Directed graphical models

Kevin P. Murphy

Last updated November 22, 2006

* Denotes advanced sections that may be omitted on a first reading.

1 Introduction

One of the key assumptions needed to make probabilistic models of large numbers of random variables is **conditional indepedence**. Recall that if X_i is conditionally independent of X_j given X_k, written $X_i \perp X_j|X_k$, it means

$$p(X_i, X_j|X_k) = p(X_i|X_k)p(X_j|X_k) \tag{1}$$

which basically means that X_i and X_j do not directly influence each other, but that their influence is mediated via X_k. (Note that X_i,X_j and X_k can be *sets* of variables.) Another way of saying this is that $X_i \perp X_j|X_k$ iff the joint factorizes as follows

$$p(X_i, X_j|X_k) = f(X_i, X_k)g(X_j, X_k) \tag{2}$$

for some functions f and g.

Probabilistic graphical models are a way of representing conditional independence assumptions using graphs. Nodes represent random variables and *lack of* edges represent conditional independence assumptions, in a way which we will define below. There are many kinds of graphical model, but the two most popular are **Bayesian (belief) networks**[1], which are based on **directed acylic graphs (DAGs)**, and **Markov networks**, aka **Markov random fields (MRFs)**, which are based on **undirected graphs**. In a directed graphical model (DGM), we can (informally) think of an edge from node X_i to node X_j as meaning X_i "causes" or "directly influences" X_j, whereas in an MRF, edges represent correlation rather than causation. In this chapter, we focus on directed graphs.

2 Example networks

The easiest way to understand DGMs is to consider a few examples.

2.1 Water sprinkler

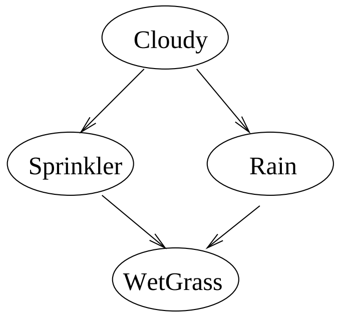
It is common when building DGMs to interpret an edge from i to j as meaning X_i causes X_j. This can be used as the basis for a theory of causality, as discussed in Section 8. However, in the more "vanilla" statistical use of these models, the edges (or lack thereof) simply encode conditional independence assumptions. We give the details below, but the basic idea is that a node is independent of its non-descendants given its parents.

Let us consider the "water sprinkler" example in Figure 1, which 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). Clouds cause rain, so there is an arc from C to R. Sprinklers are usually turned on and off by hand, or by a timer, but, for the sake of this example, suppose the sprinkler has a light sensor on it, so cloudy days cause the sprinkler to turn off, and sunny days cause it to turn on: hence there is an arc from C to S. However, there is no arc between S and R, since sprinklers do not cause rain or vice versa. Hence $S \perp R|C$. Also, there is no arc from C to W, since we assume that C does not directly cause W. More formally, we assume that

[1] There is nothing inherently Bayesian about "Bayesian networks". They are just models of probability distributions. Hence we prefer the more neutral term "directed graphical model". However, they are widely used in Bayesian statistics as a convenient notational device. They are also a useful data structure which can often be leveraged computationally.

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 1: Water sprinkler Bayes net with CPDs shown.

$W \perp C|S,R$. These two independence assumptions allow us to represent the joint as a **product of local factors**, one per node.

$$\begin{aligned}
P(C,S,R,W) &= P(C)P(S|C)P(R|S,C)P(W|S,R,C) \text{ chain rule} && (3)\\
&= P(C)P(S|C)P(R|\cancel{S},C)P(W|S,R,C) \text{ since } S \perp R|C && (4)\\
&= P(C)P(S|C)P(R|\cancel{S},C)P(W|S,R,\cancel{C}) \text{ since } W \perp C|S,R && (5)\\
&= P(C)P(S|C)P(R|C)P(W|S,R) && (6)
\end{aligned}$$

In general, if we number the nodes $1:d$ (such a total ordering always exists, since the graph is acyclic), we can write

$$p(X_{1:d}) = \prod_i p(X_i|X_{\pi_i}) \tag{7}$$

where π_i are the **parents** of node i.

Each term $p(X_i|X_{\pi_i})$ is called a **conditional probability distribution (CPD)**. In the simplest case of discrete random variables with K states each, we can represent each CPD as a table of $O(K^{f_i+1})$ numbers, where $f_i = |\pi_i|$ is the number of parents (fan-in) of node i. In this case, the CPD is called a **conditional probability table (CPT)**.

2.2 Burglar alarm

Consider the network in Figure 2. This represents a distribution on 5 variables, $p(B,E,A,J,M)$. The intuition behind the model is that either burglaries or earthquakes can trigger an alarm (this example is due to Judea Pearl who lives in Los Angeles, which is prone to both of these problems). If the alarm rings, either of your neighbors, John and Mary, will call you at work to let you know. But sometimes they don't hear the alarm, e.g., $p(M=t|A=t) = 0.7$ means the probability that Mary will call you given that there is an alarm is just 70%.

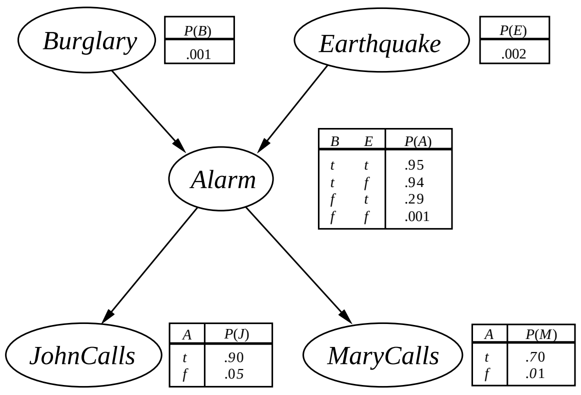


Figure 2: The burglar alarm network. Source: [RN02].

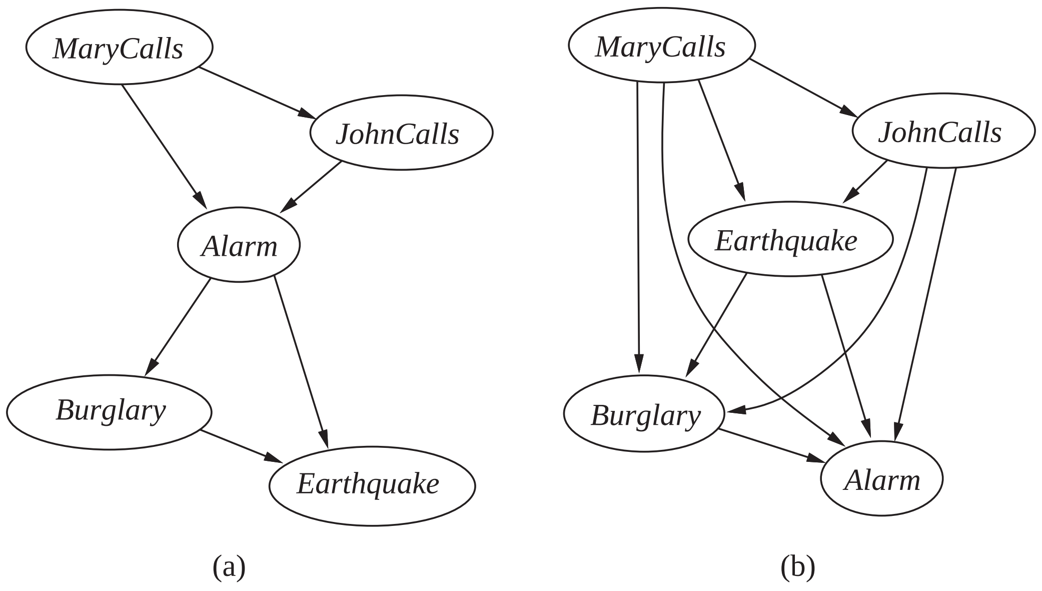


Figure 3: Two DGMs defined with different orderings. Source: [RN02].

If we pick the "wrong" (i.e., non causal) order, we will end up having to add extra arcs in order to not make any false conditional independence assertions. distribution. Recall that a DGM asserts that a node is independent of its non descendants (and hence its ancestors) given its parents. If you pick the wrong parents, they may not **screen off** your ancestors. This is illustrated in Figure 3. For example, consider the ordering M, J, E, B, A (right hand side of figure). Obviously A needs B and E as parents, since they are direct causes. But this set of parents is not enough to screen A off from its other ancestors, J and M. Specifically, we need to add an arc from J to A since $A \not\perp J|B, E$ (assuming Figure 2 is the "true" distribution). Similarly we need to add the $M \rightarrow A$ arc. The ordering M, J, E, B, A is the worst possible, since the graph is fully connected, and hence does not make any conditional independence assertions. The ordering M, J, A, B, E is also bad, but not as bad.

2.3 Naive Bayes classifiers

As another simple example, consider a naive Bayes classifier. This asserts that the features x_i are conditionally independent given the class label y. This is illustrated in Figure 4(left), which defines the following joint probability

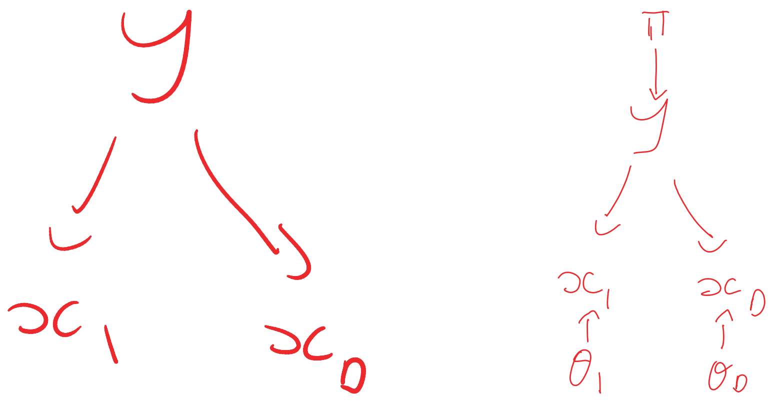


Figure 4: Naive Bayes classifier represented as a DGM. y is the class label and x_i are the features. In the version on the right, the parameters are shown explicitly.

distribution

$$p(y, x_{1:D}) = p(y) \prod_{i=1}^{D} p(x_i|y) \tag{8}$$

We can also make the parameters explicit, as in Figure 4(right):

$$p(y, x_{1:D}, \theta) = p(y|\pi) \prod_{i=1}^{D} p(x_i|y, \theta_i) \tag{9}$$

If we assume the parameters are fixed (known) constants, they are not usually explicitly represented in the model.

2.4 Markov chains

As another simple example, consider a discrete time Markov chain:

$$p(X_{1:T}) = p(X_1) \prod_{t=2}^{T} p(X_t|X_{t-1}) \tag{10}$$

If we assume the state space is discrete, we can represent $p(X_t = j|X_{t-1} = i) = A_t(i, j)$ by a transition matrix; this is just the CPT for node X_t. If we assume the chain is time invariant, then we can use the same matrix A for all time slices; this is an example of **parameter tying** (and is necessary in order to define the distribution for arbitrary number of nodes T using a constant number of parameters). The initial state of the chain can be represented by a probability vector $p(X_1 = i) = \pi_i$. Thus the parameters of the model are $\theta = (\pi, A)$. From the Bayesian point of view, parameters are just random variables, so we can add them to the graph: see Figure 5. This makes explicit the parameter tying assumption, and also the assumption that π and A are independent. Obviously if we only observe one sequence we cannot estimate π, since we will only have one data point. However, if we have many sequences,and the parameters are shared across sequences, they will be easy to learn, as we discuss below.

3 Conditional independence properties of DAGs

In this section we describe more formally how graphs encode conditional independence properties. For undirected graphs, independence corresponds to simple graph separation, but for directed graph, the story is a bit more complex, since we need to take the direction of the edges into account.

3.1 Local markov properties for directed graphs

The **local Markov property** says a node is conditionally independent of its non-descendants given its parents: see Figure 6. This means that, given any **total ordering** of the nodes (such an ordering always exists, since the graph is a

$$\begin{tikzcd}
\pi \arrow[d] & & A \arrow[dl] \arrow[d] \arrow[dr] & & \\
X_1 \arrow[r] & X_2 \arrow[r] & X_3 \arrow[r] & X_4 \arrow[r] & \cdots
\end{tikzcd}$$

Figure 5: Markov chain represented as a DGM. π is the initial state distribution and A is the transition matrix.

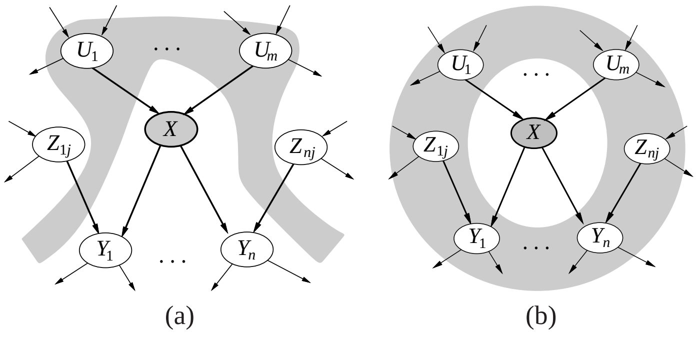


Figure 6: (a) A node is independent of its non-descendants given its parents. (b) A node is independent of all other nodes given its Markov blanket. Source: [RN02] Fig 14.4.

DAG), we have that $X_i \perp X_{pred_i} | X_{pa_i}$, where pa_i are the parents of node X_i and $pred_i$ are all the other predecessors of X_i in the ordering. Hence the **chain rule**

$$P(X_{1:N}) \quad = \quad \prod_{i=1}^{N} P(X_i | X_{1:i-1}) \tag{11}$$

simplifies to

$$P(X_{1:N}) = \prod_{i=1}^{N} P(X_i | X_{\pi_i}) \tag{12}$$

Each term only involves a node and its parents (a **family**). Hence the model is defined in terms of a product of local terms. We can therefore “mix and match” different kinds of CPDs, and build quite complex models in a modular way.

We can show that this is equivalent to an alternative local Markov property, which says that a node is conditionally independent of all others given its **Markov blanket**. The markov blanket is the parents, children, and childrens’ parents (**co-parents**): see Figure 6(b). To see why this is true, partition 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 U_i, its children Y_i, and its co-parents), and the rest of the nodes R. Then the **full conditional** is given by

$$\begin{aligned}
p(X_i|X_{-i}) &= \frac{p(X_i, X_{-i})}{\sum_x p(X_i, X_{-i})} && (13)\\
&= \frac{p(X_i, U_{1:n}, Y_{1:m}, Z_{1:m}, R)}{\sum_x p(x, U_{1:n}, Y_{1:m}, Z_{1:m}, R)} && (14)\\
&= \frac{p(X_i|U_{1:n})[\prod_j p(Y_j|X_i, Z_j)]P(U_{1:n}, Z_{1:m}, R)}{\sum_x p(X_i = x|U_{1:n})[\prod_j p(Y_j|X_i = x, Z_j)]P(U_{1:n}, Z_{1:m}, R)} && (15)\\
&= \frac{p(X_i|U_{1:n})[\prod_j p(Y_j|X_i, Z_j)]}{\sum_x p(X_i = x|U_{1:n})[\prod_j p(Y_j|X_i = x, Z_j)]} && (16)
\end{aligned}$$

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 $X_i \perp R|MB_i$, where MB_i is X_i's Markov blanket.

We can rewrite the full conditional as follows

$$p(X_i|X_{-i}) \propto p(X_i|Pa(X_i)) \prod_{Y_j \in ch(X_i)} p(Y_j|Pa(Y_j) \qquad (17)$$

The key requirement for the **Gibbs sampling** algorithm (defined later) is that we can sample from this distribution. This is easy provided each of these CPDs is conjugate. If this is not the case, we may need to use some other method to sample from this distribution, such as adaptive rejection sampling, or the Metropolis algorithm.

3.2 Global Markov properties

By chaining together local independencies, we can infer more global independencies. We will start by doing this informally by examining some examples. Then we will present an algorithm and a formal definition.

First consider a chain structure $X \rightarrow Y \rightarrow Z$. When we condition on y, are x and z independent? We have

$$p(x, y, z) = p(x)p(y|x)p(z|y) \qquad (18)$$

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?

$$p(x, y, z) = p(y)p(x|y)p(z|y) \qquad (22)$$

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

$$p(x, y, z) = p(x)p(z)p(y|x, z) \qquad (25)$$

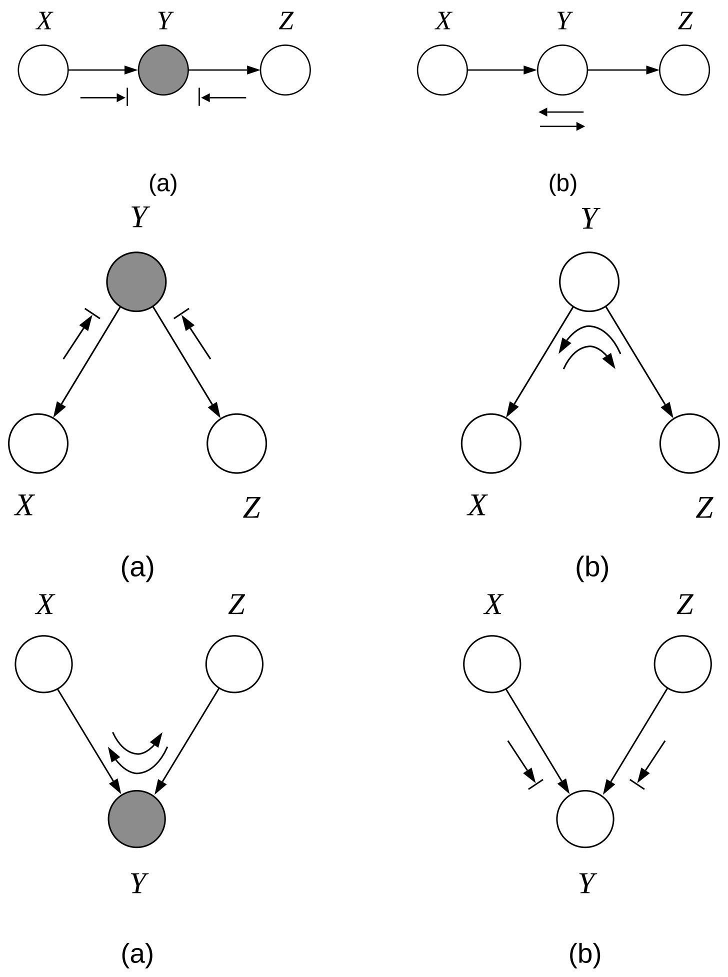


Figure 7: Bayes ball rules. A shaded node is one we condition on. If there is an arrow with a vertical bar it means the ball cannot pass through; otherwise the ball can pass through.

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 another 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).

Now we will summarize these 3 cases into the **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 vars in C. (This is like regular graph separation, but takes the direction of the edges into account.) In other words, given that all the nodes in x_C are clamped, when we wiggle nodes x_A can we change any of the node x_B? The Bayes-Ball algorithm is a such a d-separation 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 cases we considered tell us rules, which are shown in Figure 7. Notice balls can travel opposite to edge directions.

We also need the boundary conditions, which are shown in Figure 8. Note that one consequence of these boundary conditions is that if we have a v-structure $X \rightarrow Y \leftarrow Z$, and any of Y's descendants is observed, then X and Z become coupled. To see why, let YY be a descendant of Y, and suppose $P(YY|Y)$ is deterministic and one-to-one. Then if we observe YY, we effectively observe Y as well, so the parents X and Z have to compete to explain this.

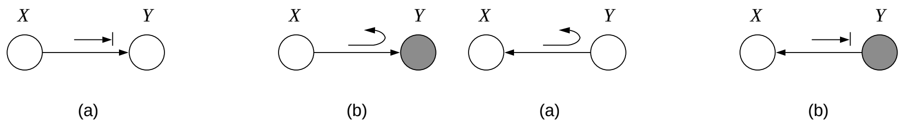


Figure 8: Bayes ball boundary conditions. A curved arrow means the ball "bounces back".

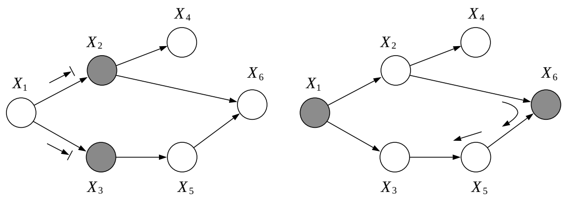


Figure 9: Two examples of Bayes ball algorithm

Two examples of the Bayes ball algorithm are shown in Figure 9. In the first one, we ask

$$x_1 \perp x_6 | \{x_2, x_3\} \quad ? \tag{26}$$

The answer is yes, since 2 and 3 block 1 from 6. In the second one, we ask

$$x_2 \perp x_3 | \{x_1, x_6\} \quad ? \tag{27}$$

The answer is no, since 6 is observed so passes the ball on from 2 to 5 and then to 3.

Let us formalize the Bayes ball algorithm. 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, $X_{i-1} \to X_i \leftarrow X_{i+1}$, then X_i or one of its descendants is in E; and
2. no other node along the path is in E

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. Then we have

Theorem 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.*

4 Graph manipulations

Sometimes it is useful to be able to reverse arcs in a DGM. In order to do so safely, we have to ensure the new graph G' does not contain any conditional independence assertions that were not present in G. For example, consider Figure 10. We see that A is not independent of E given D in graph G, written $\not\perp_G(A, E|D)$, because of the $A \to D \to B \to E$ path. When we reverse the $D \to E$ arc, we lose this path, so to be safe, we add the $A \to E$ arc explicitly to G'. Similarly $\not\perp_G(C, D|E)$ so we add the $C \to D$ arc explicitly to G'.

Another useful operation is to marginalize out a variable. Consider Figure 11 where we marginalize out H in the middle. Let G' be the resulting graph. We need to add extra edges to G' to ensure it does not represent any conditional independence assertions that we not present in G. Clearly all of H's children become dependent, because of explaining away. Also, all of H's children inherit all of H's parents, as an indirect cause. The resulting graph without the H "bottleneck" node is obviously much more densely connected. Thus we see that **hidden (latent) variables** can be used to simplify many models.

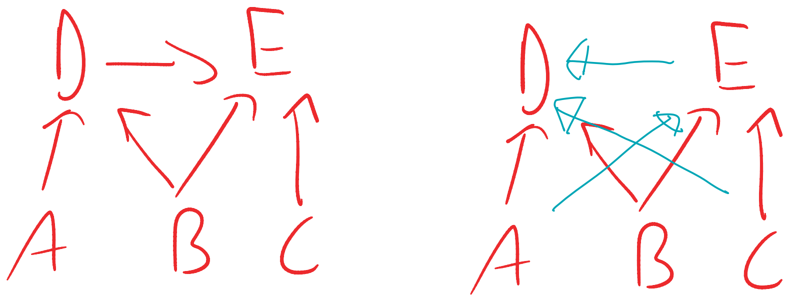


Figure 10: We reverse the arc from D to E, and have to add edges $A \rightarrow E$ and $C \rightarrow D$ to compensate.

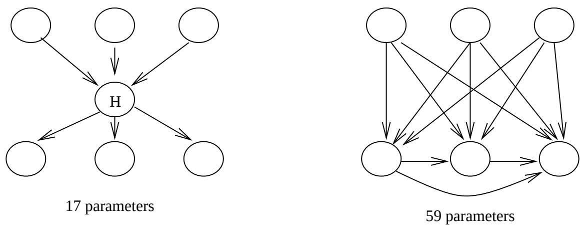


Figure 11: Example of node elimination. Marginalizing out H from the graph on the left results in the graph on the right. If all nodes are binary and all CPDs are tabular, the total parameter counts for the two models are 17 and 59.

5 Probabilistic expert systems

Because of their intuitive nature, it is possible to construct DGMs (or at least the graph structure) by hand, using expert knowledge. The results are sometimes called **probabilistic expert systems**, since they can be used to encode "rules", but they allow for exceptions.[2] Below we give some examples from the field of medicine.

5.1 Alarm net

Figure 12 shows the **alarm network**, that was used to model Intensive Care Unit (ICU) monitoring. The structure and parameters of the network were specified by human experts (we discuss how to learn the parameters and structure later). At test time, some of these nodes are observed, and the network is used to infer the values of other nodes. (This is an example of state estimation, which is discussed further below.)

5.2 QMR

Figure 14 shows the **quick medical reference (QMR)** network. The **bipartite graph structure** shows how diseases cause symptoms. The structure and parameters of the network were specified by human experts (doctors) by a process called **knowledge elicitation**.

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 with parents $U_{1:n}$. A CPT requires $O(2^n)$ parameters, since it can model arbitrary interactions amongs the parents. A simpler approach, that needs $O(n)$ parameters, would be to use logistic regression, i.e., $p(X = 1|u_{1:n}) = \sigma(w^T \vec{u})$.

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 u_i to child may fail independently at random with probability q_i. Put another way, $\lambda_i = 1 - q_i$ is the probability that u_i alone is sufficient to turn X on if all other parents were off.

[2]The existence of exceptions is what makes logic-based expert systems so fragile. For example, rain does not always cause the grass to be wet, because e.g., the grass may be covered by a tarp, the grass may be indoors, etc. Rather than writing a rule for every exception, probabilistic systems simply model such exceptions as "noise", and just try to capture the main effects. Consequently they are often simpler and more robust.

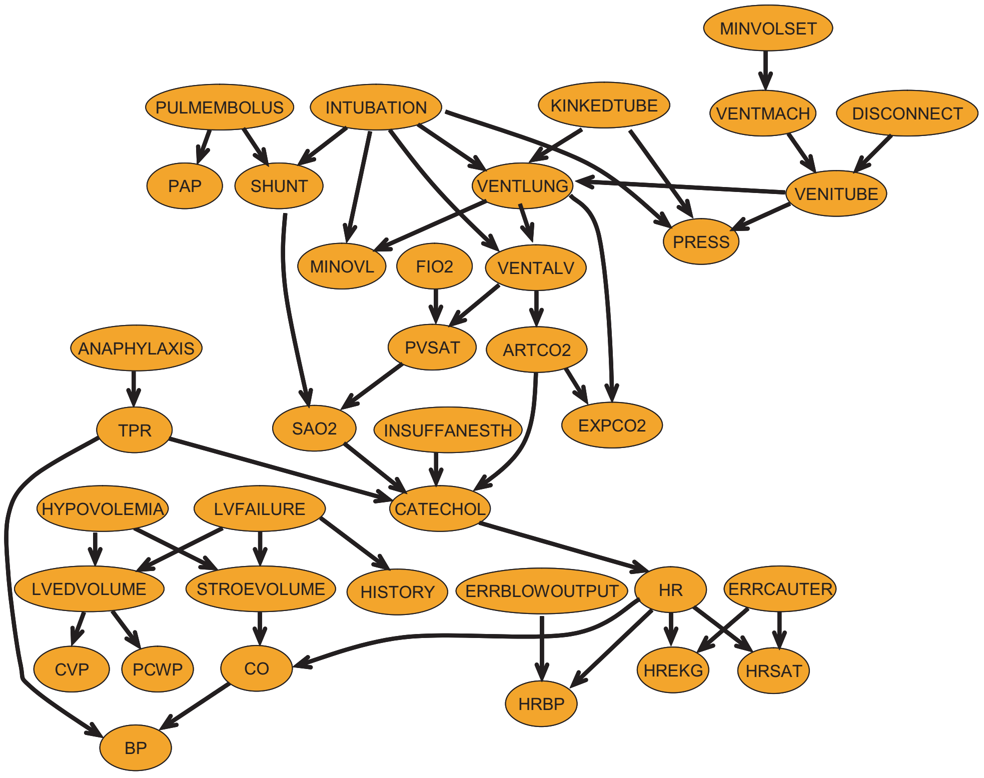


Figure 12: Alarm network.

U_1	U_2	$P(X=0\|U_1,U_2)$	$P(X=1\|U_1,U_2)$
0	0	1	0
1	0	q_1	$\lambda_1 = 1 - q_1$
0	1	q_2	$\lambda_2 = 1 - q_2$
1	1	$q_1 q_2$	$1 - q_1 q_2 = \lambda_1 + \lambda_2 - \lambda_1 \lambda_2$

Figure 13: Noisy-or CPD for 2 parents. Note that this is not a linear function of the parameters.

If we observe that $X = 1$ but all $u_i = 0$, then this contradicts the model. Hence we add a dummy **leak node** which is always on, $u_0 = 1$, and we define λ_0 to be the probability that X turns on for some reason that cannot be explained by the parents $u_{1:n}$; hence the leak node represents "all other causes". Hence the noisy-or model is

$$P(X=0|U_{1:n}) = \prod_{i:U_i=1} q_i = \prod_{i:U_i=1} (1-\lambda_i) \tag{28}$$

For example, Figure 13 shows the CPD for 2 parents. In the case of QMR, the q_i parameters were specified by hand.

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**.

5.3 Pedigree trees

Finally we consider an example that is not usually considered a probabilistic expert system, but has much in common with them, in the sense that the structure of the model is quite complex, but all the (hidden) variables are discrete, and the parameters are simple and usually assumed to be known. Thus the key problem is **state estimation** rather than parameter estimation or model selection.

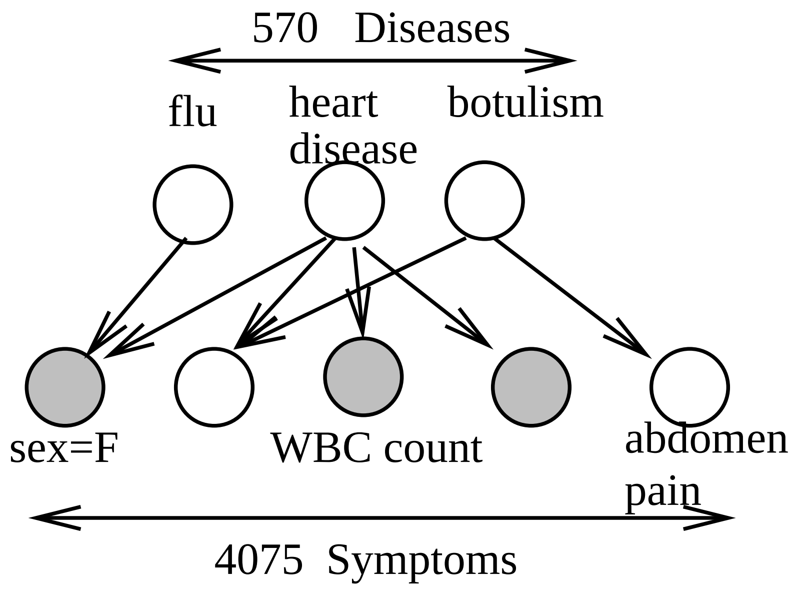


Figure 14: QMR network.

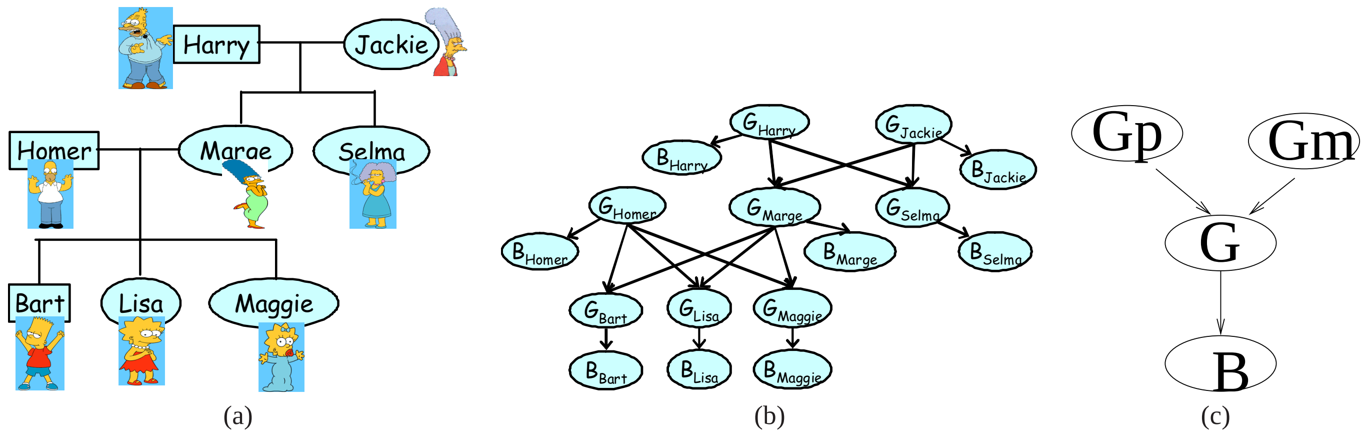


Figure 15: Genetic pedigree tree. Source: [KF06].

Suppose we are interested in inferring someone's blood type given observations of the blood types of relatives. Let $B_i \in \{a, b, o, ab\}$ be the blood type (**phenotype**) of person i. Let the (unobserved) **genotype** of person i be $G_i \in \{a, b, o\} \times \{a, b, o\}$, which represents the alleles inherited from the mother and father. If we observe i's bloodtype, we can infer their genotype, and hence their parents' genotype, and hence we can predict the bloodtype of relatives of i. (We reason from effects to causes to effects.)

Consider the family tree in Figure 15(a). This can be converted into a Bayes net as shown in Figure 15(b). This network has two kinds of connections: $G_p \rightarrow G_i \leftarrow G_m$, which models how the genotype of person i depends on their mother G_m and father G_p; and $G_i \rightarrow B_i$, which models how the genotype causes the phenotype. Mendels laws define $P(G_i|G_p, G_m)$. The phenotypic expression specifies $P(B_i|G_i)$: see Figure 16. Note that although $P(B|G)$ is deterministic, it is many-to-one, and hence hard to invert. That is, if you know B_i, you cannot (usually) uniquely infer B_i. However, we can perform probabilistic inference to estimate the genotype given N observed bloodtypes, $p(G_i|b_{1:N})$.

G	$P(B=a)$	$P(B=b)$	$P(B=o)$	$P(B=ab)$
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 16: CPD which encodes mapping from genotype to 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 $B_i = A$, there are 3 possible genotypes: $G_i = AA$, AO or OA; we can use the blood types of relatives to help disambiguate the evidence.

6 Plate notation

DGMs are widely used to represent **hierarchical Bayesian models**. These are models in which the parameters are linked together in some way; this allows us to **borrow statistical strength** across related subproblems, especially from data rich problems to data poor ones. We will see examples of this later.

From a Bayesian point of view, parameters are random variables, and are therefore just nodes in the graph. The only difference between the parameters θ and the other random variables x is that there are a fixed number of parameters (assuming the model is a parametric model[3]), but ND other random variables, where N is the number of data cases, and D is the number of variables. Another difference is that parameters are never observed. The other variables may be observed, or they may be **hidden**. If they are sometimes hidden and sometimes observed, we say we have **missing data**. If they are always hidden, we say they are **latent variables**. Thus the difference between a latent variable and a parameter is again that there are a fixed number of parameters but there may be many latent variables per data case. We shall study latent variable models later.

Given a data set $D = (x_1, \ldots, x_N)$. we usueally assume that the elements in a data set are **exchangeable**, i.e., the order of the indices does not matter:

$$p(x_1, \ldots, x_N) = p(x_{\pi(1)}, \ldots, x_{\pi(N)}) \tag{29}$$

for any **permutation** π. **de Finetti's theorem** says that (roughly speaking) data is exchangeable iff it is conditionally independent given some parameter vector θ:

$$p(x_{1:N}) = \int \prod_n p(x_n|\theta)p(\theta)d\theta \tag{30}$$

where we get to choose the form of the parametric model $p(x_n|\theta)$, and the parameter prior $p(\theta)$. This means that the assumption of exchangeability implies the existence of some prior distribution.

This factorization of the joint distribution of $p(\theta, D)$ can be represented as a tree structured DGM, with θ as the root, and the x_n as the leaves: see Figure 17(a). However, since this "motif" occurs so often, it is convenient to develop some **syntactic sugar** for it. **Plates** are little boxes we can draw around variables to represent that they are repeated, and that each one shares any common parents outside the box. See Figure 17(b). This notation can be used to concisely define quite complex models. We give some examples below.

[3]The definition of a **parametric model** is one which has a constant number of parameters independent of the data size N. A **non-parametric model** still has parameters, but the number of such parameters can grow as a function of N [**?** , p88]. An example is kernel density estimation or Gaussian processes. A **semi-parametric model** is one in which some parts are parametric, and others non-parametric.

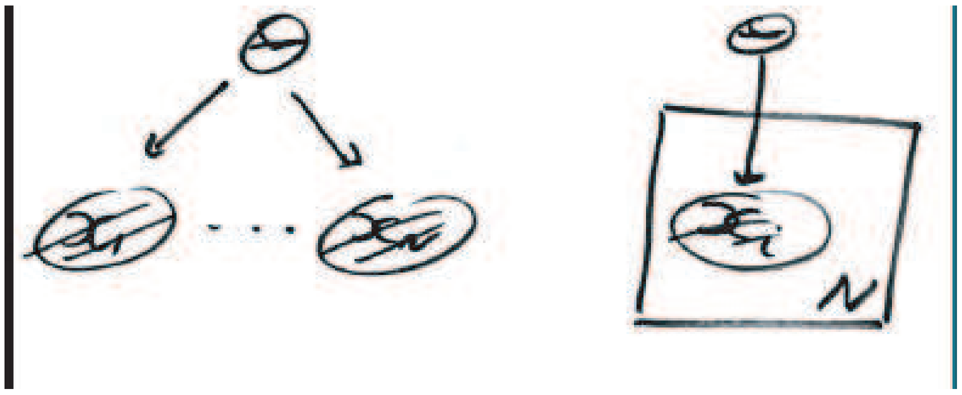


Figure 17: (a) Data points x_i are conditionally independent given parameter θ. (b) Plate notation for (a). The box represents N repetitions of the structure inside.

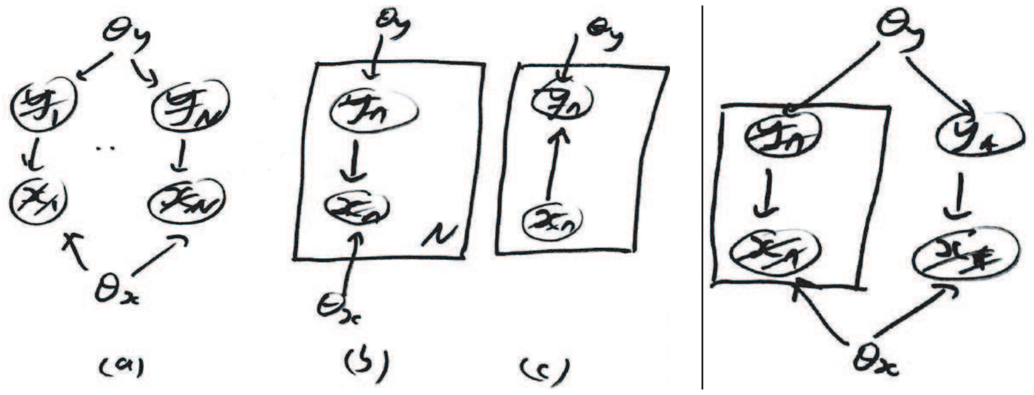


Figure 18: Plate notation for classification/ regression. (a) Training a generative classifier. The x_n and y_n nodes are shown shaded, since they are observed in the training set. (b) Plate notation for (a). The box represents N repetitions of the structure inside. (c) Discriminative classifier. (d) Applying a generative classifier to test case x^*.

6.1 Classification/ regression

Recall that in classification and regression, we fit a model to a training set of iid input/ output pairs, $D = (x_n, y_n)$. If we use a generative model, the likelihood becomes

$$p(D|\theta) = \prod_{n=1}^{N} p(y_n|\theta_y)p(x_n|y_n, \theta_x) \tag{31}$$

If we assume the parameters of the class prior θ_y and the class conditional densities θ_x are independent, we get the Bayes net in Figure 18(a). We can redraw this with plate notation as in Figure 18(b).

Learning the parameters of a discriminative model is very similar: we simply reverse the arc from y to x, so now we need to specify $p(y|x, \theta_y)$ and $p(\theta_y)$. However, we do not need to specify $p(x)$ or θ_x, since this is a conditional (discriminative) model, in which we assume the x_n are fixed constants that are always observed (**exogeneous variables**). See figure 18(c).

We can interpret the difference between full Bayesian prediction and the simpler plug-in principle using pictures. To predict future outputs y_* given future inputs $\vec{x}_*$ and the training data D, probability theory tells us

$$p(y|\vec{x}, D) = \int p(y|\vec{x}, \theta)p(\theta|D)d\theta \tag{32}$$

This amounts to integrating out the nuisance variables θ_x, θ_y in Figure 18(d). A common approximation is to use a point-estimate to the parameter posterior

$$p(\theta|D) \approx \delta(\theta - \hat{\theta}_{MAP}) \tag{33}$$

where

$$\hat{\theta}_{MAP} = \arg\max_{\theta} p(D|\theta)p(\theta) \tag{34}$$

and then use this as a **plug-in estimate** for the predictive density

$$p(y|x, D) \approx p(y|x, \hat{\theta}) \tag{35}$$

This approximation is reasonable if the posterior over parameters is well peaked at the MAP estimate. This amounts to setting the nuisance variables $\theta = (\theta_x, \theta_y)$ in Figure 18(d) to their MAP estimate, based on the training set, and then treating them as observed (known) when applied to the testing set. This requires preventing information from flowing backwards from $\vec{x}_*$ to θ_x and θ_y, since the assumption is that θ is estimated only based on the training data. Allowing the test set inputs to be used in the learning stage is called **transductive learning**. Obviously such a model can do better at predicting the test data, but it runs a greater risk of overfitting.

6.2 Nested plates

It is possible to "nest" plates one (partially) inside the other, as shown in Figure 19. This example, the first one in the BUGS manual[4], is an example of a hierarchical Bayesian analysis of Gaussian data. The data, y_{ij}, represent the weight of the i'th rat measured at age x_j. The CPDs of the bottom-most nodes are

$$\begin{aligned} y_{ij} &\sim \mathcal{N}(\mu_{ij}, \tau_c) && (36)\\ \mu_{ij} &= \alpha_i + \beta_i(x_j - \overline{x}) && (37) \end{aligned}$$

So the mean weight is modelled as a deterministic (linear) function of time. Note that the input/ covariate x_j is shared across rats i and hence is outside the i plate, and the coefficients α_i and β_i are shared across time j and so are outside of the j plate. Only the mean μ_{ij} and data y_{ij} are doubly indexed and hence inside both plates.

7 Parameter estimation from fully observed data

Parameter estimation in DGMs is particularly easy if we make two assumptions: (1) the **complete data** or **completely observed data** assumption, which means that all random variables (except the parameters) are fully observed in every data case; and (2) the **global parameter independence** assumption, which means that the parameters for each CPD are apriori independent. Given these two assumptions, it is easy to see that the parameter posterior is also factorized, and hence we can estimate the parameters for each CPD separatelt.

For example, consider the generative classifier in Figure 18(a). If we assume the parameters are a prior independent, $\theta_x \perp \theta_y$, then it is easy to see that $\theta_x \perp \theta_y|D$, since the data d-separates $\theta_x$from θ_y (because it is fully observed!). Hence the posterior factorizes

$$\begin{aligned} p(\theta|D) &\propto p(D|\theta)p(\theta) && (38)\\ &= [p(y_{1:N}|\theta_y)p(\theta_y)][p(x_{1:N}|\theta_x, y_{1:N})p(\theta_x)] && (39)\\ &\propto p(\theta_y|y_{1:N})p(\theta_x|x_{1:N}, y_{1:N}) && (40) \end{aligned}$$

So θ_y can be estimated just from the class labels $y_{1:N}$, and θ_x can be estimated from x_n, y_n pairs.

If the parent nodes y are discrete, then the parameter vector for the child, θ_x, can have a different value for each value of the parent; we shall denote this as $\theta_{x|y}$. (In the context of generative classifiers, these would be called the parameters of the **class conditional densities**.) If we assume **local parameter independence**, which means the parameters $\theta_{x|y}$ are independent, then we can separately estimate $\theta_{x|y}$ for each possible value of y.

[4]BUGS stands for "Bayesian updating using Gibbs Sampling" and is a very popular sofware package for hierarchical Bayesian modeling. See `http://www.mrc-bsu.cam.ac.uk/bugs/`.

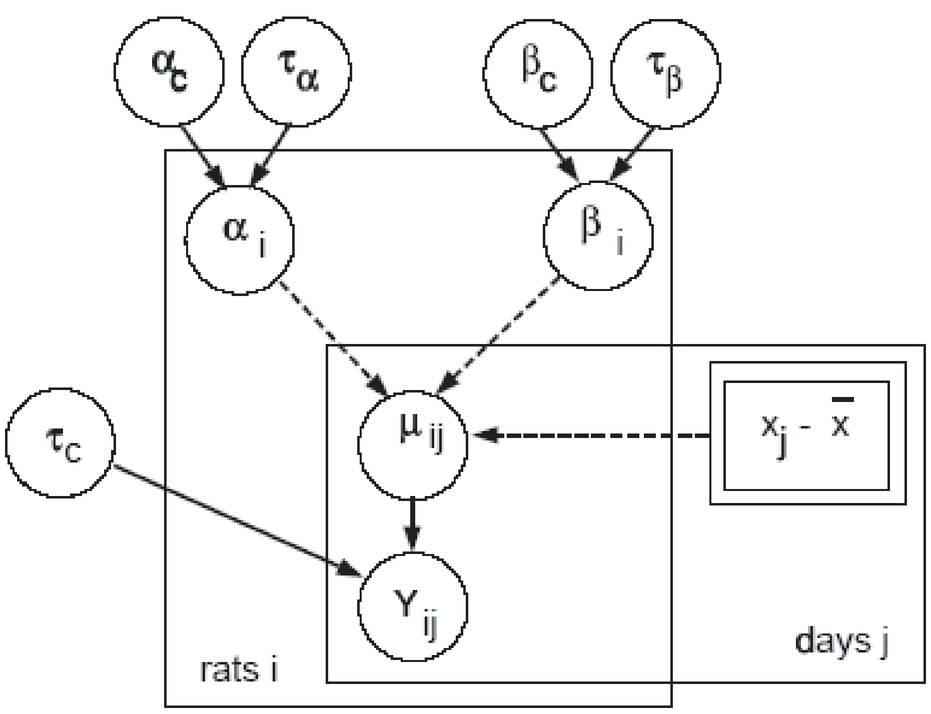


Figure 19: The rats model

For example, suppose y is binary, and x is 1D Gaussian. Let us assume global and local parameter independence. Given N iid cases, the complete probability model becomes

$$\begin{aligned} p(x_{1:N}, y_{1:N}, \theta_y, \theta_x) &= p(\theta_y)p(\theta_{x|0})p(\theta_{x|1}) \prod_{n=1}^{N} p(y_n|\theta_y)p(x_n|y_n, \theta_x) && (41) \\ &= \left[p(\theta_y) \prod_{n=1}^{N} p(y_n|\theta_y) \right] \left[p(\theta_{x|0}) \prod_{n:y_n=0} p(x_n|y_n, \theta_{x|0}) \right] \left[p(\theta_{x|1}) \prod_{n:y_n=1} p(x_n|y_n, \theta_{x|1}) \right] && (42) \\ &= \left[p(D_y|\theta_y)p(\theta_y) \right] \left[p(D_{x|0}|\theta_{x|0})p(\theta_{x|0}) \right] \left[p(D_{x|1}|\theta_{x|1})p(\theta_{x|1}) \right] && (43) \end{aligned}$$

where D_y is all the data needed to estimate θ_y, where $D_{x|0}$ is all the data needed to estimate $\theta_{x|0}$, etc. Since the likelihood and prior both factorize into a product of local terms, so does the posterior. Hence we can estimate each parameter vector separately. For example, suppose the variance is known but the mean is unknown, so $\theta_{x|y=c} = \mu_c$. If $p(\mu_c) \propto 1$ is a flat prior, we can use the MLE:

$$\hat{\mu}_c^{ML} = \arg\max_{\mu} p(D_{x|c}|\mu) = \frac{1}{N_c} \sum_{n:y_n=c} x_n \tag{44}$$

where N_c is the number of cases that are labeled with class c.

If some of the variables have **missing values**, and/or there are **hidden (latent) variables**, then the data no longer d-separates the parameters. Hence the parameter posteriors are no longer independent, and we must use more sophisticated methods, such as **EM** or **gradient descent** for MLE/MAP estimation, or **Gibbs sampling** or **variational methods** for Bayesian estimation.

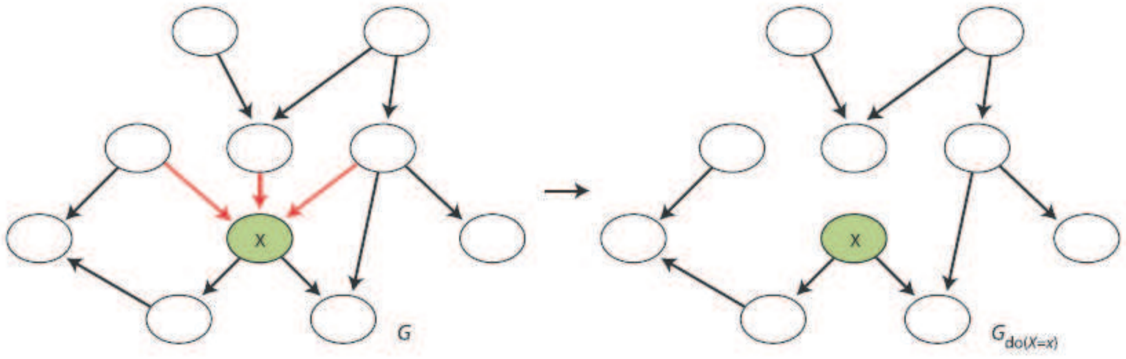


Figure 20: Surgical intervention on X. Source: [D05].

8 Causal interpretation of DGMs

DGMs are a very useful framework for making **causal models** [Pea00, SGS00]. These are (probabilistic) models which compactly encode the effects of **interventions**. A **perfect intervention** means setting a variable to some known value, say setting X_i to x_i. We need some notational convention to distinguish this from observing that X_i happens to have value x_i. We use Pearl's **do calculus** notation (as in the verb "to do") and write $\text{do}(X_i = x_i)$ to denote the event that we set X_i to x_i. A causal model defines a joint distribution $p(x)$, which can be used to make inferences of the form $p(x|(X_i = x_i))$, which is different from making inferences of the form $p(x|X_i = x_i)$.

To understand the difference between conditioning on interventions and conditioning on observations, consider a 2 node DGM $S \rightarrow Y$, in which $S = 1$ if you smoke and $S = 0$ otherwise, and $Y = 1$ if you have yellow-stained fingers. If I observe you have yellow fingers, I am licensed to infer that you are probably a smoker (since nicotine causes yellow stains):

$$p(S = 1|Y = 1) > p(S = 1) \tag{45}$$

However, if I intervene and *paint* your fingers yellow, I am no longer licensed to infer this, since I have disrupted the normal causal mechanism. Thus

$$p(S = 1|\text{do}(Y = 1)) \not> p(S = 1) \tag{46}$$

One way to model perfect interventions is to use **graph surgery**: simply cut the arcs coming into any nodes that were set by intervention. See Figure 20 for an example. A real world example of such a perfect intervention is a **gene knockout experiment**, in which a gene is "silenced" (i.e., forced to enter the "off" state).

To reason about the effects of interventions, just perform graph surgery and then perform probabilistic inference in the resulting "mutilated" graph. We state this formally as follows.

Theorem 2 (Manipulation theorem [Pea00, SGS00]) . *To compute $p(X_i|do(X_j))$ for sets of nodes i, j, perform surgical intervention on the X_j nodes and then use standard probabilistic inference in the mutilated graph.*

8.1 Simpson's paradox

We will show a dramatic example of the dangers of not thinking causally. Suppose taking a drug (cause C) decreases recovery rate (effect E) in females (F) and males ($\neg F$)

$$\begin{aligned} P(E|C, F) &< P(E|\neg C, F) \\ P(E|C, \neg F) &< P(E|\neg C, \neg F) \end{aligned}$$

but in the combined population, the drug increases recovery rate

$$P(E|C) > P(E|\neg C)$$

By the rules of probability, this is perfectly possible, as the table of numbers below shows.

	Combined				Male				Female			
	E	$\neg E$	Total	Rate	E	$\neg E$	Total	Rate	E	$\neg E$	Total	Rate
C	20	20	40	50%	18	12	30	60%	2	8	10	20%
$\neg C$	16	24	40	40%	7	3	10	70%	9	21	30	30%
Total	36	44	80		25	15	40		11	29	40	

$$p(E|C) = p(E,C)/p(c) = 20/40 = 0.5 \tag{47}$$
$$p(E|\neg C) = 16/40 = 0.4 \tag{48}$$
$$p(E|C,F) = 2/10 = 0.2 \tag{49}$$
$$p(E|\neg C,F) = 9/30 = 0.3 \tag{50}$$
$$p(E|C,\neg F) = 18/30 = 0.6 \tag{51}$$
$$p(E|\neg C,\neg F) = 7/10 = 0.7 \tag{52}$$

But the conclusion goes counter to intuition. Why? Put another way: given a new patient, do we use the drug or not? Novick wrote " The apparent answer is that when we know the gender of the patient, we do not use the drug, but if the gender is unknown, we should use the drug. Obviously that conclusion is ridiculous". (Quoted in [Pea00, p175].)

We can resolve the paradox as follows. The statement that the drug C causes recovery E is

$$P(E|\text{do}(C)) > P(E|\text{do}(\neg C)) \tag{53}$$

whereas the data merely tell us

$$P(E|C) > P(E|\neg C) \tag{54}$$

This is not a contradiction. Observing C is positive evidence for E, since more males than females take the drug, and the male recovery rate is higher (regardless of the drug). Thus Equation 54 does not imply Equation 53.

If we assume that the drug C does not cause gender F, as in Figure 21(left), then we can prove that if taking the drug is harmful in each subpopulation (male and female), then it must be harmful overall. Specifically, if we assume

$$p(E|\text{do}(C),F) < p(E|\text{do}(\neg C),F) \tag{55}$$
$$p(E|\text{do}(C),\neg F) < p(E|\text{do}(\neg C),\neg F) \tag{56}$$

then we can show

$$p(E|\text{do}(C)) < p(E|\text{do}(\neg C)) \tag{57}$$

The proof is as follows [Pea00, p181]. First we assume that drugs have no effect on gender

$$p(F|\text{do}(C)) = p(F|\text{do}(\neg C)) = p(F) \tag{58}$$

Now using the law of total probability,

$$p(E|\text{do}(C)) = p(E|\text{do}(C),F)p(F|\text{do}(C)) + p(E|\text{do}(C),\neg F)p(\neg F|\text{do}(C)) \tag{59}$$
$$= p(E|\text{do}(C),F)p(F) + p(E|\text{do}(C),\neg F)p(\neg F) \tag{60}$$

Similarly,

$$p(E|\text{do}(\neg C)) = p(E|\text{do}(\neg C),F)p(F) + p(E|\text{do}(\neg C),\neg F)p(\neg F) \tag{61}$$

Since every term in Equation 60 is less than the corresponding term in Equation 61, we conclude that

$$p(E|\text{do}(C)) < p(E|\text{do}(\neg C)) \tag{62}$$

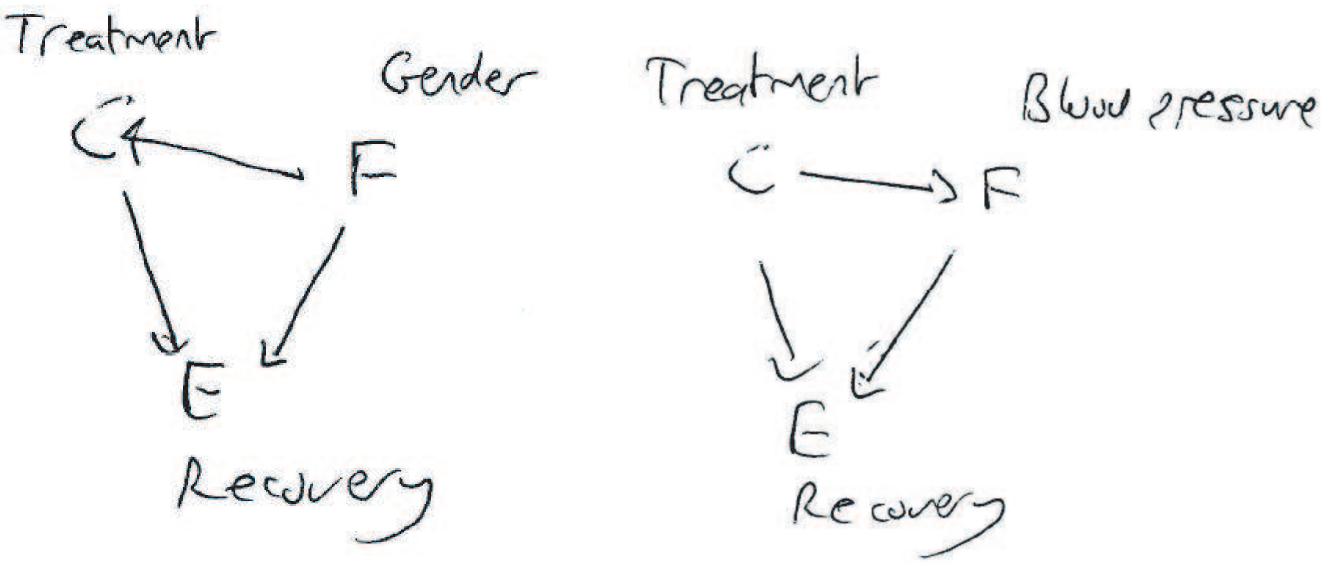


Figure 21: Two versios of the Simpson's paradox. Left: F is gender and causes C. Right: F is blood pressure and is caused by C.

To assess the effect of C on E, we have to take into account that there is a **backdoor path** from E to C via F. Pearl [Pea00, p79] proves that you have to adjust for (i.e., condition on) such backdoor variables. Intuitively, we need to be sure the effect of C on E is not due to their common cause, F. Thus we should check the $C \rightarrow E$ relationship for each value of F separately. In this example, the drug reduces E in both tables, so we should not take the drug regardless of gender.

Now consider a different cover story. Suppose we keep the data the same but interpret F as something that is affected by C, such as blood pressure. Thus F is now caused by C: see Figure 21(right). In this case, we can no longer assume

$$p(F|\text{do}(C)) = p(F|\text{do}(\neg C)) = p(F) \tag{63}$$

and the above proof breaks down. So $p(E|\text{do}(C)) - p(E|\text{do}(\neg C))$ may be positive or negaitve.

To assess the effect of C on E, we should look at the combined (C, E) table. We should not condition on F, since there is no backdoor path in this case. More intuitively, conditioning on F might block one of the causal pathways. In other words, by comparing patients with the same post-treatment blood pressure, we may mask the effect of one of the two pathways by which the drug operates to bring about recover.

Thus we see that different causal assumptions lead to different actions. In this case, the models require distinguishing the direction of arcs into/ out of the latent variable F, so we need prior domain knowledge to choose the right one.

8.2 Markov equivalence

$X \rightarrow Y$ and $X \leftarrow Y$ represent the same set of conditional independence statements (namely, none) and hence are called **Markov equivalent**. However, the **v-structure** $X \rightarrow Y \leftarrow Z$ encodes $X \perp Z$ and $X \not\perp Z|Y$, so is not Markov equivalent.

We can represent an equivalence class using a **PDAG (partially directed acyclic graph)**, 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 (see Figure 23). 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 22.

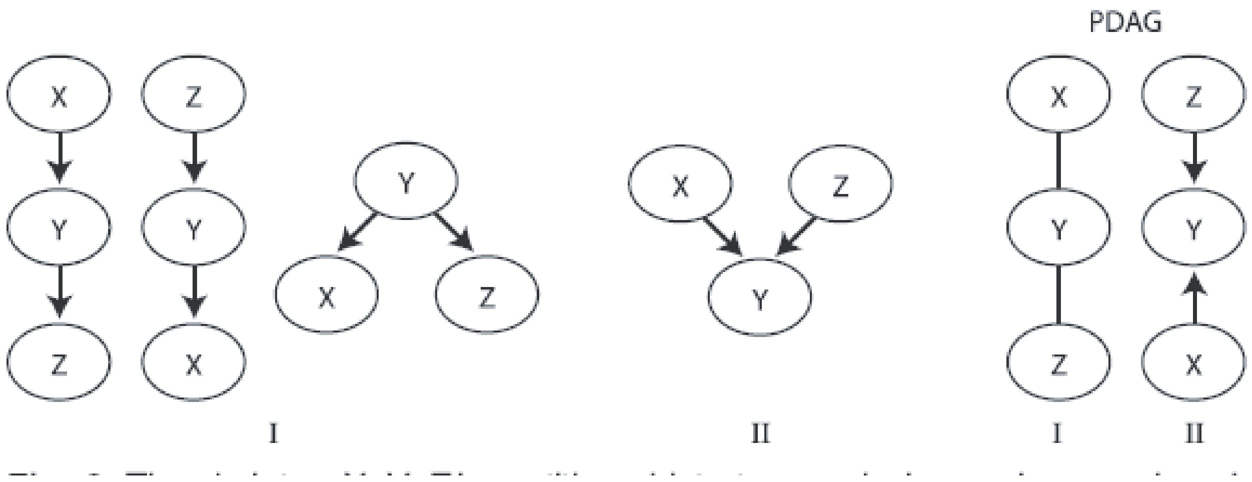


Figure 22: PDAG representation of Markov equivalent DAGs. Source:[D05]

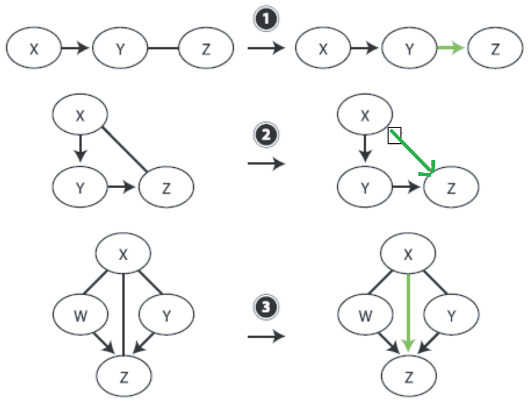


Figure 23: The 3 rules for inferring compelled edges in PDAGs. Source: [D05].

Theorem 3 (Verma and Pearl [VP90]) *Two structures are Markov equivalent if they have the same undirected skeleton and the same set of v-structures.*

9 Structure learning (model selection)*

Structure learning means inferring the graph structure given data. The simplest approach is to try to find a single best graph

$$G^* = \arg\max_G \text{score}(G) \tag{64}$$

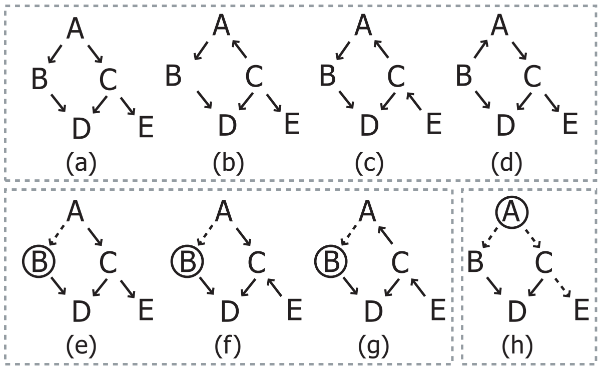


Figure 24: Top left: the "cancer network", from [FMR98]. (a-d) are Markov equivalent. (c-g) are equivalent under an intervention on B. (h) is the unique member under an intervention on A. Based on [TP01b].

Some possibilities for the **scoring function** include penalized likelihood criteria[5] such as MDL/BIC, or the (unnormalized) Bayesian posterior

$$p(G, D) = p(G)p(D|G) = p(G) \int p(D|G, \theta)p(\theta|G)d\theta \tag{65}$$

where $p(D|G)$ is called the marginal likelihood.

Alternatively, we may seek a sample of graphs from the posterior, $p(G|D)$. The reason a sample may be better than a point estimate is that there may be many graphs that have similar posterior probability, especially if $|G| \gg |D|$. These graphs differ in various ways; by looking at a sample, we can find the features they have in common.

Structure learning can be used to uncover gene regulatory networks from microarray expression data, social network structure from email traces, etc. We discuss this later.

9.1 Inferring causal structure

An important problem in causal inference is learning the causal newtork structure from data. Suppose we had an infinite data set. Then we could perfectly determine whether $A \perp B|C$ for any set of nodes A,B and C given the data. That is, we could simulate a conditional independency test **oracle**. Given such an oracle, we can identify all of the v-structures $A{\to}C{\leftarrow}B$, since they have a unique statistical signature of the form $A \perp B$ and $A \not\perp B|C$. The direction of the remaining edges will be ambiguous (since "correlation does not imply causation"), but by using the rules for compelling edges, we can infer some of their directions, too.

Even given an oracle, we can only identify the structure up to Markov equivalence. To distinguish between such members, we need to perform interventions: "no causation without manipulation". Essentially every time we perform a perfect intervention on a node, we are able to orient all edges into and out of that node. The result is called an **intervention equivalence class**. Thus by using a conditional independency test oracle to get the PDAG and then performing the "right" interventions, we can uniquely recover the generating DAG [TP01b, TP01a, EGS05]. See Figure 24 for an example.

Algorithmically, there are essentially two approaches to learning causal structure. The first is called the **constraint based approach**. It is essentially a deductive approach: we use a conditional independency test (with some fixed threshold) to answer yes or no to questions of the form $A \perp B|C$ for all sets A, B, C in increasing size of C. We then construct a PDAG that is consistent with these results. The second approach is to use standard **Bayesian model selection** techniques. It is essentially an inductive approach. We define a hypothesis space of DAGs, and evaluate their score (e.g., posterior probability) and return the "best". Since the hypothesis space is exponentially large, we need to combine the scoring function with search techniques. Although this is less computationally efficient than constraint based approaches, the Bayesian approach has the advantage that it can combine weak sources of evidence in a coherent

[5]We cannot use maximum likelihood, since that will always favor the fully connected graph.

fashion. The constraint based approach, on the other hand, relies on hard thresholding at the very first stage, and can never recover from errors made at this stage.

If there are **hidden common causes** (i.e., **confounders**), then both techniques may learn the wrong structure. For example, if the true structure is $A \leftarrow C \rightarrow B$, where C is a hidden common cause, then if we don't observe C, the best we can do is to learn the correlation between A and B.

References

[D05] D. Pe'er D. Bayesian network analysis of signaling networks: a primer. *Science STKE*, 281:14, April 2005.

[EGS05] F. Eberhardt, C. Glymour, and R. Scheines. On the number of experiments sufficient and in the worst case necessary to identify all causal relations among N variables. In *UAI*, 2005.

[FMR98] N. Friedman, K. Murphy, and S. Russell. Learning the structure of dynamic probabilistic networks. In *UAI*, 1998.

[KF06] D. Koller and N. Friedman. *Bayesian networks and beyond*. 2006. To appear.

[Pea00] J. Pearl. *Causality: Models, Reasoning and Inference*. Cambridge Univ. Press, 2000.

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

[SGS00] P. Spirtes, C. Glymour, and R. Scheines. *Causation, Prediction, and Search*. MIT Press, 2000. 2nd edition.

[TP01a] J. Tian and J. Pearl. Causal discovery from changes. In *UAI*, 2001.

[TP01b] J. Tian and J. Pearl. Causal discovery from changes: a Bayesian approach. Technical report, UCLA, 2001.

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