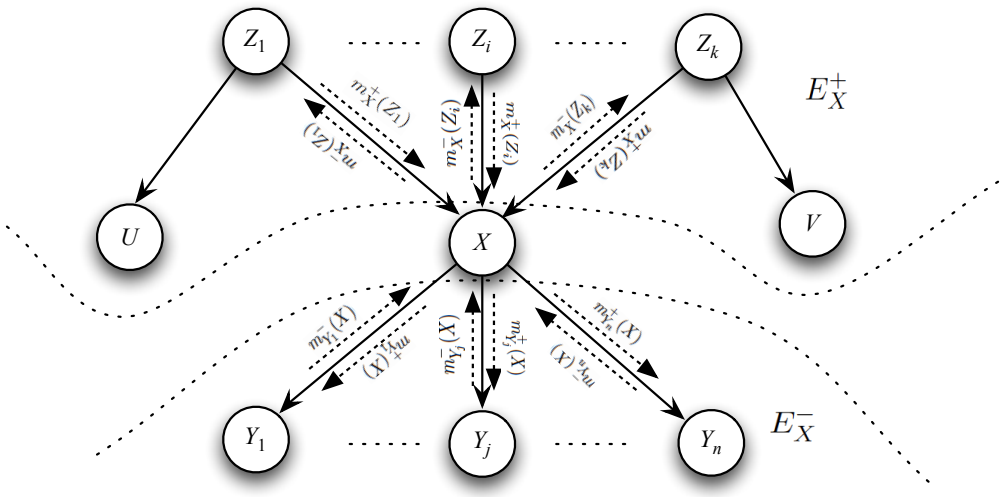


Figure 18.14. *A polytree that is used to illustrate the belief propagation algorithm. In a polytree, a node may have several children and parents, but there can be at most one undirected path that connects any two nodes.*

More detailed forms of $m_{Y_j}^-(X)$ will be provided in later paragraphs. We also denote $m_{Y_j}^+(X)$ to be the message sent from X to one of its children Y_j. For all the n children of X, we have the following messages:

$$m_{Y_1}^+(X), \ldots, m_{Y_n}^+(X). \tag{18.33}$$

Assuming the node X has exactly k parents, we define $m_X^-(Z_1), \ldots, m_X^-(Z_k)$ to be the messages sent from X to its parents $Z_1, \ldots, Z_k$. Similarly, we define $m_X^+(Z_1), \ldots, m_X^+(Z_k)$ to be the messages sent from $Z_1, \ldots, Z_k$ to X. For more details, see Figure 18.14.

Let $E_X^+ = e_X^+$ and $E_X^- = e_X^-$. We want to evaluate

$$p(x \,|\, e_X^+, e_X^-) \equiv \mathbb{P}(X = x \,|\, E_X^+ = e_X^+, E_X^- = e_X^-). \tag{18.34}$$

Since X d-separates E_X^+ and E_X^-, we have that $p(e_X^+, e_X^- \,|\, x) = p(e_X^+ \,|\, x)p(e_X^- \,|\, x)$.

We further define $E_X^+ = E_{X,Z_1}^+ \cup \cdots \cup E_{X,Z_k}^+$ where E_{X,Z_i}^+ is a subset of E_X^+ that are also ancestors of Z_i. Similarly, we define $E_X^- = E_{X,Y_1}^- \cup \cdots \cup E_{X,Y_n}^-$ where E_{X,Y_j}^- is a subset of E_X^- that are also descendants of Y_j. We define $m^+(x) \equiv p(x \,|\, e_X^+)$ to be the propagated $E_X^+ = e_X^+$ that X receives from its parents and passes on to its children, and $m^-(x) \equiv p(e_X^- \,|\, x)$ to be the propagated $E_X^- = e_X^-$ that X receives from its children and

passes on to its parents. We then have

$$p(x \mid e_X^+, e_X^-) = \frac{p(e_X^+, e_X^- \mid x)p(x)}{p(e_X^+, e_X^-)} = \frac{p(e_X^+ \mid x)p(e_X^- \mid x)p(x)}{p(e_X^+, e_X^-)} \tag{18.35}$$

$$= \frac{p(x \mid e_X^+)p(e_X^+)p(e_X^- \mid x)p(x)}{p(x)p(e_X^+, e_X^-)} \tag{18.36}$$

$$= \alpha p(x \mid e_X^+)p(e_X^- \mid x) \tag{18.37}$$

$$= \alpha m^+(x)m^-(x) \tag{18.38}$$

where the normalizing constant $\alpha = p(e_X^+)/p(e_X^+, e_X^-)$ does not depend on x.

We now explain how to evaluate $m^+(x)$ and $m^-(x)$ in a recursive way. Similarly to the definition of $m^+(x)$ and $m^-(x)$, we define the messages

$$m_{Y_j}^+(x) = p(x \mid e_{X,Y_j}^+) \quad \text{and} \quad m_{Y_j}^-(x) = p(e_{X,Y_j}^- \mid x), \tag{18.39}$$

$$m_X^+(z_i) = p(z_i \mid e_{X,Z_i}^+) \quad \text{and} \quad m_X^-(z_i) = p(e_{X,Z_i}^- \mid z_i). \tag{18.40}$$

We then have

$$m^-(x) = p(e_X^- \mid x) = p(e_{X,Y_1}^-, \ldots, e_{X,Y_n}^- \mid x) = \prod_{j=1}^n p(e_{X,Y_j}^- \mid x) = \prod_{j=1}^n m_{Y_j}^-(x), \tag{18.41}$$

where we have utilized the fact that X d-separates $Y_1, \ldots, Y_n$.

Similarly, we have

$$m^+(x) = p(x \mid e_X^+) = p(x \mid e_{X,Z_1}^+, \ldots, e_{X,Z_k}^+) \tag{18.42}$$

$$= \sum_{z_1} \sum_{z_2} \cdots \sum_{z_k} p(x \mid z_1, z_2, \ldots, z_k) p(z_1, \ldots, z_k \mid e_{X,Z_1}^+, \ldots, e_{X,Z_k}^+) \tag{18.43}$$

$$= \sum_{z_1} \sum_{z_2} \cdots \sum_{z_k} p(x \mid z_1, z_2, \ldots, z_k) \prod_{i=1}^k p(z_i \mid e_{X,Z_i}^+) \tag{18.44}$$

$$= \sum_{z_1} \sum_{z_2} \cdots \sum_{z_k} p(x \mid z_1, z_2, \ldots, z_k) \prod_{i=1}^k m_X^+(z_i). \tag{18.45}$$

Therefore, we see that given all the messages passed from the node X's parents and children, we can evaluate $p(x \mid e_X^+, e_X^-)$ easily. The remaining question is how can we efficiently collect these messages. This requires us to calculate the messages a node send to its children and parents. Without loss of generality, we only need to explain how to calculate the messages that X send to its parents and children:

$$m_X^-(Z_1), \ldots, m_X^-(Z_k), \text{ and } m_{Y_1}^+(X), \ldots, m_{Y_n}^+(X).$$

Note that what node Y_j receives from X includes both information that X gets from its parents $Z_1, \ldots, Z_k$ (i.e. E_X^+) and also from its other children $Y_1, \ldots, Y_{j-1}, Y_{j+1}, \ldots, Y_n$

(i.e. $\{E_X^- \setminus E_{X,Y_j}^-\}$). Let

$$\beta = \frac{1}{p\left(\{e_X^- \setminus e_{X,Y_j}^-\} \,|\, e_X^+\right)}.$$

To evaluate $m_{Y_j}^+(X)$, we have

$$\begin{aligned}
m_{Y_j}^+(x) &= p(x \,|\, e_{X,Y_j}^+) = p\left(x \,|\, e_X^+, \{e_X^- \setminus e_{X,Y_j}^-\}\right) && (18.46)\\
&= \frac{p\left(\{e_X^- \setminus e_{X,Y_j}^-\} \,|\, x, e_X^+\right) p\left(x \,|\, e_X^+\right)}{p\left(\{e_X^- \setminus e_{X,Y_j}^-\} \,|\, e_X^+\right)} && (18.47)\\
&= \beta \cdot p\left(\{e_X^- \setminus e_{X,Y_j}^-\} \,|\, x, e_X^+\right) p\left(x \,|\, e_X^+\right) && (18.48)\\
&= \beta \prod_{\ell \neq j} p\left(e_{X,Y_\ell}^- \,|\, x, e_X^+\right) p\left(x \,|\, e_X^+\right) && (18.49)\\
&= \beta \prod_{\ell \neq j} p\left(e_{X,Y_\ell}^- \,|\, x\right) p\left(x \,|\, e_X^+\right) && (18.50)\\
&= \beta \prod_{\ell \neq j} m_{Y_\ell}^-(x) m^+(x). && (18.51)
\end{aligned}$$

Where have used the fact that X d-separates $Y_1, \ldots, Y_n$.

We then show how to evaluate $m_X^-(Z_i)$. Note that the message $m_X^-(Z_i)$ that X passes on to one of its parents Z_i includes not only the messages X gets from its children (i.e. E_X^-) but also the messages X receives from its other parents (i.e. $\{E_X^- \setminus E_{X,Z_i}^-\}$). Let

$$\gamma = p\left(\{e_X^+ \setminus e_{X,Z_i}^+\}\right). \tag{18.52}$$

We then have

$$
\begin{aligned}
m_X^-(z_i) &= p(e_{X,Z_i}^- \mid z_i) = \sum_x \sum_{\{z_\ell\}_{\ell\neq i}} p(e_{X,Z_i}^-, x, \{z_\ell\}_{\ell\neq i} \mid z_i) && (18.53)\\
&= \sum_x \sum_{\{z_\ell\}_{\ell\neq i}} p\left(e_X^-, \{e_X^+ \setminus e_{X,Z_i}^+\}, x, \{z_\ell\}_{\ell\neq i} \mid z_i\right) && (18.54)\\
&= \sum_x \sum_{\{z_\ell\}_{\ell\neq i}} p\left(e_X^-, \{e_X^+ \setminus e_{X,Z_i}^+\} \mid x, \{z_\ell\}_{\ell\neq i}, z_i\right) p\left(x, \{z_\ell\}_{\ell\neq i} \mid z_i\right) && (18.55)\\
&= \sum_x \sum_{\{z_\ell\}_{\ell\neq i}} p\left(e_X^- \mid x\right) p\left(\{e_X^+ \setminus e_{X,Z_i}^+\} \mid \{z_\ell\}_{\ell\neq i}\right) p\left(x, \{z_\ell\}_{\ell\neq i} \mid z_i\right) && (18.56)\\
&= \sum_x \sum_{\{z_\ell\}_{\ell\neq i}} p\left(e_X^- \mid x\right) \frac{p\left(\{z_\ell\}_{\ell\neq i} \mid \{e_X^+ \setminus e_{X,Z_i}^+\}\right) \cdot \gamma}{p\left(\{z_\ell\}_{\ell\neq i}\right)} p\left(x, \{z_\ell\}_{\ell\neq i} \mid z_i\right) && (18.57)\\
&= \gamma \sum_x \sum_{\{z_\ell\}_{\ell\neq i}} p\left(e_X^- \mid x\right) p\left(\{z_\ell\}_{\ell\neq i} \mid \{e_X^+ \setminus e_{X,Z_i}^+\}\right) p\left(x \mid \{z_\ell\}_{\ell\neq i}, z_i\right) && (18.58)\\
&= \gamma \sum_x \sum_{\{z_\ell\}_{\ell\neq i}} m^-(x) \left(\prod_{\ell\neq i} m_X^+(z_i)\right) p(x \mid z_1, \ldots, z_k) && (18.59)\\
&= \gamma \sum_x m^-(x) \sum_{\{z_\ell\}_{\ell\neq i}} p(x \mid z_1, \ldots, z_k) \prod_{\ell\neq i} m_X^+(z_i). && (18.60)
\end{aligned}
$$

Given the above recursive relationship of the message-passing algorithm, the only thing left is to determine the initial values of the leaf nodes, root nodes, and evidence nodes. Let X be a node with parents $Z_1, \ldots, Z_k$ and children $Y_1, \ldots, Y_n$ (If X is a leaf node, then there is no children; If X is a root node, then there is no parents). It is easy to see that if X is initialized to be a certain evidence value e, then

$$
m_X^-(Z_1) = \cdots = m_X^-(Z_n) = m_{Y_1}^+(X) = \cdots = m_{Y_n}^+(X) = 1. \tag{18.61}
$$

If X is an un-initialized leaf node with parents $Z_1, \ldots, Z_k$, we have $m_X^-(Z_1) = \cdots = m_X^-(Z_k) = 1$. If X is an un-initialized root node with children $Y_1, \ldots, Y_n$, we have $m_{Y_1}^+(X) = \cdots = m_{Y_n}^+(X) = \mathbb{P}(X)$ with $\mathbb{P}(X = x) = p(x)$.

The belief propagation algorithm conducts exact inference for polytrees. However, if there is a cycle in the underlying undirected graph, the belief propagation algorithm will no longer work. The main reason is that when cycles exist, a node X is no longer guaranteed to d-separate its ancestors and descendants. Therefore, the messages can be propagated through multiple paths. To apply belief propagation, we need to convert the graph into a polytree T. Each node of T may correspond to a set of original variables. This is related to the *junction tree* algorithms, which can be viewed as belief propagation on a modified graph guaranteed to be a polytree. The basic idea is to eliminate cycles by clustering them into single nodes. More details about junction trees will be introduced in the next chapter

on undirected graphical models. Since the messages in the belief propagation algorithm takes the form of summing over many product terms. The belief propagation is also called the *sum-product* algorithm.

18.6.2 Max-Product and Max-Sum Algorithms

A similar algorithm is commonly referred to as *max-product*, also known as *max-sum* algorithm, which solves a related problem of maximizing a probability, or finding the most probable explanation for certain observations. Instead of attempting to calculate the marginal, the goal now is to find a joint configuration of $X = \widehat{x}$ which has *maximum a posterior probability* (the *MAP* estimate):

$$\widehat{x} = \arg\max_{x} \mathbb{P}(X = x \,|\, E = e), \tag{18.62}$$

where X is a *query node* which could possibly represents a set of random variables. E is the evidence node as before. An algorithm that solves this problem is nearly identical to belief propagation, with sums replaced by maxima in messages. The resulting procedure is called the *max-product* algorithm. Note that to avoid potential underflow, the max-product algorithm can equivalently be written in terms of log-probabilities, in which case one obtains the *max-sum* algorithm. Once again we see that in a polytree, exact MAP inference can be performed in two sweeps using the max-product algorithm. The algorithm is in fact an example of dynamic programming. The max-product or max-sum algorithm, when applied to hidden Markov models (HMMs), is known as the *Viterbi algorithm*.

18.7 Approximate Inference

The graphical models encountered in applications may have large cliques or long loops, which make exact inference intractable. In this setting we must conduct *approximate inference*. There are two popular ways for approximate inference: (i) *variational methods* and (ii) *sampling* methods. The former class of methods are deterministic and the latter class of methods are stochastic. We defer the discussions of these two families of approximation inference algorithms to later computing chapters. Another way to conduct approximate inference is to directly apply the belief propagation algorithm without worrying about the loops. Such a method is known as *loopy belief propagation* (Frey and MacKay, 1997). This algorithm can be effective on certain applications, though its convergence is not guaranteed.

18.8 Parameter Estimation

Two estimation questions arise in the context of DAGs. First, given a DAG G and data $x_1, \ldots, x_n$ from a distribution $p(x)$ consistent with G, how do we estimate $p(x)$? Second, given data $x_1, \ldots, x_n$ how do we estimate G? The first question is a standard *parameter estimation* problem. The second question is a *structure learning* problem and is similar in

approaches and terminology to model selection procedures for classical statistical models. In this section we only discuss parameter estimation with pre-fixed DAG G. For parameter estimation, one important distinction is whether all the variables are observed, or whether some of them are hidden. We discuss these two cases separately.

18.8.1 Parameter Estimation from Fully Observed Data

Let G be a DAG with vertices $V = (X_1, \ldots, X_d)$. Once G is given, the task of estimating the parameters of the joint distribution can be greatly simplified by the application of the Markov property. Suppose we use a parametric model $p(x_j \mid \pi_{x_j}; \theta_j)$ for each conditional density in (18.2), where π_{x_j} is the set of parent nodes of X_j. Let $\theta = (\theta_1, \ldots, \theta_d)$ be the set of parameters, the joint distribution in (18.2) can be written as

$$p(x; \theta) = \prod_{j=1}^{d} p(x_j \mid \pi_{x_j}; \theta_j). \tag{18.63}$$

Given n data points $\{x_1, \ldots, x_n\}$, the likelihood function is

$$\mathcal{L}(\theta) = \prod_{i=1}^{n} p(x_i; \theta) = \prod_{i=1}^{n} \prod_{j=1}^{d} p(x_{ij} \mid \pi_{x_j}; \theta_j), \tag{18.64}$$

where x_{ij} is the value of X_j for the i^{th} data point and θ_j are the parameters for the j^{th} conditional density. We can then estimate the parameters by maximum likelihood. It is easy to see that the log-likelihood decomposes according to the graph structure:

$$\ell(\theta) = \log \mathcal{L}(\theta) = \sum_{j=1}^{d} \log \left(\prod_{i=1}^{n} p(x_{ij} \mid \pi_{x_j}; \theta_j) \right) \equiv \sum_{j=1}^{d} \log \mathcal{L}_j(\theta_j) = \sum_{j=1}^{d} \ell_j(\theta_j), \tag{18.65}$$

where $\ell_j(\theta_j) = \sum_{i=1}^{n} \log p(x_{ij} \mid \pi_{x_j}; \theta_j)$ is a localized conditional likelihood for θ_j. Therefore we can maximize the contribution to the log-likelihood of each node independently. (It is straightforward to extend this to the shared parameter paradigms.)

When there is not enough information from the data points, we could also regularize the log-likelihood to avoid overfitting:

$$\widehat{\theta} = \arg\max_{\theta} \left\{ \ell(\theta) - \mathrm{Pen}(\theta) \right\}, \tag{18.66}$$

where $\mathrm{Pen}(\theta) \geq 0$ is some penalty term of θ.

18.8.2 Parameter Estimation with Hidden Variables

In many applications, observed data may not include the values of some of the variables in the DAG. We refer to these variables as hidden variables. If Z denotes be the hidden

variables, the log-likelihood can be written as

$$\ell(\theta) = \sum_{i=1}^{n} \log p(x_i; \theta) = \sum_{i=1}^{n} \log \int_{z_i} p(x_i, z_i; \theta) dz_i. \tag{18.67}$$

With hidden variables, the log-likelihood is no longer decomposable as in (18.65), and maximizing the log-likelihood in (18.67) is often difficult. This can be approached using the EM algorithm, which is discused in later chapters.

18.68 Example. Consider the graphical models in Figure18.15. The DAG has four nodes. Each node corresponds to one univariate random variable. We consider two settings: (a) all the four variables are fully observable, with data $\{(x_i, y_i, z_i, w_i)\}_{i=1}^n$; (b) only three variables X, Z, W are observable, with data $\{(x_i, z_i, w_i)\}_{i=1}^n$. Given the DAG topology, we

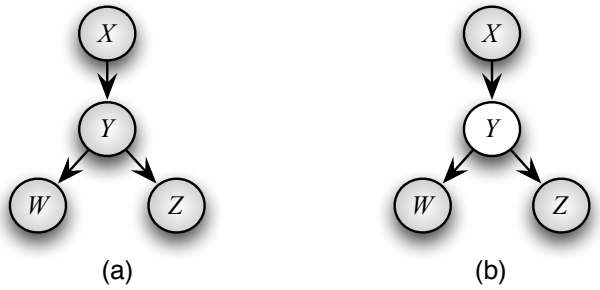


Figure 18.15. *(a) a DAG where all the four nodes are observable; (b) a DAG where only three nodes are observable and one node Y is hidden. A node is gray-colored if it is observable.*

know that the joint density has the decomposition $p(x, y, z, w) = p(w \mid y)p(z \mid y)p(y \mid x)p(x)$. We parametrize the conditional distributions as the following:

$$W \mid Y = y \sim N(\mu_1, 1)I(y = 1) + N(\mu_0, 1)I(y = 0) \tag{18.69}$$

$$Z \mid Y = y \sim N(\mu_0, 1)I(y = 1) + N(\mu_1, 1)I(y = 0) \tag{18.70}$$

$$Y \mid X = x \sim \text{Bernoulli}\left(\frac{1}{1 + \exp(-\beta_0 - \beta_1 x)}\right) \tag{18.71}$$

$$X \sim N(\mu_2, \sigma^2). \tag{18.72}$$

From the above parameterization, we see that the conditional distributions $p(w \mid y)$ and $p(z \mid y)$ share the same set of parameters μ_0 and μ_1. Let $\theta = (\mu_0, \mu_1, \beta_0, \beta_1, \mu_2, \sigma)$ and $\phi(\cdot)$ be the standard Gaussian density function. When all the four variables are observable, the joint log-likelihood of θ has the following decomposition

$$\ell(\theta) = \ell(\mu_0, \mu_1) + \ell(\beta_0, \beta_1) + \ell(\mu_2, \sigma^2) + \text{constant}, \tag{18.73}$$

where $\ell(\mu_0, \mu_1) = -\frac{1}{2}\sum_{i=1}^{n} I(y_i = 1) \cdot \left[(w_i - \mu_1)^2 + (z_i - \mu_0)^2\right] - \frac{1}{2}\sum_{i=1}^{n} I(y_i = 0) \cdot \left[(w_i - \mu_0)^2 + (z_i - \mu_1)^2\right]$, and $\ell(\beta_0, \beta_1) = -\sum_{i=1}^{n} I(y_i = 1) \log\left(1 + \exp(\beta_0 + \beta_1 x_i)\right) -$

$\sum_{i=1}^{n} I(y_i = 0) \log\left(1 + \exp(-\beta_0 - \beta_1 x_i)\right)$. We also have $\ell(\mu_2, \sigma^2) = -\frac{1}{2\sigma^2}\sum_{i=1}^{n}(x_i - \mu_2)^2$. It is easy to see that the maximum likelihood estimates take the form

$$\widehat{\mu}_0 = \frac{1}{2n}\sum_{i=1}^{n}\left[I(y_i = 0)\cdot w_i + I(y_i = 1)\cdot z_i\right] \tag{18.74}$$

$$\widehat{\mu}_1 = \frac{1}{2n}\sum_{i=1}^{n}\left[I(y_i = 1)\cdot w_i + I(y_i = 0)\cdot z_i\right] \tag{18.75}$$

$$\widehat{\mu}_2 = \frac{1}{n}\sum_{i=1}^{n} x_i \quad \text{and} \quad \widehat{\sigma^2} = \frac{1}{n}\sum_{i=1}^{n}(x_i - \widehat{\mu}_2)^2 . \tag{18.76}$$

The parameters β_0 and β_1 can also be easily estimated by solving a logistic regression using Y as output and X as input.

When Y is hidden (as in Figure 18.15 (b)), the log-likelihood no longer decomposes. Parameter estimation is more challenging and requires an iterative EM algorithm. □

18.9 Structure Learning

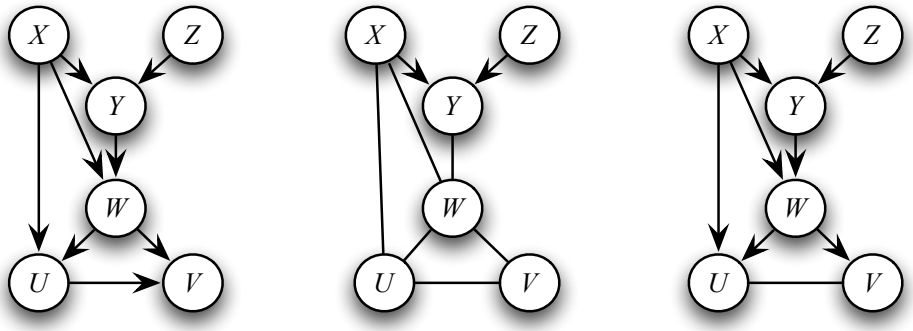
Estimating a DAG from data is very challenging due to the enormous size of the space of DAGs (the number of possible DAGs is super-exponential in the number of nodes). Existing methods can be roughly divided into two categories: (i) *constraint-based* methods and *score-based* methods. Constraint-based methods use statistical tests to learn conditional independence relationships (called constraints in this setting) from the data and prune the graph-searching space using the obtained constraints. In contrast, score-based algorithms assign each candidate DAG a score reflecting its goodness of fit, which is then taken as an objective function to be optimized. In this section, we introduce a constraint-based method, named *PC algorithm* (after its authors, Peter and Clark, see Spirtes et al. (2000)), for estimating DAGs from observed data. Under certain conditions, a sample version of the PC algorithm has been shown to be consistent even for large graphs. In the following, we first describe the population PC algorithm (i.e. we assume the true distribution is given). We then explain if only n i.i.d. observations are obtained, how can we use these samples to estimate the corresponding population quantities of the PC algorithm.

18.9.1 Representing Equivalence Classes

Let $X = (X_1, \ldots, X_d)$ be a d-dimensional random vector with distribution P. We assume that there exists a DAG G such that $\mathcal{I}(P) = \mathcal{I}(G)$. In another word, P is *faithful* to G. Without this assumption, the PC algorithm may fail.

With the faithfulness assumption, one obvious goal is to identify the DAG G from P. However, from Example 18.18, we know that there might exists another DAG G' which is Markov equivalent to G (i.e., G and G' are actually indistinguishable from P). Therefore, instead of identifying one single DAG G, we can only identify the whole Markov

equivalence class of G. From Theorem 18.17, we know that all DAGs in the equivalence class have the same skeleton and the set of unshielded colliders. Motivated by this theorem, the whole equivalence class can be compactly represented by the skeleton graph with unshielded colliders marked (All the edges are undirected except the edges corresponding to unshielded colliders). One such example is shown in Figure 18.16.



(a) original DAG (b) skeleton + unshielded collider (c) CPDAG

Figure 18.16. *(a) The original DAG G; (b) The skeleton of G with unshielded colliders; (c) The CPDAG corresponds to G. The figure shows that CPDAG can be more informative than just graph skeleton annotated with unshielded colliders.*

Figure 18.16 (a) shows an original DAG G which has only one unshielded collider $X \longrightarrow Y \longleftarrow Z$. Figure 18.16 (b) shows the skeleton of G annotated with the unshielded collider. Every DAG that is Markov equivalent to G should have the same skeleton and the unshielded collider. Such a graph, containing both directed and undirected edges, is called *partially directed acyclic graph (PDAG)*. The definition of PDAG is:

18.77 Definition. *(*PDAG*) A partially directed acyclic graph or PDAG is an acyclic graph containing both directed and undirected edges and one can not trace a cycle by following the direction of directed edges and any direction for undirected edges.*

However, the representation in Figure 18.16 (b) is not compact at all. There are 6 undirected edges in the PDAG in Figure 18.16 (b). Since each edge have two possible directions, the potential number of DAGs represented by this graph is $2^6 = 64$. However, if we think more carefully, the directions of many undirected edges can in fact be determined based on the PDAG representing the skeleton and all the unshielded colliders. For example, the edge connecting Y and W must be $Y \longrightarrow W$. Otherwise we will get another unshielded collider $Z \longrightarrow Y \longleftarrow W$, which contradicts the fact that the equivalence class has only one unshielded collider. Similarly, we get two more directed edges $W \longrightarrow U$ and $W \longrightarrow V$. Given the path $X \longrightarrow Y \longrightarrow W$, we immediately get the edge $X \longrightarrow W$ since any element in the equivalence class must be a DAG (which does not contain loops). Similarly, given $X \longrightarrow W \longrightarrow U$, the edge connecting X and U must be $X \longrightarrow U$. Therefore, the directions of many undirected edges in the skeleton can be determined using simple rules (e.g. we can not introduce new unshielded colliders or cycles.). Therefore the size of the

equivalence class can be greatly reduced. For the PDAG in Figure 18.16 (b), the only edge that we can not determine its directionality is U—V, which lead to two potential DAGs.

We define a special PDAG, named *completed PDAG (CPDAG)*, to compactly represent an equivalent class:

18.78 Definition. (CPDAG) *Let G be a DAG and K be a PDAG. K is a completed PDAG (or CPDAG) with respect to the equivalence class of G: if (i) $\text{skeleton}(K) = \text{skeleton}(G)$; (ii) K contains a directed edge $X \longrightarrow Y$ if and only if any DAG G' that is Markov equivalent to G contains the same directed edge $X \longrightarrow Y$.*

In other words, if an edge is directed in a CPDAG, all DAGs in the equivalent class agree on the direction of this edge. If an edge is undirected, then there are at least two DAGs within the equivalence class, such that they disagree on the direction of this edge.

Given a distribution P that is faithful to G, we can only identify the CPDAG of G. The population PC algorithm takes a distribution P as input and return a CPDAG. The algorithm starts from a complete, undirected graph and recursively deletes edges based on conditional independence decisions. This yields an undirected skeleton graph which can then be partially directed and further extended to represent the CPDAG. The algorithm has two parts: (i) identify the DAG skeleton; (ii) identify the CPDAG. After introducing the population PC algorithm, We describe a sample version PC algorithm that can reliably recover the true CPDAG purely based on observational data for Gaussian models.

18.9.2 PC Algorithm, Step 1: Identifying the Skeleton

Let P be a distribution that is faithful to G. We want to construct an undirected graph S such that $S = \text{skeleton}(G)$. The algorithm is based on evaluating conditional independence relationships of the form

$$X_i \perp\!\!\!\perp X_j \mid A \tag{18.79}$$

for different subsets of variables A. For finite data, these evaluations are based on statistical tests of conditional independence. The basic idea is if X_i and X_j are adjacent in G, then $X_i \perp\!\!\!\perp X_j \mid A$ does not hold for any A. This is summarized in the next theorem.

18.80 Theorem. *Let G be a DAG and P be a distribution that is faithful to G. If X_i and X_j are adjacent in G, then the conditional independence test $X_i \perp\!\!\!\perp X_j \mid A$ fails for all $A \subset V \setminus \{i, j\}$. On the other hand, if X_i and X_j are not adjacent in G, then either $X_i \perp\!\!\!\perp X_j \mid \pi(X_i)$ or $X_i \perp\!\!\!\perp X_j \mid \pi(X_j)$, where $\pi(X_i), \pi(X_j)$ are the parent sets of X_i and X_j in the DAG G.*

Proof. For the first part, with no loss of generality, we assume a directed edge $X \longrightarrow Y$ is in

G. For any $A \subset V \backslash \{i, j\}$, there is no way for A to d-separate X_i and X_j. Since P is faithful to G, we know that X_i and X_j must be conditionally dependent for any $A \subset V \setminus \{i, j\}$.

For the second part, we consider two cases: (i) X_j is a descendant of X_i and (ii) X_j is not a descendant of X_i. By the definition of d-separation, in the first case we can show that $X_i \perp\!\!\!\perp X_j \mid \pi(X_j)$, and in the second case we have $X_i \perp\!\!\!\perp X_j \mid \pi(X_i)$. □

From Theorem 18.80, we see that to determine whether X_i and X_j has an undirected edge in skeleton(G), we need to check whether there exists $A \subset V \setminus \{i, j\}$, such that $X_i \perp\!\!\!\perp X_j \mid A$. If we can find such an A, it is enough to ensure that X_i and X_j are not adjacent. The corresponding A is called a *separating set* for X_i and X_j. In the next section, we will show that these separating sets are useful for determining unshielded colliders. If X_i and X_j are conditionally independent given A, they must be conditionally independent given any superset of A. Therefore, we can search the set of possible separating sets in order of increasing size.

To make sure that X_i and X_j are actually adjacent in G, we need to exhaustively check that $X_i \perp\!\!\!\perp X_j \mid A$ fails for all $A \subset V \setminus \{i, j\}$. This is computationally intensive. The corresponding algorithm is shown in Figure 18.17. where adj(C, i) represents the adjacency nodes of X_i in an undirected graph C. The outer loop $k = 0, 1 \ldots, d$ indexes the size of the separating sets. In the next paragraph, we will show that the number of iterations of this outer loop is upper bounded by the maximum degree of skeleton(G). Also, it is easy to see that if skeleton(G) is sparse, the algorithm will converge much faster.

Let $K \equiv$ the maximum degree of skeleton(G). The following theorem shows the correctness of Algorithm 18.17.

18.81 Theorem. *Let G be a DAG and P be a distribution that is faithful to G. Then the algorithm in Figure 18.17 correctly reconstruct skeleton(G). Furthermore, let*

$$\ell_{stop} \equiv \textit{the maximum reached value of } \ell \textit{ in the outer loop}. \tag{18.82}$$

Then either $\ell_{stop} = K - 1$ or $\ell_{stop} = K$.

Proof. Let C_{out} be the final output from the algorithm. We know the algorithm only removes an edge X_i—X_j from the current skeleton graph if a separating set $A \subset V \setminus \{i, j\}$ is found such that $X_i \perp\!\!\!\perp X_j \mid A$. From Theorem 18.80, we know that all the removed edges do not belong to skeleton(G). Therefore skeleton$(G) \subset C_{\text{out}}$. For the other direction, if for any edge X_i—$X_j \in C_{\text{out}}$, it follows that in G, X_i and X_j are not d-separated given any subset of the adjacencies of X_i or any adjacencies of X_j in C_{out}. Since the adjacencies of X_i is a superset of $\pi(X_i)$ and the adjacencies of X_j is a superset of $\pi(X_j)$, X_i and X_j are not d-separated given $\pi(X_i)$ and $\pi(X_j)$ in G. It then follows from Theorem 18.80 that X_i and X_j must be adjacent in G.

We now show that $\ell_{stop} = K - 1$ or $\ell_{stop} = K$. Since we just proved that $C_{\text{out}} =$ skeleton(G), it's obvious that $\ell_{stop} \leq K$. We now show that $\ell_{stop} \geq K - 1$. Suppose the contrary. Then $\ell_{stop} \leq K - 2$. We could then continue with a further iteration in the

Population PC Algorithm, Step 1: Skeleton Identification

Input: Vertex set V and joint distribution P.
Initialize $C \leftarrow$ complete undirected graph on V; $\mathcal{A}_{ij} \leftarrow \Phi$ for any $i, j \in V$.
Loop through $k = 0, 1, \ldots, d$:

- **Loop** through any two adjacent nodes X_i, X_j, such that $|\text{adj}(C, i) \setminus \{j\}| \geq k$:
  - **For** every $A \in |\text{adj}(C, i) \setminus \{j\}|$ with $|A| = k$
    If $X_i \perp\!\!\!\perp X_j \mid A$
    * Remove X_i—X_j from C;
    * $\mathcal{A}_{ij} \leftarrow A$;
    * **Break**;
    End if
  - **End for**

Output: Estimated skeleton C and the class of separating sets $\mathcal{A}$.

Figure 18.17. *The population PC algorithm on skeleton identification.*

algorithm since $\ell_{stop} + 1 \leq K - 1$ and there is at least one node X_j with neighborhood-size $|\text{adj}(G, j)| = K$; that is, the reached stopping level would at least be $K - 1$ which contradicts the assumption that $\ell_{stop} \leq K - 2$. □

To understand the time complexity of the PC algorithm for skeleton identification, the algorithm uses N conditional independence checks where N is at most

$$N \leq \binom{d}{2} \sum_{k=1}^{K} \binom{d-1}{k} \leq \frac{d^{K+1}}{2(d-1)}. \tag{18.83}$$

Here K is the maximal degree of any vertex in skeleton(G); the worst case complexity is thus exponential.

18.9.3 PC Algorithm, Step 2: Identifying the CPDAG

Once the skeleton C_{out} and the class of separating sets $\mathcal{A}$ are obtained, the second step of the PC algorithm is to identify all the unshielded colliders and further identify the CPDAG. We consider all of the triples X_i—X_k—X_j with X_i and X_j nonadjacent in C_{out} as *candi-*

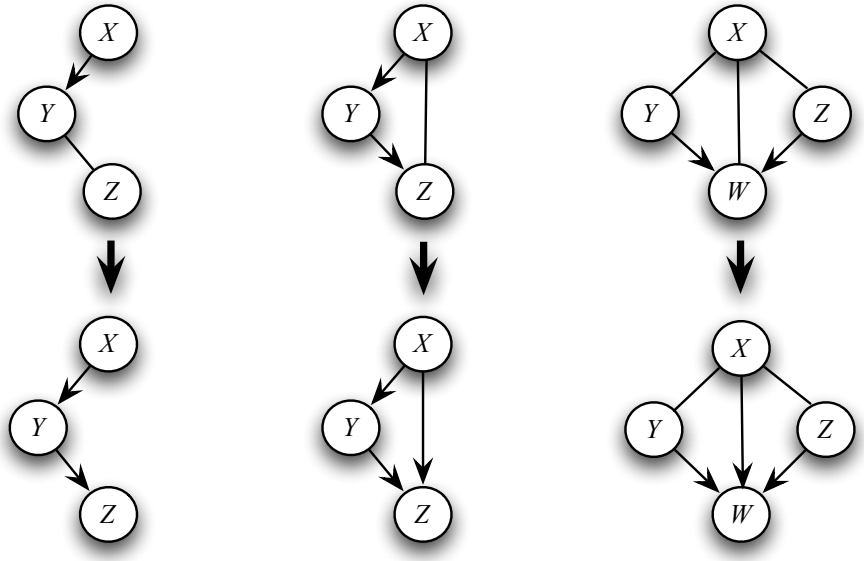
date unshielded colliders. The following theorem gives necessary and sufficient conditions under which a candidate is actually an unshielded collider.

18.84 Theorem. *Let G be a DAG and P be a distribution that is faithful to G. Assume we use P as input and the PC algorithm part 1 outputs an identified skeleton C and a class of separating sets $\mathcal{A}$. A candidate unshielded collider X_i—X_k—X_j in C is an unshielded collider $X_i \longrightarrow X_k \longleftarrow X_j$ if and only if $X_k \notin \mathcal{A}_{ij}$.*

Proof. Since X_i and X_j are nonadjacent, there exists a non-empty set $A = \mathcal{A}_{ij}$ such that $X_i \perp\!\!\!\perp X_j \mid A$. First, we show that if X_i—X_k—X_j is an unshielded collider $X_i \longrightarrow X_k \longleftarrow X_j$, then $X_k \notin A$. Suppose on the contrary that $X_k \in A$, then *conditioning* on A, since X_k is also conditioned on, there is no way for X_i and X_j to be d-separated. Thus it is impossible that $X_i \perp\!\!\!\perp X_j \mid A$, which leads to contradiction.

Next, we show that if X_i—X_k—X_j is not an unshielded collider, then $X_k \in A$. To see this, since X_i—X_k—X_j is not an unshielded collider, there are only three possible cases: $X_i \longrightarrow X_k \longrightarrow X_j$, $X_i \longleftarrow X_k \longleftarrow X_j$, and $X_i \longleftarrow X_k \longrightarrow X_j$. In any case, to d-separate X_i and X_j, X_k must be conditioned on. Therefore, we have $X_k \in A$. □

From Theorem 18.84, it is easy for us to evaluate a candidate unshielded collider X_i—X_k—X_j; we only need to check if X_k belongs to the separating set of X_i and X_j.



(a) R1: unshielded collider (b) R2: acyclicity (c) R3: hybrid rule

Figure 18.18. *Three basic configurations for which directions of certain edges can be determined: (a) R1: unshielded collider rule; (b) R2: acyclicity rule. (c) R3: hybrid rule.*

Once the skeleton and all the unshielded colliders are given, the next task is to identify

the whole CPDAG. This requires us to infer the directions of certain edges based on existing knowledge. Three simple rules are provided in Figure 18.18. The first rule R1 is based on the idea that if a local configuration $X \longrightarrow Y—Z$ is not an unshielded collider, then the edge connecting Y and Z must be $Y \longrightarrow Z$. Otherwise, we will form an unshielded collider. The second rule R2 is based on the idea that the graph is a DAG, thus no cycles are allowed. The third rule R3 is in fact a combined application of R1 and R2. To see this, given the upper configuration in R3, let's assume we have a directed edge $W \longrightarrow X$. Then by R2, we know that the directions of edge $Y—X$ and $Z—X$ must be $Y \longrightarrow X$ and $Z \longrightarrow X$. This forms an unshielded collider $Y \longrightarrow X \longleftarrow Z$, which violates R1. These three rules can be applied in a dynamic way. Given a PDAG, a rule applies whenever a subgraph in the PDAG can be found that matches the upper parts of the three rules. In that case, we modify this subgraph by applying the rule and orienting a previously undirected edge. Such an operation then creates a new PDAG. The new PDAG may create another subset that match one of the three rules, leading to the orientation of more edges. We could then proceed this process until we obtain a PDAG on which none of these three rules can apply. This PDAG is then exported as the final identified CPDAG. The convergence of this procedure is obvious. We will show its correctness in a later theorem.

Population PC Algorithm, Step 2: CPDAG Identification

Input: The identified skeleton C and a class of separating sets $\mathcal{A}$.
Initialize $K \leftarrow C$.
For every pair of nonadjacent variables X_i, X_j with common neighbor X_k:

> **If** $k \notin \mathcal{A}_{ij}$ **Then** Replace $X_i—X_k—X_j$ by $X_i \longrightarrow X_k \longleftarrow X_j$;

End for
Loop until converge:

> Find a subgraph in K on which any rule R1-R3 in Figure 18.18 can apply;
>
> Apply the rule on the subgraph and add in the corresponding directions;

End loop
Output: Identified CPDAG K.

Figure 18.19. *The population PC algorithm on CPDAG identification.*

The algorithm in Figure 18.19 summarizes the whole process. With the estimated skeleton and the class of separating sets as input. The algorithm have two parts. In the first part, every candidate unshielded colliders are examined and all unshielded colliders are identi-

fied. In the section part, many undirected edges in the obtained PDAG are further oriented until no updates can be made. The final PDAG is then output as the identified CPDAG.

The next theorem secures the correctness of the algorithm in Figure 18.19, which was first proved by Meek (1995).

18.85 Theorem. *Let G be a DAG and P be a distribution that is faithful to G. Then the algorithm in Figure 18.19 correctly reconstruct the CPDAG of G.*

Proof Idea. In Theorem 18.81, we have already shown that the identified skeleton is correct. We only need to show all the directed and undirected edges in the output CPDAG K is indeed the same as the true CPDAG of G. To establish this result we need to show three properties of the obtained PDAG: (i) Property 1: the final graph returned by the algorithm is acyclic; (ii) If an edge $X \longrightarrow Y$ appears in K, then this edge appears in all DAGs of the equivalence class of G; (iii) If an undirected edge X—$Y \in K$, then we can find two DAGs G_1 and G_2. Both G_1 and G_2 are Markov equivalent to G and $X \longrightarrow Y \in G_1$ and $X \longleftarrow Y \in G_1$. The first two properties are straightforward. The third property requires additional machinery and is omitted here. More details can be found in (Meek, 1995). □

The time complexity of the second part of the PC algorithm is no larger than that of the first part on skeleton identification. Therefore, the total time complexity of the algorithm is $O(d^{K+1})$. However, one thing to keep in mind is that in applications, such a worst case scenario is seldom met.

18.9.4 PC Algorithm: Sample Version

In the population version PC algorithm, the only place where we utilize the population quantities is on the conditional independence query $X_i \perp\!\!\!\perp X_j \mid A$ with $A \subset V \setminus \{i, j\}$. If we only have observed data points, we need to test whether X_i and X_j are conditional independent given A. For Gaussian DAGs, such a test is very easy to construct.

Under the Gaussian DAG assumption, Theorem 18.29 says that

$$X_i \perp\!\!\!\perp X_j \mid A \text{ if and only if } \rho_{i,j \mid A} = 0, \tag{18.86}$$

where $\rho_{i,j \mid A}$ is the partial correlation between X_i and X_j given A. Therefore, to query whether $X_i \perp\!\!\!\perp X_j \mid A$, we only need to test the hypothesis

$$H_0 : \rho_{i,j \mid A} = 0 \text{ vs. } H_1 : \rho_{i,j \mid A} \neq 0. \tag{18.87}$$

Let $\widehat{\rho}_{i,j \mid A}$ be the sample partial correlation which can be calculated using the recursive formula (18.31). In order to test whether $\widehat{\rho}_{i,j \mid A} = 0$, we apply Fisher's Z-transform and define

$$\widehat{z}_{ij \mid A} = \frac{1}{2} \log \left(\frac{1 + \widehat{\rho}_{i,j \mid A}}{1 - \widehat{\rho}_{i,j \mid A}} \right) \text{ and } z_{ij \mid A} = \frac{1}{2} \log \left(\frac{1 + \rho_{i,j \mid A}}{1 - \rho_{i,j \mid A}} \right). \tag{18.88}$$

Classical distribution theory in the Gaussian case characterizes the asymptotic distribution of $\widehat{z}_{ij\,|\,A}$:

$$(\sqrt{n-|A|-3})(\widehat{z}_{ij\,|\,A} - z_{ij\,|\,A}) \xrightarrow{D} N(0,1), \tag{18.89}$$

where $|A|$ is the cardinality of A . Under the null hypothesis we have $z_{ij\,|\,A} = 0$, which suggests a level α test that we reject the null hypothesis if

$$(\sqrt{n-|A|-3})|\widehat{z}_{ij\,|\,A}| > \Phi^{-1}\left(1-\frac{\alpha}{2}\right), \tag{18.90}$$

where $\Phi(\cdot)$ is the cumulative function for the standard Gaussian random variable. The sample PC algorithm is almost identical to the population PC algorithm with the population conditional independence query on whether $X_i \perp\!\!\!\perp X_j \,|\, A$ replaced by a finite sample level α test (18.90).

Sample Version of the PC Algorithm

The Sample version PC algorithm is identical to the population PC algorithm but replace the conditional independence query on whether $X_i \perp\!\!\!\perp X_j \,|\, A$ by a finite sample level α test:

$$(\sqrt{n-|A|-3})|\widehat{z}_{ij\,|\,A}| > \Phi^{-1}\left(1-\frac{\alpha}{2}\right). \tag{18.91}$$

Figure 18.20. *The sample version PC algorithm.*

The large-sample properties of this sample version PC algorithm has been analyzed by Kalisch and Bühlmann (2007). They have the following assumptions:

(A1) The distribution P is multivariate Gaussian and faithful to the DAG G even when the dimension d increases with the sample n.

(A2) $d = O(n^a)$ for some $0 \le a < \infty$.

(A3) Let $q = \max_{1\le j\le d} |\text{adj}(G, j)|$. Then $q = O(n^{1-b})$ for some $0 < b \le 1$.

(A4) $\inf\{|\rho_{i,j\,|\,A}|; i, j, A \text{ with } \rho_{i,j\,|\,A} \neq 0\} \ge c_n$, where $c_n^{-1} = O(n^\gamma)$ for some $0 < \gamma < b/2$. Also $\sup_{i,j,A} |\rho_{i,j\,|\,A}| \le M < 1$. Here $0 < b \le 1$ is as in (A3).

18.92 Theorem. (Kalisch and Bühlmann (2007)) *Under Assumptions (A1) - (A4), we denote by* Γ *the true CPDAG. Let* $\widehat{\Gamma}_n^{\alpha_n}$ *be the estimated CPDAG from the sample PC algorithm*

with sample size n and level α_n for each conditional independence test. Then, there exists $\alpha_n \to 0$ as n goes to infinity, such that

$$\mathbb{P}\left(\widehat{\Gamma}_n^{\alpha_n} = \Gamma\right) = 1 - O(\exp(-Cn^{1-2d})) \to 1 \text{ as } n \to \infty \text{ for some } 0 < C < \infty,$$

where $d > 0$ is as in (A4).

18.9.5 Analysis of Cell-Signaling Networks

We apply the PC algorithm on a flow-cytometry dataset from Sachs et al. (2005) with $p = 11$ variables and $n = 7,466$ data points. Each data point corresponds to a cell and the variables correspond to the expression levels of proteins. The abbreviated variable names are: `Raf`, `Mek`, `Plcg`, `PIP2`, `PIP3`, `P44/42`, `Akt`, `PKA`, `PKC`, `P38`, `Jnk`. The only tuning parameter for the PC algorithm is the level α of the conditional independence tests. The larger the value α is, the sparser the estimated CPDAG will be. In this example, we use $\alpha = 0.01$ and the estimated CPDAG is shown in Figure 18.21.

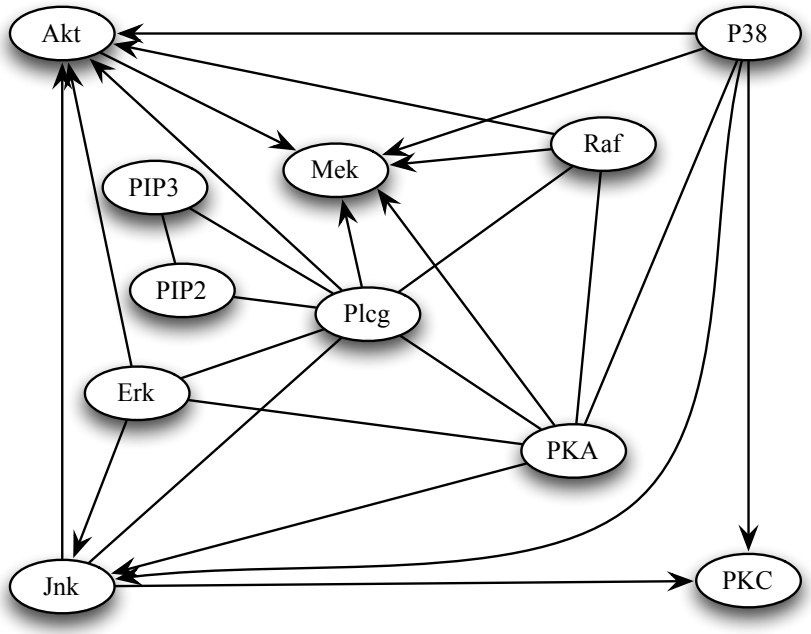


Figure 18.21. *An estimated CPDAG from a flow cytometry dataset, with $d = 11$ protein measured on $n = 7,466$ cells. The network structure is estimated using the PC algorithm with level $\alpha = 0.01$.*

In this example, the estimated CPDAG contains many undirected edges. We see that the variable `Mek` is pointed to by all variables that are connected with it. In the next chapter, we will estimate undirected Gaussian graphs on this same dataset, and discuss the relationships between directed acyclic graphs and undirected graphs.

18.10 Bibliographic Remarks

There are a number of texts on DAGs including Edwards (1995) and Jordan (2003). The first use of DAGs for representing causal relationships was by Wright (1934). Modern treatments are contained in Spirtes et al. (2000) and Pearl (2000). Robins et al. (2003) discuss the problems with estimating causal structure from data. Nice treatments on graphical model inference appears in Bishop (2007) and Alpaydin (2004). A very thorough and excellent discussion of DAGs can be found in Koller and Friedman (2009).

Exercises

18.1 Complete the proof of Theorem 18.5.

18.2 Show the equivalence of the following two statements:

- $p(x \mid y, z) = p(x \mid z)$ for all x, y and z
- $p(x, y \mid z) = p(x \mid z)p(y \mid z)$ for all x, y and z.

18.3 Prove Theorem 18.17.

18.4 Prove Theorem 18.26.

18.5 Prove Proposition 18.30.

18.6 Let X, Y and Z have the following joint distribution:

	$Y=0$	$Y=1$
$X=0$	.405	.045
$X=1$	.045	.005

$Z=0$

	$Y=0$	$Y=1$
$X=0$	.125	.125
$X=1$	.125	.125

$Z=1$

(a) Find the conditional distribution of X and Y given $Z = 0$ and the conditional distribution of X and Y given $Z = 1$.
(b) Show that $X \perp\!\!\!\perp Y \mid Z$.
(c) Find the marginal distribution of X and Y.
(d) Show that X and Y are not marginally independent.

18.7 Consider the three DAGs in Figure 18.12 without a collider. Prove that $X \perp\!\!\!\perp Z \mid Y$.

18.8 Consider the DAG in Figure 18.12 with a collider. Prove that $X \perp\!\!\!\perp Z$ and that X and Z are dependent given Y.

18.9 Let $X \in \{0, 1\}$, $Y \in \{0, 1\}$, $Z \in \{0, 1, 2\}$. Suppose the distribution of (X, Y, Z) is Markov to: $X \longrightarrow Y \longrightarrow Z$. Create a joint distribution $p(x, y, z)$ that is Markov to this DAG. Generate 1000 random vectors from this distribution. Estimate the distribution from the data using maximum likelihood. Compare the estimated distribution to the true distribution. Let $\theta = (\theta_{000}, \theta_{001}, \ldots, \theta_{112})$ where $\theta_{rst} = \mathbb{P}(X = r, Y = s, Z = t)$. Use the bootstrap to get standard errors and 95 percent confidence intervals for these 12 parameters.