Directed graphical models

Kevin P. Murphy

Last updated November 7, 2007

1 Graphical models

We have already seen how **conditional independence** (CI) assumptions help to represent joint distributions in terms of smaller pieces (see the chapter on naive Bayes classifiers, Chapter **??**). We give a very simple example in Figure 1, where we show how the joint distribution $p(X, Y)$ can be represented in terms of the marginal distributions $p(X)$ and $p(Y)$ if we assume that X and Y are (unconditionally) independent, $X \perp Y$.

Graphical models provide a way to represent CI assumptions pictorially, in terms of graph (network) structures. The nodes represent random variables, and the (lack of) edges represent CI assumptions. For the example in Figure 1, we would draw a graph with two nodes, X and Y, with no edge between them, representing the fact that $X \perp Y$.

To define a *specific* probability model, we need to associate parametric functions with the nodes in the graph. We will explain how to do this in detail below, but as a very simple example, for the model in Figure 1, we would associate $p(x)$ with node X and $p(y)$ with node Y, where $p(x)$ and $p(y)$ are tables of 6 numbers each. We will write the resulting probability model as $p(\mathbf{x}|G, \boldsymbol{\theta})$, where G is the graph structure, $\boldsymbol{\theta}$ are the parameters of the model, and $\mathbf{x} = (x_1, \ldots, x_d)$ are the nodes in the graph.

Note that $\boldsymbol{\theta}$ may encode additional CI relations that are not encoded in the graph. For example, in Figure 1, if the graph were $X - Y$ (meaning X and Y are dependent), but the joint table $p(X, Y)$ were a diagonal matrix, then $X \perp Y$ would still hold, but this would be by virtue of the specific parameter values θ in the table $p(X, Y)$, rather than because of the graph structure. One of the principles of graphical modeling is to try to make as many CI relations graphically explicit as possible. A distribution that does not contain any non-graphical CI relations is said to be **faithful** to the graph. Many real-world problems contain **context-specific independence**, which essentially means that the graph structure changes depending on the values of the nodes. This cannot be captured using standard graphical modeling techniques.

There are three main kinds of graphical model:

- **Directed graphical models** (DGMs), also called **Bayesian networks** or **belief networks**. (The term "belief" is sometimes used to represent a (posterior) probability distribution.) These models require that the graph is a **directed acyclic graph** (DAG). Note that what makes "Bayesian networks" Bayesian is just the fact that they are a convenient way to represent probability distributions, which it is at the core of Bayesian statistics. It is possible to use frequentist parameter estimation techniques in conjunction with "Bayesian networks", as we will see in Chapter **??**.

- **Undirected graphical models** (UGMs), also called **Markov random fields** (MRFs) or **Markov networks**. These can use any undirected graph structure.

- **Chain graphs**, which are a combination of DGMs and UGMs, and have directed and undirected edges. Loosely speaking, they are DAGs, but some nodes have internal structure represented as an undirected graph.

These model classes have different **expressive power**, in the sense that they represent different sets of distributions, as we will explain below. See Figure 2.

Each graph represents a set of CI relations (in a way which we shall shortly define); let us call these $I(G)$. Any given probability distribution p also defines a set of CI relations; let us call these $I(p)$. We say that a graph G is an **I-map** (independence map) for distribution p if $I(G) \subseteq I(p)$. In other words, the graph does not make any assertions of CI which are not true of the distribution. This allows us to use the graph as a safe proxy for p when reasoning about

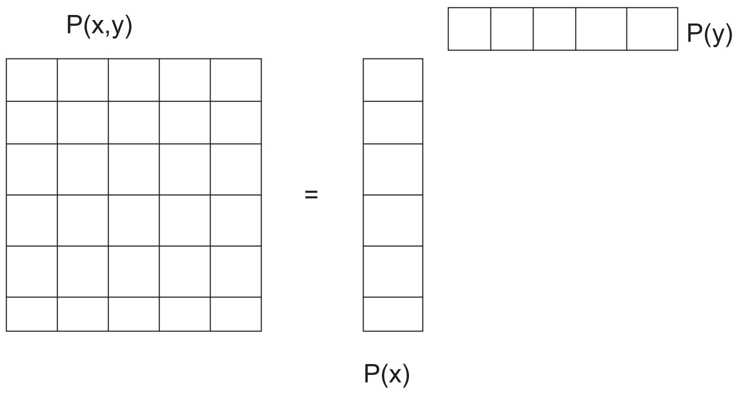


Figure 1: Computing $p(x, y) = p(x)p(y)$, where $X \perp Y$. Here X and Y are discrete random variables; X has 6 possible states (values) and Y has 5 possible states. A general joint distribution on two such variables would require $(6 \times 5) - 1 = 29$ parameters to define it (we subtract 1 because of the sum-to-one constraint). By assuming (unconditional) independence, we only need $(6 - 1) + (5 - 1) = 9$ parameters to define $p(x, y)$. Source: Sam Roweis.

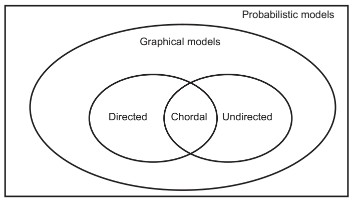


Figure 2: Venn diagram representing relationships between different kinds of graphical models.

p's CI properties. This is helpful for designing algorithms that work for large classes of distributions, regardless of the specific numerical parameters θ. 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 say that a graph G is a **perfect map** of a distribution p if $I(G) = I(p)$. In other words, the graph captures all and only the CI relations of the distribution. It turns out that DGMs and UGMs are perfect maps for different sets of distributions. In this sense, neither is more powerful than the other as a representation language. However, there are some distributions that can be perfectly modeled by either a DGM or a UGM; the resulting graphs are called **decomposable** or **chordal**. We shall define these concepts later.

A simple example of a decomposable graph is a **chain**: directed chains and undirected chains represent the same set of probability distributions, namely those that satisfy the (first order) **Markov property**

$$X_{t-1} \perp X_{t+1} | X_t \quad t = 2, \ldots, d-1 \tag{1}$$

In words, this says that the future X_{t+1} is independent of the past X_{t-1} given the present X_t. See Figure 3. Another example of a decomposable graph is a **tree**. For example, the graphs in Figure 4 represent distributions that satisfy the CI relations

$$X_i \perp X_j | Y \quad i, j \in \{1, \ldots, d\}, i \neq j \tag{2}$$

In the context of naive Bayes classifiers, this says that the features X_j are conditionally independent given the class label Y.

The graph structure reflects our beliefs about the domain. If the graph structure is "wrong", the resulting model will be a poor prediction of the future. There are various ways of assessing **goodness of fit** of models, including graphical models; see e.g., [CDLS99] for a discussion. Also, we can use model selection techniques, such as **cross validation**, to choose between different model structures. We will discuss this later.

In this chapter, we focus on DGMs, and return to UGMs in Chapter **??**.

$X_1 \to X_2 \to X_3 \to X_d$
$X_1 - X_2 - X_3 - X_d$

Figure 3: Directed and undirected chain.

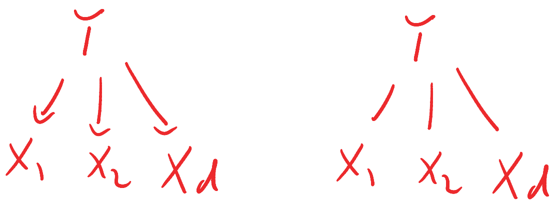


Figure 4: Directed and undirected tree structure.

2 DGMs provide a compact representation of joint probability distributions

By the chain rule of probability, any joint distribution can be written as follows

$$p(x_{1:d}) = p(x_1)p(x_2|x_1)p(x_3|x_1,x_2)\dots p(x_d|x_{1:d-1}) \tag{3}$$

for an arbitrary ordering of the variables. In Section 3.5 below, we will show that a DAG G implies the following CI relations:

$$X_j \perp X_{\text{pred}_j \setminus \pi_j} | X_{\pi_j} \tag{4}$$

where π_j are the **parents** of node j in G and pred$_j$ are the predecessors of j in any **topological ordering** of G. (This is an ordering of the nodes in which parents preceed their children.) Note that $\pi_j \cup \text{pred}_j = \{1, \dots, j-1\}$. Hence we can simplify Equation 3 to

$$p(x_{1:d}) = p(x_1)p(x_2|x_{\pi_1})p(x_3|x_{\pi_3})\dots p(x_d|x_{\pi_d}) \tag{5}$$

where each of the terms $p(x_j|x_{\pi_j})$ is called a **conditional probability distribution** (CPD).

Let us consider the "water sprinkler" example in Figure 5. This defines a joint distribution over 4 binary variables, representing whether it is cloudy or not (C), whether it is raining or not (R), whether the water sprinkler is on or not (S), and whether the grass is wet or not (W). It is common to use the heuristic that we should add an edge from X_{π_j} to X_j if X_{π_j} are all the immediate causes of X_j. Since clouds cause rain, there is an arc from C to R. Since we are assuming this is a light-sensitive sprinkler, there is an arc from C to S. Finally, since either the sprinkler or the rain can cause the grass to be wet, there are arcs from S and R to W.

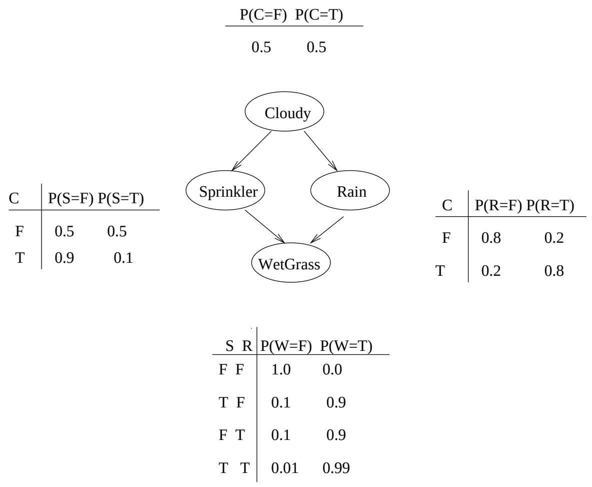
This graph encodes various CI assumptions. For example, the lack of arc between S and R means $S \perp R|C$, since C is a predecessor of S and R in any topological ordering. Similarly, the lack of arc between C and W means $W \perp C|S, R$. Informally, this means that the clouds do not have any direct effect on the wet grass; rather, this effect is mediated via S and R. Sometimes we say that S and R **screen off** W from C. These two CI assumptions allow us to represent the joint as a product of local factors, one per node:

$$\begin{aligned}
P(C,S,R,W) &= P(C)P(S|C)P(R|S,C)P(W|S,R,C) \text{ chain rule} && (6)\\
&= P(C)P(S|C)P(R|\cancel{S},C)P(W|S,R,C) \text{ since } S \perp R|C && (7)\\
&= P(C)P(S|C)P(R|\cancel{S},C)P(W|S,R,\cancel{C}) \text{ since } W \perp C|S,R && (8)\\
&= P(C)P(S|C)P(R|C)P(W|S,R) && (9)
\end{aligned}$$

If we multiply all these CPDs together, we get the joint distribution shown in Table 1.

In this example, each CPD $p(X_j|X_{\pi_j})$ is represented as a multidimensional table. These are called **conditional probability tables** (CPTs). Formally, each row of the table are the parameters of a conditional multinomial distribution; we have one row for each possible conditioning case, i.e., combination of parent values. For example, for node W, we have

$$p(W = w|S = s, R = r, \boldsymbol{\theta}_w) = Mu(w|\boldsymbol{\theta}_{w,s,r}, 1) \tag{10}$$



P(C=F)	P(C=T)
0.5	0.5

C	P(S=F)	P(S=T)
F	0.5	0.5
T	0.9	0.1

C	P(R=F)	P(R=T)
F	0.8	0.2
T	0.2	0.8

S	R	P(W=F)	P(W=T)
F	F	1.0	0.0
T	F	0.1	0.9
F	T	0.1	0.9
T	T	0.01	0.99

Figure 5: Water sprinkler Bayes net with CPDs shown. T and F stand for true and false.

C	S	R	W	Probability
0	0	0	0	0.200
0	0	0	1	0.000
0	0	1	0	0.005
0	0	1	1	0.045
0	1	0	0	0.020
0	1	0	1	0.180
0	1	1	0	0.001
0	1	1	1	0.050
1	0	0	0	0.090
1	0	0	1	0.000
1	0	1	0	0.036
1	0	1	1	0.324
1	1	0	0	0.001
1	1	0	1	0.009
1	1	1	0	0.000
1	1	1	1	0.040

Table 1: Joint distribution defined by the water sprinkler model. Here 0 represents False and 1 represents True.

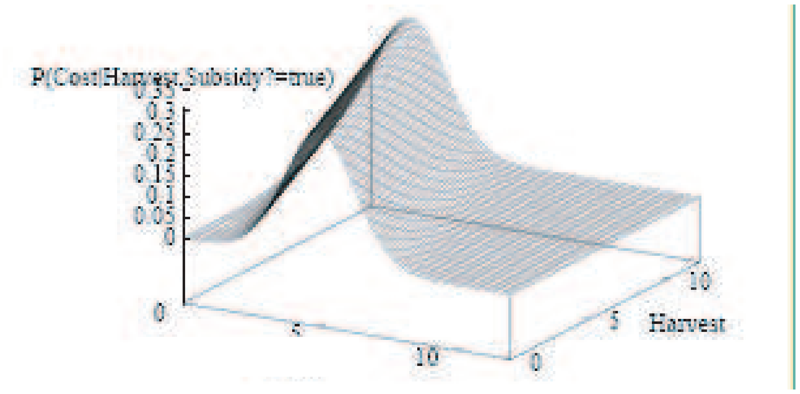


Figure 6: Linear Gaussian CPD $p(y|x) = \mathcal{N}(Y|a + bx, \sigma^2)$, where a is the offset and b is the slope. Source: [RN02] Figure 14.6a(a).

where $\boldsymbol{\theta}_w$ are all the parameters for node W, and each row is indexed by s and r. For the **root** nodes, which have no parents, the CPTs are just unconditional multinomial distributions.

The number of parameters in this representation is

$$(K_j - 1) \prod_{k \in \pi_j} K_k = O(K^{1+|\pi_j|}) \tag{11}$$

where K_j is the number of states for node j, and $K = \max_j K_j$. (We subtract 1 because each row of the CPT sums to one.) In the water sprinkler example, there are

$$1 + (2-1) \times 2 + (2-1) \times 2 + (2-1) \times 4 = 1 + 2 + 2 + 4 = 9 \tag{12}$$

free parameters. In contrast, an unconstrained joint distribution on 4 binary variables has $2^4 - 1 = 15$ parameters. In general, a DGM can have exponentially fewer parameters than an unconstrained distribution. Thus the DGM requires less space to store in memory, and it will require less data (smaller sample size) to learn its parameters from data.

2.1 Compact representations of CPDs

CPTs require a number of parameters that is exponential in the number of parents. We give an example of a more compact representation of a CPD for discrete variables in Section 5.2.

If the child node is continuous valued, we replace the multinomial distribution with a different kind of distribution, say Gaussian. Also, if the parents are continuous, we cannot use a different parameter value for every possible parent configuration; instead we will need a functional mapping from parent values to child values. For example, if W, S and R represent the amount of wet grass/ sprinkler flow/ rain fall, we could use

$$p(w|s, r, \boldsymbol{\theta}_w) = \mathcal{N}(w|a + b_1 s + b_2 r, \sigma^2) \tag{13}$$

where $\boldsymbol{\theta}_w = (a, \mathbf{b}, \sigma^2)$ are the parameters of W's CPD. We can write this more compactly using scalar product notation:

$$p(w|s, r, \boldsymbol{\theta}_w) = \mathcal{N}(w|\boldsymbol{\beta}^T[1; s; r], \sigma^2) \tag{14}$$

where $\boldsymbol{\beta} = (a; b_1; b_2)$ and $[1; s; r]$ are column vectors. This is a simple example of **linear regression**, where the mean of the child is a linear function of its parents, and the variance is a constant (see Figure 6). If all the CPDs are Gaussian or linear-Gaussian, the corresponding joint distribution $p(x_{1:d})$ is **multivariate Gaussian**. Unlike the water sprinkler example, we cannot represent this as a table, but we *can* represent it compactly in terms of a mean vector and covariance matrix: see Chapter **??**.

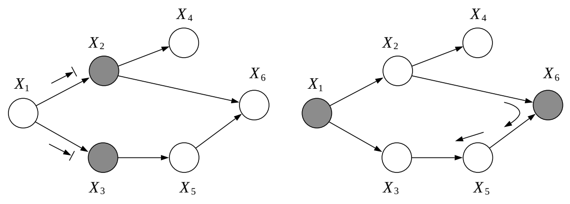


Figure 7: Two examples of the Bayes ball algorithm

3 Conditional independence properties encoded by DAGs

We have said that DAGs encode a set of CI relations. There are various ways to "read off" such relations from the graph; we will consider three below.[1] The goal is to be able to answer questions of the form

$$X_A \perp X_B | X_S \tag{15}$$

for any (disjoint) set of nodes A, B, and S. Each method will give the same answer, so you are free to pick whichever is most convenient.

3.1 d-separation

We say $X_1 - X_2 \cdots - X_n$ is an **active path** in a DAG G given evidence E if

1. Whenever we have a **v-structure**, $X_{i-1} \rightarrow X_i \leftarrow X_{i+1}$, then X_i or one of its descendants is in E; and
2. no other node along the path is in E

In other words, if any of the nodes along the path are observed, then they must be at the bottom of a v-structure. We also say X is **d-separated** from Y given E if there is no active path from any $x \in X$ to any $y \in Y$ given E. (This is like regular graph separation, but takes the direction of the edges into account.) Then we have

Theorem 3.1. *$x_A \perp x_B | x_C$ if every variable in A is d-separated from every variable in B conditioned on all the variables in C.*

Consider the DAG in Figure 7. Suppose we want to know if

$$x_1 \perp x_6 | \{x_2, x_3\} \tag{16}$$

We shade the nodes that we are conditioning on, namely x_2 and x_3: see Figure 7(left). We see that 1 is d-separated from 6, since the evidence breaks the 1-2-4 and the 1-3-5 paths (makes them inactive). Hence $x_1 \perp x_6 | \{x_2, x_3\}$ is true.

Now suppose we want to know if

$$x_2 \perp x_3 | \{x_1, x_6\} \tag{17}$$

We shade nodes 1 and 6 in Figure 7(right). Now there is an active path 2-6-5-3, since $x_6 \in E$. Hence $x_2 \not\perp x_3 | \{x_1, x_6\}$.

3.2 Bayes ball algorithm

To check if $x_A \perp x_B | x_C$ we need to check if every variable in A is d-separated from every variable in B conditioned on all variables in C. The **Bayes ball algorithm** is a simple way to implement this test. We shade all nodes x_C, place "balls" at each node in x_A (or x_B), let them bounce around according to some rules, and then ask if any of the balls reach any of the nodes in x_B (or x_A). The three main rules are shown in Figure 8. Notice that balls can travel opposite to edge directions. We also need the boundary conditions, which are shown in Figure 9.

[1]Reading off CI relations is tricky, because one needs to worry about the directionality of the edges. There are several ways to do this. In contrast, in the UGM case, there will be only one way.

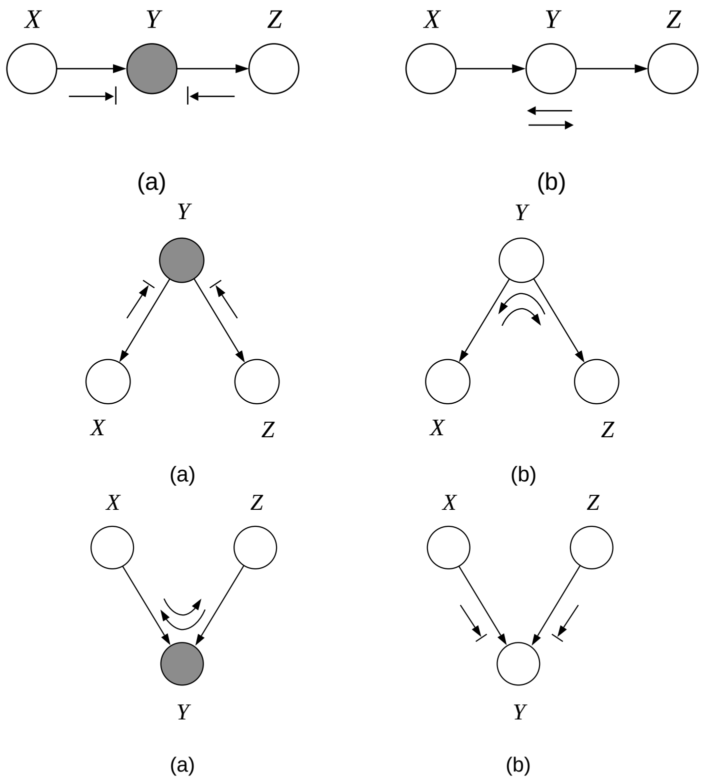


Figure 8: Bayes ball rules. A shaded node is one we condition on. If there is an arrow with a vertical bar it means the ball cannot pass through; otherwise the ball can pass through.

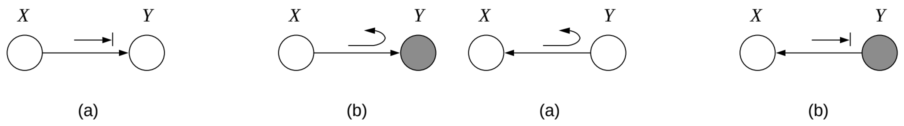


Figure 9: Bayes ball boundary conditions. A curved arrow means the ball “bounces back”.

We can derive the 3 main rules as follows. First consider a chain structure $X \rightarrow Y \rightarrow Z$. When we condition on y, are x and z independent? We have

$$p(x,y,z) = p(x)p(y|x)p(z|y) \tag{18}$$

which implies

$$\begin{aligned} p(x,z|y) &= \frac{p(x)p(y|x)p(z|y)}{p(y)} && (19) \\ &= \frac{p(x,y)p(z|y)}{p(y)} && (20) \\ &= p(x|y)p(z|y) && (21) \end{aligned}$$

and therefore $x \perp z|y$. So observing the middle node of chain breaks it in two. Think of x as the past, y as the present and z as the future.

Now consider the structure $X \leftarrow Y \rightarrow Z$. When we condition on y, are x and z independent?

$$p(x,y,z) = p(y)p(x|y)p(z|y) \tag{22}$$

which implies

$$\begin{aligned} p(x,z|y) &= \frac{p(x,y,z)}{p(y)} && (23) \\ &= \frac{p(y)p(x|y)p(z|y)}{p(y)} = p(x|y)p(z|y) && (24) \end{aligned}$$

and therefore $x \perp z|y$ So observing a root node separates its children.

Finally consider a **v-structure** $X \rightarrow Y \leftarrow Z$. When we condition on y, are x and z independent? We have

$$p(x,y,z) = p(x)p(z)p(y|x,z) \tag{25}$$

so we see that x and z are *marginally independent*, but given y they are *conditionally dependent*. This important effect is called **explaining away** (and is also known as Berkson's paradox). Thus observing a child at the bottom of a v-structure makes its parents become inter-dependent.

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).

3.3 Convert to UGM

As we discuss in Chapter **??**, given an *undirected* graph H, we say that $X_A \perp_p X_B|X_S$ in the distribution p iff A separates B from S in H, i.e., if we cut all the edges touching nodes in S, there will no paths from any node in A to any node in B. We will write this as $A \perp_H B|S$. (We use X_j for the random variables, and j for the nodes in the graph.)

One strategy for inferring CI relations in DAGs is to convert the DAG to a UGM, and then apply the above method. This conversion process has two steps:

1. Form the **ancestral** graph of G with respect to $U = \{A, B, S\}$. This means we remove all nodes from G that are not in U or are not ancestors of U. Let the result be denoted by $A = \text{ancestral}(G, U)$. See Figure 10 for an example. (The reason we need to form the ancestral graph is to prevent us from concluding that hidden nodes (not part of A, B or S) at the bottom of v-structures cause their parents to become dependent.)

2. Form the **moral** graph of A. This means we connect together all "unmarried" (disconnected) parents who share a common child, by adding **moral edges**, thus converting the **family** (parents and children) into a fully connected **clique**. See Figures 11,12 for some examples. Let the result be denoted by $H = \text{moral}(A)$. (The reason we need to moralize the graph is to capture the fact that parents become dependent given their children, due to explaining away.)

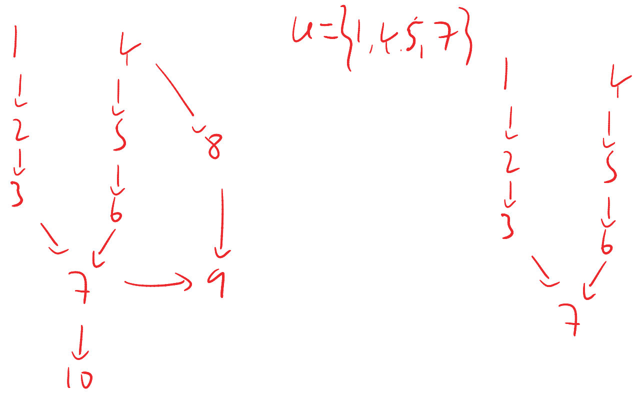


Figure 10: Example of how to form an ancestral graph.

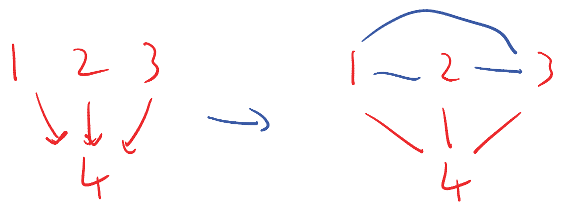


Figure 11: Example of how to form a moral graph. The blue edges are the newly added moral edges.

It can then be shown that

$$X_A \perp_p X_B | X_S \iff A \perp_H B | S \tag{26}$$

where

$$H = \text{moral}(\text{ancestral}(G, A \cup B \cup S)) \tag{27}$$

is the corresponding UGM. For example, combining Figure 10 and Figure 12, we can conclude that

$$X_1 \perp X_4 | \{X_5, X_7\} \tag{28}$$

3.4 Markov blankets

The above methods allow us to determine if $X_A \perp X_B | X_S$ for any sets of nodes A, B and S. Suppose $A = \{i\}$. The minimal set of nodes that renders i independent of all the rest of the nodes is called i's **Markov blanket**, MB_i:

$$X_i \perp X_{R_i} | X_{MB_i} \tag{29}$$

where

$$R_i = \{1, \ldots, d\} \setminus \{i\} \setminus MB_i \tag{30}$$

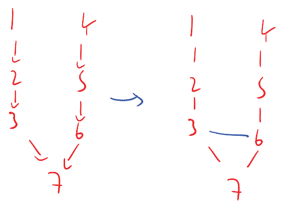


Figure 12: Example of how to form a moral graph. The blue edges are the newly added moral edges.

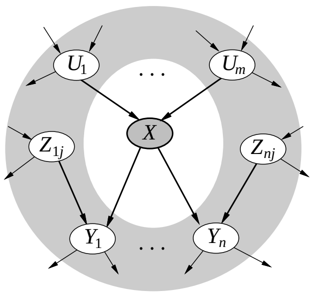


Figure 13: A node X is independent of all other nodes given its Markov blanket, which include its parents U, children Y and coparents Z. Source: [RN02] Fig 14.4.

are the rest of the nodes excluding i and its Markov blanket. Below we will show that i's Markov blanket are its parents $U_1, \ldots, U_m$, its children $Y_1, \ldots, Y_n$, and its **co-parents**, i.e., other nodes Z who are also parents if the Y_j. (The reason i depends on the co-parents is because of explaining away.) See Figure 13.

To see why this is true, partition all the nodes into X_i and the other nodes, X_{-i}. We can partition the other nodes X_{-i} in those that involve X_i (namely its parents, its children, and its co-parents), and the other nodes, O. Then the **full conditional** is given by

$$\begin{aligned}
p(X_i|X_{-i}) &= \frac{p(X_i, X_{-i})}{\sum_x p(X_i = x, X_{-i})} && (31)\\
&= \frac{p(X_i, U_{1:n}, Y_{1:m}, Z_{1:m}, O)}{\sum_x p(X_i = x, U_{1:n}, Y_{1:m}, Z_{1:m}, O)} && (32)\\
&= \frac{p(X_i|U_{1:n})[\prod_j p(Y_j|X_i, Z_j)]P(U_{1:n}, Z_{1:m}, O)}{\sum_x p(X_i = x|U_{1:n})[\prod_j p(Y_j|X_i = x, Z_j)]P(U_{1:n}, Z_{1:m}, O)} && (33)\\
&= \frac{p(X_i|U_{1:n})[\prod_j p(Y_j|X_i, Z_j)]}{\sum_x p(X_i = x|U_{1:n})[\prod_j p(Y_j|X_i = x, Z_j)]} && (34)\\
&\propto p(X_i|Pa(X_i)) \prod_{Y_j \in ch(X_i)} p(Y_j|Pa(Y_j)) && (35)
\end{aligned}$$

so the terms that do not involve X_i cancel out from the numerator and denominator. We are left with a product of terms that include X_i in their "scope". This proves that $X_i \perp X_{R_i}|MB_i$.

The ability to sample from the full conditional distributions $p(X_i|X_{-i})$ of each node will prove crucial to the **Gibbs sampling** algorithm (see Section **??**).

3.5 Local Markov property

One consequence of the global Markov properties definined by d-separation is that a node is independent of all its predecessors in the total ordering, given its parents:

$$X_j \perp X_{\text{pred}_{j\setminus\pi_j}} | X_{\pi_j} \qquad (36)$$

This was the basis of deriving Equation 5.

3.6 Markov equivalence

Consider the 3 DGMs in Figure 14(left). These all represent the same set of CI statements, namely

$$X \perp Z|Y, \quad X \not\perp Z \qquad (37)$$

Hence these graphs are called **Markov equivalent**. However, the v-structure $X \rightarrow Y \leftarrow Z$ encodes $X \perp Z$ and $X \not\perp Z|Y$, which represents a different set of CI assumptions.

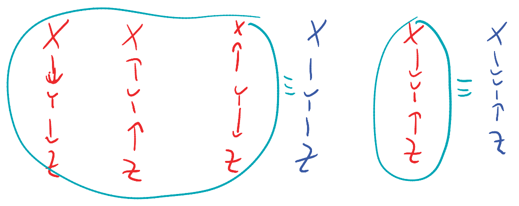


Figure 14: PDAG representation of Markov equivalent DAGs.

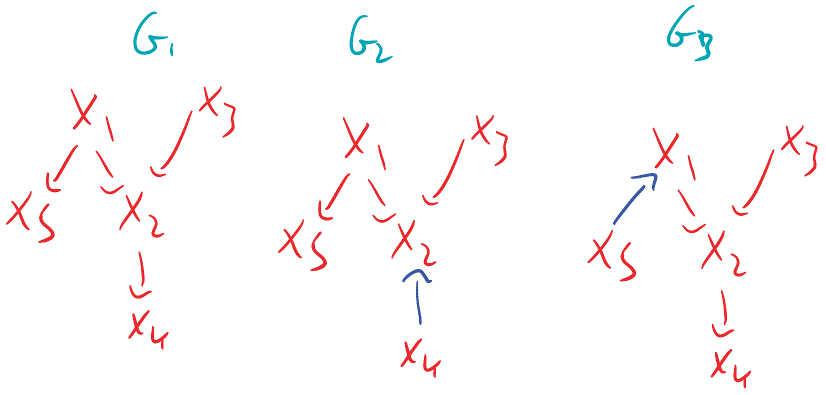


Figure 15: Three DAGs.

We can represent an equivalence class using a **partially directed acyclic graph** (PDAG), aka **essential graph** in which edges some edges are directed and some undirected. The undirected ones represent reversible edges; any combination is possible so long as no new v-structures are created. The directed edges are called **compelled edges**, since changing their orientation would change the v-structures and hence change the equivalence class. For example, the PDAG $X - Y - Z$ represents $\{X \rightarrow Y \rightarrow Z, X \leftarrow Y \leftarrow Z, X \leftarrow Y \rightarrow Z\}$ which encodes $X \not\perp Z$ and $X \perp Z|Y$. See Figure 14.

One can show the following.

Theorem 3.2 (Verma and Pearl [VP90])**.** *Two structures are Markov equivalent if they have the same undirected skeleton and the same set of v-structures.*

For example, referring to Figure 15, we see that $G_1 not \equiv G_2$, since reversing the $2 \rightarrow 4$ arc creates a new v-structure. However, $G_1 \equiv G_3$, since reversing the $1 \rightarrow 5$ arc does not create a new v-structure.

As a consequence of this theorem, we should not try to interpret the direction of the arrows in terms of **causality**.

4 Bayesian models

Each CPD $p(X_j|X_{\pi_j}, \theta_j)$ depends on the parameters θ_j. (In the case of CPTs, these are the numbers in the table.) We can show these parameters explicitly as nodes in the graph, as shown in Figure 16(left). This is very natural from a Bayesian standpoint, which treats parameters just like other random variables. In fact, the only difference between parameters and "regular" variables is that we assume the number of parameters is fixed, whereas the number of regular variables increases as we get more data cases. This is illustrated in Figure 16(right). We see that θ_j is a parent of all X_{ij}, where $i = 1 : n$ indexes the data cases. This is because we assume the data is iid, so each $\mathbf{x}_i$ is drawn from the same distribution, and hence has the same parameters.

A more concise notation for representing iid repetition is obtained by drawing a box around all the repeated variables, with the number of repetitions indicated in the bottom right corner. See Figure 17(left). All nodes outside the box pointing in are assumed to point all copies inside in the obvious way. This is called **plate notation**.

We see in the above examples that the parameters for each CPD are independent. This assumption is called **global parameter independence**. If all the regular variabes are observed (i.e., we have **complete data**, or no **missing data**), all the parameters are d-separated from each other, so the parameters will also be independent in the posterior. For the water sprinkler example, we have

$$\begin{aligned} p(\theta|D) &\propto p(\theta)p(D|\theta) && (38)\\ &= p(\theta_c)\prod_i p(c_i|\theta_c) \times p(\theta_s)\prod_i p(s_i|c_i,\theta_s) && (39)\\ &\quad \times p(\theta_r)\prod_i p(r_i|c_i,\theta_r) \times p(\theta_w)\prod_i p(w_i|s_i,r_i,\theta_s) && (40)\end{aligned}$$

Hence we can estimate the parameters for each node separately.

In the case of tabular CPDs, it is common to also assume that the multinomial parameters defining each row of the table are independent. This is called **local parameter independence**. See Figure 17(right). If the data is complete, the posterior will factorize across nodes and across conditioning cases. For example, for the R node in the water sprinkler example, if we associate a Dirichlet prior with each row of the distribution, we have

$$\begin{aligned} p(\boldsymbol{\theta}_R|D) &= \prod_{k=0}^{1} p(\boldsymbol{\theta}_{R|C=k})\prod_{i=1}^{n} p(r_i|\theta_{R|C=k})^{I(c_i=k)} && (41)\\ &= \prod_k Dir(\boldsymbol{\theta}_{R|C=k}|\boldsymbol{\alpha}_{R|C=k}) Mu(\mathbf{n}_{R,C=k}|\boldsymbol{\theta}_{R|C=k}, n_{C=k}) && (42)\\ &= \prod_k Dir(\boldsymbol{\theta}_{R|C=k}|\boldsymbol{\alpha}_{R|C=k} + \mathbf{n}_{R,C=k}) && (43)\end{aligned}$$

where

$$\mathbf{n}_{R,C=k} = (\sum_{i=1}^{n} I(R_i = 0, C_i = k), I(R_i = 1, C_i = k)) \quad (44)$$

is the vector of sufficient statistics for node R derived from those cases in which $C = k$. As usual with the Dirichlet-multinomial distribution, Bayesian inference amounts to simply updating the hyper-parameters by counting. In Figure 18, we give an example where we sequentially update the distribution over $\boldsymbol{\theta}_R$ after seeing each data case. (In this case, all the nodes are binary, so the Dirichlet-multinomial becomes beta-Bernoulli.)

A slightly more complex example is shown in Figure 19. This illustrates **nested plates**: e.g., X_{ij} is doubly indexed and is therefore inside both the i and j plates. This model corresponds to the following factorization of the joint distribution

$$p(\boldsymbol{\pi}, \boldsymbol{\theta}, D) = p(\boldsymbol{\pi})\prod_{j=1}^{d}\prod_{c=1}^{c} p(\boldsymbol{\theta}_{jc}) \times \prod_{i=1}^{n}\prod_{j=1}^{d} p(x_{ij}|y_i, \boldsymbol{\theta}_j) \quad (45)$$

We can further simplify the last term using the indicator trick as follows:

$$\prod_{j=1}^{d} p(x_{ij}|y_i, \boldsymbol{\theta}_j) = \prod_{j=1}^{d}\prod_{c=1}^{C} p(x_{ij}|\boldsymbol{\theta}_{jc})^{I(y_i=c)} \quad (46)$$

Finally, Figure 20 illustrates the difference between the Bayesian approach to predicting future values $\tilde{x}$ and the plug-in approach. In the Bayesian approach, we compute

$$p(\tilde{x}|D) = \int p(\tilde{x}|\theta)p(\theta|D)d\theta \quad (47)$$

which corresponds to integrating out the unknown variable θ in Figure 20. In the plug-in approach, we first estimate θ based on the data, and then “clamp it”, and use this fixed value to predict the future:

$$p(\tilde{x}|D) \approx p(\tilde{x}|\hat{\theta}(D)) \quad (48)$$

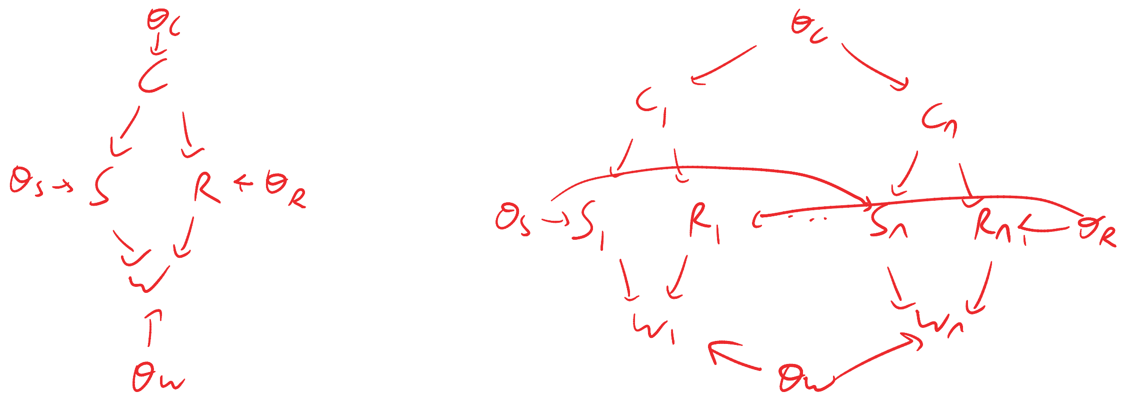


Figure 16: Left: The sprinkler network with parameters shown explicitly. Right: the same network unrolled for n cases.

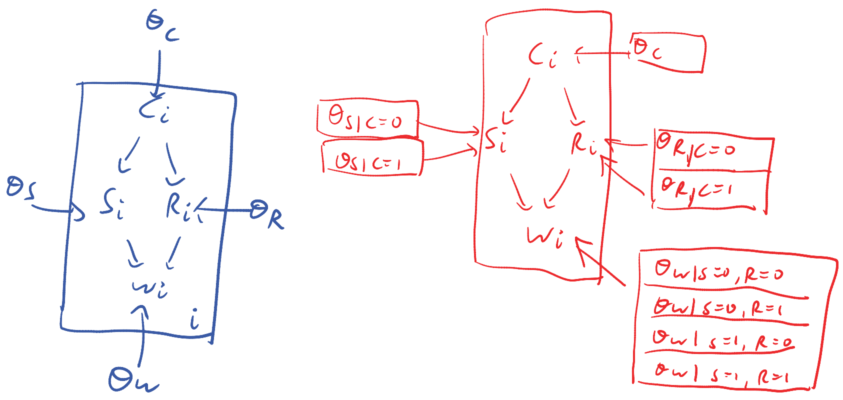
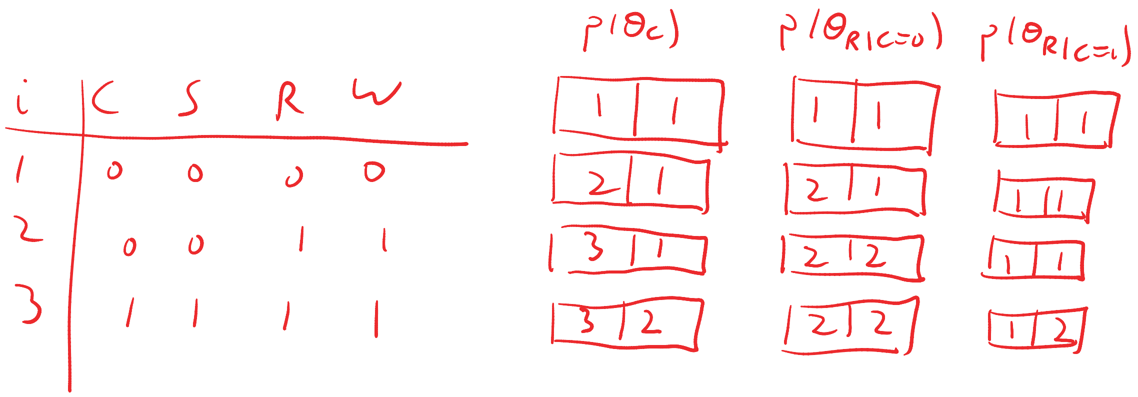


Figure 17: Left: The unrolled sprinkler network in plate notation. Right: illustration of local parameter independence.



i	C	S	R	W
1	0	0	0	0
2	0	0	1	1
3	1	1	1	1

p(θ_C)	p(θ_R|C=0)	p(θ_R|C=1)
1 | 1	1 | 1	1 | 1
2 | 1	2 | 1	1 | 1
3 | 1	2 | 2	1 | 1
3 | 2	2 | 2	1 | 2

Figure 18: Sequential updating of $p(\boldsymbol{\theta}_R|D)$ in the water sprinkler network. We start with a $Dir(1, 1)$ prior distribution on each row of the CPT. We then update either $\boldsymbol{\theta}_{R|C=0}$ or $\boldsymbol{\theta}_{R|C=1}$ depending on whether the conditioning case (value of C_i).

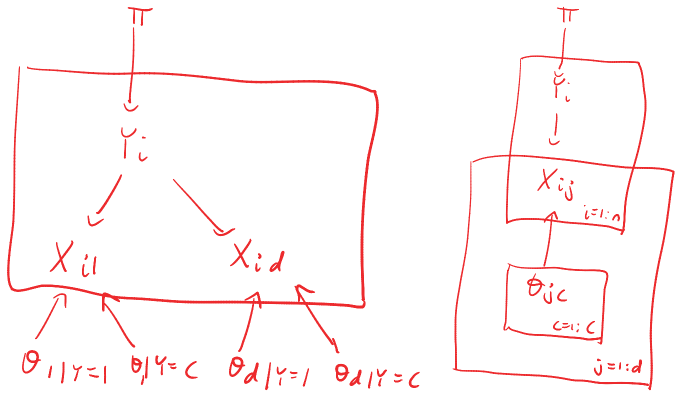


Figure 19: Naive Bayes classifier. Left: plate over cases $i = 1 : n$. Right: additional plates over features $j = 1 : d$ and classes $c = 1 : C$.

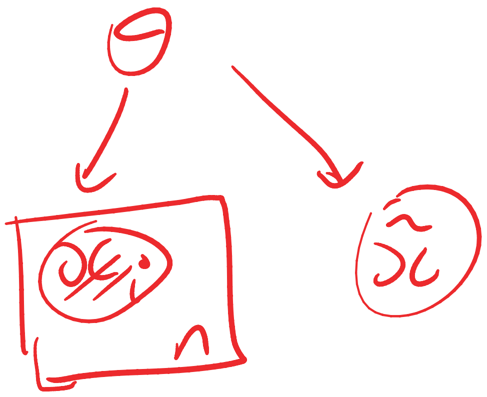


Figure 20: Predicting future data $\tilde{x}$ based on observed training data x_i, $i = 1 : n$.

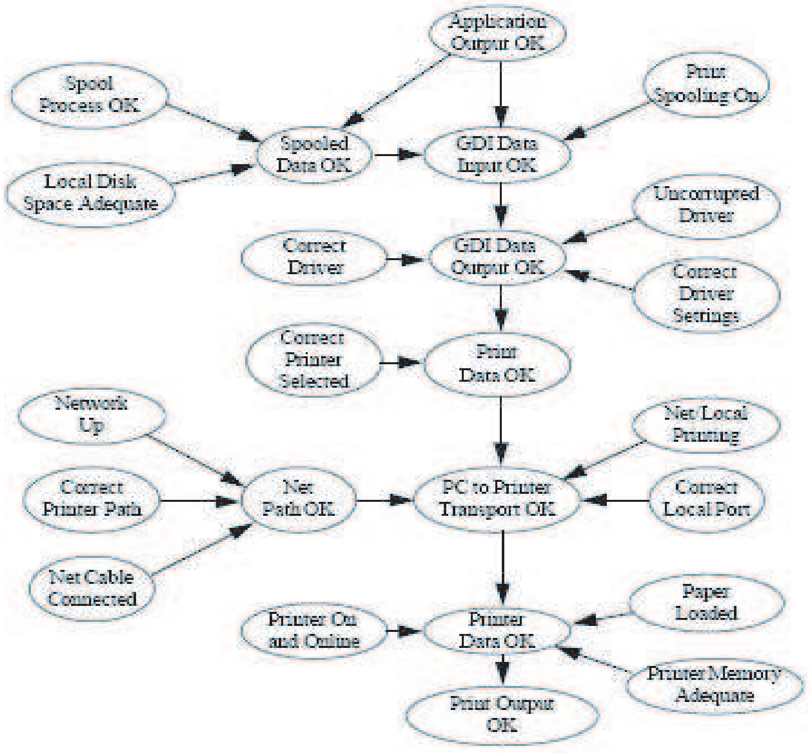


Figure 21: Part of the MS Windows Hardware troubleshooter. Source: [HB94].

5 More complex examples of DGMs

We will use the DGM notation throughout the book to describe the CI assumptions behind different statistical models. Such models are often sufficiently simple that they can be described without using graphical model notation. In this section, we look at a few examples where the graph structure is crucial to understanding the model.

5.1 Troubleshooters in Microsoft Windows

Microsoft Windows uses DGMs to perform **fault diagnosis** and **trouble shooting** of various kinds. An example from the Windows 2000 Hardware troubleshooter is shown in Figure 21. This example is concerned with modeling printer failures. A typical task, given some observed data on some of the nodes, is to infer the most likely causes of this data. An additional task is to recommend actions to the user; this requires the use of **decision theory**. DGMs can be extended to this setting by adding **action**/ **decision** nodes and **utility** nodes; the result is called a **decision diagram** or **influence diagram**. See e.g., [CDLS99] for details.

5.2 QMR

Figure 24 shows the **quick medical reference (QMR)** network. The **bipartite** graph structure shows how diseases cause symptoms. In QMR, all nodes are binary. However, since many of the leaves (symtpoms) have high fan-in (i.e., many parents), the number of parameters that would be needed to represent the CPDs in tabular form would be prohibitive. Consider a leaf X_j with parents Z_{π_j}. A CPT requires $O(2^n)$ parameters, since it can model arbitrary interactions amongs the parents. An approach that only needs $O(n)$ parameters is to use **logistic regression** (see Section **??**). However, the approach actually used in QMR was to use **noisy-OR** CPDs. This is similar to logistic regression, but is restricted to binary nodes. Specifically, the noisy-or assumption is that if a parent is "on", then the the child will also be on (since it is an or-gate), but the link from each parent Z_k to child X_j may fail independently at random with probability q_{kj}. So the only way for the child to be off is if the "wires" from all parents which are on fail

Z_1	Z_2	$P(X_j=0\|Z_1,Z_2)$	$P(X_j=1\|Z_1,Z_2)$
0	0	1	0
1	0	q_{1j}	$1-q_{1j}$
0	1	q_{2j}	$1-q_{2j}$
1	1	$q_{1j}q_{2j}$	$1-q_{1j}q_{2j}$

Figure 22: Noisy-or CPD for 2 parents. Note that this is not a linear function of the parameters.

B	Z_1	Z_2	$P(X_j=0\|Z_1,Z_2)$	$P(X_j=1\|Z_1,Z_2)$
1	0	0	q_{0j}	$1-q_{0j}$
1	1	0	$q_{0j}q_{1j}$	$1-q_{0j}q_{1j}$
1	0	1	$q_{0j}q_{2j}$	$1-q_{0j}q_{2j}$
1	1	1	$q_{0j}q_{1j}q_{2j}$	$1-q_{0j}q_{1j}q_{2j}$

Figure 23: Noisy-or CPD for 2 parents with leak node.

independently at random. Thus

$$p(X_j=0|Z_{\pi_j}) = \prod_{k\in\pi_j} q_{kj}^{I(Z_k=1)} = \prod_{k\in\pi_j:Z_k=1} q_{kj} \tag{49}$$

For example, Figure 22 shows the CPD for 2 parents.

If we observe that $X_j=1$ but all its parents are off, then this contradicts the model. Hence we add a dummy **leak node** or background node B, which is always on, this represents “all other causes”. The parameter q_{0j} represents the probability that the background leak will be inhibited:

$$p(X_j=0|Z_{\pi_j}) = q_{0j}\prod_{k\in\pi_j} q_{kj}^{I(Z_k=1)} \tag{50}$$

See Figure 23 for an example.

At test time, the goal is to infer the diseases given the symptoms. Some of the symptoms are not observed, and therefore may be removed, since they do not convey any information about their parents (the diseases); this is called **barren node removal**. For example, consider a small model with 3 diseases and 5 symptoms. Suppose symptoms 3 and 5 are not measured. Then

$$\begin{aligned} p(z_{1:3}|x_1,x_2,x_4) &\propto \sum_{x_3}\sum_{x_5} p(z_{1:3},x_{1:5}) && (51)\\ &= p(z_{1:3})p(x_1|z_{1:3})p(x_2|z_{1:3})p(x_4|z_{1:3})\left[\sum_{x_3}p(x_3|z_{1:3})\right]\left[\sum_{x_5}p(x_5|z_{1:3})\right] && (52)\\ &= p(z_{1:3})p(x_1|z_{1:3})p(x_2|z_{1:3})p(x_4|z_{1:3}) && (53)\end{aligned}$$

since $\sum_{x_3} p(x_3|z_{1:3}) = 1$. See Figure 25. Despite this trick, state estimation in the QMR model can be very slow. For general CPDs, it would take about $O(2^w)$ time, where w is the size of the largest clique in the moral graph (after barren node removal). For many of the hard test cases for which QMR is used (called the **CPSC** cases), $w \sim 151$, so exact inference would be intractable. The **quickscore** algorithm exploits properties of the noisy-OR to perform exact inference in $O(2^p)$ time, where p is the number of positive findings (leaf nodes in the “on” state). However, even this can be too slow, since $p > 20$ for many of the CPSC cases. Many approximate inference methods have been applied to this model. See e.g., [JJ99] and references therein.

5.3 Pedigree analysis

Consider the problem of **genetic linkage analysis**, which is concerned with finding the location of a disease-causing gene given observed characteristics of relatives and a known **family tree** structure.

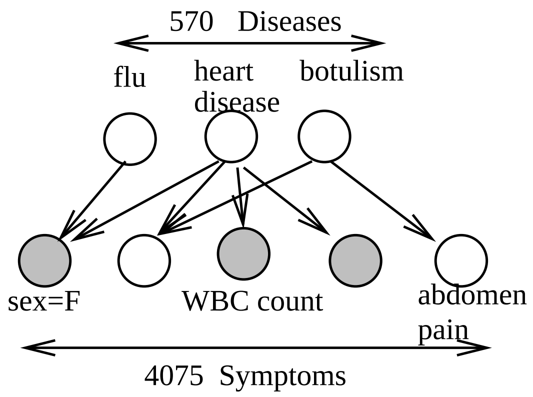


Figure 24: QMR network.

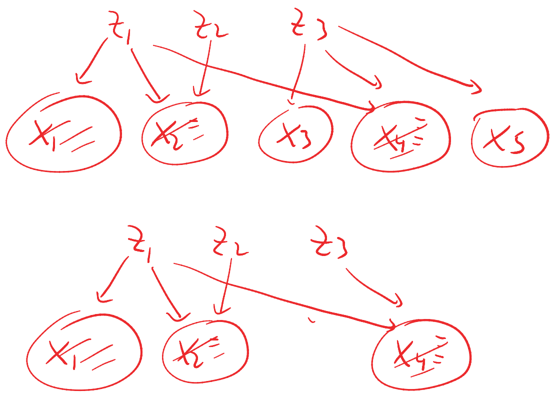


Figure 25: Barren node removal in a small QMR-style network.

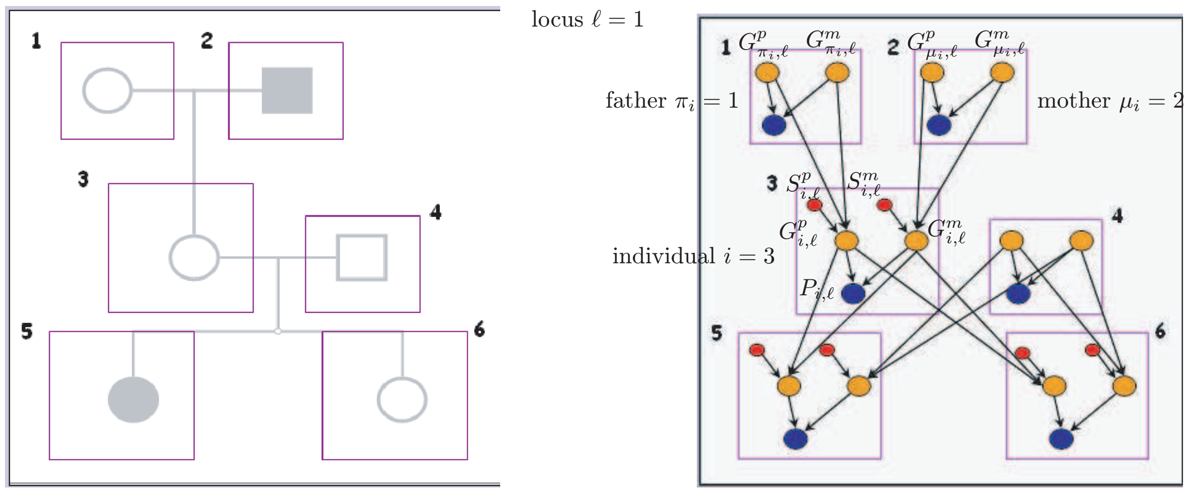


Figure 26: Left: family tree, circles are females, squares are males. Right: equivalent DGM for a single locus. Blue nodes $P_{i,\ell}$ is the observed phenotype for individual i at locus ℓ. All other nodes are hidden. Yellow nodes $G^{p/m}_{i,\ell}$ is the paternal/ maternal allele. Red nodes $S^{p/m}_{i,\ell}$ is the paternal/ maternal selection variable (which do not exist for the founder (root) nodes). Source: [FGL00].

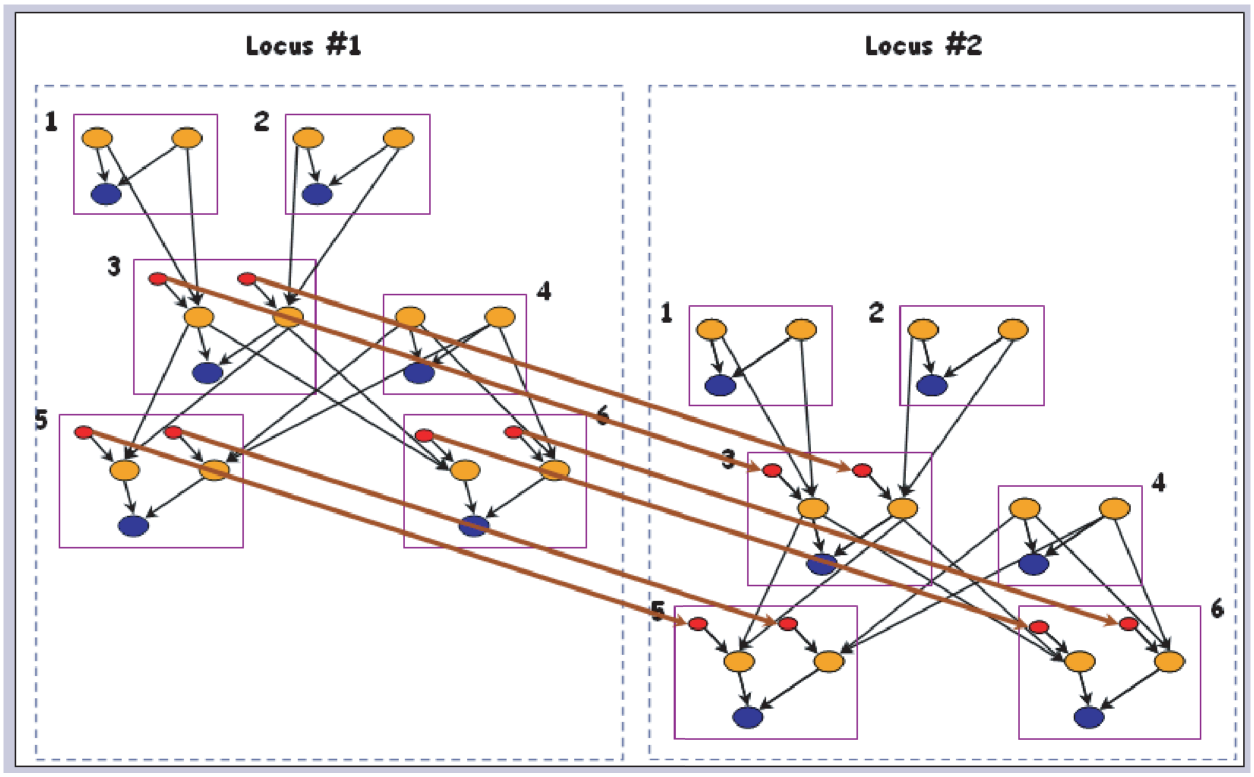


Figure 27: DGM for two loci. The red selection variables are linked across loci, $S^{p/m}_{i,\ell} \rightarrow S^{p/m}_{i,\ell+1}$. Source: [FGL00].

G_p	G_m	$p(P=a)$	$p(P=b)$	$p(P=o)$	$p(P=ab)$
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	1
o	a	1	0	0	0
o	b	0	1	0	0
o	o	0	0	1	0

Figure 28: CPD which encodes mapping from genotype to phenotype (bloodtype). This is a deterministic, but many-to-one, mapping. For example, A dominates O, so if a person has genotype AO or OA, their phenotype will be A. But AA also produces blood type A. So if we observe $P_i = A$, there are 3 possible genotypes: $G_i = A, A$, A, O or O, A. We can use the blood types of relatives to help disambiguate the evidence.

Each person i carries two copies of each gene ℓ, one from their mother (maternal gene $G^m_{i,\ell}$) and one from their father (paternal gene $G^p_{i,\ell}$). Each gene comes in several "versions" called **alleles**. For example, the gene for blood type can be of 3 types, $G^{p/m}_{i,\ell} \in \{A, B, O\}$. The maternal and paternal copies of each gene together produce a **phenotype**, $P_{i,\ell}$. (We are only considering single-gene causes for simplicity). For example, there are 4 types of blood a person can have: $P_{i,\ell} \in \{A, B, O, AB\}$. The mapping from genotype to phenotype is determined by the **pentrance model**, $p(P_{i,\ell}|G^m_{i,\ell}, G^p_{i,\ell})$. See figure 28 for an example.

Let π_i be the father of i. A person i's paternal gene $G^p_{i,\ell}$ can come from π_i's maternal copy $G^m_{\pi_i,\ell}$ or paternal copy $G^p_{\pi_i,\ell}$. This is determined by a hidden selection or **haplotype** variable, $S^p_{i,\ell}$, as follows:

$$p(G^p_{i,\ell}|G^p_{\pi_i,\ell}, G^m_{\pi_i,\ell}, S^p_{i,\ell}) = \begin{cases} \delta(G^p_{i,\ell} - G^p_{\pi_i,\ell}) & \text{if } S^p_{i,\ell} = 0 \\ \delta(G^p_{i,\ell} - G^m_{\pi_i,\ell}) & \text{if } S^p_{i,\ell} = 1 \end{cases} \quad (54)$$

Thus $S^p_{i,\ell}$ causes $G^p_{i,\ell}$ to switch between the two parent nodes deterministically. A similar equation holds for $p(G^m_{i,\ell}|G^p_{\mu_i,\ell}, G^m_{\mu_i,\ell}, S^m_{i,\ell})$, where μ_i is i's mother. This is called the **transmission model**. The root nodes in the tree are called **founder nodes**, and have priors $p(G^m_{i,\ell})$ representing the frequency of the allele types of gene ℓ in the general population. See Figure 26.

Now consider two adjacent loci on the paternal chromosome, say $G^p_{i,\ell}$ and $G^p_{i,\ell+1}$. If ℓ was inherited from the paternal π_i, then $\ell + 1$ will also be inherited from the paternal π_i, unless there was a **recombination event** (crossover between the chromosomes) between ℓ and $\ell + 1$. Let the probability of this event be θ_ℓ; this depends on the distance between ℓ and $\ell+1$ on the chromosome. We define $p(S^p_{i,\ell+1} = 0|S_{i,\ell} = 1) = \theta_\ell$ and $p(S^p_{i,\ell+1} = 0|S_{i,\ell} = 0) = 1-\theta_\ell$; this is called the **recombination model**. The result is a set of **Markov chains** connecting the S^p_i variables for each person i. We have a similar set of Markov chains for the S^m_i variables. See Figure 27.

Finally we are ready to solve the linkage analysis problem. Suppose all the parameters of the model, including the distance between all the loci, are known. The only unknown is the location of the disease-causing gene. If there are L loci, we construct $L + 1$ models, in which the disease-causing gene is postulated to lie between each of the adjacent markers. We compute the likelihood of the observed data under model j, $p(P_{1:N}|M_j, \theta)$, for $j = 1 : L + 1$, and pick the most likely one. Note that computing the likelihood requires marginalizing out all the hidden S and G variables, which can be computationally intractable if the number of individuals and/or loci is large. See [FG02] and the references therein for some exact methods based on the **variable elimination** algorithm (see Section **??**), which is also called **peeling** in this context. Unfortunately, even this exact method is often intractable. See e.g., [CAAK06] for some approximate methods.

References

[CAAK06] Martijn AR Leisink Cornelis A Albers and Hilbert J Kappen. The Cluster Variation Method for Efficient Linkage Analysis on Extended Pedigrees. *BMC Bioinformatics*, 7, 2006.

[CDLS99] R. G. Cowell, A. P. Dawid, S. L. Lauritzen, and D. J. Spiegelhalter. *Probabilistic Networks and Expert Systems*. Springer, 1999.

[FG02] M. Fishelson and D. Geiger. Exact genetic linkage computatins for general pedigrees. *BMC Bioinformatics*, 18, 2002.

[FGL00] N. Friedman, D. Geiger, and N. Lotner. Likelihood computation with value abstraction. In *UAI*, 2000.

[HB94] D. Heckerman and J.S. Breese. Causal Independence for Probability Assessment and Inference Using Bayesian Networks. *IEEE Trans. on Systems, Man and Cybernetics*, 26(6):826–831, 1994.

[JJ99] T.S. Jaakkola and M.I. Jordan. Variational probabilistic inference and the QMR-DT network. *J. of AI Research*, 10:291–322, 1999.

[RN02] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 2002. 2nd edition.

[VP90] T. Verma and J. Pearl. Equivalence and synthesis of causal models. In *UAI*, 1990.