Bayesian networks

Charles Elkan

October 23, 2012

Important: These lecture notes are based closely on notes written by Lawrence Saul. Text may be copied directly from his notes, or paraphrased. Also, these typeset notes lack illustrations. See the classroom lectures for figures and diagrams.

Probability theory is the most widely used framework for dealing with uncertainty because it is quantitative and logically consistent. Other frameworks either do not lead to formal, numerical conclusions, or can lead to contradictions.

1 Random variables and graphical models

Assume that we have a set of instances and a set of random variables. Think of an instance as being, for example, one patient. A random variable (rv for short) is a measurement. For each instance, each rv has a certain value. The name of an rv is often a capital letter such as X. An rv may have discrete values, or it may be real-valued.

Terminology: Instances and rvs are universal in machine learning and statistics, so many other names are used for them depending on the context and the application. Generally, an instance is a row in a table of data, while an rv is a column. An rv may also be called a feature or an attribute. It is vital to be clear about the difference between an rv and a possible value of the rv. Values are also called outcomes. One cell in a table of data contains the particular outcome of the column rv for the row instance. If this outcome is unknown, we can think of a special symbol such as "?" being in the cell. Here, "unknown" and "?" and "missing" are all synonyms.

A graphical model is a diagram that describes the relationships between random variables. Formally, a graphical model is a graph whose nodes are rvs. For

example, we might have five nodes named flu, meningitis, fever, cough, and stiff neck (draw picture with directed edges).

Intuitively, if there is an edge between two rvs, then the value of each rv directly provides information about the value of the other. If there is a path between two rvs, then their values are informative about each other indirectly. If there is no path between two rvs, then their values are not correlated. Later, we will make this notion of being informative mathematically formal using probability theory. We will see how graphical models represent correlation, independence, and conditional independence.

A directed graphical model is one where the edges form a directed acyclic graph (DAG). A graphical model of this type is called a Bayesian network (BN). BNs are also called belief networks, and causal networks. Often, when a BN is used to describe a real-world scenario, a directed edge goes from a cause to a symptom. However, causation is a delicate issue that will be discussed separately later.

Here are some examples of BNs.

- Symptoms are observable medical conditions such as fever, cough, jaundice, and pain. Causes are diagnoses, for example infections including tuberculosis, flu, meningitis, and pneumonia.

  Because the infections are not mutually exclusive, they are represented as separate rvs. The same is true for the symptoms.

- Symptoms are grayscale images, perhaps 28 by 28 grids, and causes are digits 0 to 9. This describes the famous MNIST dataset. Each of the 28^2 pixels is a separate rv.

  The digits are mutually exclusive, so they are represented by a single rv with ten possible values. Edges go from this rv to each pixel rv. It makes the scenario more complicated, but also more realistic, if there is an edge between each pair of neighboring pixels. Note that by definition an edge is only allowed between two rvs, not between three or more.

- Symptoms are word counts and the cause is a binary rv whose two values are spam and not-spam. Each word in the dictionary is a separate rv, whose value is an integer. For each instance, most symptom rvs have the value zero, because in any given email message, the count of most words is zero.

Edges go from causes to symptoms. However, a common type of reasoning is to

figure out the probability of each possible cause, given a value for each possible symptom.

Note again the distinction between an rv and its possible values. For example, in the email example there is one rv, which we may call "status" which has two possible values, spam and not-spam. Values must be mutually exclusive and exhaustive.

For a given BN, the values of different rvs can be known for different instances. Figuring out the unknown values given known values is called inference. In general, we cannot work out a deterministic value when the actual value of an rv is unknown. We can only work out a distribution, that is a range of values with associated probabilities. This fact makes reasoning with BNs cumbersome difficult compared to reasoning with approaches that are not probabilistic. However, probabilistic formulations describe the world more accurately. Figuring out the parameters of the BN, given instances, is called learning.

End of the lecture on Tuesday October 2.

Various BNs can capture many common data modeling scenarios, including standard supervised and unsupervised learning situations.

Supervised learning means training a classifier, where a classifier is a function that predicts labels. In this scenario we have input rvs and one discrete-valued output rv, called a label or class variable. Often, each input rv is real-valued, named X_1 to X_d, and the output rv is named Y. If $Y \in \{1, \ldots, c\}$ then there are c classes. Typically, the input rvs are symptoms while the label is a cause, so arrows in the BN go from the label to each input rv.

Label prediction means inferring the value of Y, given values for X_1 to X_d. An algorithm that performs label prediction is called a classifier. Classifier learning means creating such a classifier based on training data. Training data consist of n instances, for each of which X_1 to X_d and Y all have known values. It is especially easy to draw a picture when the dimensionality $d = 2$ and the number of classes $c = 2$.

Unsupervised learning generally means inducing patterns that somehow explain or summarize data. Clustering is a special case of unsupervised learning The BN here is actually the same as for multiclass classification. The difference is that the value of the class rv is never observed. Notice the difference in technical meaning between the phrases clustering, class prediction, and classifier learning. In ordinary English, the word classification means the same as clustering. In machine learning, it tends to mean classifier learning.

2 Elements of probability theory

Consider a single discrete rv named X, and suppose that its possible basic outcomes are x_1 to x_m. Basic outcomes must be mutually exclusive and exhaustive. Each outcome, also called a value, has a probability $p(X = x_i)$. The foundational axioms of probability theory say that

$$\begin{aligned} p(X = x_i) &\geq 0 \\ \sum_{i=1}^{m} p(X = x_i) &= 1 \\ p(X = x_i \text{ or } X = x_j) &= p(X = x_i) + p(X = x_j) \text{ if } x_i \neq x_j. \end{aligned}$$

The last equation says that the probabilities of mutually exclusive events can be added.

The notation $p(X = x_i|Y = y_j)$ means the probability that the outcome of X is x_i given that it is certain that the outcome of Y is y_j. This is called the probability of $X = x_i$ conditioned on $Y = y_j$. The vertical bar indicates conditioning. In general, $p(X = x_i|Y = y_j) \neq p(X = x_i)$. For example, let M be a random variable whose value is a month of the year, and let W be a random variable whose value is either "sunny" or "rainy." Numerically, for San Diego we might have

$$\begin{aligned} p(W = \text{sunny}) &= 0.8 \\ p(W = \text{sunny}|M = \text{august}) &= 0.9 \\ p(W = \text{sunny}|M = \text{may}) &= 0.5. \end{aligned}$$

If there exist values x and y such that $p(X = x|Y = y) \neq p(X = x)$ then we say that the rvs X and Y are dependent. If no such values exist for X and Y, then the rvs are independent. For example, let the value of the rv D be one of the seven days of the week. If $p(W = \text{sunny}|D = \text{monday}) = p(W = \text{sunny})$ and so on for every day of the week, then W and D are independent.

A fact such as $Y = y$ is called an event. The laws of probability theory remain true when conditioning on any event. Specifically,

$$\begin{aligned} p(X = x_i|Y = y) &\geq 0 \\ \sum_{i=1}^{m} p(X = x_i|Y = y) &= 1 \\ p(X = x_i \text{ or } X = x_j|Y = y) &= p(X = x_i|Y = y) + p(X = x_j|Y = y) \text{ if } x_i \neq x_j. \end{aligned}$$

Note that the second equation above involves a sum over i, that is over all possible outcomes of X. Nothing specific can be said about the sum over the outcomes of Y, that is $\sum_j p(X = x|Y = y_j)$.

A joint probability is the probability of a conjunction:

$$p(X = x, Y = y) = p(X = x \text{ and } Y = y).$$

The product rule says that for any values x and y

$$p(X = x, Y = y) = p(X = x|Y = y)p(Y = y).$$

The order of the conjuncts in a conjunction does not matter, so it is also true that

$$p(X = x, Y = y) = p(Y = y|X = x)p(X = x).$$

Marginalizing means summing over the alternative values of an rv, which eliminates the rv from further consideration. Specifically,

$$p(X = x_i) = \sum_j p(X = x_i, Y = y_j)$$

and

$$p(X = x_i, Y = y_j) = \sum_k p(X = x_i, Y = y_j, Z = z_k)$$

and so on.

It is cumbersome to be explicit about the possible values of random variables. We use a shorthand notation called implied universality: when the symbol for an rv, for example X, appears where an event is needed syntactically, then $X = x_i$ is meant for every outcome x_i of X. For example,

$$p(X, Y) = p(X)p(Y|X)$$

means

$$\forall x_i \; \forall y_j \; p(X = x_i, Y = y_j) = p(X = x_i)p(Y = y_j|X = x_i).$$

With this notation, the generalized product rule is

$$p(A, B, C, D, \ldots) = p(A)p(B|A)p(C|A, B)p(D|A, B, C) \ldots$$

Conversely, when the symbol for an outcome, for example z_j, appears where an event is needed syntactically, then the event $Z = z_j$ is meant, where Z is the

relevant random variable. If a random variable B is binary, then by default its possible outcomes are 0 and 1, and saying "B is true" or just B means $B = 1$.

Using implied universality, Bayes' rule is

$$p(X|Y) = \frac{p(Y|X)p(X)}{p(Y)}$$

and the generalized Bayes rule is

$$p(X|Y,Z) = \frac{p(Y|X,Z)p(X|Z)}{p(Y|Z)}.$$

The generalized Bayes rule is an example of how conditioning on an event essentially creates a new, restricted probability universe within which all the rules of probability theory remain valid.

3 An example of a Bayesian network

This section goes through a classic example of a BN. The example illustrates how the rules of probability theory are applied for inference, and also shows concretely all the details of an especially simple BN.

The BN involves three binary rvs, whose names are B, E, and A, for burglary, earthquake, and alarm. The BN has two directed edges, one from B to A and one from E to A. These edges capture the fact that the alarm can be triggered by either a burglary or an earthquake. The edges point to the node A, because the alarm is essentially a symptom of which burglary and earthquake are potential causes.

There is no edge between B and E because these are independent rvs. Although that is highly unlikely, it is possible for B and E to both be true.

To specify the BN fully, we need to specify the distributions of B and E, and we need to specify how the distribution of A depends on the four possible combined values of B and E. In a real application, some or all of these distributions would be learned from data. For now, we will posit them based on our own human background knowledge.

To specify the distribution of B, we just need to say that $p(B = 1) = 0.001$. It follows then that $p(B = 0) = 1 - p(B = 1) = 0.999$. Similarly, we can say that $p(E = 1) = 0.002$ and infer that $p(E = 0) = 0.998$. Note that these numbers are arbitrary, chosen just for the purposes of this example. In general, it is difficult for humans to estimate probabilities accurately. Moreover, for the probability of

b	e	$p(A=1\|B=b,E=e)$	$p(A=0\|B=b,E=e)$
0	0	0.001	$\vdots$
0	1	0.94	$1-p(A=1\|B=b,E=e)$
1	0	0.29	$\vdots$
1	1	0.95	$\vdots$

Table 1: Conditional probability table for the "alarm" random variable.

an event to be well-defined, the event must be defined carefully. Here, the event $E = 1$ might mean "an earthquake happens with shaking of magnitude 4.0 or higher at the location of my house during a given 24 hour period." A probability of 0.002 means 1 in 500, which is unrealistically high for San Diego. Nevertheless, we will continue with these numbers.

We need to specify $p(A|B = b, E = e)$ for all possible binary values b and e. We do this with what is called a conditional probability table (CPT). For the current scenario, see Table 1.

End of the lecture on Thursday October 4.

To see how the Bayesian network can be used for reasoning, consider first $p(B = 1)$, then $p(B = 1|A = 1)$, and then $p(B = 1|A = 1, E = 1)$. In words, we have three questions: what is the baseline probability of burglary, what is its probability given that the alarm is triggered, and finally, what is the probability of burglary given that the alarm is triggered and also that an earthquake did happen?

To answer the first question, by assumption, $p(B = 1) = 0.001$. To answer the second question, we can use Bayes' rule, which says

$$p(B = 1|A = 1) = p(A = 1|B = 1)p(B = 1)/p(A = 1).$$

We have

$$\begin{aligned} p(A=1|B=1) &= \sum_{e=0}^{1} p(A=1,E=e|B=1) \\ &= \sum_{e=0}^{1} p(A=1|E=e,B=1)p(E=e|B=1) \end{aligned}$$

$$
\begin{aligned}
&= \sum_{e=0}^{1} p(A=1|E=e,B=1)p(E=e) \\
&= p(A=1|E=0,B=1)p(E=0) + p(A=1|E=1,B=1)p(E=1) \\
&= 0.29 \cdot 0.998 + 0.95 \cdot 0.002 \\
&= 0.29132.
\end{aligned}
$$

We can calculate $p(A=1)$ using marginalization:

$$p(A=1) = p(A=1,B=1) + p(A=1,B=0).$$

Again by marginalization,

$$
\begin{aligned}
p(A=1,B=0) &= p(A=1,B=0,E=0) + p(A=1,B=0,E=1) \\
&= p(A=1|B=0,E=0)p(B=0)p(E=0) \\
&\quad + p(A=1|B=0,E=1)p(B=0)p(E=1) \\
&= 0.001 \cdot 0.999 \cdot 0.998 + 0.94 \cdot 0.999 \cdot 0.002 \\
&= 0.002875122
\end{aligned}
$$

Similarly,

$$
\begin{aligned}
p(A=1,B=1) &= p(A=1,B=1,E=0) + p(A=1,B=1,E=1) \\
&= p(A=1|B=1,E=0)p(B=1)p(E=0) \\
&\quad + p(A=1|B=1,E=1)p(B=1)p(E=1) \\
&= 0.29 \cdot 0.001 \cdot 0.998 + 0.95 \cdot 0.001 \cdot 0.002 \\
&= 0.00029132
\end{aligned}
$$

Putting the results above together,

$$p(A=1) = 0.002875122 + 0.00029132$$

and

$$
\begin{aligned}
p(B=1|A=1) &= p(A=1|B=1)p(B=1)/p(A=1) \\
&= 0.29132 \cdot 0.001/(0.002875122 + 0.00029132) \\
&= 0.092 \text{ approximately.}
\end{aligned}
$$

Why is this answer reasonable? An earthquake is twice as likely as a burglary, and an earthquake is more than three times as effective than a burglary at triggering

the alarm. Hence, given that the alarm is triggered, a quake is much more likely than a burglary.

Usually, there is more than one way of calculating a probability. Assuming that the denominator is not zero, $p(B = 1|A = 1) = p(A = 1, B = 1)/p(A = 1)$. As before,

$$\begin{aligned} p(A = 1) &= p(A = 1, B = 1) + p(A = 1, B = 0) \\ &= p(A = 1, B = 0, E = 0) + p(A = 1, B = 0, E = 1) \\ &\quad + p(A = 1, B = 1, E = 0) + p(A = 1, B = 1, E = 1) \\ &= 0.002875122 + 0.00029132. \end{aligned}$$

We get $p(B = 1|A = 1) = 0.00029132/(0.002875122 + 0.00029132)$ which is the same answer as before.

To answer the third question,

$$\begin{aligned} p(B = 1|A = 1, E = 1) &= \frac{p(B = 1, A = 1, E = 1)}{p(B = 1, A = 1, E = 1) + p(B = 0, A = 1, E = 1)} \\ &= \frac{0.95 \cdot 0.001 \cdot 0.002}{0.95 \cdot 0.001 \cdot 0.002 + 0.94 \cdot 0.999 \cdot 0.002} \\ &= 0.00101 \text{ approximately.} \end{aligned}$$

We have that

$$p(B = 1) < p(B = 1|A = 1) > p(B = 1|A = 1, E = 1).$$

This pattern of numerical probabilities is called "explaining away." One piece of information, that is the fact $A = 1$, increases the probability of $B = 1$, but a second, additional piece of information, that is $E = 1$, reduces the probability of $B = 1$. Intuitively, if $E = 1$ is known, then it is enough to explain $A = 1$, so $B = 1$ is not needed as an explanation, and its probability returns to being ciose to its baseline, which is 0.001 here.

Explaining away is an example of what is called nonmonotonic reasoning. In classical logic, reasoning can only be monotonic. Knowing additional true facts can only increase the number of facts that can be deduced.

4 Bayesian networks in general

The joint distribution of n binary random variables has $2^n - 1$ parameters, and in general specifying a distribution over n discrete random variables requires a

number of parameters that is exponential in n. A Bayesian network is a more compact representation, for which efficient algorithms exist for inference.

Definition. A Bayesian network (BN) is a directed acyclic graph (DAG)o where nodes are random variables, edges represent conditional dependence, and a conditional probability tables (CPT) exists for each node. A CPT specifies the probability of each outcome of a node, for each combination of outcomes of the parents of the node.

What is critical about a BN is that the *absence* of an edge indicates conditional independence, thus reducing the number of parameters needed to specify the joint distribution. Let the nodes of the BN be the rvs X_1 to X_n. Because the BN is a DAG, we can order the nodes so that the parents of each node X_i are a subset of X_1 to X_{i-1}. Let this subset be denoted parents(X_i) or pa(X_i) for short. By definition, the joint distribution represented by the BN is

$$p(X_1, \ldots, X_n) = \prod_{i=1}^{n} p(X_i|\text{pa}(X_i)).$$

This decomposition is a simplification compared to the universal decomposition

$$p(X_1, \ldots, X_n) = \prod_{i=1}^{n} p(X_i|X_1, \ldots, X_{i-1}).$$

To construct a BN that captures knowledge about a certain domain, there are four steps:

(i) Identify the relevant random variables.

(ii) Identify an appropriate ordering of the random variables.

(iii) Taking the nodes in order from X_1 to X_n, make the parents of X_i be the smallest subset of X_1 to X_{i-1} on which X_i is dependent.

(iv) For each X_i, specify the CPT for $p(X_i|\text{pa}(X_i))$.

When Bayesian networks were first introduced, researchers expected their structure and the parameter values inside CPTs to be obtained by interviewing experts, in a process called knowledge engineering. Nowadays, learning CPT parameters (step iv) from data is more common. There is research on learning the BN structure (step iii) from data, but in many applications human experts can provide this knowledge.

End of the lecture on Tuesday October 9, which also covered Simpson's paradox as explained in the sample solution to the October 9 quiz.

For step (ii), the way to select an ordering is to identify the variables that are the most basic causes first, then the variables that they influence directly, and so on. Usually, this procedure will yield only a partial ordering. If so, the ordering can be extended arbitrarily to be total. If a different ordering is selected, the BN will still be correct, assuming that step (iii) is performed correctly, but it will be less parsimonious than it could be. In the worst case, a wrong ordering can lead to the universal decomposition shown above.

Bayesian networks have many advantages. They are complete, consistent, compact representations of joint distributions, with no redundancy. For binary random variables, if the maximum number of parents of any node is k, then $O(n2^k)$ numerical parameters are needed to represent a joint distribution over 2^n combinations of outcomes. Typically, the number of known instances is far smaller than this exponential number, so it is pointless to try to learn this many parameters from data. A Bayesian network makes it feasible to learn a high-dimensional joint distribution from data, by encoding assumptions about the distribution in its structure.

A related advantage of Bayesian networks is that they provide an elegant separation of quantitative and qualitative knowledge. The DAG of a BN encodes independence relationships, while the CPTs encode numerical relationships. As mentioned above, often the qualitative knowledge is acquired from human experts, while the quantitative knowledge is learned from data.

How should a CPT be represented? Consider a node Y with parents X_1 to X_k. If each random variable has v possible outcomes, then a full lookup table requires $(v-1)v^k$ numbers, namely $p(Y = y|X_1 = x_1, \ldots, X_k = x_k)$ for every possible combination of outcomes x_1 to x_k, and all outcomes y except one. The probability for the remaining outcome of Y can be computed from the fact

$$\sum_y p(Y = y|X_1 = x_1, \ldots, X_k = x_k) = 1.$$

Some important special cases of CPT that require fewer parameters. CPTs can be deterministic. Suppose that all random variables are binary. The CPT for an "and" node is by definition

$$p(Y = 1|X_1 = x_1, \ldots, X_k = x_k) = \prod_{i=1}^{k} x_i$$

and the CPT for an "or" is

$$p(Y = 0|X_1 = x_1, \ldots, X_k = x_k) = \prod_{i=1}^{k}(1 - x_i).$$

Note that all x_i above are either zero or one, so the products are also either zero or one.

A so-called "noisy or" is a useful special case where only k numbers $p_i \in [0, 1]$ are used to parameterize the 2^k entries of a CPT. Specifically, a noisy-or CPT is defined to be

$$p(Y = 0|X_1 = x_1, \ldots, X_k = x_k) = \prod_{i:x_i=1} (1 - p_i) = \prod_{i=1}^{k}(1 - p_i)^{x_i}.$$

Consider the probability that $Y = 1$ when $X_i = 1$ and all other $X_j = 0$:

$$p(Y = 1|X_1 = \ldots = X_{i-1} = 0, X_i = 1, X_{i+1} = \ldots = X_k = 0) = 1-(1-p_i) = p_i.$$

Intuitively, p_i is the probability that X_i being on by itself makes Y be on.

A sigmoid CPT is a different parameterization that uses the same number of parameters, which are also called weights. Again for binary random variables, a sigmoid CPT says

$$p(Y = 1|X_1 = x_1, \ldots, X_k = x_k) = \sigma(\sum_{i=1}^{k} w_i x_i)$$

where the sigmoid function is

$$\sigma(z) = \frac{1}{1 + e^{-z}}.$$

This function converts a real number in the range $-\infty$ to $+\infty$ to a number between zero and one, with $\sigma(0) = 0.5$. A sigmoid CPT is more flexible than a noisy-or in at least two important ways. First, if the weight w_i is negative, then the influence of X_i is reversed: the value $X_i = 1$ makes the probability that $Y = 1$ smaller. Second, the parent random variables X_i can be real-valued; they do not need to be binary.

A sigmoid CPT is also called a logistic regression model. Generally, logistic regression is preferable to the noisy-or model when parameters are learned from data, but it may be easier for human experts to suggest parameter values for a noisy-or model.

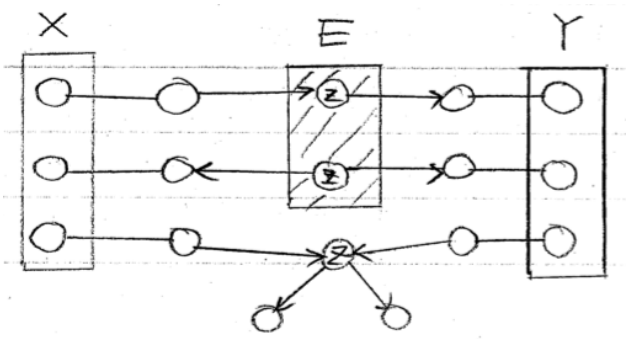

Figure 1: Three alternative ways of being d-separated. Figure drawn by Lawrence Saul.

5 The logic of conditional independence

In a Bayesian network a node is conditionally independent of its ancestors that are not its parents:

$$p(X_i|X_1, \ldots, X_{i-1}) = p(X_i|\text{pa}(X_i)).$$

Remember that arrows go from parents to children, and that the set $\text{pa}(X_i)$ is a subset of $\{X_1, \ldots, X_{i-1}\}$. What other independence relationships are true? Specifically, let A, B, and E refer to sets of nodes. When are the variables in the set A independent of those in B, conditional on the evidence set E?

Reminder: The variable M is independent of N conditional on Z if, when the outcome of Z is known, then the outcome of N does not provide additional information about the outcome of M. Conditional independence is symmetric: if M is independent of N given Z then N is independent of M given Z. "Conditional on the evidence set E" means assuming that the outcome of each variable in E is known.

The concept that relates conditional independence and the graph structure of a Bayesian network is called d-separation, where "d" means direction-dependent.

Definition. An undirected path (sometimes called a trail) between nodes M and N is d-separated by the set of nodes E if and only if there exists a node Z in the undirected path such that one of the three cases shown in Figure 1 is true. In these cases, essentially, Z blocks the flow of information between an outcome for M and an outcome for$N$.

1. In the first case, Z is in E and is the middle node in a directed subpath of

length two somewhere between M and N. Because the outcome of Z is known, the outcome of M does not provide additional information about the outcome of N, and vice versa.

2. In the second case, z is in E and influences M and N separately. Again because the outcome of Z is known, M and N are independent.

3. In the third case, M and N both influence Z. However, neither Z nor any of its descendants is in E. Hence, nothing is known about the value of Z, so M and N remain independent.

The definition of d-separation provides the answer to the question asked above, when are the variables in the set A independent of those in B, conditional on the evidence set E? In other words, when does $p(A, B|E) = p(A|E)p(B|E)$?

Theorem. The sets A and B are independent conditional on E if and only if every undirected path from a node in A to a node in B is d-separated by E.

End of the lecture on Thursday October 11.

The third case in the definition of d-separation is the least obvious one. Before Judea Pearl in the 1980s, no one had explained it clearly. The node Z is called a collider because it has two incoming edges. In this case, knowing the outcome of the cause M provides no information about the outcome of the cause N, if the effect Z is unknown. However, if the outcome of the effect is known, then knowing about one cause does provide information about the other ("explaining away" is a special case). Knowing about a descendant of Z implies knowing something about Z, although in general it does not imply knowing the outcome of Z deterministically.

Proving the theorem is not easy, but algorithms do exist to test efficiently whether or not two sets of nodes are d-separated by a third set. Here, we will just look at an example. Consider five binary random variables, B for burglary, E for earthquake, A for alarm as before, and J for John phones and M for Mary phones. The only edges are from B sand E to A, and then from A to J and M.

- Is B independent of M given A? Note that the following are equivalent ways of expressing this conditional independence:

$$\begin{aligned} p(B|M, A) &= p(B|A) \\ p(B, M|A) &= p(B|A)p(M|A) \\ p(M|B, A) &= p(M|A). \end{aligned}$$

One thing that all probabilities in the equations above have in common is that they are conditioned on A. The answer is yes, that is all the equations are true, because of the first case in the definition of d-separation. We can say informally that A is an intermediate cause between B and M, and the outcome of A is known.

- Are J and M independent given A? The answer is yes because of the second case in the definition of d-separation. Informally, A is a shared cause of J and M, and the outcome of A is known.

- Are B and E independent? Yes. Note that this question is about unconditional independence, and the answer is obvious given that B and E have no parents. However, we need to check that the definition of d-separation gives the correct answer. There is an undirected path between B and E that goes through A. A is a shared effect, but its outcome is not known, so the third case in the definition of d-separation is applicable.

- Given A, are B and E independent? No. Given A means that the value of A is known. The undirected path with edges from B to A and E to A does not satisfy any of the three d-separation cases.

End of the lecture on Tuesday October 16. See also the sample solution to the day's quiz.

6 Inference in polytrees

We need a general algorithm for doing inference given a Bayesian network. For any set E of evidence nodes, and any set Q of query nodes, we want an algorithm that computes $p(Q|E)$. What this means is that we want to compute the conditional probability $p(Q = q|E = e)$ for any fixed sets of outcomes q and e. It turns out that an efficient algorithm exists when the Bayesian network is a polytree, but not otherwise. Efficient means that the running time of the algorithm is a polynomial function of the size of the DAG and of the CPTs.

Definition. A polytree is a singly connected graph: between any two nodes there exists exactly one undirected path.

Note that some definitions of a polytree allow it to be a forest, that is, between some nodes there may exist no undirected path.

The problem can be reduced to computing $p(X = x|E = e)$ where X is one query node and $E = \{E_1, E_2, \ldots\}$ is a set of evidence nodes, because the chain rule can be applied to any query set Q. (Unfortunately, there is no similar chain rule for an evidence set E.) If X is in E then X is E_i for some i and trivially

$$p(X = x|E = e) = I(x = e_i)$$

where e_i is the outcome of E_i given in the condition $E = e$.

Assume that X is not in E. Let its parents be U_i for $i = 1$ to $i = m$ and let its children be Y_j for $j = 1$ to $j = n$. For each Y_j let the other parents of X be Z_{jk}. These other parents are often called spouses, but coparents would be a better name.

A critical observation is that the entire polytree, except X itself, can be divided into $m + n$ regions that do not overlap, one region for each parent and one region for each child. The evidence nodes inside the parent region i are called E_i^+ and the evidence nodes inside the child region j are called E_j^-. The set E^+ is the union of the E_i^+ sets and E^- is the union of the E_j^- sets, so $E = E^+ \cup E^-$.

Remember that $p(A|B, C) = p(A|B)$ is one way of saying that A and C are independent conditional on B. We will invoke d-separation several times to show conditional independence, and hence simplify an expression like $p(A|B, C)$ into $p(A|B)$. Because the Bayesian network is a polytree, whenever we invoke d-separation to argue that any two sets A and C are conditionally independent, there is exactly one undirected path between each node in A and each node in C. For each such pair of nodes, we just have to find one node on that path for which one of the three cases of d-separation is true.

Now we are ready to start working out $p(X|E)$. This is

$$\begin{aligned} p(X|E) &= p(X|E^-, E^+) \\ &= \frac{p(E^-|X, E^+)p(X|E^+)}{p(E^-|E^+)} \text{ by Bayes' rule} \\ &= \frac{p(E^-|X)p(X|E^+)}{p(E^-|E^+)} \end{aligned}$$

The last equality follows from d-separation for the sets E^- and E^+. For each node in E^- and each node in E^+, the node X d-separates this pair, by case 1 of d-separation.

The denominator above equals the sum of the numerators because the sum $\sum_x p(X = x|E) = 1$, since the sum is over all possible outcomes x of X. There-

fore, we will evaluate explicitly the numerators only. We will compute the denominator as

$$p(E^-|E^+) = \sum_x p(E^-|X=x)p(X=x|E^+).$$

Let us evaluate the second factor in the numerator first:

$$\begin{aligned} p(X|E^+) &= \sum_u p(X, U=u|E^+) \text{ by marginalization over all parents of } X \\ &= \sum_u p(X|U=u, E^+)p(U=u|E^+) \text{ by the chain rule.} \end{aligned}$$

We have $p(X|U=u, E^+) = p(X|U=u)$ by d-separation case 1 or 2 for X and each evidence node in E^+. Also,

$$p(U=u|E^+) = \prod_i p(U_i=u_i|E^+)$$

by d-separation case 3, and $p(U_i=u_i|E^+) = p(U_i=u_i|E_i^+)$ by d-separation case 3 again, where E_i^+ is the set of evidence nodes connected to parent U_i but not by a path through X. In other words, E_i^+ is subset of evidence nodes inside region i of the polytree. We get

$$p(X|E^+) = \sum_u p(X|U=u) \prod_i p(U_i=u_i|E_i^+).$$

Each factor $p(X|U=u)$ can be evaluated directly by lookup in the CPT for node X. Each factor $p(U_i=u_i|E_i^+)$ can be evaluated by calling the algorithm recursively. The recursive call is on a smaller problem instance because U_i is closer than X to a root of the polytree.

Now we need to evaluate $p(E^-|X)$. By case 2 of d-separation this is

$$p(E^-|X) = \prod_j p(E_j^-|X)$$

where the set E_j^- consists of the evidence nodes in region j. Specifically, any node in one downstream region is conditionally independent of any node in another downstream region given X, because X is a known common cause.

End of the lecture on Thursday October 18.

Each expression $p(E_j^-|X)$ can be decomposed by an argument similar to the one above for $p(X|E^+)$ The decomposition is based on marginalization over all possible combinations of outcomes for the child Y_j and the spouses Z_{jk}. Again in a similar way to above, the decomposition leads to recursive calls of the algorithm. Details will be added to these lecture notes at a later time.

7 Inference in loopy networks

The efficient, exact algorithm on the previous section for evaluating $p(X|E)$ is well-defined only for polytrees. A loopy network is a Bayesian network that is not a polytree. A simple example is a network with two disease nodes and two symptom nodes, where edges go from each disease node to each symptom node. There are various ways to reduce inference in a loopy network to inference in a polytree, but unfortunately none are both efficient and exact.

7.1 Merging nodes

One approach is to group nodes together into a larger node. Conside the example of a disease node, symptom nodes, and a "visit doctor" node. Merge all the symptom nodes into one giant random variable whose set of outcomes is the cross-product of the outcome sets of the individual symptom nodes. After the merger, the network is no longer loopy.

One example is not enough to specify an algorithm. In general, if a network is loopy, then there exist two nodes X and Y with two different undirected paths between them. At least one of these paths must contain a node that is not part of the other path. Then, the network can be simplified by merging a node that is unique to one path with a node from the other path.

Example 1: $A \to B \to C$ and $A \to C$. Merge B and C or merge A and B.

Example 2: $A \to B_1 \to B_2 \to D$ and $A \to C_1 \to C_2 \to D$. Merge B_2 and C_2, then merge B_1 and C_1.

When merging two nodes, the set of values of the new node is the cross-product of the sets of values of the original nodes. The new network is equivalent to the original network in the sense that it represents the same knowledge about the unmerged nodes. However, the new network cannot be used directly to answer questions about nodes that were merged. Hence, different mergers are needed to answer different questions.

A new merged node depends on all nodes that either original node depends on. The CPT of the new node grows in size accordingly. Overall, converting a loopy network into a polytree may increase its size exponentially. It is NP-hard to find the sequence of node mergers that makes the final polytree the smallest possible.

7.2 Cutset conditioning

The idea here is to assign values to some nodes such that the remaining nodes constitute a polytree. We then apply the polytree algorithm to these. The final answer is obtained as a weighted average, where each weight is the probability of one combination of assigned values. For example, in the disease/symptom/visit network. we can instantiate $D = 0$ and $D = 1$.

Again, one example is not enough to specify an algorithm. In general, a cutset is a set of nodes such that the network becomes a polytree when these nodes and their incoming and outgoing edges are removed. Cutset conditioning is the process of finding such a set S and then using the identity

$$p(X|E) = \sum_S p(X, S|E) = \sum_S p(X|S, E)p(S|E).$$

Separately for each combination of values of the nodes in S, we compute $p(X|S, E)$ and $p(S|E)$. Each of these computations is done on the original network where each node in S has a separate replica for each edge that it is part of.

Example: Consider the disease/symptom/visit scenario again. Suppose there are four nodes and four edges $D \to S, D \to T, S \to V, T \to V$. Intuitively, D is a disease, S and T are symptoms, and V is visiting the doctor. We can condition on S and compute

$$p(D|V) = \sum_S p(D, S|V) = \sum_S p(D|S, V)p(S|V).$$

End of the lecture on Tuesday October 23. See also the sample solution to the day's quiz.

8 Randomized inference methods

A Bayesian network can be viewed as a generative model, meaning that it specifies how to generate, step by step, a random sample from the joint distribution that it

describes. Let the universe U be the set of all nodes $\{X_1, \ldots, X_n\}$. It is easy to draw samples from $p(U)$. But how can we use samples to evaluate $p(Q|E)$ where Q and E are subsets of U?

We cannot directly draw samples q from the distribution $p(Q|E)$. The simplest approach is called rejection sampling. We generate n samples from $p(U)$. Let $n(e)$ be the number (also called the count) of these samples for which $E = e$ and let $n(q, e)$ be the number of samples for which $E = e$ and also $Q = q$. Note that both these counts are obtained from the same n samples. Estimate $p(Q = q|E = e)$ as $n(q, e)/n(e)$. By the law of large numbers, this ratio will converge to the desired probability.

The weakness of rejection sampling is that in most applications it is highly inefficient. Samples for which $E \neq e$ are generated but discarded. If the event $E = e$ is rare, and/or $Q = q$ is rare, rejection sampling is not useful in practice. A more sophisticated approach is called likelihood weighting. The idea is that we force $E = e$ to be true, instead of waiting for it to happen. As an example of likelihood weighting, consider the loopy network $X \to Y, X \to E, Y \to Q, E \to Q$. This network has the same structure as the disease/symptom/visit network seen above. Draw samples $\{x_i, y_i\}$ for $i = 1$ to $i = n$, but clamp the node E to have the outcome $e_i = e$ always. For each i, sample q_i from $p(Q|Y = y_i, E = e)$.

The difficulty now is that the value of E has not been drawn according to the correct probability $p(E|X)$. To compensate, we need to change the weights of samples. If $p(E = e|X = x_i)$ is high, then the sample should have a high weight. Specifically, the estimate of $p(Q = q|E = e)$ is

$$\frac{\sum_{i=1}^{n} I(q_i = q)p(E = e|X = x_i)}{\sum_{i=1}^{n} p(E = e|X = x_i)}.$$

Note that computing the weight $p(E = e|X = x_i)$ is fast because this is just a lookup in the CPT for E.

Likelihood weighting is more efficient than rejection sampling because it does not matter if the event $E = e$ is rare. However, it is still slow for estimating small probabilities, that is if the event $Q = q$ is rare. Also, suppose that $p(E = e|X = x)$ is small for frequent values x of X, but large for some rare value of X. Then the estimate will have high randomness even though the total number of samples n is large.

Markov chain Monte Carlo, abbreviated MCMC, is even faster than likelihood weighting. It is based on a definition and a theorem.

Definition. The Markov blanket B_X of X consists of the parents, children, and spouses of X.

Theorem. X is conditionally independent of all nodes outside B_X, given B_X.

The MCMC algorithm to estimate $p(Q = q|E = e)$ is as follows. Fix the evidence nodes E to their observed values e. Initialize all other nodes to random values. Then, repeat n times: Pick a non-evidence node X at random. Use Bayes' rule to compute $p(X|B_X)$. Sample a new value for X from this distribution.

The estimate of $p(Q = q|E = e)$ is $n(Q = q)/n$. It can be proved that this ratio converges to the true value in the limit. The fundamental difference between likelihood weighting and MCMC is that the former samples non-evidence nodes from $p(X|\text{pa}(X))$ and the latter from $p(X|B_X)$.

9 Markov models

Material in this section will be covered later in the quarter.

Bayesian networks are useful for modeling systems whose state changes over time. Here, the rvs are indexed by discrete time. That is, time t is an integer subscript on the names of the rvs. We can use X_t to denote the observed data at time t, and Y_t to denote the state of the system at time t.

In a BN for modeling sequences X_t, the configuration of the edges (i.e., arrows) represents what simplifying assumptions we make. The most common assumption is called the Markov assumption, named after Andrey Markov (1856 to 1922). It is that each state Y_t depends only on the previous state Y_{t-1}. In other words, the earlier history Y_1 to Y_{t-2} is irrelevant.

In a hidden Markov model (HMM), the states Y_t are never observed. They must be inferred from the observations X_t. HMMs are the standard approach to speech recognition. Intuitively, an observation X_t is a short window, maybe 20 milliseconds, of audio. A state Y_t is a phoneme such as "ba."

Models that include the Markov assumption are also the foundation of learning to behave optimally over time, a research area that is called reinforcement learning. Here the word "reinforcement" means reward or benefit. The fundamental issue is the trade-off between short-term and long-term reward.

A Markov decision process (MDP) is a description of an infinite sequence of rvs where some rvs are the state of the environment, some rvs are actions selected by an agent, and some rvs are short-term rewards received by the agent. The goal of the agent is to maximize the sum of the short-term rewards. (Draw picture of infinite sequence of rvs here.)

Note that the MDP is an explicit model only of the environment. From the point of view of the MDP, actions are rvs that are deterministic and exogenous.

The MDP does not model how the agent selects actions. Inside the agent, the action depends on the current state and anticipated rewards. However, the MDP does not represent this dependency.

Formally, an MDP is a tuple (S, A, P, R) where S is a state space, A is an action space, P is a transition model with $P(s, a, s')$ being the probability of making a transition from s to s' on taking action a, and R is a reward model. A transition model P is a function such that $P(s, a, s')$ is the probability of going from state s to state s' when taking action a. A reward model R is a function such that $R(s, a, s')$ is the immediate benefit associated with going from s to s' and taking action a.

The state and action spaces S and A may be discrete or real-valued, and may be high-dimensional. The transition model P and the reward model R are assumed to be unknown, so the agent must learn them from experience. The state, action, reward and next state are observed during the experience from which P and R are learned.

In general, the MDP has an infinite horizon, so for finiteness, future rewards must be discounted exponentially with a discount factor $\gamma < 1$. For problems with a goal or sink state, the discount factor can equal 1.

A stationary policy is a mapping $\pi : S \to \Omega(A)$ where $\Omega(A)$ is the space of probability distributions over the action space; $\pi(a; s)$ is the probability of choosing action a in state s. A deterministic stationary policy is a mapping $\pi : S \to A$, with $\pi(s)$ being the specific action that the agent always takes in state s.