SINGAPORE UNIVERSITY OF TECHNOLOGY AND DESIGN

50.007 Machine Learning, Fall 2015
Lecture Notes for Week 12

Bayesian Networks (III)

Last update: Tuesday 1st December, 2015 20:42

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 (supervised learning), and we have to search over possible structures (model selection).

Learning Model Parameters

Suppose now that we have d discrete variables, $X_1, \ldots, X_d$ where $X_i \in \{1, \ldots, r_i\}$ and m *complete* observations $D = \{(x_1^{(t)}, \ldots, x_d^{(t)}), t = 1, \ldots, m\}$. 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.

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

$$P(X_1 = x_1, \ldots, X_d = x_d) = \prod_{i=1}^{d} P(X_i = x_i | \mathbf{X}_{pa_i} = \mathbf{x}_{pa_i}) = \prod_{i=1}^{d} \theta_i(x_i | \mathbf{x}_{pa_i}) \tag{1}$$

where $\theta_i(x_i|\mathbf{x}_{pa_i})$ 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 $\{1, \ldots, r_2\}$, respectively, then the table $\theta_3(x_3|x_1, x_2)$ is given by

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

Discussions How many parameters are in the table? What is the number of *free* (or *independent*) parameters here?

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

$$l(D;\theta;G) = \sum_{t=1}^{m} \log \left[\prod_{i=1}^{d} \theta_i(x_i^{(t)}|\mathbf{x}_{pa_i}^{(t)}) \right] = \sum_{t=1}^{m}\sum_{i=1}^{d} \log \theta_i(x_i^{(t)}|\mathbf{x}_{pa_i}^{(t)}) = \sum_{i=1}^{d} \left[\sum_{t=1}^{m} \log \theta_i(x_i^{(t)}|\mathbf{x}_{pa_i}^{(t)}) \right] \tag{3}$$

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 the table above (*i.e.*, (2)).

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

$$\sum_{t=1}^{m} \log \theta_i(x_i^{(t)}|\mathbf{x}_{pa_i}^{(t)}) = \sum_{x_i,\mathbf{x}_{pa_i}} \text{Count}\Big((x_i, \mathbf{x}_{pa_i}) \text{ in } D\Big) \log \theta_i(x_i|\mathbf{x}_{pa_i}) \tag{4}$$

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

$$\hat{\theta}_i(x_i|\mathbf{x}_{pa_i}) = \frac{\text{Count}\Big((x_i, \mathbf{x}_{pa_i}) \text{ in } D\Big)}{\text{Count}\Big((\mathbf{x}_{pa_i}) \text{ in } D\Big)}, x_i, \in \{1, \ldots, r_1\} \tag{5}$$

where $\text{Count}\Big((\mathbf{x}_{pa_i}) \text{ in } D\Big)$ is the number of times we see the pattern $\mathbf{x}_{pa_i}$ in D:

$$\text{Count}\Big((\mathbf{x}_{pa_i}) \text{ in } D\Big) = \sum_{x_i'} \text{Count}\Big((x_i', \mathbf{x}_{pa_i}) \text{ in } D\Big)$$

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

(Optional) Learning with Incomplete Data

Now let us consider the case where we have incomplete observations. Suppose in additional to d observed discrete variables $\mathbf{X} = \{X_1, \ldots, X_d\}$, we have h unobserved discrete variables $\mathbf{Y} = \{Y_1, \ldots, Y_h\}$. Such variables are also called *hidden variables* or *latent variables*.

In the ideal case, each sample should be of the form $D = \{(x_1^{(t)}, \ldots, x_d^{(t)}, y_1^{(t)}, \ldots, y_h^{(t)}), t = 1, \ldots, m\}$. However, the hidden variables are not observed. As a result, we only have samples of the following form as our observations: $\{(x_1^{(t)}, \ldots, x_d^{(t)}), t = 1, \ldots, m\}$.

How do we estimate the model parameters in this case? We cannot optimize the joint probability $P(\mathbf{X}, \mathbf{Y})$ but we can instead optimize $P(\mathbf{X})$. This objective function was exactly what we used when using the EM algorithm for learning model parameters!

Thus, the steps are as follows. We first randomly initialize the model parameters. Next, given the model parameters, we can use Gibbs sampling algorithm learned in the last lecture to generate the samples for $\mathbf{Y}$ variables. Together with $\mathbf{X}$ values, we now have complete data samples including both $\mathbf{X}$ and $\mathbf{Y}$ values. This step corresponds to the E-step in the EM algorithm, where we are interested in finding the distribution of $\mathbf{Y}$ given $\mathbf{X}$ (or: the soft assignments of $\mathbf{Y}$ to $\mathbf{X}$).

Next, given a large collection of samples, we can re-estimate the model parameters using the maximum likelihood estimator described above. This corresponds to the M-step in EM.

After the model parameters are updated, we can then move to the next iteration, and return to the E-step. We again use the Gibbs sampling algorithm to generate new $(\mathbf{X}, \mathbf{Y})$ samples using the new model parameters. These samples will then be used for learning new parameters in the M-step next. We can repeat these two steps until convergence, and return the final model parameters.

Learning the Graph Structure

Given the ML parameter estimates $\hat{\theta}_i(x_i|\mathbf{x}_{pa_i})$ shown above, we can evaluate the resulting maximum value of the log-likelihood $l(D; \theta, G)$. 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,

$$G_0: \quad l(D;\theta,G_0) = \sum_{t=1}^{m} \log \theta_1(x_1^{(t)}) + \sum_{t=1}^{m} \log \theta_2(x_2^{(t)}) \tag{6}$$

$$G_1: \quad l(D;\theta,G_1) = \sum_{t=1}^{m} \log \theta_1(x_1^{(t)}) + \sum_{t=1}^{m} \log \theta_2(x_2^{(t)}|x_1^{(t)}) \tag{7}$$

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*

$$BIC(D;\hat{\theta},G) = l(D;\hat{\theta},G) - \frac{dim(G)}{2}\log(m) \tag{8}$$

where $dim(G)$ specifies the number of (independent) parameters in the model, and m is the number of data points in the training set. In our case this is given by

$$dim(G) = \sum_{i=1}^{d}(r_i - 1)\prod_{j\in pa_i} r_j \tag{9}$$

where each term in the sum corresponds to the size of the probability table $\theta_i(x_i|\mathbf{x}_{pa_i})$ 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

$$score(i|pa_i;D) = \sum_{t=1}^{m} \log \hat{\theta}_i(x_i^{(t)}|\mathbf{x}_{pa_i}^{(t)}) - \frac{1}{2}\left[(r_i - 1)\prod_{j\in pa_i} r_j\right]\log(m) \tag{10}$$

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

$$BIC(D;\hat{\theta},G) = \sum_{i=1}^{d} score(i|pa_i;D) \tag{11}$$

While we can pre-compute the scores $score(i|pa_i;D)$ 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).

Discussions How to design a dynamic programming algorithm for structure learning in Bayesian networks?

Figure 1 characterizes the structure learning steps.

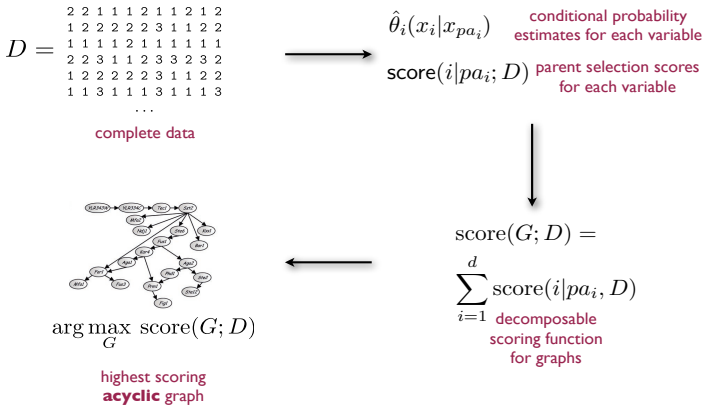

Figure 1: A summary of Bayesian network structure learning steps

Learning Objectives

You need to know:

1. How to count the effective number of parameters for a given Bayesian network
2. How to learn the parameters of Bayesian networks of a certain structure from complete data based on a collection of samples
3. How to learn the structures of Bayesian networks from data using BIC