2

Introducing Bayesian Networks

2.1 Introduction

Having presented both theoretical and practical reasons for artificial intelligence to use probabilistic reasoning, we now introduce the key computer technology for dealing with probabilities in AI, namely **Bayesian networks.** Bayesian networks (BNs) are graphical models for reasoning under uncertainty, where the nodes represent variables (discrete or continuous) and arcs represent direct connections between them. These direct connections are often causal connections. In addition, BNs model the quantitative strength of the connections between variables, allowing probabilistic beliefs about them to be updated automatically as new information becomes available.

In this chapter we will describe how Bayesian networks are put together (the **syntax**) and how to interpret the information encoded in a network (the **semantics**). We will look at how to model a problem with a Bayesian network and the types of reasoning that can be performed.

2.2 Bayesian network basics

A **Bayesian network** is a graphical structure that allows us to represent and reason about an uncertain domain. The nodes in a Bayesian network represent a set of random variables, $\mathbf{X} = X_1, ..X_i, ...X_n$, from the domain. A set of directed **arcs** (or links) connects pairs of nodes, $X_i \rightarrow X_j$, representing the direct dependencies between variables. Assuming discrete variables, the strength of the relationship between variables is quantified by conditional probability distributions associated with each node. The only constraint on the arcs allowed in a BN is that there must not be any directed cycles: you cannot return to a node simply by following directed arcs. Such networks are called directed acyclic graphs, or simply **dags**.

There are a number of steps that a **knowledge engineer**[1] must undertake when building a Bayesian network. At this stage we will present these steps as a sequence; however it is important to note that in the real-world the process is not so simple. In Chapter 10 we provide a fuller description of BN knowledge engineering.

[1] Knowledge engineer in the jargon of AI means a practitioner applying AI technology.

Throughout the remainder of this section we will use the following simple medical diagnosis problem.

Example problem: Lung cancer. *A patient has been suffering from shortness of breath (called* dyspnoea*) and visits the doctor, worried that he has lung cancer. The doctor knows that other diseases, such as tuberculosis and bronchitis, are possible causes, as well as lung cancer. She also knows that other relevant information includes whether or not the patient is a smoker (increasing the chances of cancer and bronchitis) and what sort of air pollution he has been exposed to. A positive X-ray would indicate either TB or lung cancer.*[2]

2.2.1 Nodes and values

First, the knowledge engineer must identify the variables of interest. This involves answering the question: what are the nodes to represent and what values can they take, or what state can they be in? For now we will consider only nodes that take discrete values. The values should be both **mutually exclusive** and **exhaustive**, which means that the variable must take on exactly one of these values at a time. Common types of discrete nodes include:

- Boolean nodes, which represent propositions, taking the binary values true (T) and false (F). In a medical diagnosis domain, the node *Cancer* would represent the proposition that a patient has cancer.
- Ordered values. For example, a node *Pollution* might represent a patient's pollution exposure and take the values $\{$*low, medium, high*$\}$.
- Integral values. For example, a node called *Age* might represent a patient's age and have possible values from 1 to 120.

Even at this early stage, modeling choices are being made. For example, an alternative to representing a patient's exact age might be to clump patients into different age groups, such as $\{$*baby, child, adolescent, young, middleaged, old*$\}$. The trick is to choose values that represent the domain efficiently, but with enough detail to perform the reasoning required. More on this later!

TABLE 2.1
Preliminary choices of nodes and values for the lung cancer example.

Node name	Type	Values
Pollution	Binary	$\{$*low, high*$\}$
Smoker	Boolean	$\{T, F\}$
Cancer	Boolean	$\{T, F\}$
Dyspnoea	Boolean	$\{T, F\}$
X-ray	Binary	$\{$*pos, neg*$\}$

[2] This is a modified version of the so-called "Asia" problem Lauritzen and Spiegelhalter, 1988, given in §2.5.3.

For our example, we will begin with the restricted set of nodes and values shown in Table 2.1. These choices already limit what can be represented in the network. For instance, there is no representation of other diseases, such as TB or bronchitis, so the system will not be able to provide the probability of the patient having them. Another limitation is a lack of differentiation, for example between a heavy or a light smoker, and again the model assumes at least some exposure to pollution. Note that all these nodes have only two values, which keeps the model simple, but in general there is no limit to the number of discrete values.

2.2.2 Structure

The structure, or topology, of the network should capture qualitative relationships between variables. In particular, two nodes should be connected directly if one affects or causes the other, with the arc indicating the direction of the effect. So, in our medical diagnosis example, we might ask what factors affect a patient's chance of having cancer? If the answer is "Pollution and smoking," then we should add arcs from *Pollution* and *Smoker* to *Cancer*. Similarly, having cancer will affect the patient's breathing and the chances of having a positive X-ray result. So we add arcs from *Cancer* to *Dyspnoea* and *XRay*. The resultant structure is shown in Figure 2.1. It is important to note that this is just one possible structure for the problem; we look at alternative network structures in §2.4.3.

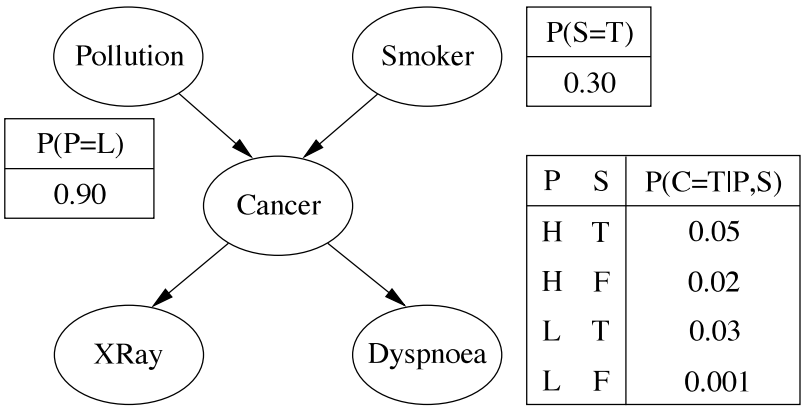

FIGURE 2.1: A BN for the lung cancer problem.

Structure terminology and layout

In talking about network structure it is useful to employ a family metaphor: a node is a **parent** of a **child**, if there is an arc from the former to the latter. Extending the

metaphor, if there is a directed chain of nodes, one node is an **ancestor** of another if it appears earlier in the chain, whereas a node is a **descendant** of another node if it comes later in the chain. In our example, the *Cancer* node has two parents, *Pollution* and *Smoker*, while *Smoker* is an ancestor of both *X-ray* and *Dyspnoea*. Similarly, *X-ray* is a child of *Cancer* and descendant of *Smoker* and *Pollution*. The set of parent nodes of a node X is given by *Parents(X)*.

Another useful concept is that of the **Markov blanket** of a node, which consists of the node's parents, its children, and its children's parents. Other terminology commonly used comes from the "tree" analogy (even though Bayesian networks in general are graphs rather than trees): any node without parents is called a **root** node, while any node without children is called a **leaf** node. Any other node (non-leaf and non-root) is called an **intermediate node**. Given a causal understanding of the BN structure, this means that root nodes represent original causes, while leaf nodes represent final effects. In our cancer example, the causes *Pollution* and *Smoker* are root nodes, while the effects *X-ray* and *Dyspnoea* are leaf nodes.

By convention, for easier visual examination of BN structure, networks are usually laid out so that the arcs generally point from top to bottom. This means that the BN "tree" is usually depicted upside down, with roots at the top and leaves at the bottom![3]

2.2.3 Conditional probabilities

Once the topology of the BN is specified, the next step is to quantify the relationships between connected nodes – this is done by specifying a conditional probability distribution for each node. As we are only considering discrete variables at this stage, this takes the form of a conditional probability *table* (CPT).

First, for each node we need to look at all the possible combinations of values of those parent nodes. Each such combination is called an **instantiation** of the parent set. For each distinct instantiation of parent node values, we need to specify the probability that the child will take each of its values.

For example, consider the *Cancer* node of Figure 2.1. Its parents are *Pollution* and *Smoking* and take the possible joint values $\{< H,T >, < H,F >, < L,T >, < L,F >\}$. The conditional probability table specifies in order the probability of cancer for each of these cases to be: $< 0.05, 0.02, 0.03, 0.001 >$. Since these *are* probabilities, and must sum to one over all possible states of the *Cancer* variable, the probability of no cancer is already implicitly given as one minus the above probabilities in each case; i.e., the probability of no cancer in the four possible parent instantiations is $< 0.95, 0.98, 0.97, 0.999 >$.

Root nodes also have an associated CPT, although it is degenerate, containing only one row representing its prior probabilities. In our example, the prior for a patient being a smoker is given as 0.3, indicating that 30% of the population that the

[3] Oddly, this is the antipodean standard in computer science; we'll let you decide what that may mean about computer scientists!

doctor sees are smokers, while 90% of the population are exposed to only low levels of pollution.

Clearly, if a node has many parents or if the parents can take a large number of values, the CPT can get very large! The size of the CPT is, in fact, exponential in the number of parents. Thus, for Boolean networks a variable with n parents requires a CPT with 2^{n+1} probabilities.

2.2.4 The Markov property

In general, modeling with Bayesian networks requires the assumption of the **Markov property**: there are no direct dependencies in the system being modeled which are not already explicitly shown via arcs. In our *Cancer* case, for example, there is no way for smoking to influence dyspnoea except by way of causing cancer (or not) — there is no hidden "backdoor" from smoking to dyspnoea. Bayesian networks which have the Markov property are also called **Independence-maps** (or, **I-maps** for short), since every independence suggested by the lack of an arc is real in the system.

Whereas the independencies suggested by a lack of arcs are generally required to exist in the system being modeled, it is not generally required that the arcs in a BN correspond to real dependencies in the system. The CPTs may be parameterized in such a way as to nullify any dependence. Thus, for example, every fully-connected Bayesian network can represent, perhaps in a wasteful fashion, any joint probability distribution over the variables being modeled. Of course, we shall prefer **minimal models** and, in particular, **minimal I-maps**, which are I-maps such that the deletion of any arc violates I-mapness by implying a non-existent independence in the system.

If, in fact, every arc in a BN happens to correspond to a direct dependence in the system, then the BN is said to be a **Dependence-map** (or, **D-map** for short). A BN which is both an I-map and a D-map is said to be a **perfect map**.

2.3 Reasoning with Bayesian networks

Now that we know how a domain and its uncertainty may be represented in a Bayesian network, we will look at how to use the Bayesian network to reason about the domain. In particular, when we observe the value of some variable, we would like to **condition** upon the new information. The process of conditioning (also called **probability propagation** or **inference** or **belief updating**) is performed via a "flow of information" through the network. Note that this information flow is *not* limited to the directions of the arcs. In our probabilistic system, this becomes the task of computing the posterior probability distribution for a set of **query** nodes, given values for some **evidence** (or **observation**) nodes.

2.3.1 Types of reasoning

Bayesian networks provide full representations of probability distributions over their variables. That implies that they can be conditioned upon any subset of their variables, supporting any direction of reasoning.

For example, one can perform **diagnostic reasoning**, i.e., reasoning from symptoms to cause, such as when a doctor observes *Dyspnoea* and then updates his belief about *Cancer* and whether the patient is a *Smoker*. Note that this reasoning occurs in the *opposite* direction to the network arcs.

Or again, one can perform **predictive reasoning**, reasoning from new information about causes to new beliefs about effects, following the directions of the network arcs. For example, the patient may tell his physician that he is a smoker; even before any symptoms have been assessed, the physician knows this will increase the chances of the patient having cancer. It will also change the physician's expectations that the patient will exhibit other symptoms, such as shortness of breath or having a positive X-ray result.

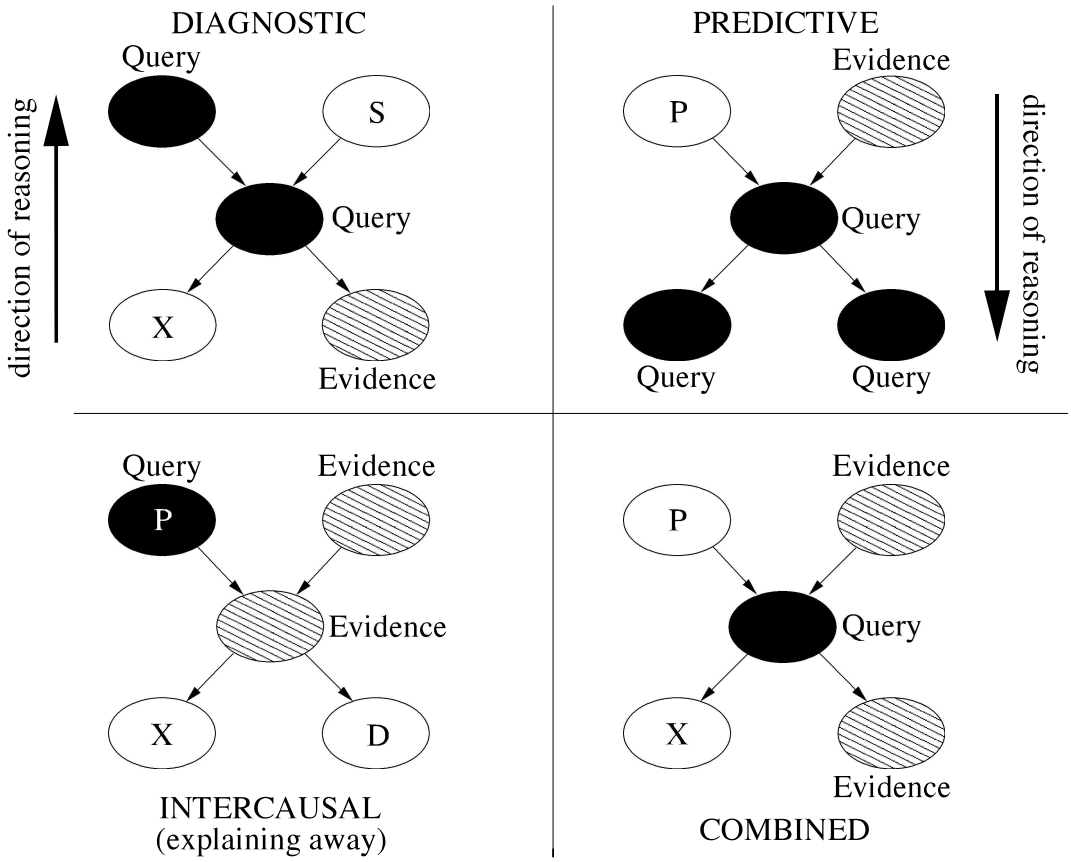

FIGURE 2.2: Types of reasoning.

A further form of reasoning involves reasoning about the mutual causes of a common effect; this has been called **intercausal reasoning**. A particular type called

explaining away is of some interest. Suppose that there are exactly two possible causes of a particular effect, represented by a **v-structure** in the BN. This situation occurs in our model of Figure 2.1 with the causes *Smoker* and *Pollution* which have a common effect, *Cancer* (of course, reality is more complex than our example!). Initially, according to the model, these two causes are independent of each other; that is, a patient smoking (or not) does not change the probability of the patient being subject to pollution. Suppose, however, that we learn that Mr. Smith has cancer. This will raise our probability for both possible causes of cancer, increasing the chances both that he is a smoker and that he has been exposed to pollution. Suppose then that we discover that he is a smoker. This new information explains the observed cancer, which in turn *lowers* the probability that he has been exposed to high levels of pollution. So, even though the two causes are initially independent, with knowledge of the effect the presence of one explanatory cause renders an alternative cause less likely. In other words, the alternative cause has been *explained away*.

Since any nodes may be query nodes and any may be evidence nodes, sometimes the reasoning does not fit neatly into one of the types described above. Indeed, we can combine the above types of reasoning in any way. Figure 2.2 shows the different varieties of reasoning using the Cancer BN. Note that the last combination shows the simultaneous use of diagnostic and predictive reasoning.

2.3.2 Types of evidence

So Bayesian networks can be used for calculating new beliefs when new information – which we have been calling **evidence** – is available. In our examples to date, we have considered evidence as a definite finding that a node X has a particular value, x, which we write as $X = x$. This is sometimes referred to as **specific evidence**. For example, suppose we discover the patient is a smoker, then *Smoker=T*, which is specific evidence.

However, sometimes evidence is available that is not so definite. The evidence might be that a node Y has the value y_1 *or* y_2 (implying that all other values are impossible). Or the evidence might be that Y is *not* in state y_1 (but may take any of its other values); this is sometimes called a **negative evidence**.

In fact, the new information might simply be any new probability distribution over Y. Suppose, for example, that the radiologist who has taken and analyzed the X-ray in our cancer example is uncertain. He thinks that the X-ray looks positive, but is only 80% sure. Such information can be incorporated equivalently to Jeffrey conditionalization of §1.5.1, in which case it would correspond to adopting a new posterior distribution for the node in question. In Bayesian networks this is also known as **virtual evidence**. Since it is handled via likelihood information, it is also known as **likelihood evidence**. We defer further discussion of virtual evidence until Chapter 3, where we can explain it through the effect on belief updating.

2.3.3 Reasoning with numbers

Now that we have described qualitatively the types of reasoning that are possible using BNs, and types of evidence, let's look at the actual numbers. Even before we obtain any evidence, we can compute a prior belief for the value of each node; this is the node's prior probability distribution. We will use the notation Bel(X) for the posterior probability distribution over a variable X, to distinguish it from the prior and conditional probability distributions (i.e., $P(X)$, $P(X|Y)$).

The exact numbers for the updated beliefs for each of the reasoning cases described above are given in Table 2.2. The first set are for the priors and conditional probabilities originally specified in Figure 2.1. The second set of numbers shows what happens if the smoking rate in the population increases from 30% to 50%, as represented by a change in the prior for the *Smoker* node. Note that, since the two cases differ only in the prior probability of smoking ($P(S = T) = 0.3$ versus $P(S = T) = 0.5$), when the evidence itself is about the patient being a smoker, then the prior becomes irrelevant and both networks give the same numbers.

TABLE 2.2
Updated beliefs given new information with smoking rate 0.3 (top set) and 0.5 (bottom set).

Node	No	Reasoning Case				
P(S)=0.3	Evidence	Diagnostic	Predictive	Intercausal		Combined
		D=T	S=T	C=T	C=T S=T	D=T S=T
Bel(P=high)	0.100	0.102	0.100	0.249	0.156	0.102
Bel(S=T)	0.300	0.307	1	0.825	1	1
Bel(C=T)	0.011	0.025	0.032	1	1	0.067
Bel(X=pos)	0.208	0.217	0.222	0.900	0.900	0.247
Bel(D=T)	0.304	1	0.311	0.650	0.650	1
P(S)=0.5						
Bel(P=high)	0.100	0.102	0.100	0.201	0.156	0.102
Bel(S=T)	0.500	0.508	1	0.917	1	1
Bel(C=T)	0.174	0.037	0.032	1	1	0.067
Bel(X=pos)	0.212	0.226	0.311	0.900	0.900	0.247
Bel(D=T)	0.306	1	0.222	0.650	0.650	1

Belief updating can be done using a number of exact and approximate inference algorithms. We give details of these algorithms in Chapter 3, with particular emphasis on how choosing different algorithms can affect the efficiency of both the knowledge engineering process and the automated reasoning in the deployed system. However, most existing BN software packages use essentially the same algorithm and it is quite possible to build and use BNs without knowing the details of the belief updating algorithms.