12.1 Introduction

A Bayesian network is a graph-based model of joint multivariate probability distributions that captures properties of conditional independence between variables. Such models are attractive for their ability to describe complex stochastic processes, and since they provide clear methodologies for learning from (noisy) observations.

Bayesian networks are a useful tool. First, they are particularly useful for describing processes composed of locally interacting components; that is, the value of each component directly depends on the values of a relatively small number of components. Second, statistical foundations for learning Bayesian networks from observations, and computational algorithms to do so are well understood and have been used successfully in many applications. Finally, Bayesian networks provide models of causal influence: although Bayesian networks are mathematically defined strictly in terms of probabilities and conditional independence statements, a connection can be made between this characterization and the notion of direct causal influence. (Heckerman, Meek & Cooper 1997; Pearl & Verma 1991; Spirtes, Glymour & Scheines 1993).

All of these features make this model very suitable for applications such as Gene Array analysis, which we shall present later. First, let us present the general concepts of Bayesian networks.

12.1.1 Representing Distribution with Bayesian Networks

Consider a finite set $X=\{X_1, \ldots, X_n\}$ of random variables where each variable $X_i$ may take on a value $x_i$ from a certain domain. In this paper, we use capital letters, such as $X$, $Y$, $Z$, for variable names and lowercase letters $x$, $y$, $z$, to denote specific values taken by those variables. Sets of variables are denoted by boldface capital letters $X$, $Y$, $Z$, and assignments of values to the variables in these sets are denoted by boldface lowercase letters $x$, $y$, $z$. We denote $I(X;Y|Z)$ to mean $X$ is independent of $Y$ conditioned on $Z$.

A Bayesian Network is a representation of a joint probability distribution. This representation consists of two components. The first component, $G$, is a directed acyclic graph (DAG) whose vertices correspond to the random variables $X_1, \ldots, X_n$. The second component, $\Theta$, describes a conditional distribution for each variable, given its parents in $G$. Together, these two components specify a unique distribution on $X$.

The graph $G$ represents conditional independence assumptions that allow the joint distribution to be decomposed, economizing on the number of parameters. The graph $G$ encodes the Markov Assumption:

\[ (*) \quad \text{Each variable is independent of its non-descendants, given its parents in } G. \]

By applying the chain rule of probabilities and properties of conditional independencies, any joint distribution that satisfies $(*)$ can be decomposed into the product form:
Where $\text{Pa}^G(X_i)$ is the set of parents of $X_i$ in $G$.

$$P(X_1, \ldots, X_n) = \prod_{i=1}^{n} P(X_i | \text{Pa}^G(X_i)) \quad (12-1)$$

A graph $G$ specifies a product form as (12-1). To fully specify a joint distribution, we also need to specify each of the conditional probabilities in the product form. The second part of the Bayesian network describes these conditional distributions, $P(X_i | \text{Pa}^G(X_i))$ for each variable $X_i$. We denote the parameters that specify these distributions by $\theta$. The qualitative part (the graph itself) and the quantitative part (set of conditional probability distributions) together define a unique distribution in the factored form that we saw in (12-1).

Notice that the requirement for the graph to be directed and acyclic (DAG) prevents the network from being of a “Feedback” form, such as we have seen in the neural network case.
In specifying these conditional distributions we can choose from several representations. The choice of representation depends on the type of variables we are dealing with, whether they are discrete variables or continuous variables. In any case, the Markov assumption allows us to achieve a very compact representation of probability distributions via conditional independence.

As opposed to the CART model that we have studied, here, given a variable \( X_i \), we are interested only in his parents \( Pa^G(X_i) \).

### 12.1.2 Equivalence Classes of Bayesian Networks

A Bayesian network structure implies a set of independence assumptions in addition to (*). Let \( \text{Ind}(G) \) be the set of independence statements (of the form \( X \) is independent of \( Y \) given \( Z \)) that hold in all distributions satisfying these Markov assumptions. These can be derived as consequences of (*) (see (Pearl 1988)).

More than one graph can imply exactly the same set of independencies. For example, consider graphs over two variables \( X \) and \( Y \). The graphs \( X \to Y \) and \( Y \to X \) both imply the same set of independencies (i.e., \( \text{Ind}(G) = \emptyset \)). Two graphs \( G \) and \( G' \) are equivalent if \( \text{Ind}(G) = \text{Ind}(G') \). That is, both graphs are alternative ways of describing the same set of independencies. An example for 3 variables is given in Figure 2.

![Graph 1](A → C → B)

\[ P(x) = P(A)P(C|A)P(B|C) \]

![Graph 2](A → C ← B)

\[ P(x) = P(C)P(A|C)P(B|C) \]

In the same way

This notion of equivalence is crucial, since when we examine observations from a distribution, we cannot distinguish between equivalent graphs. We can characterize equivalence classes of graphs using a simple representation. In particular, these results establish that equivalent graphs have the same underlying undirected graph but might disagree on the direction of some of the edges. More specifically, two graphs are equivalent if they have the same underlying undirected graph and same v-structures (a v-structure is any two directed edges terminating at the same node) (Pearl and Verma, 1991).

Moreover, an equivalence class of network structures can be uniquely represented by a partially directed graph (PDAG), where a directed edge \( X \to Y \) denotes that all members of the equivalence class contain the edge \( X \to Y \). An undirected edge \( X \sim Y \) denotes that some members of the class contain the edge \( X \to Y \), while others contain the edge \( X \leftarrow Y \). Given a DAG \( G \), the PDAG representation of its equivalence class can be constructed efficiently (Chickering 1995).
12.1.3 Learning Bayesian Networks

When we have a Bayesian network, inference becomes a very simple action. The compact representation, using conditional independencies, allows simple estimations. When trying to estimate a variable value, all we need is its parent’s values.

Learning is the inverse action: we have several observations of the variables and we try to conclude the “real” underlying network. This “reverse engineering” of a network is a practical method of knowledge acquisition, as the data achieving becomes more and more available nowadays.

The problem of learning a Bayesian network can be stated as follows: Given a training set \( D = (x_1, \ldots, x_M) \) of independent instances of \( X \), find a network, \( B = \langle G, \theta \rangle \), that best matches \( D \) (or has the best prediction ability for future observations).

![Learning Bayesian Networks](Image)

The learning problem for Bayesian networks can be categorized as follows:

1. Parameter Estimation: learning the parameters (the probability distributions) for a given graph and a training data set.
2. Model Selection: learning the graph structure (the dependencies between the variables)

The learning problem can also be categorized by the completeness or incompleteness of the data set. In the complete data scenario, each observation contains a value for every variable in the network. In the incomplete data scenario, some values may be missing.
In this lecture we will only address the complete data scenario.

### 12.2 Parameter Estimation

In this section we assume that the graph $G$ (the model) is known. Each node in the graph corresponds to a random variable $X_i$ (total of $n$ variables). We denote the $n$-tuple of random variables by $X$. We are given a set of training samples (which we assume to be i.i.d.) $D = (x_1, \ldots, x_M)$, where each $x^m$ is a single sample of the $n$-tuple of random variables $X$ (we denote by $x^i_m$ the value of the $i$-th variable in the $m$-th sample). We denote by $P^m_d$, the tuple of values in the $m$-th sample for the parents of $X_i$ in the network.

Our goal is to find $\theta$, the set of best network parameters. The definition of “best” is subject to interpretation and we present several well-known approaches to this issue.

**NOTE:** In this entire section the graph $G$ is known and thus all probabilities are conditional on this graph. This will be omitted from formulas in this section (but will be addressed in model selection).

#### 12.2.1 Maximum Likelihood Estimator (MLE) Approach

In the maximum likelihood estimator approach, we are searching for the set of parameters $\theta$ that maximizes the likelihood function of the network. Starting from the alarm example depicted in Figure 1, we have training data of the form:

$$D = \begin{bmatrix} x^1_j \\ x^2_j \\ \vdots \\ x^M_j \end{bmatrix} = \begin{bmatrix} E^1 & B^1 & R^1 & A^1 & C^1 \\ E^2 & B^2 & R^2 & A^2 & C^2 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ E^M & B^M & R^M & A^M & C^M \end{bmatrix}$$  \hspace{1cm} (12-2)

The likelihood function is defined as:

$$L(\theta : D) = P(D|\theta) = \prod_{m=1}^{M} P\left(E^m, B^m, R^m, A^m, C^m|\theta\right)$$

Using the network structure and independence statements we obtain:
\[ L(\theta : D) = \prod_{m=1}^{M} P\left(E^m | \theta\right) P\left(B^m | \theta\right) P\left(A^m | E^m, B^m, \theta\right) P\left(R^m | E^m, \theta\right) P\left(C^m | A^m, \theta\right) = \]
\[ = \prod_{m=1}^{M} P\left(E^m | \theta_e\right) \times \prod_{m=1}^{M} P\left(B^m | \theta_b\right) \times \prod_{m=1}^{M} P\left(A^m | E^m, B^m, \theta_A\right) \times \prod_{m=1}^{M} P\left(R^m | E^m, \theta_R\right) \times \prod_{m=1}^{M} P\left(C^m | A^m, \theta_{c_i}\right) = \]
\[ = L_e(\theta_e : D) L_b(\theta_b : D) L_A(\theta_A : D) L_R(\theta_R : D) L_{c_i}(\theta_{c_i} : D) \]

Where \( \theta_X \) is the collection of all parameters relevant for the calculation of the probability of \( X \) given its parents. We have used the conditional independence embodied in the network structure to move from row-by-row calculation of the likelihood over \( D \) in (12-2), to column-by-column calculation.

For a general Bayesian network, the likelihood is defined as:
\[ L(\theta : D) = P(D | \theta) = \prod_{m=1}^{M} P\left(x^m | \theta\right) \]

Using equation (12-1) we obtain the following:
\[ L(\theta : D) = \prod_{m=1}^{M} \prod_{i=1}^{n} P\left(x^m_i | Pa^m_i, \theta\right) = \prod_{i=1}^{n} \left( \prod_{m=1}^{M} P\left(x^m_i | Pa^m_i, \theta\right) \right) = \prod_{i=1}^{n} L_i(\theta_i : D) \]

Where \( \theta_i \) is the collection of all parameters relevant for the calculation of the probability of \( X_i \) given its parents.

This means that the likelihood function is the product of the per-node likelihood functions. Selecting \( \theta \) that maximizes the total likelihood is equivalent to selecting each \( \theta_i \) to maximize the appropriate per-node likelihood. This is due to the fact that different parameters control each node – so each \( \theta_i \) is allowed to vary independently.

We say that the likelihood function is decomposable, in a sense that the problem of maximizing the likelihood may be decomposed into \( n \) problems of maximizing the local likelihoods.
The problem is therefore reduced to the problem of maximizing $L_i(\theta_i : D)$.

### 12.2.1.1 The Multinomial Case

A common scenario for Bayesian networks is when all random variables are multinomials. Each random variable $X_i$ has a finite number of possible outcomes. For each possible outcome of its parents, each possible outcome for $X_i$ has a specific probability.

A *configuration* of $Pa(X_i)$ (the parents of $X_i$) is a tuple of values, where each value describes the outcome of a single parent. For example, in the Bayesian network given in Figure 1 (the alarm example), the parents of the variable “Alarm” are “Earthquake” and “Burglary”. A single configuration for these parents, for example, corresponds to a single row in the table given in Figure 1 (for example, configuration $<e,b>$ means that both an earthquake and a burglary occurred).

We denote by $S_i$ the set of all possible configurations for $Pa(X_i)$. We denote by $S_{i,p}$ the set of possible outcomes for $X_i$ given configuration $p$ for $Pa(X_i)$.

For the multinomial case, our parameters are the probabilities assigned to each possible outcome of $X_i$ in each possible configuration of $Pa(X_i)$. For a configuration $p \in S_i$ and value $x \in S_{i,p}$ we denote by $\theta_{i,p,x}$ the parameter defining the probability:

$$
\theta_{i,p,x} = P(X_i = x | Pa_i = p)
$$

In the alarm example, the parameters are the values in the table given in Figure 1, so for example, $\theta_{\text{alarm, ce, b}, \text{e}} = 0.8$.

The $\theta_i$ mentioned in previous results is simply the collection of all $\theta_{i,p,x}$ per a specific node $i$, and $\theta$ is the collection of all $\theta_i$. We denote by $N_{i,p,x}$ the number of times value $x$ for $X_i$ and configuration $p$ for $Pa(X_i)$ appeared in the training data $D$. By changing multiplication order, this gives us the following result for the per-node likelihood:

$$
L_i(\theta_i : D) = \prod_{n=1}^{M} P(x^n | Pa^n_i, \theta) = \prod_{p \in S_i, x \in S_{i,p}} \prod_{\theta_{i,p,x}}^{N_{i,p,x}}
$$

We conclude that $N_{i,p,x}$ are sufficient statistics for the likelihood calculation. It has already been shown (Lecture 2) that in such multinomial cases, the likelihood $L_i(\theta_i : D)$ is maximized by choosing parameters $\hat{\theta}_{i,p,x}$ according to the relative frequencies of configurations in the training data:

$$
\hat{\theta}_{i,p,x} = \frac{N_{i,p,x}}{N_{i,p}}
$$
12.2.2 Bayesian Approach

In the Bayesian approach, we wish to obtain a confidence level for each possible parameter configuration. Unlike the maximum likelihood approach where we selected a single parameter configuration to maximize the likelihood, in the Bayesian approach we will calculate the probability of each parameter configuration, given the training data set D.

12.2.2.1 Bayesian Inference

Assuming a training data set D and a graph G, we wish to calculate $P(\theta | D)$. Using Bayes rule we obtain:

$$P(\theta | D) = \frac{P(D | \theta) P(\theta)}{P(D)} = \frac{L(\theta : D) P(\theta)}{P(D)} \quad (12-3)$$

A similar result is obtained for probability density functions in continuous cases (which usually is the case for $\theta$).

- The prior distribution $P(\theta)$ must be chosen to complete the Bayesian approach, and we shall discuss one family of such distributions (Dirichlet) soon.
- The likelihood $L(\theta : D)$ is usually easily calculated (we have previously seen one such example for the multinomial case).
- The marginal likelihood $P(D)$ is given as in integral over the previous two probabilities.

$$P(D) = \int P(D | \theta) P(\theta) d\theta = \int L(\theta : D) P(\theta) d\theta$$

Let us look for example at the binomial case (heads or tails), where the single parameter $0 \leq \theta \leq 1$ is the probability for heads, and the prior distribution for $\theta$ is uniform over $[0,1]$ (density function 1). For training data D of $N$ samples comprised of $N_H$ heads and $N_T$ tails, we have:

$$L(\theta : D) = P(D | \theta) = \theta^{N_H} (1 - \theta)^{N_T}$$

$$P(D) = \int L(\theta : D) P(\theta) d\theta = \int \theta^{N_H} (1 - \theta)^{N_T} d\theta = \frac{1}{N+1} \binom{N}{N_H}$$

The last result was proven in lecture 2. Using Bayes rule we obtain the following posterior probability density function for $\theta$ given D:
12.2.2 Bayesian Prediction

Given the Bayesian inference calculation for $P(\theta|D)$ we can now predict the outcome of a new sample. In many cases this is a more accurate goal than learning $\theta$ itself. When learning a model we are not necessarily interested in the model itself – but rather in the ability to predict and evaluate future data. For Bayesian networks, this means that upon receiving a new sample, we wish to be able to evaluate the probability of this sample.

While in the maximum likelihood approach we chose a single Bayesian network and can therefore calculate the probability of new samples using this network, in the Bayesian approach we need to average over all possible networks using the posterior probability given in (12-3). We will calculate the distribution $P(x^{M+1}|D)$ - the distribution of the next sample given our knowledge of the training samples (and the graph):

$$P(x^{M+1}|D) = \int P(x^{M+1}|\theta, D) P(\theta|D) d\theta = \int P(x^{M+1}|\theta) P(\theta|D) d\theta \quad (12-4)$$

The final equality is due to the independence of different observations, given the parameters and the graph.

In the example given for a binomial variable with uniform priors we obtain:

$$P(x^{M+1} = H|D) = \int_{\theta} P(x^{M+1} = H|\theta) P(\theta|D) d\theta =$$

$$\int_{\theta} \theta (N+1) \binom{N}{N_H} \theta^{N_H} (1-\theta)^{N-N_H} d\theta =$$

$$(N+1) \binom{N}{N_H} \frac{1}{N+2} \binom{N+1}{N_H+1} = \frac{N_H+1}{N+2}$$

Now we can actually construct a new Bayesian network with the parameter

$$\hat{\theta} = \frac{N_H+1}{N+2}$$

For new samples, this network will calculate exactly the probability of the sample using the Bayesian approach. Note that we have found this network not by searching for a single set of “best” parameters; it is in a sense the “expected” network given the training data and prior distribution. The term “expected” here is appropriate – formula (12-4) yields exactly the expectation of $P(x^{M+1}|\theta)$ over networks distributed according to the posterior distribution for $\theta$. 

$$P(\theta|D) = \frac{L(\theta:D) P(\theta)}{P(D)} = (N+1) \binom{N}{N_H} \theta^{N_H} (1-\theta)^{N-N_H}$$
12.2.2.3 Choice of Prior Distribution

In theory we are free to choose any prior distribution we wish. However, we would like to use prior distributions that satisfy parameter independence, meaning that the set of parameters \( \theta_{i,\beta} \) for the distribution of a certain \( X_i \) given a certain configuration \( \tilde{\beta} \) for \( \text{Pa}(X_i) \) is independent of any other such set of parameters. Such prior distributions yield simpler calculations because all probability calculations can be decomposed, according to the network structure, into the product of independent per-node-and-parent-configuration calculations.

Using a prior distribution that respects parameter independence allows us to decompose the learning problem of the entire parameter set into smaller learning problems, one per node and parent configuration.

12.2.2.4 Dirichlet Priors

In the multinomial case each such set of parameters \( \theta_{i,\beta} \) is simply the set of probabilities \( \theta_{i,\beta,x} \) of \( X_i \) giving outcome \( x \) given configuration \( \tilde{\beta} \) for \( \text{Pa}(X_i) \). A common family of prior distributions in this case is Dirichlet priors.

A Dirichlet distribution with hyper-parameters \( \alpha_1, \alpha_2, \ldots, \alpha_r \) is a distribution over parameters of an \( r \)-valued multinomial distribution \( \phi = (\phi_1, \phi_2, \ldots, \phi_r) \) (the sum of \( \phi \) is 1 of course). The probability density of \( \phi \) is given by:

\[
\text{Dir}(\phi) = B(\alpha_1, \alpha_2, \ldots, \alpha_r) \prod_{i=1}^{r} \phi_i^{\alpha_i-1}
\]

Where \( B(\alpha_1, \alpha_2, \ldots, \alpha_r) \) is a constant independent of \( \phi \), that serves as a normalization factor for \( \text{Dir} \), ensuring that \( \text{Dir} \) is a probability density function (it’s integral over all \( \phi \) must be 1). It can be verified that this means that:

\[
B(\alpha_1, \alpha_2, \ldots, \alpha_r) = \frac{\Gamma\left(\sum_{i=1}^{r} \alpha_i\right)}{\prod_{i=1}^{r}\Gamma(\alpha_i)} \quad (\Gamma \text{ is the Gamma function})
\]

However, this will not be critical for this lecture. Note that for \( r = 2 \), the Dirichlet distribution is exactly a Beta distribution. Also note that if \( \alpha_i = 1 \) for all \( i \), we have a uniform distribution over \( \phi \).

Let \( Y \) be an \( r \)-valued multinomial variable, and let \( \phi \) be its associated probabilities. We are given a set \( D \) of \( N \) i.i.d samples of \( Y \), and we denote by \( N_k \) the number of appearances of the \( k \)-th possible outcome of \( Y \) in \( D \). Assuming a \( \phi \) has a Dirichlet prior distribution with hyper-parameters \( (\alpha_1, \alpha_2, \ldots, \alpha_r) \), the posterior distribution given the prior and \( D \) is given by:
\[
P(\varphi|D) = \frac{P(D|\varphi)P(\varphi)}{P(D)} = \frac{B(\alpha_1, \alpha_2, \ldots, \alpha_r)}{P(D)} \left( \prod_{k=1}^{r} \varphi_k^{N_k} \right) \left( \prod_{k=1}^{r} \varphi_k^{\alpha_k-1} \right) = \\
\frac{B(\alpha_1, \alpha_2, \ldots, \alpha_r)}{P(D)} \prod_{k=1}^{r} \theta_k^{\alpha_k+N_k-1} = \\
C \cdot \text{Dir}(\alpha_1 + N_1, \alpha_2 + N_2, \ldots, \alpha_r + N_r)
\]

Where \( C \) is a constant independent of \( \varphi \). Because \( \text{Dir}(\alpha_1 + N_1, \alpha_2 + N_2, \ldots, \alpha_r + N_r) \) is a probability density function whose integral is 1, and likewise for \( P(\varphi|D) \), the only possible \( C \) is \( C = 1 \):

\[
P(\varphi|D) = \text{Dir}(\alpha_1 + N_1, \alpha_2 + N_2, \ldots, \alpha_r + N_r) \quad (12-5)
\]

We conclude that the posterior per parameter set is also a Dirichlet distribution, with hyper-parameters that are the sum of prior hyper-parameters and sample appearance counts.

We wish to calculate Bayesian prediction in this case. Note that for any \( r \)-valued multinomial variable \( Y \) with values \((v_1, \ldots, v_r)\), with Dirichlet prior for parameters \( \varphi \), we have:

\[
P(Y = v_i) = \Phi_i \text{Dir}(\alpha_1, \alpha_2, \ldots, \alpha_r) = \frac{\alpha_i}{\sum_{j=1}^{r} \alpha_j}
\]

Since the posterior distribution given this prior and a sample set \( D \) is also a Dirichlet distribution, we have:

\[
P(Y = v_i|D) = \Phi_i \text{Dir}(\alpha_1 + N_1, \alpha_2 + N_2, \ldots, \alpha_r + N_r) = \frac{\alpha_i + N_i}{\sum_{j=1}^{r} (\alpha_j + N_j)} \quad (12-6)
\]

So the prediction probabilities are actually estimated by the relative frequencies of outcomes in the sample data, but we “adjust” the number of appearances by adding \( \alpha_k \).

Note that if \( \alpha_k = 1 \) for all \( k \) then we obtain the exact same result we have seen in the uniform distribution case, as expected.

Back to Bayesian networks, assume a parameter independent prior distribution, where each node \( X_i \) and each parent configuration \( \hat{p} \) is a multinomial with possible outcome set \( S_{i,\hat{p}} \). Assume a Dirichlet prior distribution on the multinomial parameters with hyper-parameters \( \alpha_{i,\hat{p},x} \) (for each \( x \in S_{i,\hat{p}} \)). We can apply results (12-5) and (12-6) on a per-node-and-parent-configuration basis to obtain:

\[
P(\theta_{i,\hat{p}}|D, Pa_i = \hat{p}) = \text{Dir}\left( \left\{ \alpha_{i,\hat{p},x} + N_{i,\hat{p},x} \mid x \in S_{i,\hat{p}} \right\} \right)
\]

And
So by defining a Bayesian network with parameters

\[
P(X_i = x | D, Pa_i = \tilde{p}) = \frac{\alpha_{i,\tilde{p},x} + N_{i,\tilde{p},x}}{\sum_{s \in S_{\tilde{p}}} (\alpha_{i,s,x} + N_{i,s,x})}
\]

We have a network useful for future prediction.

### 12.2.3 Parameter Estimation Summary

We have seen two approaches to parameters estimation in a Bayesian network. For multinomial Bayesian networks, given a training data set D and appropriate appearance counts of the various configurations in the data set, we have obtained that the following are the “best” parameters:

\[
\hat{\theta}_{i,\tilde{p},x} = \frac{N_{i,\tilde{p},x}}{N_{i,\tilde{p}}}, \quad (\text{MLE Approach})
\]

\[
\tilde{\theta}_{i,\tilde{p},x} = \frac{\alpha_{i,\tilde{p},x} + N_{i,\tilde{p},x}}{\sum_{s \in S_{\tilde{p}}} (\alpha_{i,s,x} + N_{i,s,x})}, \quad (\text{Bayesian Prediction Approach using Dirichlet Priors})
\]

Note that both estimators are asymptotically equivalent (as the number of training samples grows, the Dirichlet hyper-parameters become insignificant). Additionally, both estimators may be used in an “online” learning manner, where the learnt parameters are updated as new data is obtained (by keeping track of the appearance counts).

### 12.3 Model Selection

In the previous section we described the problem of parameter estimation: given the network structure (a DAG G representing the independencies between the variables in our domain) and a data set D, find the best parameters defining the conditional probability functions in the network.

In this section we will address the problem of learning the graph structure from the data set D.

Once again, we only address the complete data scenario.

#### 12.3.1 Effects of Learning an Inaccurate Graph

If a learning algorithm finds an inaccurate graph G, where the correct graph is actually G’, then there are two possible cases:

1) G describes an independence relation between X_i and X_j given Z (a set of variables), that isn’t described by G’. In this case G must have a missing edge, relative to G’. Note that the missing edge is not necessarily the edge between X_i and X_j.

Comment: If you have used additional sources for this section, please specify them in the references list.
The absence of an edge from X to Y in G (this edge exists in G') means that the calculation of Y's probability in the Bayesian network doesn't take into account the value of X. No method of fitting the parameter from the data can compensate for this absence.

2) The graph G' describes an independence relation between X_i to X_j given Z which isn't described by G. This means that an extra edge exists in G relative to G'.

Having unnecessary edges is undesired because:

a. It increases the number of parameters to be estimated and therefore the dimension of the problem.

b. It increases the chance for overfitting

c. Understanding of the real-world dependencies from the network structure is more difficult.

d. It complicates probability calculation for an observation using the network.

12.3.2 Score-based search

Learning a model for a Bayesian network is performed by assigning a score for each possible model, where the score describes how well the model fits the data, or is able to predict future data. We will present two approaches to the score, MLE and Bayesian. The goal of the learning algorithm is to find a high scoring network (ideally, the highest scoring network).

12.3.2.1 Selecting a Scoring Function

A scoring function S(G,D) assigns a score for each graph G given a training data set D. We would like to select a scoring function that respects several properties.

Definition (score equivalent): A scoring function S(G,D) is said to be score equivalent if for any pair of equivalent DAG, G and H, S(G,D) = S(H,D).

Definition (decomposable): A scoring function is decomposable if it can be written as a sum of measures, each of which is a function of a one node and its parents.

\[ S(G,D) = \sum_{i} s(X_i, Pa^G(X_i), D_i) \]

Where s assigns a score to the fact that X_i's parents are the set Pa\(^G\)(X_i), using only D_i – the part of the data set D including only samples for X_i and Pa\(^G\)(X_i). Using a decomposable score allows us to perform local changes to the graph and to adjust the score based only on the score changes in the affected nodes without recalculating the entire score for the entire graph.

Definition (locally consistent): Given a data set D, containing M observations which are i.i.d, let G be any DAG and G' be the DAG created by adding the edge X_j → X_i to G. We denote by P the real-world probability distribution we are trying to model.

We say that the scoring function S is locally consistent if for M large enough, the following holds:
1. If $X_j$ is not independent of $X_i$ given $Pa^G_j(X_j)$ in $P$ then $S(G', D) > S(G, D)$
2. If $X_j$ is independent of $X_i$ given $Pa^G_j(X_j)$ in $P$ then $S(G', D) < S(G, D)$

Local consistency of scores allows to use local improvements to progress in the search space while increasing the fitness of the model to the real world distribution. Chickering showed that the Bayesian scoring criteria is locally consistent.

### 12.3.2.2 Maximum Likelihood Score

The *entropy* of $x$ given a distribution $p$ is

$$H_p(x) = -\sum p(x) \log p(x)$$

The *mutual information* between a set of variables $Y$ and $Z$ is defined as:

$$MI_p(Y, Z) = H_p(Y) - H_p(Y | Z) = \sum_{y,z} p(y,z) \log \frac{p(y,z)}{p(y) \cdot p(z)}$$

If $Y$ and $Z$ are independent then $MI_p(Y, Z) = 0$.

If $Y$ is totally predictable given $Z$ then $MI_p(Y, Z) = H_p(Y)$

The maximum likelihood score for a structure is defined as the likelihood using the best (MLE-wise) parameter setting for that structure. This can be expressed in terms of entropy and mutual information.

$$S(G) = \log(L(G, D)) = m \sum_{i=1}^{n} MI_p(X_i, Pa^G_i(X_i)) - H_p(X_i)$$

Where $p$ is the probability distribution generated by the parameters for $G$ that maximize likelihood for the training data. As we have seen, these parameters are simply the relative frequencies of observations in the training data.

Note that adding edges can only improve the score, because:

$$MI_p(Y, Z) \leq MI_p(Y, Z \cup \{X_i\})$$

Therefore, the best scoring graph will be a fully connected graph. Additionally, this means that the scoring function isn’t *locally consistent* but it’s *score equivalent* and *decomposable*

As always, when using a learning algorithm one must worry about overfitting. The MLE score also does not automatically prevent overfitting. We should therefore prevent it ourselves by limit the number of parents a node can have, or by adjusting the score taking into account the number of parameters in the model. This is in accordance with Occam’s Razor principle. Such a score usually maintains the score’s decomposability.

### 12.3.2.3 Bayesian Score

In the Bayesian approach we assign a score to a model by estimating its probability given the training data set and a prior distribution on models. We therefore wish to estimate $P(G | D)$. We know from Bayes rule that:

$$P(G | D) = \frac{P(D | G) \cdot P(G)}{P(D)}$$
And $P(D)$ is a constant factor which doesn't depend on $G$, and therefore has no effect on our search. $P(G)$ is the prior distribution over the network structures. $P(D|G)$ is the marginal likelihood, given by:

$$P(D|G) = \int P(D|G, \theta) * P(\theta|G)$$

Where

1. $P(D|G, \theta)$ is the likelihood
2. $P(\theta|G)$ is the prior over parameters, given the model

So we shall define the score as:

$$\text{score}(G) = \log \{ P(D|G) \} + \log \{ P(G) \}$$

It's important to choose the right prior and to let two equivalent graphs have the same prior. This score is known to be locally consistent, score equivalent, and decomposable.

## 12.3.3 Search Algorithms

For most sensible scoring functions, the problem of learning graph structure is known to be hard. Chickering shows that for the Bayesian score the problem is NP complete (Chickering 1996). Therefore most model selection algorithms use search heuristics.

Most heuristic search algorithms are composed of three elements:

1. Search space and search space operators
2. Scoring method for a graph in the search space
3. Initial graph selection
4. Search technique

### 12.3.3.1 Search space and search space operators

Usually the search for the right graph structure is done on the collection of possible DAGs. The operators used to traverse the graph space

1. Adding an edge
2. Removing an edge
3. Reversing an edge
Notice that not all of the edge reversal operations are legit as we are not allowed to create cycles in the graph.

One must take into account, when searching in this space, that the problem of equivalence between graphs makes the search algorithm more complex. The scoring method and priors used must take this issue into consideration.

As a consequence of the above, the search is sometimes performed on equivalence class graphs (PDAG). Such algorithms use different search space operators and scoring functions (Chickering 2002).

Typically any general purpose heuristic search algorithm may be used. To name a few:

1. **Greedy Hill Climbing**
   
   Always choose the operator that produces the network with the best score. Such algorithms might converge to a local maxima. This can be improved using:

   1.1. **Random starting points** - always keep the best traversed result, occasionally jump to a new random starting point.
   
   1.2. **Taboo list** - keep in memory k most visited structures and always choose a move which doesn't generate one of them.

2. **Best First Search (A*)**

3. **Simulated Annealing**
Escape local maxima by allowing some apparently “bad” moves, but gradually decrease their size and frequency:

3.1. Pick a start state $s$

3.2. Pick a temperature parameter $T$, which will control the probability of a random move

3.3. Repeat:

3.3.1. Select a random successor $s'$ of the current state

3.3.2. Compute $\Delta E = \text{score}(S') - \text{score}(S)$

3.3.3. If $\Delta E \geq \text{Threshold}$ make the move to $s'$

3.3.4. Else move to $s'$ with probability $\frac{\Delta E}{e^{T}}$

3.3.5. According to the score change and the time passed change the temperature $T$ (as the search is near the end the temperature will be low)

If the temperature is decreased slowly enough, it can be shown that the algorithm converges asymptotically to the best solution with probability 1.

When the temperature is high, the algorithm is in an exploratory phase (all moves have about the same value). When the temperature is low, the algorithm is in an exploitation phase (the greedy moves are most likely).

4. Others…

12.3.3.2 Initial Graph Selection

The initial graph selection can affect the speed of the search algorithm. The following methods are typically used:

1. Random graph
2. Empty graph
3. Best scoring tree (can be found using maximum spanning tree, where the weights of the edges are the mutual information between the nodes)
4. Mutual Information based graph

Let $C_{ij} = MI_p \left( X_i, X_j \right)$. This method computes $C_{ij}$ for every pair $i, j$ and sorts them. A threshold is chosen (the “knee” of the graph of the sorted $C_{ij}$’s) and the model graph will contain all edges from $i$ to $j$ such that $C_{ij} >$ threshold (breaking cycles when necessary).

12.4 Application to Gene Expression Analysis

A central goal of molecular biology is to understand the regulation of protein synthesis and its reactions to external and internal signals. All the cells in an organism carry the same genomic
data, yet their protein makeup can be drastically different both temporally and spatially, due to regulation. Protein synthesis is regulated by many mechanisms at its different stages. These include mechanisms for controlling transcription initiation, RNA splicing, mRNA transport, translation initiation, post-translational modifications, and degradation of mRNA/protein. One of the main junctions at which regulation occurs is mRNA transcription. A major role in this machinery is played by proteins themselves, that bind to regulatory regions along the DNA, greatly affecting the transcription of the genes they regulate.

In recent years, technical breakthroughs in spotting hybridization probes and advances in genome sequencing efforts lead to development of DNA microarrays, which consist of many species of probes, either oligonucleotides or cDNA, that are immobilized in a predefined organization to a solid phase. By using DNA microarrays researchers are now able to measure the abundance of thousands of mRNA targets simultaneously. Unlike classical experiments, where the expression levels of only a few genes were reported, DNA microarray experiments can measure all the genes of an organism, providing a “genomic” viewpoint on gene expression.

The bayesian network model is a suitable tool for discovering interactions between genes based on multiple expression measurements. Such model is attractive for it’s ability to describe complex stochastic processes, and since it provides clear methodologies for learning from (noisy) observations.

This is a learning problem, as defined above, where the inputs are measurements of gene expression under different conditions. While achieving enormous amount of gene expression data for one experiment, each experiment is very expensive to execute. The amount of samples, even in the largest experiments in the foreseeable future, does not provide enough information to construct a full detailed model with high statistical significance. The curse of dimensionality here plays a major role, as we get expressions of thousands of genes for a very few samples.

The output of the learning procedure is a model of gene interactions which uncover the mutual transcription interactions of the DNA. This is the regulatory or the transcription network.

A causal network models not only the distribution of the observations, but also the effects of interventions: if X causes Y, then manipulating the value of X affects the value of Y. On the other hand, if Y is a cause of X, then manipulating Y will not affect X. Thus, although X→Y and X←Y are equivalent bayesian networks, they are not equivalent as causal networks. In our biological domain assume X is a transcription factor of Y. If we knockout gene X then this will affect the expression of gene Y, but a knockout of gene Y has no effect on the expression of gene X.

A causal network can be interpreted as a bayesian network when we are willing to make the Causal Markov Assumption: given the values of a variable’s immediate causes, it is independent of its earlier causes. When the casual Markov assumption holds, the causal network satisfies the Markov independencies of the corresponding bayesian network.

We present our modeling assumptions: we consider probability distributions over all possible states of the system in question (a cell or an organism and its environment). We describe the state of the system using random variables. These random variables denote the expression level of individual genes. In addition, we can include random variables that denote other attributes that affect the system, such as experimental conditions, temporal indicators (i.e., the time/stage that
the sample was taken from), background variables (e.g., which clinical procedure was used to get a biopsy sample), and exogenous cellular conditions.

We attempt to build a model which is a joint distribution over a collection of random variables (Figure 4). If we had such a model, we could answer a wide range of queries about the system. For example, does the expression level of a particular gene depend on the experimental condition? Is this dependence direct, or indirect? If it is indirect, which genes mediate the dependency? Not having a model at hand, we want to learn one from the available data and use it to answer questions about the system.

Figure 4 displays an example of the graphical display of Markov features. This graph shows a “local map” for the gene SVS1. The width (and color) of edges corresponds to the computed confidence level. An edge is directed if there is a sufficiently high confidence in the order between the genes connected by the edge. This local map shows that CLN2 separates SVS1 from several other genes. Although there is a strong connection between CLN2 to all these genes, there are no other edges connecting them. This indicates that, with high confidence, these genes are conditionally independent given the expression level of CLN2.

Comment: What are Markov features and what is the relation to Bayesian networks – explain more clearly
Add a references section, listing all of the sources used for writing this scribed, and all of the sources mentioned in the text.