Computational Genomics **Fall Semester, 2003**

Lecture 8: December 17, 2003

Lecturer: Irit Gat-Viks *Scribe: Tal Peled and David Burstein*

8.1 Exploiting Independence Property

8.1.1 Introduction

Our goal is to model biological processes given biological data. It is essential for such a model to be of probabilistic nature, since measurements are noisy and molecular biology itself has stochastic characteristics. In this lecture Bayesian networks [4, 8] will be introduced [1], these provide graphical representation of joint probability distributions in a compact way.

These networks are useful for describing systems composed of locally interacting components, that is, the value of each component depends on the values of a small number of other components. These networks also provide us a model of causal influence, as will be discussed later. One of the advantages of using Bayesian networks is the fact that 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. These applications include medical and fault diagnosis expert systems, monitoring systems, information access technologies, speech recognition systems, and finally analysis and classification for biological sequencing.

8.1.2 Basic Probability Rules

- **Product Rule:**

$$P(A, B) = P(A \mid B) \cdot P(B) = P(B \mid A) \cdot P(A) \tag{8.1}$$

  Where $P(A)$ is the probability of A, $P(A, B)$ is the joint probability of both A *and* B, and $P(A \mid B)$ denotes the probability of A, given B.

- **Independence:** Independence between A and B will be denoted as $I(A; B)$. If such an independence exists, then:

$$P(A, B) = P(A) \cdot P(B) \tag{8.2}$$

  Or alternatively

$$P(A \mid B) = P(A), \quad P(B \mid A) = P(B) \tag{8.3}$$

[1] This lecture is based on presentations of Nir Friedman

- **Conditional Independence:** If A and B are independent given known C:

$$P(A, B \mid C) = P(A \mid C) \cdot P(B \mid C) \tag{8.4}$$

Or alternatively:

$$P(A \mid B, C) = P(A \mid C), \quad P(B \mid A, C) = P(B \mid C) \tag{8.5}$$

Such an independence will be denoted as $I(A; B \mid C)$.
Note that $I(A; B) \Rightarrow I(A; B \mid C)$: if A and B are independent, then they will be independent given any known C. Nevertheless, $I(A; B \mid C) \not\Rightarrow I(A; B)$: if A and B are independent given known C, it does *not* mean A and B are independent in any case.

- **Total probability Theorem:** Let $B_1, \ldots, B_n$ be a set of disjoint events, which their union is the sample space: $\bigcup_{i=1}^{n} B_i = \Omega, \ \forall i \neq j . B_i \bigcap B_j = \phi$.

$$P(A) = \sum_{i=1}^{n} P(A, B_i) = \sum_{i=1}^{n} P(B_i) \cdot P(A \mid B_i) \tag{8.6}$$

The last formula is for the case B is a *discrete* variable, in case B is a *continuous* variable, the $\int$ (integral) function will be used to sum:

$$P(A) = \int P(A, B)\ dB = \int P(B) \cdot P(A \mid B)\ dB$$

- **Bayes Rule:**

$$P(A \mid B) = \frac{P(B \mid A) \cdot P(A)}{P(B)} \qquad P(A \mid B, C) = \frac{P(B \mid A, C) \cdot P(A \mid C)}{P(B \mid C)} \tag{8.7}$$

- **Chain Rule:**

$$P(X_1, \ldots, X_n) = P(X_1 \mid X_2, \ldots, X_n) \cdot P(X_2 \mid X_3, \ldots, X_n) \cdot \ \ldots \ \cdot P(X_{n-1} \mid X_n) \cdot P(X_n) \tag{8.8}$$

8.1.3 Exploiting Independence Property

Independence of variables can be used to simplify significantly the representation of complex joint distributions. Consider the representation of the joint distribution of n variables, for example n tosses of a coin. Even in this simple binary example, explicit representation of the joint distribution will require 2^n entries, for all possible assignment of heads / tails to the tosses.

X_1	X_2	X_3	X_4	$P(X_1, X_2, X_3, X_4)$
0	0	0	0	0.0625
0	0	0	1	0.0625
0	0	1	0	0.0625
0	0	1	1	0.0625
0	1	0	0	0.0625
0	1	0	1	0.0625
0	1	1	1	0.0625
1	0	0	0	0.0625
1	0	0	1	0.0625
1	0	1	0	0.0625
1	0	1	1	0.0625
1	1	0	0	0.0625
1	1	0	1	0.0625
1	1	1	1	0.0625

Table 8.1: Joint distribution representation of four fair tosses

Example 8.1: Presenting Distribution of four Fair Tosses

Let $X_i = 1$ denote heads in toss number i. The joint distribution of four tosses is represented in table 8.1.

This is obviously a wasteful representation. If the independence of each toss is exploited, a much more compact representation can be achieved. Such a representation would include only the probabilities of each toss (table 8.2). In this degenerate case the tosses are of a fair coin, so the probability to get heads in each toss is 0.5.

	$X_i = 0$	$X_i = 1$
X_1	0.5	0.5
X_2	0.5	0.5
X_3	0.5	0.5
X_4	0.5	0.5

Table 8.2: Independent distribution representation of four fair tosses

The joint distribution $P(x_1, x_2, x_3, x_4)$ would be computed using this table as $P(X_1 = x_1) \cdot P(X_2 = x_2) \cdot P(X_3 = x_3) \cdot P(X_4 = x_4)$. In most biological applications, the majority of the variables are not completely independent, but exploiting the existing independencies can be used to achieve a rather simple representation of complex dependencies.

First a simple example will be presented to explain the principles of this method.

Example 8.2: Two Variables Conditional Probability Distribution (CPD)

Consider the problem of determining whether a suspect is guilty. The suspect is standing trial, and the judge finds him either guilty or innocent. The probability space is the joint distribution over these two variables: whether the suspect is **g**uilty, G, and whether the **j**udge will find him guilty, J. Assuming theses two variables are binary (0 - not guilty; 1 - guilty), the joint distribution has four entries. Denote the case that $G = 0$ by g^0, the probability the suspect is not guilty, and in the same manner we use g^1, j^0, j^1. A possible joint distribution of G and J is described in table 8.3.

$\boldsymbol{G}$	$\boldsymbol{J}$	$\boldsymbol{P(G,J)}$
g^0	j^0	0.38
g^0	j^1	0.02
g^1	j^0	0.06
g^1	j^1	0.54

Table 8.3: Joint distribution of two variables (the suspect-judge problem)

This is the *joint distribution representation* which presents all the different assignments of valid values to the variables: The probability that the suspect is innocent and the judge finds him innocent is 0.38, the probability that he is innocent, and yet found guilty by the judge is 0.02, the probability that the suspect is guilty but found innocent is 0.06, etc...

The same joint distribution can be alternatively displayed in a somewhat more natural manner. Instead of specifying the various joint entries, the product rule (equation 8.1) is used to get $P(J,G) = P(J \mid G) \cdot P(G)$, so $P(G)$ and $P(J \mid G)$ are to be represented. This will require the use of two tables, one representing the distribution of G, and the other representing the *conditional probability distribution (CPD)* of J given G. This representation is denoted as a *factorial representation* and is presented in table 8.4.

The distribution $P(J \mid G)$ represents the probability for the judge to find the suspect guilty in the two possible alternatives: In the case of g^0 - suspect not guilty and in case of g^1 - suspect is guilty. It is easy to see in this representation the false negative probability: $P(j^0 \mid g^1)$ - the judge found the suspect innocent although he was guilty (10% in this example), and the false positive probability: $P(j^1 \mid g^0)$ - the probability of an innocent suspect to be found guilty (5%).

It's also easy to switch between the tables using the product rule(equation 8.1), for example in the joint distribution representation appears $\mathrm{P}(g^0 j^0) = 0.38$ (suspect found innocent rightfully). This can be achieved from the factorial representation using the product rule $P(J,G) = P(J \mid G) \cdot P(G)$, $P(g^0 j^0) = P(j^0 \mid g^0) \cdot P(g^0) = 0.95 \cdot 0.4 = 0.38$

$\boldsymbol{G}$	$\boldsymbol{P(G)}$
$\boldsymbol{g^0}$)	0.4
$\boldsymbol{g^1}$)	0.6

$\boldsymbol{P(J \mid G)}$	$\boldsymbol{j^0}$	$\boldsymbol{j^1}$
$\boldsymbol{g^0}$	0.95	0.05
$\boldsymbol{g^1}$	0.1	0.9

Table 8.4: Factorial representation of two variable distribution (suspect-judge problem)

Switching from joint distribution representation to factorial is done simply by summing the relevant probabilities: $P(g^0) = P(g^0 j^0) + P(g^0 j^1) = 0.38 + 0.02 = 0.4$

It is instructive to consider the number of independent parameters needed in each of the representations. The joint distribution representations requires three independent parameters, as there are four entries, and their sum is known to be 1, so the fourth completes to 1. The factorial representation consists of three probability distribution:

1. $P(G)$, so $P(g^0) + P(g^1) = 1$.
2. $P(J \mid g^0)$, so $P(j^0 \mid g^0) + P(j^1 \mid g^0) = 1$.
3. $P(J \mid g^1)$, so $P(j^0 \mid g^1) + P(j^1 \mid g^1) = 1$.

Therefore, only three parameters are required. Note that in this example the two representations use the same number of parameters, but as will be seen further, the factorial method enables a more compact representation.

Bayesian networks will be formally defined further on, in Section 8.2, yet it will be useful to consider how a Bayesian network of this example will be represented. A Bayesian network in this case will have two nodes, representing the two variables G and J, and a single edge from G to J, representing J depends on G (the *direction of dependence* in this model is from J to G). Figure 8.1 shows the Bayesian network in this simple case:

Example 8.3: Three Variables Conditional Probability Distribution (CPD)

Now a more complicated setting will be considered, where the suspect is being tested on a polygraph during the investigation. As polygraph results are not adequately reliable, they are not admissible in court, and therefore the judge does *not* see the results. This means his verdict is independent of the polygraph's results. Now there are three variable to consider: G, J and the outcome of the polygraph, L. Assuming the polygraph results are also binary, the joint distribution has eight entries. Table 8.5 presents a possible joint distribution of G, J and L.

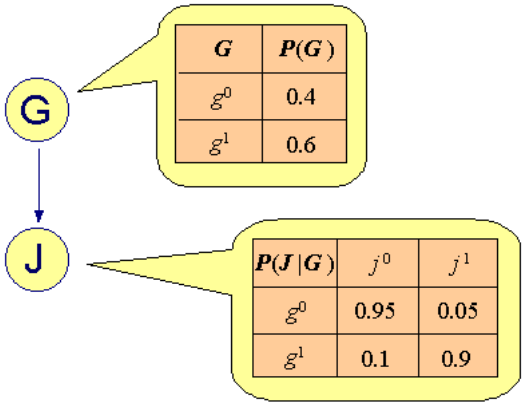

Figure 8.1: Bayesian network for the suspect-judge example. Each variable in the network is associated with a local probability distribution

G	J	L	$P(G,J)$
g^0	j^0	l^0	0.304
g^0	j^0	l^1	0.076
g^0	j^1	l^0	0.016
g^0	j^1	l^1	0.004
g^1	j^0	l^0	0.006
g^1	j^0	l^1	0.054
g^1	j^1	l^0	0.054
g^1	j^1	l^1	0.486

Table 8.5: Joint distribution of three variables (the suspect-judge-polygraph problem)

The result of the polygraph and the judge's verdict clearly depends on whether the suspect is guilty or not. The polygraph outcome and the verdict are also dependent: if it is given that one of them found the suspect guilty, it increases the probability that the other will also find him guilty.

However, there is a *conditional* independence: If it is known whether the suspect is guilty, the outcome of the polygraph does no longer give information about the verdict, and vice versa, assume it is known the suspect is not guilty, the fact that the judge found him guilty, for example, does not increase the probability the polygraph will show the same. Denoted by l^1 the polygraph outcome which states that the suspect is guilty and l^0 otherwise. Then, $P(l^1 \mid g^1, j^1) = P(l^1 \mid g^1)$. More generally it can be assumed that $I(J; L \mid G)$, this means if G is given, L and J are independent. Note that this assumption is based on the fact that the judge and the polygraph take into considerations completely different aspects of the case.

The conditional independence allows to provide a compact representation of the joint distribution, as seen before. Based on the product rule (8.1), the joint distribution is:

$$P(G, J, L) = P(J, L \mid G) \cdot P(G)$$

The conditional probability assumed implies that:

$$P(J, L \mid G) = P(J \mid G) \cdot P(L \mid G)$$

Hence,

$$P(G, J, L) = P(J \mid G) \cdot P(L \mid G) \cdot P(G) \tag{8.9}$$

Equation 8.9 leads to the desired factorial representation. Therefore, in order to specify fully the joint distribution of this case, we have to know $P(G)$, $P(J \mid G)$ and $P(L \mid G)$. Note that $P(G)$ and $P(J \mid G)$ distributions stays the same as in example 8.2 (table 8.4). This means that expanding the previous example to include also L required only the distribution of $P(L \mid G)$ to be added, this distribution is displayed in table 8.6. That is in contrast to the eight entries joint distribution representation, in which the whole distribution has to be recalculated.

$\boldsymbol{P(L \mid G)}$	$\boldsymbol{l^0}$	$\boldsymbol{l^1}$
$\boldsymbol{g^0}$	0.8	0.2
$\boldsymbol{g^1}$	0.1	0.9

Table 8.6: The conditional $P(L \mid G)$ distribution which together with table 8.4 ,matches the joint distribution described in table 8.5.

Together these conditional probability distributions fully specify the joint distribution. For example: $P(g^0, j^0, l^1) = P(g^0) \cdot P(j^0 \mid g^0) \cdot P(l^1 \mid g^0) = 0.4 \cdot 0.95 \cdot 0.2 = 0.076$. Unlike the previous example, this one demonstrates a situation in which the factorial representation is more compact: only five independent parameters suffice to fully specify the whole distribution with conditional dependencies (for example $P(g^0)$, $P(j^0 \mid g^0)$, $P(j^0 \mid g^1)$, $P(l^0 \mid g^0)$, $P(l^0 \mid g^1)$, since the other only completes to 1), whilst seven parameters are required using the joint representation (the eighth parameter completes to 1). Note that we succeed to reduce the number of parameters due to conditional independence between the variables. Moreover, the factorial representation has another important property, which was mentioned before, *modularity*. When adding a new variable to a model, L in this example, only the local probability model for L needs to be added, while the local probabilities of G and J are reused. In joint distribution, this is not the case, and all the probabilities have to be recalculated, as the distribution changes entirely.

The Bayesian network corresponding to this scenario (figure 8.2) contains an additional node in its network, representing L, and an edge from G to L, denoting the direction of dependence: L depends on G. Note that given the value of the parent G, both L and J are independent.

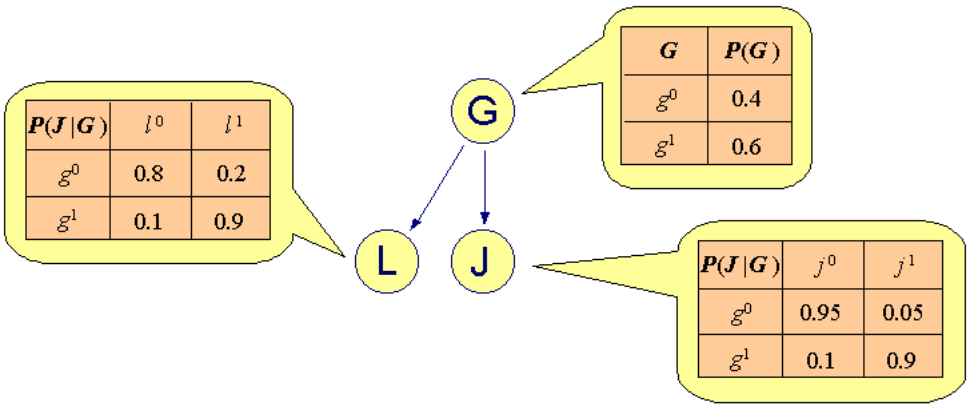

Figure 8.2: Bayesian network for the suspect-judge-polygraph example

8.2 Bayesian Networks

8.2.1 Representing Distributions with Bayesian Networks

Suppose we are given a set of assertions and a variety of ways in which they support each other. Each assertion establishes a value for an attribute and is of the form $(X_i = x_i)$, that is, "Variable X_i has value x_i". The variables are $X_1, \ldots, X_n$. We would know everything we need to know about the world described by these assertions if we had the joint probability $P(X_1, \ldots, X_n)$. From this probability function we could compute any other probability such as $P(X2)$ or $P(X2 \mid X3, X5)$. Unfortunately, even when assuming for simplicity that the variables are binomial, the representation complexity of $P(X_1, \ldots, X_n)$ is, as we've seen, 2^n, which is impractical even for small value of n.

Bayesian Networks simplify this problem by taking advantage of existing causal connections between assertions, and of assumptions about conditional independence. 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 nodes correspond to the random variables $X_1, \ldots, X_n$, and its edges correspond to dependencies and their directions. This component is know as the *Qualitative part*.
- The second component, describes the *local probability model, the conditional probability distribution (CPD)* for each variable, given its parents in G. Let X_i be a variable and $\mathbf{Pa}(X_i)$ its parents in G, the CPD of X_i is the distribution of $P(X_i \mid \mathbf{Pa}(X_i))$. This part is the *Quantitative part*

Together, these two components specify a unique distribution over $X_1, \ldots, X_n$. The graph G represents conditional independence assumptions that allow the joint distribution to be

decomposed, economizing on the number of parameters. Note that the graph G encodes the *Markov Assumption: Each variable X_i is independent of its non-descendants, given its parents in G* (it is still dependent on its descendants). It is a natural assumption for many causal processes. If the parents of an event, meaning the events which directly caused it, are known, it is independent of the events which effect its parents, or of any other event, except the events which it is the cause for, or the cause for one of their ancestors.

By applying the chain rule of probabilities and properties of conditional independencies, any joint distribution can be decomposed to the *product form* according to the Markov assumption on it's Bayesian network. Suppose w.l.o.g (without loss of generality) $X_1, \ldots, X_n$ are arranged in reversed topological order, i.e. if X_j is a descendant of X_i in the network then $j < i$. According to the chain rule (equation 8.8):

$$P(X_1, \ldots, X_n) = P(X_1 \mid X_2, X_3, \ldots, X_n) \cdot P(X_2 \mid X_3, X_4, \ldots, X_n) \cdot \ldots \cdot P(X_{n-1} \mid X_n) \cdot P(X_n)$$

Since each X_i does not depend on any of its non-descendants given its parents, and any variable X_j that is a descendant of X_i has a lower index ($j < i$) due to the reversed topological order, then $P(X_i \mid X_{i+1}, \ldots, X_n) = P(X_i \mid \mathbf{Pa}(X_i))$, where $\mathbf{Pa}(X_i)$ is the set of parents of X_i in graph G. Thus, we get the chain rule for Bayesian networks:

$$P(X_1, \ldots, X_n) = \prod_{i=1}^{n} P(X_i \mid \mathbf{Pa}(X_i)), \tag{8.10}$$

Example 8.4: A Simple Bayesian Network with Five Variables

In Figure 8.3 we can see a simple example of a Bayesian network structure. This network describes the connections between the following events:

- B - There is a **B**urglary.
- A - The **A**larm goes off.
- E - There is an **E**arthquake.
- R - There is a **R**adio report of an earthquake.
- C - Mr. Watson, the neighbor, **C**alls to inform us he heard the alarm.

The alarm is set to detect earthquakes and attempts of burglary and. Naturally, there is a probability of a false alarm to occur, or for some malfunction to cause the alarm not to operate when it should. Mr. Watson is the neighbor, and in case he hears the alarm he should call us to inform the alarm had gone off. He might, of course, not hear it or mistakenly think he had heard it (though it hadn't gone off). In addition the local radio

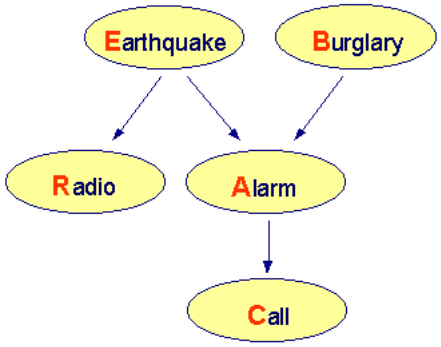

Figure 8.3: Simple Bayesian Network with Five Variables

station usually reports any case of an earthquake.
Specifying the joint distribution of these events, requires $2^5-1 = 31$ parameters. This is quite a high number for such a simple system. Consider an expert system for monitoring intensive care patients, which measures over 37 different properties. Joint distribution representation of such a system will require at least 2^{37} parameters. This is not feasible. Instead, we will use a Bayesian network to represent such systems (figure 8.3).

Following is the list of independencies of this network:

- $I(E, B)$ - Given the parents of E (none), E is conditionally independent of its non-descendant: B.
- $I(B, (E, R))$ - Given the parents of B (none), B is conditionally independent of its non-descendants: E and R.
- $I(R, (B, A, C) \mid E)$ - Given the parents of R (E), R is conditionally independent of its non-descendants: B, A and C.
- $I(A, R \mid (B, E))$ - Given the parents of A (B, E), A is conditionally independent of its non-descendant: R.
- $I(C, (R, B, E) \mid A)$ - Given the parents of C (A), C is conditionally independent of its non-descendants: R, B and E.

Consider the first independence, $I(E, B)$: E has no parents, so in any case it is independent of B. Indeed there is no dependence between the events of a burglary and an earthquake. The last independence, $I(C, (R, B, E) \mid A)$ means that if the parents of C, which is A, is known then C is independent of R, E and B. This correlates with the rational of the events: if it is known that the alarm broke off (A), then whether the neighbor will or will not call (C) because of it, does not depend on the radio report (R), the earthquake

(E) or the burglary (B). This is true despite the fact that the earthquake and the burglary might be the reason for the alarm.

According to the chain rule (equation 8.8), the joint distribution is:

$$P(C, A, R, E, B) = P(C \mid A, R, E, B) \cdot P(A \mid R, E, B) \cdot P(R \mid E, B) \cdot P(E \mid B) \cdot P(B)$$

Which requires 31 parameters. Alternatively, using independencies in the Bayesian network (equation 8.10), the joint distribution of the five events is

$$P(C, A, R, E, B) = P(C \mid A) \cdot P(A \mid B, E) \cdot P(R \mid E) \cdot P(E) \cdot P(B)$$

This distribution requires only 10 independent parameters.

Complexity Analysis

To fully specify a joint distribution, we need to specify the conditional probabilities in the product form. The quantitative part of the Bayesian network describes these conditional distributions, $P(X_i \mid \mathbf{Pa}(X_i))$ for each variable X_i. Suppose the variables have k possible values (If the variables were binomial then $k = 2$). Each one of these conditional distributions contains $k^{|\mathbf{Pa}(X_i)|}$ independent parameters. Let l be the maximum number of possible parents, then for each variable X_i, there are $\leq k^l$ parameters to be stored. Hence for n variables, the representation complexity is $O(n \cdot k^l)$. Note that for a joint distributions with many variables, each of which has only few dependencies, then $l \ll n$ and the representation complexity is much better then the $O(k^n)$ representation complexity in the joint distributions representation.

8.2.2 Representing Markov chains and HMM as Bayesian networks

A Markov chain can be treated as a simple case of a Bayesian network. A Markov chain is a

Figure 8.4: A Markov chain represented as a Bayesian network

triplet Q, p, A, where Q is a set of states, p is the probabilities of the initial state, and A is the state transition probabilities, which contains for each two states $s, t \in Q$ the probability $a_{st} \equiv P(x_i = t \mid x_{i-1} = s)$. $X = (x_1, \ldots, x_n)$ is a random process (for additional details refer to HMM scribe 5). An equivalent Bayesian network will be a chain of variables $X_1, \ldots, X_n$ (figure 8.4), thus X_{i-1} is the parent of X_i. Each variable may attain any value $s \in Q$. The

CPD of X_1 will be p. The CPD of variable X_i is $P(X_i \mid X_{i-1}) = a_{st}$, i.e. the probability of observing value s in variable X_i is dependent on the value in it's parent X_{i-1} in the Bayesian network. Therefore computing $P(X) = P(x_1) \cdot \prod_{i=2}^{L} a_{x_{i-1}x_i}$ using Markov chain, is equivalent to the factorial representation of the joint distribution in the Bayesian network: $P(X) = P(X_1 = x_1) \cdot \prod_{i=2}^{L} P(X_i = x_i \mid X_{i-1} = x_{i-1})$.

Hidden Markov model (HMM), can also be represented using Bayesian networks. In addition to the Markov chain, it contains emission probabilities, which will be represented as additional nodes (Y_i) in the Bayesian network (figure 8.5). The emission node will have no descendants, as nothing depends on them. Each variable Y_i may attain any value $y_i \in \Sigma$. The CPD of Y_i will be represented as the emission probability, i.e. $P(Y_i = y_i \mid X_i = x_i) = e_{x_i}(y_i)$. The CPD of each variable X_i remains the same as in the Markov chain model.

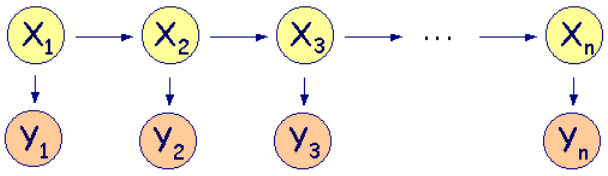

Figure 8.5: A HMM as a Bayesian network

As demonstrates, it's possible to represent HMM using Bayesian network, but this representation is wasteful, since a_{kl} and $e_k(b)$ appear identically n time, once in each node. In the CpG problem, for example, locating CpG islands in a sequence of 2,000 nucleotides, will require an HMM with two states, but the Bayesian network of the same problem will contain 2,000 nodes representing the probabilities of the states in each one of the nucleotides, and another 2,000 for the emissions probabilities. Therefore for random processes with memory of length 1, it will be easier to use HMM rather then Bayesian networks. However, in HMM each state depends only on the state before it, and more complicated dependencies cannot be expressed. As the Bayesian network model generalizes, and represents high-order dependencies, it should be used in cases of complicated dependencies.

8.3 Inference in Bayesian networks

8.3.1 Introduction to Inference

So far it has been shown that Bayesian networks can be used to represent probability distributions in a compact and intuitive manner. As it is so, Bayesian networks should contain information that would answer any query about the distributions represented by them. This section is devoted to the way such queries can be answered, using the data available by the

network. Given a Bayesian network for the variables $X_1, \ldots, X_n$, frequent types of queries we are interested in are $P(x_i), P(x_i \mid X_j = x_j), P(x_1, \ldots, x_n \mid X_j = x_j)$ etc.

Bayesian networks can be used to answer these queries, since the joint distributions can be generated using the chain rule for Bayesian networks (equation 8.10). The joint distribution can then be used to make the necessary calculations for the query. A naïve solution to compute $P(X_i)$ based on the total probability theorem (equation 8.6), is $P(X_i) = \sum_{x_1} \cdots \sum_{x_{i-1}} \cdot \sum_{x_{i+1}} \cdots \sum_{x_n} P(X_1, \ldots, X_n)$ Using the joint distribution representation, for binomial variables, the complexity of solving this is exponential ($O(2^n)$). For variables with up to k possible value, it is $O(k^n)$. Avoiding exponential complexity, was one of the main reasons to use Bayesian networks, and in the following section it will be demonstrated how the use of them might reduce the complexity. The inference problem in Bayesian networks is NP-hard. It means that in the *worst case* the best algorithm available to solve these problems is exponential (assuming $P \neq NP$). In other words - performing the computations on all the joint distribution entries is the best that can be done in worst case. But it turns out that the worst case comes up only very rarely. So, although the complexity of Bayesian networks algorithms in *worst case* is exponential, in practice, a simple method can be used to make the computations in most practical cases quite fast. This technique is called *Variable elimination* algorithm.

8.3.2 Variable Elimination algorithm

The Basic Idea

The idea of eliminating variables will be demonstrated through a basic inference task on a simple Bayesian network. Consider the "chain" network in figure 8.6. Suppose we want to

Figure 8.6: Simple Bayesian network in form of a nodes chain

calculate $P(A = a, D = d)$, the distribution of the joint probability of A and D: Using the total probability equation (8.6), we get:

$$P(a, d) = \sum_b \sum_c P(a, b, c, d)$$

Note that each addition is used to eliminate one variable: Summing for all the probabilities of c on the joint distribution, $P(a, b, c, d)$, will yield $P(a, b, d)$, thus eliminating C. In the same manner summing up all the probabilities of b on $P(a, b, d)$ will eliminate b, yielding the

requested probability $P(a,d)$. Using the factorial representation, instead of the joint one, will give us:

$$P(a,d) = \sum_b \sum_c P(a,b,c,d) = \sum_b \sum_c P(d \mid c)P(c \mid b)P(b \mid a)P(a)$$

This representation embodies the power of the Bayesian network, as it uses the conditional probabilities instead of the joint, and these will be used to simplify the additions and reduce the complexity.
First, c will be eliminated. Note that when summing for all $C = c$ on the conditional probabilities, some terms need not to be summed, as their CPDs do not contain the variable C. That means we can omit these terms from the sum on c, thus reducing the complexity, as less calculation has to be performed to accomplish that sum:

$$P(a,d) = \sum_b \sum_c P(d \mid c)P(c \mid b)\overbrace{P(b \mid a)P(a)}^{\text{No } c \text{ in CPDs}} = \sum_b P(b \mid a)P(a) \sum_c P(d \mid c)P(c \mid b)$$

Now, in order to complete the elimination of c, we will use the chain rule (equation 8.8), applying $\sum_c P(d \mid c)P(c \mid b) = P(d \mid b)$:

$$P(a,d) = \sum_b P(b \mid a)P(a)\overbrace{\sum_c P(d \mid c)P(c \mid b)}^{P(d|b)} = \sum_b P(b \mid a)P(a)P(d \mid b)$$

c was eliminated from this equation, and the factor of $P(d \mid b)$ was calculated. $P(d \mid b)$ is an *intermediate factor* of the algorithm. The next step will be to eliminate B. As before, this will be done by summing on every $B = b$, after excluding the terms that do not include b (in this case $P(a)$) from the addition.

$$P(a,d) = \sum_b P(b \mid a)\overbrace{P(a)}^{\leftarrow} P(d \mid b) = P(a)\overbrace{\sum_b P(d \mid b)P(b \mid a)}^{P(d|a)} = P(d \mid a)P(a)$$

b was eliminated from this equation, and the intermediate factor $P(d \mid a)$ was calculated. Next, in order to compute $P(a,d)$, we only need to multiply $P(a)$ by the intermediate factor $P(d \mid a)$, which was already calculated.

Generalization

The technique presented can be generalized to solve any specific distribution of a variable, or joint distribution of a subset of variables: Let $P(x_1)$ be the distribution we're looking for

in a network of n variables. First the query will be written in the following form:

$$P(x_1) = \sum_{x_n} \cdots \sum_{x_3} \sum_{x_2} \prod_i P(x_i \mid pa_i)$$

Where pa_i are the parents of x_i in the Bayesian network. In the last example, $\prod_i P(x_i \mid pa_i) = P(d \mid c)P(c \mid b)P(b \mid a)P(a)$, After having the query in that form, perform iteratively:

- Move all irrelevant terms outside of innermost sum.
- Perform innermost sum, getting a new intermediate factor.
- Insert the intermediate factor into the product

So far we've seen how to answer queries about the distribution of a variable, or the joint distribution of few variables. Consider a conditional probability query, let X_1 be the query variable given $X_k = x_k$, so the distribution of $P(X_1 \mid x_k)$ is requested. The query will be managed in a similar way as in the previous case: First $P(X_1, x_k)$ and $P(x_k)$ will be calculated, as follows. Note that for $P(X_1, x_k)$ we sum neither on X_1 nor on x_k.

$$P(X_1, x_k) = \sum_{x_n} \cdots \sum_{x_{k+1}} \sum_{x_{k-1}} \cdots \sum_{x_2} \prod_{i=1}^{n} P(x_i \mid pa_i)$$

$$P(X_k = x_k) = \sum_{x_n} \cdots \sum_{x_{k+1}} \sum_{x_{k-1}} \cdots \sum_{x_1} \prod_{i=1}^{n} P(x_i \mid pa_i)$$

Next $P(X_1 \mid X_k)$ is found, using the product rule (equation 8.1): $P(X_1 \mid x_k) = \dfrac{P(X_1, x_k)}{P(X_k = x_k)}$.

Complexity Analysis

Let's observe the complexity of solving the queries presented. As mention in section 8.3.1, solving such queries in a Bayesian network is NP-hard. Therefore, in the worst case, time complexity is expected to be exponential. But most practical cases do not yield such networks. In the following calculations the variables will be treated at first as binomial, to simplify the calculations. In order to find the distribution of X_1, the following expression needs to be calculated: $P(x_1) = \sum_{x_n} \cdots \sum_{x_3} \sum_{x_2} P(x_2, x_3, \ldots, x_n)$.
In the naïve case, all the additions will be performed on the joint distribution. Since we sum over all the possible combination of $x_1, \ldots, x_n$, this will cost $O(2^n)$. In the multinomial case, where X_i has k possible values, the complexity will be $O(k^n)$.

We've seen that time complexity might be reduced, if the factorial representation is used and each sum is done only on the relevant factors. In this method we start with the

expression $P(x_1) = \sum_{x_n} \cdots \sum_{x_3} \sum_{x_2} \prod_i P(x_i \mid pa_i)$, and then perform $O(n)$ additions, each on its relevant factors. Let's observe the complexity of the X_i-th summation: We sum only on the relevant factors of the product, let r_i be the number of variables in the intermediate factor generated in summation of x_i. For example, summing $\sum_c P(A \mid c) \cdot P(c \mid b, d)$ will generate the intermediate factor $P(A \mid b, d)$ with $r = 3$. In the intermediate factor there are $O(2^{r_i})$ possible combinations, and each one is calculated by multiplying $O(r_i)$ factors. Therefore summing X_i will cost $O(r_i \cdot 2^{r_i})$. Let $r = \max_{i=1}^{n}\{r_i\}$. Summing $O(n)$ variables will cost $O(n \cdot r \cdot 2^r)$. In the multinomial case, the complexity will be $O(n \cdot r \cdot k^r)$. Note that this is significantly smaller than $O(k^n)$ only if each variable has a small number of parents in the network (few parents means few dependencies, which will give factors with few variables, hence a low r).

Finally note that the number of factors, and hence the complexity of the elimination, strongly depend on the order of elimination. Even in the simple elimination example on the chain network described in figure (8.6), containing only the four variables A, B, C and D this fact is manifested. consider the following order of elimination:

$$\begin{aligned}
P(d) &= \sum_c \sum_b \sum_a P(a,b,c,d) = \sum_c \sum_b \sum_a P(d \mid c)P(c \mid b)P(b \mid a)P(a) \\
&= \sum_c \sum_b P(d \mid c)P(c \mid b) \overbrace{\sum_a P(b \mid a)P(a)}^{P(b)} = \sum_c \sum_b P(d \mid c)P(c \mid b)P(b) \\
&= \sum_c P(d \mid c) \overbrace{\sum_b P(c \mid b)P(b)}^{P(c)} = \sum_c P(d \mid c)P(c) = P(d)
\end{aligned}$$

Each one of the products being summed contains 1 factor at most: Just one variable except the one that we sum upon. Alternatively, consider the following order of elimination:

$$\begin{aligned}
P(d) &= \sum_a \sum_b \sum_c P(a,b,c,d) = \sum_a \sum_b \overbrace{\sum_c P(d \mid c)P(c \mid b)}^{P(d|b)} P(b \mid a)P(a) \\
&= \sum_a \overbrace{\sum_b P(d \mid b)P(b \mid a)}^{P(d|a)} P(a) = \sum_a P(d \mid a)P(a) = P(d)
\end{aligned}$$

In this elimination all the products except the last contains 2 factors, hence the complexity of this elimination is greater, since the order of elimination wasn't optimal. Choosing an efficient order of elimination can be done using a *clique tree algorithm*, which will not be discussed here.

8.4 Learning Bayesian Networks

8.4.1 Introduction [5]

The amount of available information is growing rapidly. However, knowledge acquisition is an expensive process. Learning allows us to construct models from raw data, which can then be used for many tasks. Bayesian networks in particular, which convey conditional independencies, enable us to capture the structure of many real-world distributions.

8.4.2 The Learning Problem

The problem of learning a Bayesian network can be stated as follows:
Let m be the number of samples and n the number of variables. Given a training set $D = (X_1, \ldots, X_m)$, where $X_i = (x_{i1}, \ldots, x_{in})$, and prior information, find a network that *best matches* D. The problem of learning a Bayesian network can be learning the CPD (the parameters) given a certain structure, as in figure 8.7, or learning both the distributions and the graph's structure (the dependencies), as in figure 8.8. In addition, the learning problem can be divided into two other cases: complete data and incomplete data. In complete data, all parameter values are known, whereas in the case of incomplete data, some of the values in the vectors $X_i = (x_{i1}, \ldots, x_{in})$ are missing.
The means of handling these aforementioned cases, are described in table 8.7.

	Known Structure	**Unknown Structure**
Complete Data- All instances of the variables are known	Statistical parametric estimation (closed-form equations)	Discrete optimization over structures (discrete search)
Incomplete Data- Not all instances of the variables are known	Parametric optimization (EM, gradient descent...)	Combined (structural EM, mixture models...)

Table 8.7: Means of learning the network in different cases

We will focus on complete data for the rest of the talk. We start with the case of a known structure, whose parameters we wish to learn and estimate.

8.4.3 The Likelihood Method

One way of finding an estimator for a parameter is the likelihood method. According to this method, given a sequence of samples, we look for the parameter's value that seems most

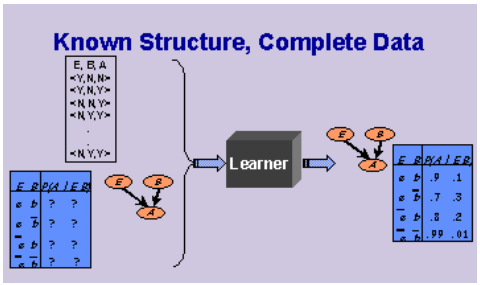

Figure 8.7: Learning from a known structure and complete data

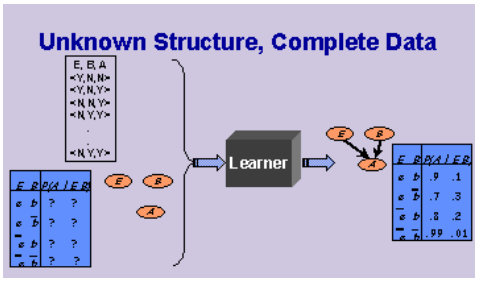

Figure 8.8: Learning from an unknown structure and complete data

"likely" to have caused this certain sequence of samples (i.e. gives our sample the highest probability compared to all other possible values of the parameter). In other words, we try to deduce the value of the parameter **after** a certain sample has been received, namely the value of the parameter that best explains the given sample results.

Likelihood Definitions

- **The Likelihood Function** [7, 6]
  Let $X_1, X_2, \ldots, X_m$ be samples from a population with a certain distribution with an unknown parameter θ and let $X_i = x[i]$ be the training data- the sample results obtained, each result independent of the others. The likelihood function is defined as:

$$L(\theta : D) = P(D \mid \theta) = P(X_1 = x[1], X_2 = x[2], \ldots, X_m = x[m] \mid \theta) = \prod_{i=1}^{m} P(x[i] \mid \theta)$$

  where θ gets all the possible values of the parameter.

- **Maximum Likelihood Estimator**
  Given a sample result, $\hat{\theta}$ is the maximum likelihood estimator (MLE) for the parameter θ if it holds that $\forall\theta.L(\hat{\theta}) \geq L(\theta)$. In other words, $\hat{\theta}$ is the value of the parameter θ which maximizes the likelihood function.

8.4.4 Example: Binomial Experiment

A coin can land on one of two positions: H or T, each with an unknown probability. We denote by θ the unknown probability $P(H)$.
Estimation Task: given a sequence of samples $x[1], x[2], \ldots, x[m]$ we want to estimate the probability $P(H) = \theta$ and $P(T) = 1 - \theta$.
Suppose we performed six independent Bernolli experiments and received the following sample results: H, H, H, H, T, T. We denote by X_i the result of the i-th coin toss. The likelihood function for the obtained results is:

$$L(\theta : D) = P(D \mid \theta) = P(X_1 = H, X_2 = H, X_3 = H, X_4 = H, X_5 = T, X_6 = T \mid \theta) \overset{(*)}{=}$$

$$= P_\theta(X_1 = H)P_\theta(X_2 = H)P_\theta(X_3 = H)P_\theta(X_4 = H)P_\theta(X_5 = T)P_\theta(X_6 = T) =$$

$$= \theta^4(1 - \theta)^2$$

where (*) is because of independent coin tosses.
Note the following:

- For a known θ, we would have an exact numerical value denoting the probability to get the training data results.
- The coefficient $\binom{n}{k}$ in the binomial distribution is omitted here, since we have the exact sample sequence and for each result in the sequence we are interested only in the probability of obtaining that certain result.
  In addition, even if we write the coefficient, it will eventually "disappear" from the equation, as explained in the steps to come.

We can formulate the following general case:
Let N_H be the number of occurrences of H and N_T be the number of occurrences of T in the training set, so that $N_H + N_T = m$.

$$L(\theta : D) = P(D \mid \theta) = {\theta_H}^{N_H} \cdot {\theta_T}^{N_T}$$

where θ_H is the probability of the outcome H and θ_T is the probability of the outcome T.

As described earlier, the MLE principle is to choose the value of the parameter that maximizes the likelihood function. Applying the MLE principle means we have to find the maximum point of the likelihood function. In order to do this, we will first switch to the log (or ln) of the likelihood function. This is done in order to make the derivative calculations easier (instead of a complicated derivative of a long product term the log turns the product into summation, thus making the calculation of the derivative simpler). The logarithm is a monotonically increasing function and therefore the maximum point of the function $L(\theta)$ is the same as the maximum point of the function $\log L(\theta)$.

Returning to finding the maximum point of our likelihood function, we get:

$$\ln L(\theta) = 4\ln\theta + 2\ln(1-\theta)$$

$$\frac{d\ln L(\theta)}{d\theta} = \frac{4}{\theta} - \frac{2}{1-\theta}$$

$$\frac{4}{\theta} - \frac{2}{1-\theta} = 0$$

$$\hat{\theta} = 2/3$$

or in the general case:

$$\hat{\theta} = \frac{N_H}{N_H + N_T}$$

We can now see that had we written the coefficient, it would have been omitted anyway, after applying the log and finding the derivative.
Using the above general equation, we can easily find the MLE estimation for other cases. For example, given $(N_H, N_T) = (3, 2)$, the MLE estimation is $\frac{3}{5} = 0.6$.

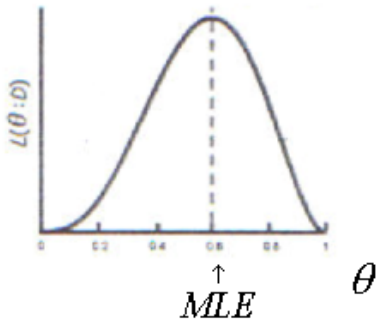

Figure 8.9: The maximum likelihood estimation

8.4.5 Learning Parameters for Bayesian Networks

When learning parameters for Bayesian networks, the training data has a form as in figure 8.10.

$$D = \begin{pmatrix} X_1 \\ \vdots \\ X_m \end{pmatrix} = \begin{pmatrix} x_{11} & \cdots & x_{1n} \\ \vdots & & \vdots \\ x_{m1} & \cdots & x_{mn} \end{pmatrix} = \begin{pmatrix} E[1] & B[1] & A[1] & C[1] \\ \vdots & \vdots & \vdots & \vdots \\ E[m] & B[m] & A[m] & C[m] \end{pmatrix}$$

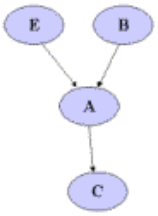

Figure 8.10: The training data, including the network structure

Since we assume i.i.d (independent identically distributed) samples, the likelihood function is:

$$L(\theta : D) = \prod_m P(E[m], B[m], A[m], C[m] : \theta) \overset{(*)}{=} \underbrace{\prod_m \begin{pmatrix} P(E[m] : \theta) \cdot \\ P(B[m] : \theta) \cdot \\ P(A[m] \mid B[m], E[m] : \theta) \cdot \\ P(C[m] \mid A[m] : \theta) \end{pmatrix}}_{\text{the factorial representation}}$$

where (*) is due to switching to the factorial representation, by definition of the network in figure 8.10. Rewriting the terms (looking at the columns of the training data instead of the rows) we get:

$$L(\theta : D) = \prod_m P(E[m] : \theta) \prod_m P(B[m] : \theta) \prod_m P(A[m]|B[m], C[m] : \theta) \prod_m P(C[m]|A[m] : \theta)$$

As can be seen from the calculation above, we have gone from the product of the probabilities of all nodes for each m, to the product of the likelihood of each node (given its parents, if it has any).
Generalizing for any Bayesian network we get:

$$L(\theta : D) = \prod_m P(x_1[m], \ldots, x_n[m] : \theta) \underbrace{=}_{\text{network factorization}}$$

$$= \prod_i \prod_m P(x_i[m] \mid Pa_i[m] : \theta_i)$$
$$= \prod_i L_i(\theta_i : D)$$

where Pa_i is the set of node X_i's parents.
We can see here the **decomposition principle**: the likelihood decomposes according to the structure of the network meaning that we can calculate each node's log-likelihood separately

and sum them all to get the log-likelihood of the entire network. The decomposition enables us to have independent estimation problems. In other words, since the parameters for each variable are not dependent, they can be estimated independently of each other. As a result, we can use the MLE formula we already described and get $\hat{\theta}_{x_i|pa_i} = \frac{N_{x_i|pa_i}}{N_{x_i=0|pa_i}+N_{x_i=1|pa_i}}$ where pa_i is a certain combination of the parents (for example, see figure 8.11).

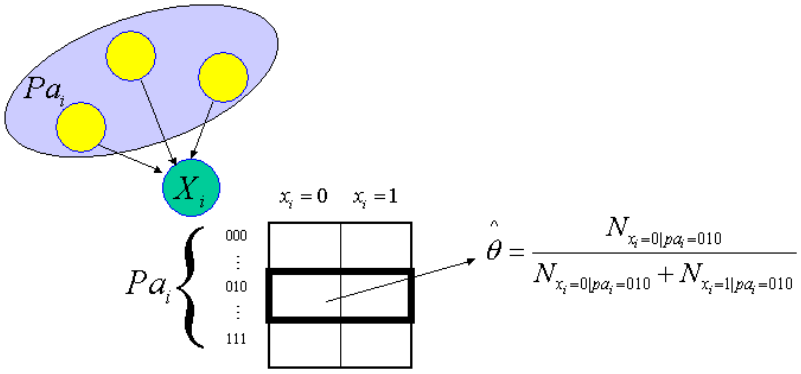

Figure 8.11: Demonstrating the decomposability principle- the variable X_i has three parents, which are the members of the set Pa_i. Each row of the variable's CPD is a different combination of the parents' values and the columns are the different values X_i can receive. Each row is X_i's distribution given the parents' specific combination (the probabilities in each row sum up to 1) and the parameter is estimated according to the formula for $\hat{\theta}$. In this figure, we wish to estimate the value of $\theta = P(x_i = 0)$ for a given parental combination 010.

8.4.6 Likelihood Function: Multinomials

Up to now we dealt with binomial variables that had either the value 0 or the value 1 ("failure" or "success"). We now make the transition to the multinomial case.
Suppose our variable X can have the values $1, 2, \ldots, k$. We would like to learn the parameters $\theta_1, \ldots, \theta_k$ (for example each θ_i corresponds to the probability of getting a specific number on a die). We denote by $N_1, \ldots, N_k$ the number of times each outcome is observed. We get the following likelihood function:

$$L(\theta : D) = \prod_{j=1}^{k} \theta_j^{N_j}$$

where θ_j is the probability of the outcome j and N_j is the number of times the outcome j is observed in D.
As in the binomial case, we have once again omitted the coefficient (this time $\frac{N!}{N_1!N_2!,\ldots,N_k!}$, $N = N_1 + \ldots + N_k$) for it "falls off" after the log likelihood function is derived.

In the same way as before, we get that the MLE is

$$\hat{\theta}_j = \frac{N_j}{\sum_{l=1}^{k} N_l}$$

In Bayesian networks, when we assume that $P(x_i \mid pa_i)$ is multinomial, we get further decomposition. From equation 8.11 we get:

$$L(\theta_i : D) = \prod_m P(x_i[m] \mid Pa_i[m] : \theta_i)$$

We now write the above product expression in a slightly different way, by grouping products according to a certain combination of the parents' values (for example, in the network depicted in figure 8.10, a combination of A's parents E and B may be $E = 1$ and $B = 2$). We once again denote by pa_i each combination of the parents and go over all such combinations. Now, for each pa_i we multiply only observations where $Pa_i[m] = pa_i$. Thus,

$$= \prod_{pa_i} \prod_{m:Pa_i[m]=pa_i} P(x_i[m] \mid pa_i : \theta_i)$$

Looking at the second product expression, we now "group" it even further- for each combination of the parents, we take a certain value of x_i and raise its probability to the power of that certain value's number of appearances:

$$= \prod_{pa_i} \prod_{x_i} P(x_i \mid pa_i : \theta_i)^{N(x_i,pa_i)} = \prod_{pa_i} \prod_{x_i} {\theta_{x_i|pa_i}}^{N(x_i,pa_i)}$$

For each combination pa_i of the parents of x_i we get an independent likelihood function $\prod_{x_i} {\theta_{x_i|pa_i}}^{N(x_i,pa_i)}$. After performing log on the last expression and finding its derivative we get that the MLE is:

$$\hat{\theta}_{x_i|pa_i} = \frac{N(x_i, pa_i)}{N(pa_i)}$$

This is essentially as in figure 8.11, but for the multinomial case.

8.4.7 Bayesian Learning

Unlike the MLE approach, the Bayesian approach doesn't estimate the most likely θ, but rather takes into account all possible θs with their probabilities. We can represent our uncertainty about the sampling process using a Bayesian network, as seen in figure 8.12.
All the data are dependent on the certain θ that "created" them. As seen in the Markov

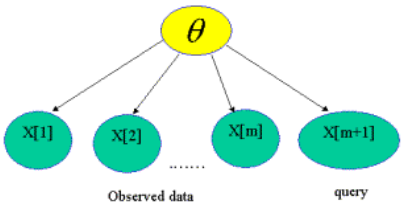

Figure 8.12: Representing Bayesian prediction as inference in Bayesian network

assumption, given θ (the parent) the values of the different $x[i]$ are independent.
The Bayesian prediction is inference in this network:

$$P(x[m+1] \mid x[1],\ldots,x[m]) \overset{(*)}{=} \int_0^1 P(x[m+1] \mid \theta, x[1],\ldots,x[m])P(\theta \mid x[1],\ldots,x[m])d\theta$$

where (*) is because of the conditional probability and total probability theorem for a continuous variable θ ($0 \leq \theta \leq 1$). Since the first probability term in the integral is for a given θ, $x[m+1]$ is independent of its non-descendants and we can therefore write:

$$= \int_0^1 P(x[m+1] \mid \theta)P(\theta \mid x[1],\ldots,x[m])d\theta \tag{8.11}$$

Now let us look at the second term of the integral.

$$\underbrace{P(\theta \mid x[1],\ldots,x[m])}_{\text{posterior}} \underbrace{=}_{\text{Bayes rule}} \frac{\overbrace{P(x[1],\ldots,x[m] \mid \theta)}^{\text{likelihood}}\overbrace{P(\theta)}^{\text{prior}}}{\underbrace{P(x[1],\ldots,x[m])}_{\text{probability of data}}} \tag{8.12}$$

The **prior** is some earlier information we have regarding the random variable θ.
Using the total probability theorem, we may rewrite the denominator and get:

$$P(x[1],\ldots,x[m]) = \int_0^1 P(x[1],\ldots,x[m] \mid \theta)P(\theta)d\theta$$

and therefore the expression on the right hand side of equation 8.12 can be written as

$$\frac{P(x[1],\ldots,x[m] \mid \theta)P(\theta)}{\int_0^1 P(x[1],\ldots,x[m] \mid \theta)P(\theta)d\theta}$$

Note that $P(x[1],\ldots,x[m])$ does not depend on θ (it goes over all values of θ in its equivalent integral expression). As a result, it behaves as a constant in formula 8.11 and can be taken out of the integral there (see the example in the following section, 8.4.8).

8.4.8 Binomial Example: MLE vs. Bayesian Learning

We return to our coin toss example. Recall that $P(H) = \theta$. This time, however, we also have prior information that the distribution of θ is uniform in $[0, 1]$. Therefore, $P(\theta) = f(\theta) = 1$ (no substantial information about θ). In addition, the data is $(N_H, N_T) = (4, 1)$ (there are four heads and one tail).
According to the MLE general case found earlier, the MLE for $P(X = H)$ is $\frac{N_H}{N_H+N_T} = \frac{4}{5} = 0.8$.
The Bayesian prediction is:

$$P(x[m+1] = H \mid D) = \int_0^1 P(x[m+1] = H \mid \theta) P(\theta \mid x[1], \ldots, x[m]) d\theta =$$

$$= \int_0^1 P(x[m+1] = H \mid \theta) \frac{P(\theta) P(x[1], \ldots, x[m] \mid \theta)}{P(x[1], \ldots, x[m])} d\theta =$$

$$= \int_0^1 \frac{P(x[m+1] = H \mid \theta) P(\theta) P(x[1], \ldots, x[m] \mid \theta) d\theta}{\int_0^1 P(\theta) P(x[1], \ldots, x[m] \mid \theta) d\theta} =$$

$$= \frac{\int_0^1 \theta \cdot 1 \cdot \theta^{N_H} (1-\theta)^{N_T} d\theta}{\int_0^1 1 \cdot \theta^{N_H} (1-\theta)^{N_T} d\theta} = \frac{\int_0^1 \theta \cdot \theta^4 \cdot (1-\theta) d\theta}{\int_0^1 \theta^4 \cdot (1-\theta) d\theta} =$$

$$= \frac{\int_0^1 (\theta^5 - \theta^6) d\theta}{\int_0^1 (\theta^4 - \theta^5) d\theta} = \frac{\frac{1}{6} - \frac{1}{7}}{\frac{1}{5} - \frac{1}{6}} = \frac{5}{7} = 0.7142$$

Since our earlier knowledge was θ's uniform distribution, we were not "tempted" to believe, after only 5 tosses, that $P(H)$ is as extreme as 0.8, but rather slightly lower, toward the uniform 0.5. We can see that the prior has given us a more "balanced" result. It seems as though we have added two more tosses, one of which was H, namely the prior has given us a more established result, based on our earlier knowledge.

The differences between the MLE approach and the Bayesian approach are summarized in table 8.8.

8.4.9 Dirichlet Priors

In example 8.4.8, we saw an example of a uniform prior in $[0, 1]$, and thus $P(\theta) = f(\theta) = 1$. Let us now look at the following density function (recall that θ is a random variable), called **Dirichlet's function**:

$$P(\theta) = \frac{(\sum_{j=1}^k \alpha_j - 1)!}{(\alpha_1 - 1)!(\alpha_2 - 1)! \ldots (\alpha_k - 1)!} \cdot \prod_{j=1}^k \theta_j{}^{\alpha_j - 1}$$

MLE approach	**Bayesian approach**
Assume that there is an unknown but fixed parameter θ.	Represents uncertainty about the unknown parameter θ.
Estimate θ with some confidence.	Uses probability to quantify this uncertainty. Unknown parameters as random variables.
Prediction using the estimated parameter value.	Prediction follows from the rules of probability- expectation over the unknown parameters.

Table 8.8: The differences between the MLE and Bayesian approaches

This is the density function of a k-dimensional random variable θ with hyperparameters $\alpha_1, \ldots, \alpha_k$. If we know these hyperparameters, we will know the density function explicitly as a function of $\theta_1, \ldots, \theta_k$. Note that we can "drop" the coefficient (it is a normalization constant) and get:

$$P(\theta) \propto \prod_{j=1}^{k} {\theta_j}^{\alpha_j - 1}$$

We now use the Dirichlet distribution as a prior. Recall that the likelihood function for multinomial data is:

$$L(\theta : D) = P(D \mid \theta) = \prod_{j=1}^{k} {\theta_j}^{N_j}$$

From equation 8.12 we get:

$$P(\theta \mid D) \overset{(*)}{\propto} P(\theta)P(D \mid \theta) \propto \prod_{j=1}^{k} {\theta_j}^{\alpha_j - 1} \prod_{j=1}^{k} {\theta_j}^{N_j} = \prod_{j=1}^{k} {\theta_j}^{\alpha_j + N_j - 1} \tag{8.13}$$

We can see that the posterior density function $P(\theta \mid D)$ has the same form as the prior $P(\theta)$, with hyperparameters $\alpha_j + N_j - 1, \ldots, \alpha_k + N_k - 1$. Therefore we can conclude that Dirichlet is **closed** under multinomial sampling (i.e. both the prior and posterior distributions are Dirichlet). Note that in the transition marked (*) we disregarded the denominator $P(D)$ from equation 8.12. This denominator and the normalization constant, which was "dropped" from Dirichlet's function, give us the normalization constant of our new Dirichlet distribution. Since the posterior is Dirichlet, we get:

$$P(x[m+1] = j \mid D) \overset{(*)}{=} \int \theta_j \cdot P(\theta \mid D) d\theta \overset{(**)}{=} \frac{\alpha_j + N_j}{\sum_l (\alpha_l + N_l)}$$

where (*) is from equation 8.11 and (**) is the result of solving the integral with $P(\theta \mid D)$, which appears in equation 8.13. We can learn from the above expressions that α indicates

the "degree of influence" the prior has on the posterior result. The larger α is, the greater the effect the prior has, because it means we have simulated a situation with more experiments (for each j we have $\alpha_j + N_j$ experiments instead of the N_j we had in the likelihood function) and our "added" experiments α_j have a greater part out of the whole $\alpha_j + N_j$ experiments. We can now generalize:

$$P(x_i[m+1] = j \mid pa_i, D) = \frac{\alpha(x_i = j, pa_i) + N(x_i = j, pa_i)}{\sum_l (\alpha(x_i = l, pa_i) + N(x_i = l, pa_i))} = \frac{\alpha(x_i = j, pa_i) + N(x_i = j, pa_i)}{\alpha(pa_i) + N(pa_i)}$$

Learning Multinomial Parameters: Summary
We count $N(x_i, pa_i)$ and get the following estimations:

$$\underbrace{\hat{\theta}_{x_i|pa_i} = \frac{N(x_i, pa_i)}{N(pa_i)}}_{\text{MLE}} \qquad \underbrace{\hat{\theta}_{x_i|pa_i} = \frac{\alpha(x_i, pa_i) + N(x_i, pa_i)}{\alpha(pa_i) + N(pa_i)}}_{\text{Bayesian (Dirichlet)}}$$

Both can be implemented in an on-line manner by accumulating counts.

8.4.10 Learning Structure

As described in section 8.4.2 (The Learning Problem), given a training set D, our goal is to find a network that best matches D. This time, however, the structure is unknown and we wish to learn it on top of learning the parameters. We therefore describe the following **optimization problem**:
Given an input of:

- Training data
- A scoring function (including priors) to measure our suggested network's match with the data
- Set of possible structures- a set of networks (DAGs, trees, etc) from which we need to find the best structure

we need to output the network (or networks) that maximize the score.
As before, we would like to take advantage of the decomposability property, and therefore we look for a scoring function such that the score of the network will be the sum of the nodes' scores. An example for such a score is the **BDE score**:

$$P(G \mid D) \propto P(D \mid G)P(G) \overset{(*)}{=} P(G) \int P(D \mid G, \theta)P(\theta \mid G)d\theta$$

where (*) holds due to the total probability theorem for $P(D \mid G)$. Following particular assumptions, the score satisfies the decomposability property:

$$Score(G : D) = \sum_i Score(x_i \mid Pa_i^G : D)$$

The problem of finding the optimal structure is NP-hard and we therefore address the problem by using heuristic searching. We traverse the space of possible networks, looking for high-scoring structures. This can be done by using searching techniques, such as:

- Simulated Annealing
- Greedy hill-climbing- we start with a certain structure and we add modifications until we reach a local maximum. In this technique, we save time by not going over all the possibilities.

The typical operations performed in our heuristic search can be seen in figure 8.13- we can add, remove or reverse the direction of an edge, thus creating a new structure, which has a new score we can calculate. Traversing all possible edges and applying the above modifications, we can find the highest-scoring new structure and reiterate the process from it. The number of edge modifications in each step of the greedy hill-climbing algorithm is $O(n^2)$, where n is the number of nodes (because there are $O(n^2)$ edges). Once again, the importance of the decomposability property is demonstrated- each modification of a certain edge is local to the node that particular edge enters. The decomposability enables us to update the score of either only one node (in case of an addition or removal of an edge) or two nodes (in case of a reverse in an edge's direction), without having to recalculate the entire network's score.

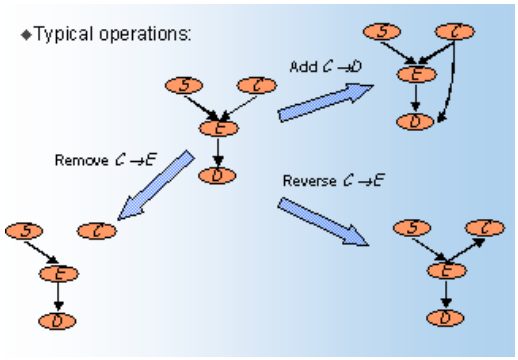

Figure 8.13: The typical operations performed in the heuristic search

8.5 Applications Example: DNA Binding Sites Motifs

8.5.1 Introduction

Our goal in this example is to analyze and predict DNA binding sites.
Representing Sites: Given a collection of known binding sites, which bind a certain transcription factor, we wish to develop a representation of those sites that can be used to search new sequences and reliably predict where additional binding sites occur. So far in the course we have already studied four methods to achieve this:

- **Consensus** sequence- the sequence that represents the collection of binding sites. For more details refer to the multiple alignment scribe, sections 4.2.2 and 4.2.3.
- **Profile-** profiles of binding site motifs are frequently called **PSSM** (Position Site Specific Matrix). For more details refer to the multiple alignment scribe, section 4.4.1.
- **Profile HMM-** generalizes the profile method. For more details refer to the Hidden Markov Models scribe, section 5.2 (Baum-Welch training).
- **Bayesian Networks** (modeling independencies)- this method is unique in modeling dependencies between nodes which are not adjacent to each other.

8.5.2 Representing Sites example

In order to build any model, we need a *Training set*. In our example the training set is a set of sequences which are known to be binding sites, as specified in table 8.9.

Training set
AGACT
AGTCT
ACAGT
ACTGT
AGTCT

Table 8.9: Training set of DNA binding sites

If we have more data than what is used for the training set, that data enables us to estimate the quality of our model. In this example, we know that the genome sequences in table 8.10 are binding sites, and the sequences in table 8.11 are known non-binding sites.

In the following definitions *True* and *False* refer to whether in reality the data is, or is not, a binding site, while *Positive* and *Negative* refer to the **prediction** of the model: whether the model predicts the data to be, or not to be, a binding site.

Additional binding sites
1) AGACT
2) AGTCT
3) ACAGT
4) ACTGT

Table 8.10: Genome sequences, known as binding sites

Non-Binding sites
5) ACTCT
6) AGTGT
7) ACACT
8) AGAGT

Table 8.11: Genome sequences known as non-binding sites.

Definition *Sensitivity* - The ratio of the true positives out of all the true data. Meaning, how much of all the data that is really in the set, was predicted as such by the model.

Definition *Specificity* - The ratio of the true positives out of all the positives. Meaning, how much of the data which was predicted by the model to be in the set, is really in it.

A good model is one that will show high ratios in both indexes.

Consensus Representation

The consensus of the training set (table 8.9) is AGTCT. Searching in the known genome sequences (tables 8.10 and 8.11) allowing no mismatches will give:

- 1 true positive (sequence 2, which is identical to the consensus).
- 3 true negatives (1, 3, 4).
- No false positives.

This is perfect specificity (1), but low sensitivity ($\frac{1}{4}$). On the other hand, searching with the consensus allowing one mismatch will give us the following:

- 2 true positive (1, 2).
- 2 true negatives (3, 4).

- 2 false positives (5, 6).

Which gives us a specificity of $\frac{1}{2}$ and a sensitivity of $\frac{1}{2}$.

PSSM

The PSSM matrix contains the score of each nucleotide in each position, which is derived from their frequency in each location in the training set. A PSSM matrix for the training set is described in table 8.12. Using the PSSM for a high probability search will yield:

	1	**2**	**3**	**4**	**5**
A	1	0	2/5	0	0
G	0	3/5	0	2/5	0
C	0	2/5	0	3/5	0
T	0	0	3/5	0	1

Table 8.12: PSSM matrix of training set in table 8.9

- 4 true positive (all of the known binding sites).
- No true negative.
- 4 false positives (all of the known non-binding sites).

This is a specificity of $\frac{1}{2}$, but a perfect sensitivity (1).

Bayesian Networks

Note that in the sequences of the training set if position 2 is G then position 4 is C, and vice versa: if position 2 is C, then position 4 is G. The reason for the PSSM's low specificity stems from the fact that it cannot represent such dependencies, and so it accepts sequences which have either C or G in both positions. Bayesian networks are able to express such dependencies, and therefore will be ideal for this example. The Bayesian network for this example's training set is presented in figure 8.14.

Running an high probability search on the Bayesian network model will give us:

- 4 true positive (all of the known binding sites).
- No true negative.
- No false positives.

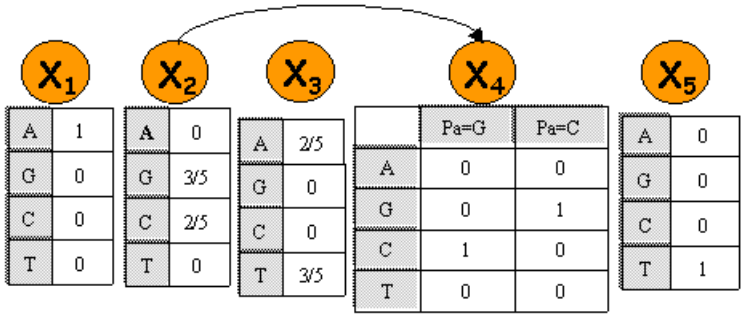

Figure 8.14: Bayesian network produced using the training set on table 8.9. This model is achieved if no priors are used in the learning process (see section 8.4.2)

This is a perfect specificity (1) and perfect sensitivity (1) !

Note that the Bayesian method gave such high results, since in this hypothetical example, dependencies existed between the nucleotides of the binding sites. It is not certain that such dependencies exist in nature as well. The next section will present an attempt to find out if there are dependencies of this kind in nature.

8.5.3 Modeling Dependencies in Protein-DNA Binding Sites

This section describes the application of Bayesian network by Barash *et al.* (2003) [1] for modeling dependencies in protein-DNA binding sites. From the TRANSFAC database [3] they acquired a set of 95 transcription factors, for each at least 20 short binding sites were known. For each transcription factor two models were created, one applying PSSM profile, and one applying Bayesian networks (in order to simplify the process the Bayesian network was restricted to a form of tree). Having the models, the goal is to estimate the generalization of each models created. This is not as simple as demonstrated in section 8.5.2, since no additional binding sites are available. In such cases, where only few sequences are available and are all used for the training set, we can exclude from the training set a small subset of sequences, use the rest of the sequences to create the model, and then test each of the excluded sequences for its probability given the model. Repeating that process for each small subset will give us an indication about the quality of the model. This technique is called *Cross Validation*. Barash *et al.* (2003) [1] used cross validation in the following manner: each one of the binding sites was excluded from the training set, a model was created and then the log likelihood to receive the excluded sequence given the model was calculated. Afterward the average log-likelihood of all the sequences was calculated, and used to evaluate the quality of the model.

One of the transcription factors that were examined by Barash *et al.* (2003) was *Arabidopsis* ABA binding factor 1. 49 examples were used for its modeling using profile and Bayesian networks models. Figure 8.15 is a graphical representation of the profile created. Its

cross validation test average log-likelihood was -19.93. The Bayesian network model (figure 8.16) received a higher average log-likelihood of -18.47. Notice the Bayesian network shows high dependency between positions 5 and 6, this dependency couldn't be expressed in the profile model, and is probably partially responsible for the higher likelihood in the Bayesian model. In order to further check the quality of Bayesian networks vs. profiles, 105 yeast

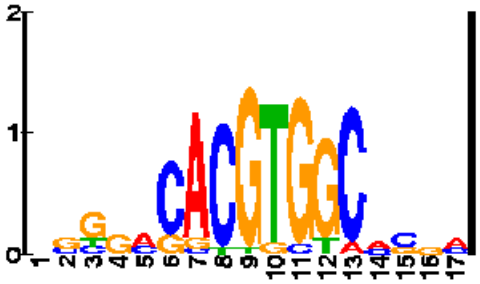

Figure 8.15: Profile of Arabidopsis ABA binding factor 1, based on 49 examples

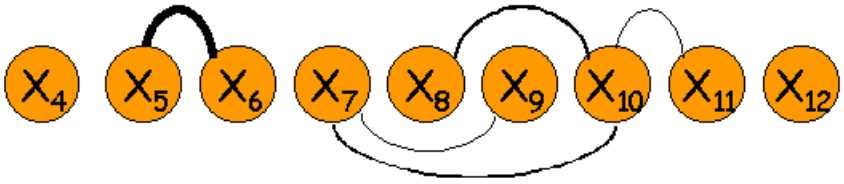

Figure 8.16: Bayesian network for Arabidopsis ABA binding factor 1, based on 49 examples

transcription factors data sets from Lee *et al.* (2002) [2] were used. The data set of each transcription factor included the strength of binding to each genome promoter. Using this information a profile and a Bayesian network were created for each transcription factor, and their sensitivity and specificity were calculated. Figure 8.17 presents the difference between the Bayesian network (tree) and the profile in specificity and in sensitivity. Note that in most cases in which there was a difference between the two models the Bayesian network gave better results. These findings imply that in the transcription factor's binding sites there are indeed certain dependencies between the nucleotides in the different locations.

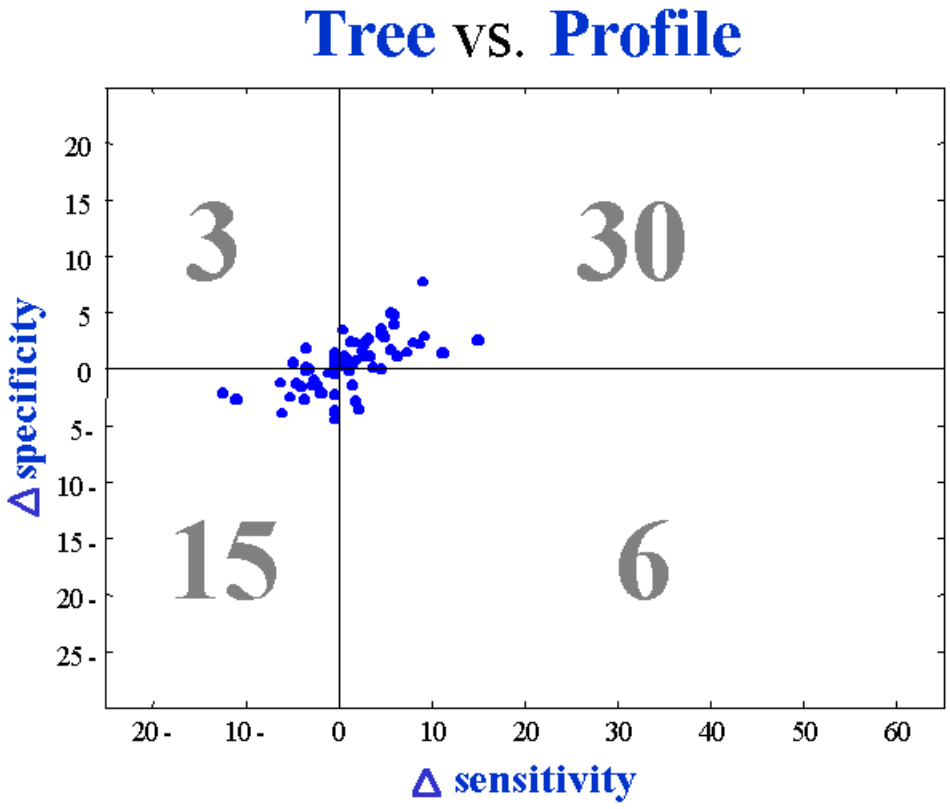

Figure 8.17: Difference between Bayesian tree and profile of yeast binding sites. In 30 out of 54 sites, there is an improvement both in sensitivity and in specificity by using the Bayesian method instead of the profile.

Bibliography

[1] Y. Barash, G. Elidan, N. Friedman, and T. Kaplan. Modeling dependencies in protein-dna binding sites. *RECOMB*, 2003.

[2] TI. Lee et al. Transcriptional regulatory networks in saccharomyces cerevisiae. *Science*, 298:799–804, 2002.

[3] W. Wingender et al. The transfac system on gene expression regulation. *Nuc. Acids Res.*, 29:281–283, 2001.

[4] J. Finn. *An Introduction to Bayesian Networks*. UCL Press, 1996.

[5] N. Friedman and D. Koller. http://www.cs.huji.ac.il/ nir/nips01-tutorial/.

[6] CM. Grinstead and JL. Snell. *Introduction to Probability*. American Mathematical Society, second edition, 1997.

[7] T. Leviatan and A. Raviv. *Introduction to Probability and Statistics- Statistical Inference*. Amichai Publishing House, 1997.

[8] J. Pearl. *Probabilistic Reasoning in Intelligent Systems*. San Mateo CA. Kaufmann, 1988.