Massachusetts Institute of Technology
Department of Electrical Engineering and Computer Science

6.S064 Introduction to Machine Learning

Phase 4: Bayesian networks (lecture 19)

Bayesian networks: graph and independence

We have already emphasized that the graph structure in Bayesian networks represents useful qualitative properties about the variables. Specifically, the graph encodes independence statements about how the variables relate to each other. We will need a criterion for reading such independence properties from the graph without consulting the underlying probability distribution. The subtlety here is that we cannot just pick any criterion we like. The probability distribution we will associate with the graph must be consistent with all the properties we can derive from the graph. Otherwise the graph would "lie" and wouldn't be useful to us.

We had previously introduced the graph as specifying the parents of each node $i = 1, \ldots, d$. In the graph, we call nodes j which start directed edges or arcs to node i as parents of i. Formally, we say that $j \in pa_i$. The choice of these parents, i.e., the choice of parent sets pa_i, is unconstrained except for the fact that we cannot introduce directed cycles. In other words, the graph must be an acyclic directed graph (DAG). In terms of variables $X_1, \ldots, X_d$, we motivated the graph as specifying which other variables each X_i directly depends on, i.e., variables whose values we would have to know prior to drawing a value for X_i. We write the set of parents as variables using notation $X_{pa_i} = \{X_j\}_{j \in pa_i}$. Once we know the parents, we can write (factor) the probability distribution over all the variables as

$$P(X_1 = x_1, \ldots, X_d = x_d) = \prod_{i=1}^{d} P(X_i = x_i | X_{pa_i} = x_{pa_i}) \tag{1}$$

where, for some nodes, $pa_i = \{\}$ (the empty set) and $P(X_i = x_i | X_{pa_i} = x_{pa_i})$ reduces to $P(X_i = x_i)$ (no other variable need to be consulted prior to sampling a value for X_i). You should convince yourself that any directed acyclic graph must have at least one node without any parents.

The above factorization resembles an application of the chain rule. First, we write the variables in some order such that the parents of each variable always come before the variable itself in the ordering. This is always possible for a directed acyclic graph (in fact, there are often a large number of such "consistent" orderings). Then we simply "drop" dependences in the conditional probabilities to get the above factorization. These simplifications represent independence assumptions about the variables. To simplify the notation, let's assume that the simple lexicographic ordering works for our graph. In

other words, assume that $pa_i \subseteq \{1, \ldots, i-1\}$, $i = 1, \ldots, d$. Then, by applying the chain rule, we could always write (without any assumptions)

$$P(X_1 = x_1, \ldots, X_d = x_d) = \prod_{i=1}^{d} P(X_i = x_i | X_{i-1} = x_{i-1}, \ldots, X_1 = x_1) \tag{2}$$

When we construct the distribution for a particular graph, we are assuming that

$$P(X_i = x_i | X_{i-1} = x_{i-1}, \ldots, X_1 = x_1) = P(X_i = x_i | X_{pa_i} = x_{pa_i}) \tag{3}$$

where $pa_i \subseteq \{1, \ldots, i-1\}$. This is an independence assumption. Specifically, define $npa_i = \{1, \ldots, i-1\} \setminus pa_i$ as the "preceding non-parents". By factoring the joint distribution according to Eq.(1), i.e., dropping dependences except for the parents, we make the assumption that X_i is independent of X_{npa_i} given values for the parents X_{pa_i}. These independence statements, one statement per variable, would suffice to specify the directed graph (dropping all arcs from preceding non-parents). But there are many other independence statements that are implied by these. How do we read all of these off the graph directly?

Independence from the graph: D-separation

As an example, consider a slightly extended version of the alarm model we had discussed before. The model is given in Figure 1a, with an additional binary variable L. This could be whether we "leave work" as a result of hearing/learning about the alarm. We will now define a procedure for answering questions such as: are R and B independent of each other? Are R and B independent of each other if we know (given) A? And so on.

The general procedure involves three simple steps that we will motive and illustrate in relation to the graph in Figure 1a.

1. First we only keep the *ancestral graph* of the variables of interest. The ancestral graph includes all the variables of interest as well as every other variable we can get to by traversing in the reverse direction of the arrows (parents, parents' parents, and so on). Why do we want this graph? Common ancestors are variables that might correlate our variables of interest (loosely speaking, you can think of them as common underlying causes). We need to know whether such variables exist. In contrast, it is unnecessary to look at variables downstream (these represent properties we could but did not measure).

   The variables we care about here are R, B, and A. The ancestral graph includes these variables as well as all the variables (ancestors) you can get to by starting from one of these variables and following the arrows in the reverse direction (their parents, their parents' parents, and so on). The ancestral graph in our case is given in Figure 1b.

2. *Moralize* the resulting ancestral graph. This operation simply adds an *undirected edge* between any two variables in the ancestral graph that have a common child ("moralization" = "marry the parents"). In case of multiple parents, they are connected pairwise, i.e., by adding an edge between any two parents. See Figure 1c. Moralization is needed to take into account induced dependences discussed earlier (observed effects of multiple causes). You wouldn't actually need to moralize the whole ancestral graph. It would suffice to do so for the parents of the conditioning variables (in this case A).

3. Once we have completed the above two steps, we are nearly done. We can now read off the independence property from the graph. If all the paths between R and B go via A then R and B are independent given A (this is NOT true in our graph). So R and B are *dependent* given A (recall induced dependence). Note that, after moralization, we no longer pay attention to the direction of the arcs. For clarity, you could change all the direct edges into undirected edges at this point. This would give the resulting *undirected graph* in Figure 1d from which you can read the answer.

Let's go back to the previous examples to make sure we can read off the properties we claimed from the graphs. For example, if we are interested in asking whether X_1 and X_2 are marginally independent (i.e., given nothing) in the model in Figure **??**, we would create the graph transformations shown in Figure 2. Similarly, to establish that X_1 and X_2 become dependent with the observation of X_3, we would ask whether X_1 and X_2 are independent given X_3 and get the transformations in Figure 3. The nodes are not separated by X_3 and therefore not independent.

Learning Bayesian networks

There are two problems we have to solve in order to estimate Bayesian networks from available data. We have to estimate the parameters given a specific graph structure, and we have to search over possible structures (model selection).

Suppose now that we have d discrete variables, $X_1, \ldots, X_n$, where $X_i \in \{1, \ldots, r_i\}$, and n *complete* observations $D = \{(x_1^{(t)}, \ldots, x_d^{(t)}), t = 1, \ldots, n\}$. In other words, each observation contains a value assignment to all the variables in our model. This is a simplification and models in practice (e.g., mixture models, HMMs) are intended to be estimated from incomplete data. We will also assume that the conditional probabilities we need to specify for a Bayesian network are *fully parameterized*. This means, e.g., that in $P(X_1 = x_1 | X_2 = x_2)$ we can select the probability distribution over X_1 separately (without additional constraints) for each possible value of the parent x_2. Models used in practice often have parametric constraints so that, for example, "similar" values of X_2 would lead to "similar" distributions over X_1. In our case here, the model interprets the value of each variable just as a symbol without any such constraints.

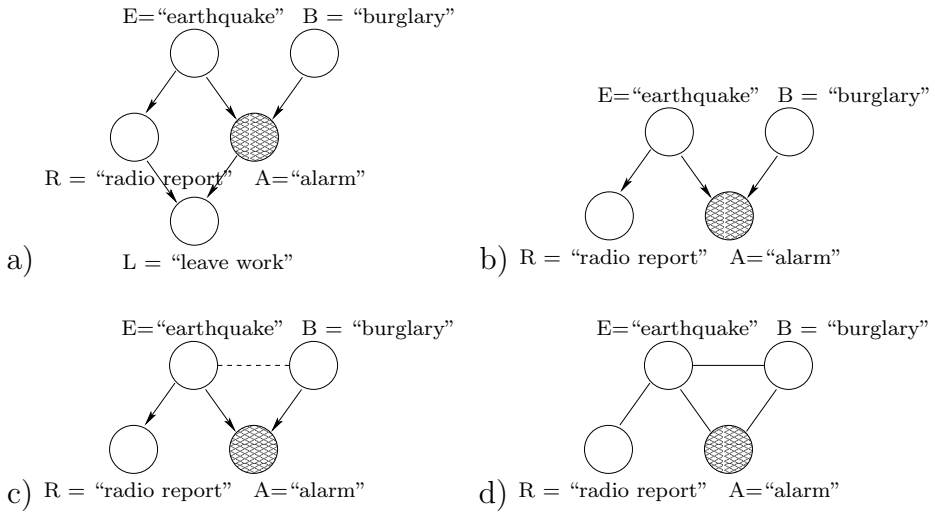

Figure 1: a) Burglary model, extended, b) ancestral graph of R, B, and A, c) moralized ancestral graph, d) resulting undirected graph.

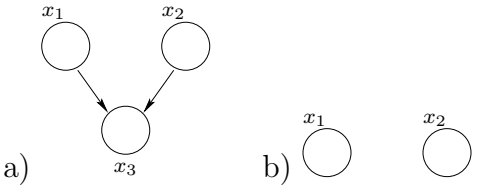

Figure 2: a) Bayesian network model, b) ancestral graph of x_1 and x_2, already moralized and undirected.

Given an acyclic graph G over d variables, we already know that we can write down the associated joint distribution as

$$P(X_1 = x_1, \ldots, X_d = x_d) = \prod_{i=1}^{n} P(X_i = x_i | X_{pa_i} = x_{pa_i}) = \prod_{i=1}^{n} \theta_i(x_i | x_{pa_i}) \tag{4}$$

where $\theta_i(x_i|x_{pa_i})$ are the probability tables that we must estimate. For example, if $X_3 \in \{1, \ldots, r_3\}$ has two parents, X_1 and X_2, each taking values in $\{1, \ldots, r_1\}$ and

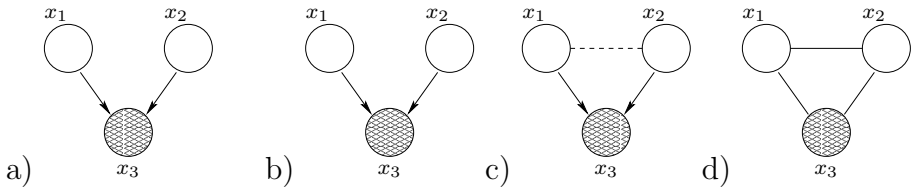

Figure 3: a) Bayesian network model, b) ancestral graph of x_1 and x_2 given x_3, c) moralized ancestral graph, d) resulting undirected graph.

$\{1, \ldots, r_2\}$, respectively, then the table $\theta_3(x_3|x_1, x_2)$ is given by

$$X_3|X_1, X_2 : \begin{array}{c|ccc} X_1, X_2 & 1, & \ldots, & r_3 \\ \hline 1,1 & \theta_3(1|1,1), & \ldots & \theta_3(r_3|1,1) \\ 2,1 & \theta_3(1|2,1), & \ldots & \theta_3(r_3|2,1) \\ \cdots & \cdots, & \ldots & \cdots \\ r_1,1 & \theta_3(1|r_1,1), & \ldots & \theta_3(r_3|r_1,1) \\ r_1,2 & \theta_3(1|r_1,2), & \ldots & \theta_3(r_3|r_1,2) \\ \cdots & \cdots, & \ldots & \cdots \\ r_1,r_2 & \theta_3(1|r_1,r_2), & \ldots & \theta_3(r_3|r_1,r_2) \end{array} \tag{5}$$

Each row of the table sums to one since $\sum_{x_3=1}^{r_3} \theta_3(x_3|x_1, x_2) = 1$ for any setting of x_1 and x_2. There are exactly $r_1 r_2 (r_3 - 1)$ parameters in this table that can be chosen independently (the sum to one constraint removes one from each row).

Maximum likelihood parameter estimation

We can now write down the log-likelihood of observed data $D = \{(x_1^{(t)}, \ldots, x_d^{(t)}), t = 1, \ldots, n\}$ for any particular Bayesian network structure, i.e., for any particular graph G. It is given by

$$l(D; \theta, G) = \sum_{t=1}^{n} \log \left[\prod_{i=1}^{d} \theta_i(x_i^{(t)}|x_{pa_i}^{(t)}) \right] = \sum_{t=1}^{n} \sum_{i=1}^{d} \log \theta_i(x_i^{(t)}|x_{pa_i}^{(t)}) = \sum_{i=1}^{d} \left[\sum_{t=1}^{n} \log \theta_i(x_i^{(t)}|x_{pa_i}^{(t)}) \right] \tag{6}$$

where we grouped the terms by variable (given their parents) in order to highlight the fact that the associated parameters can be set separately from those pertaining to other variables. The maximum likelihood estimation of the model parameters then reduces to the problem of estimating individual tables such as the one in Eq.(5).

The table $\theta_i(x_i|x_{pa_i})$ specifies a multinomial distribution over x_i for each setting of the parent variables x_{pa_i}. As a result, we can maximize the likelihood analogously to estimating multinomial parameters we have seen before. Indeed,

$$\sum_{t=1}^{n} \log \theta_i(x_i^{(t)}|x_{pa_i}^{(t)}) = \sum_{x_i, x_{pa_i}} n_{i,pa_i}(x_i, x_{pa_i}) \log \theta_i(x_i|x_{pa_i}) \tag{7}$$

where $n_{i,pa_i}(x_i, x_{pa_i})$ gives the number of observations in data D for which $X_i = x_i$ and $X_{pa_i} = x_{pa_i}$. So, if we fix x_{pa_i}, then $n_{i,pa_i}(\cdot, x_{pa_i})$ specifies the counts for a multinomial $\theta_i(\cdot|x_{pa_i})$. The corresponding maximum likelihood parameter estimate is simply (analogously to a single multinomial)

$$\hat{\theta}_i(x_i|x_{pa_i}) = \frac{n_{i,pa_i}(x_i, x_{pa_i})}{\sum_{x_i'} n_{i,pa_i}(x_i', x_{pa_i})}, \quad x_i \in \{1, \ldots, r_1\} \tag{8}$$

Repeating the procedure for each setting of x_{pa_i}, and for different variables, yields the maximum likelihood parameter estimates $\hat{\theta}_i(x_i|x_{pa_i})$, $i = 1, \ldots, d$.

Learning the graph structure

Given the ML parameter estimates $\hat{\theta}_i(x_i|x_{pa_i})$ shown above, we can evaluate the resulting maximum value of the log-likelihood $l(D; \hat{\theta}, G)$. Note that this value depends on the graph G (which specifies the conditional probability tables we can use) as well as on the data. As with other models we have seen (e.g., mixture models), we cannot use this log-likelihood value alone for deciding which model (graph) is the best one. We must use a model selection criterion to decide between the graphs.

Consider a simple case of just two variables X_1 and X_2. We can evaluate the log-likelihood of the data D for three different graphs, 1) G_0 where X_1 and X_2 are independent, 2) G_1 where X_1 is a parent of X_2, and 3) G_2 where X_2 is a parent of X_1. The graphs G_1 and G_2 are equivalent in the sense that they make the same set of independence assumptions about the variables, i.e., none. They would therefore always result in the same value of log-likelihood, and we cannot distinguish between them. However, the problem is that G_0 would almost always result in a lower log-likelihood value than G_1 or G_2, regardless of whether X_1 and X_2 were independent. Why is that? Let's say we are choosing between G_0 and G_1. For each observation in D, we predict values for X_1 in the same way, using a table $\theta_1(x_1)$ since X_1 has no parents in either graph. But they differ in how observed values of X_2 are predicted. Specifically,

$$G_0: \quad l(D; \theta, G_0) = \sum_{t=1}^{n} \log \theta_1(x_1^{(t)}) + \sum_{t=1}^{n} \log \theta_2(x_2^{(t)}) \tag{9}$$

$$G_1: \quad l(D; \theta, G_1) = \sum_{t=1}^{n} \log \theta_1(x_1^{(t)}) + \sum_{t=1}^{n} \log \theta_2(x_2^{(t)}|x_1^{(t)}) \tag{10}$$

For G_1, we could always choose $\theta_2(x_2|x_1)$ such that it takes the same value regardless of x_1. This would correspond to having only $\theta_2(x_2)$. In other words, G_0 is "contained" in G_1. But this is hardly optimal for G_1 in terms of the log-likelihood. As a result, $l(D;\hat{\theta}, G_1) \geq l(D;\hat{\theta}, G_0)$ since G_1 has more parameters (more degrees of freedom to fit to the data) and we would never select G_0, whether it is correct or not.

To remedy the situation, and appropriately compare two different Bayesian networks, we must use a model selection criterion such as the Bayesian Information Criterion

$$BIC(D;\hat{\theta}, G) = l(D;\hat{\theta}, G) - \frac{dim(G)}{2}\log(n) \tag{11}$$

where $dim(G)$ specifies the number of (independent) parameters in the model. In our case this is given by

$$dim(G) = \sum_{i=1}^{d}(r_i - 1)\prod_{j\in pa_i} r_j \tag{12}$$

where each term in the sum corresponds to the size of the probability table $\theta_i(x_i|x_{pa_i})$ minus the number of associated normalization constraints. We can now search for the graph G that maximizes $BIC(D;\hat{\theta}, G)$ similarly to selecting the number of mixture components.

We can write the criterion in a bit more convenient form. Note that both the log-likelihood value and the BIC penalty term decompose according to the variables. In other words, we can define

$$\text{score}(i|pa_i; D) = \sum_{t=1}^{n}\log\hat{\theta}_i(x_i^{(t)}|x_{pa_i}^{(t)}) - \frac{1}{2}\left[(r_i - 1)\prod_{j\in pa_i} r_j\right]\log(n) \tag{13}$$

as the BIC score for selecting parents pa_i for node i so that

$$BIC(D;\hat{\theta}, G) = \sum_{i=1}^{d}\text{score}(i|pa_i; D) \tag{14}$$

While we can pre-compute the scores $\text{score}(i|pa_i; D)$ for each node and each possible choice of parents, the main difficulty is that the graph has to be acyclic. Indeed, maximizing the overall BIC score with respect to the graph (the choice of parents for each node) is provably hard for this reason, even if we limit the number of parents that each variable can take to be just two. A number of algorithms are available, however, from local search (changing individual edges to maximize the BIC score) to exact dynamic programming algorithms (which scale exponentially in the number of variables but remain feasible up to 25-30 variables).

Figure 4 characterizes the structure learning steps.

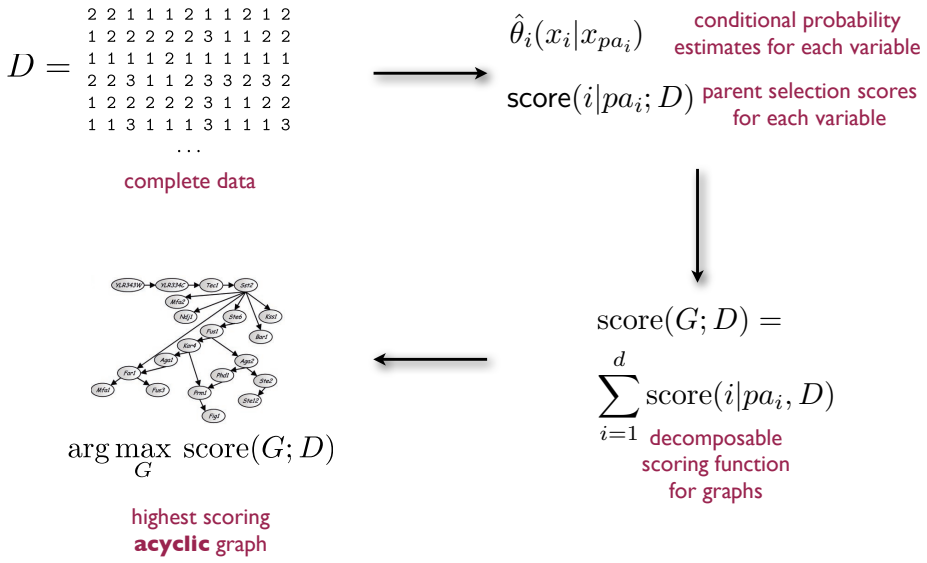

Figure 4: A summary of Bayesian network structure learning steps