Analysis of Gene Expression Data **Spring Semester, 2005**

Lecture 10: June 22, 2007

Lecturer: Irit Gat-Viks *Scribe: Ofer Lavi and Ran Potshter*[1]

10.1 Exploiting Structure in Probability Distributions

10.1.1 Introduction

A major goal in biology 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. The problem is that biological processes are composed of a large number of parameters and representing their probabilistic properties requires an exponential, thus very large, number of variables. In this lecture Bayesian networks [7, 19] will be introduced[2]. These networks take advantage of the local dependence occurring in biological processes in order to represent their probability distribution 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. Bayesian networks also provide us a graphical, intuitive model of causal influence, as will be discussed later.

One advantage in using Bayesian networks is the fact that the statistical foundations for learning Bayesian networks from observations, and computational algorithms to do so are well understood and have been used successfully for 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.

This lecture starts with a probability primer, describing the important definitions and results needed to understand Bayesian networks. Later, Bayesian networks are introduced and an example to their compactness of representation is given. Methods to compute probability distributions based on a Bayesian network are than presented. Once the Bayesian networks were described as a compact way to represent a stochastic (local) system, methods to deduce the structure and relations of a Bayesian network based on experimental data are given. The lecture ends with an example application of Bayesian networks to gene expression analysis.

[1]Based on scribes Uri Avni and Liza Potikha, April 2005, Erez Yaffe and Dan Cohen, June 2004 and Tal Peled and David Burstein, December 2003

[2]This lecture is partially based on presentations of Nir Friedman

10.1.2 A probability primer

Basic notions and notations

In the following lecture we will deal with discrete probabilities defined over a finite *sample space*. The *sample space* Ω is a group containing all of the possible *world states*. A *world state* defines everything that can be known about the world and the value of any measurable parameter. Any biological experiment is unaware of the current world state and can only measure different parameters resulting from it. Each world state has a given probability to occur and this probability may differ from one world state to another.

A random variable X is a parameter that has a given value for each world state (i.e. it is a function on the group Ω). $P(X = x)$ is the marginal probability for X to be measured as x in a random world state. Thus $P(X = x)$ is the sum of probabilities of the world states in which X will be measured to be x. Many times, the name of the random variable is omitted so $P(x)$ will be used instead of $P(X = x)$.

Given a group of values A for a random variable X, we will note the chance that one of these values will be measure for X by $P(X = A)$ or $P(A)$. It is obvious that if $A = x_1...x_n$ then $P(A) = \sum_{i=1}^{n} P(x_i)$. Notice also that if X and Y are completely unrelated, i.e. *independent*, then $P(x, y) = P(x) \cdot P(y)$ and $P(A, B) = P(A) \cdot P(B)$. This independence will be noted $I(X; Y)$ or $I(A; B)$ when A is related to X and B to Y .

Joint and conditional distribution

Given two random variables, X and Y, we would like to know their *joint* distribution $P(X = x, Y = y)$, i.e. the probability that in the same time X is measured to be x and Y is measured to be y. As before, this could be written as $P(x, y)$ even though x and y are values of (possibly) very different parameters. If A and B are groups of possible values for X and Y, respectively, $P(X = A, Y = B)$ or $P(A, B)$ is the probability for X to have a value from A and Y from B. It is obvious that if $A = \{x_1...x_n\}$,$B = \{y_1...y_m\}$ then $P(A, B) = \sum_{i=1,j=1}^{i=n,j=m} P(x_i, y_j)$.

We would like now to know the probability of measuring $X = x$ given that we have already measure Y to be y. It is obvious that the probability may differ from $P(X = x)$. Think of the probability of someone speaking Hebrew if we already knew that he is from Mars. It should be much smaller than the overall probability for him to speak Hebrew (unless there is something we don't know about Mars...). The above probability is the conditional probability $P(X = x|Y = y)$ or $P(x|y)$. It is logical that the probability of $X = x$ and $Y = y$ should be equal for the probability for "$Y = y$" and "$X = x$ given that $Y = y$". These are

completely unrelated variables (even if X and Y are related) so :

$$P(x, y) = P(y)P(x|y) \tag{10.1}$$

and

$$P(A, B) = P(B)P(A|B) \tag{10.2}$$

We can interchange X and Y and get $P(A, B) = P(A) \cdot P(B|A)$. From these results the important Bayes rule can be inferred :

$$P(A \mid B) = \frac{P(B \mid A) \cdot P(A)}{P(B)} \tag{10.3}$$

Also notice that if I(A;B), that is X and Y are independent then

$$P(A \mid B) = \frac{P(A, B)}{P(B)} = \frac{P(A) \cdot P(B)}{P(B)} = P(A) \tag{10.4}$$

This makes sense, because X doesn't depend on Y so knowing Y doesn't change the probability distribution of X.

Conditional Independence

It is possible that two random variables X and Y are dependent, but given a value of some other random variable Z they become independent. This is the case when X and Y are both dependent on Z but otherwise unrelated to each other. Once we fix the value of Z, there is no remaining information to create dependence between the distributions of X and Y . But without fixing Z we could infer from the value of X the value of Z and using the value of Z to learn about the possible values of Y so they are dependent. This case is called *conditional independence* and will be marked $I(X; Y \mid Z)$ or $I(A; B \mid C)$. In this case it is obvious that :

$$P(A, B \mid C) = P(A \mid C) \cdot P(B \mid C) \tag{10.5}$$

Or alternatively:

$$P(A \mid B, C) = P(A \mid C), P(B \mid A, C) = P(B \mid C) \tag{10.6}$$

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

The sample space can be partitioned based on the value of one random variable Y. Suppose that B is the set composed of the possible values for Y. B can be partitioned into a disjoint sets B_i so that $\bigcup_{i=1}^{n} B_i = B$. This breaks the sample space accordingly. The total probability theorem states that

$$P(A) = \sum_{i=1}^{n} P(A, B_i) = \sum_{i=1}^{n} P(B_i) \cdot P(A \mid B_i) \tag{10.7}$$

The chain Rule

Using the fact that $P(X_1, \ldots, X_n) = P(X_1 \mid X_2, \ldots, X_n) \cdot P(X_2, \ldots, X_n)$ we could write $P(X_1, \ldots, X_n)$ in the following way:

$$P(X_1, \ldots, X_n) = P(X_1 \mid X_2, \ldots, X_n) \cdot P(X_2 \mid X_3, \ldots, X_n) \cdot \ldots \cdot P(X_{n-1} \mid X_n) \cdot P(X_n) \tag{10.8}$$

This equation is called the *chain rule*.

10.1.3 Exploiting independence property

Independence of variables can be used to significantly simplify the representation of complex joint distributions. To this end, we will consider the following example of six different events related to a woman's pregnancy.

The pregnancy test example - two variables conditional probability distribution

Consider two binary random variables, G and D, related to the pregnancy state of a woman:

- G stands for whether the woman is pregnant or not
- D stands for the doctor's pregnancy test result for this woman

Since these variables are binary, there are two possible events for each of them and a total of four joint events ($4 = 2^2$). Table 10.1 presents the joint distribution of the variables, $P(G, D)$.

However, these events are not independent, and clearly the result of the doctor's pregnancy test depends on whether the woman is really pregnant or not. The four values of the joint events' probability can be easily computed if we had the distribution of $P(g)$ - the real pregnancy state of the woman, and the distribution of the conditional probability of $P(d|g)$ - or what are the chances of the pregnancy test to be positive or negative given the real

G	D	$P(G, D)$
0	0	0.54
0	1	0.06
1	0	0.02
1	1	0.38

Table 10.1: Joint distribution of pregnancy events

pregnancy state of the woman. Let g_0 denote the event of the woman being not pregnant and g_1 denote the complement event of the woman being pregnant. In a similar manner, let d_0 denote the event of a negative pregnancy test result conducted by the doctor, and d_1 denote a positive result.

Breaking the joint distribution into two distributions, one of $P(g)$ and one of $P(d|g)$ is called a factorial representation. Tables 10.2 and 10.3 present this factorial representation.

g_i	$P(G = g_i)$
g_0	0.6
g_1	0.6

Table 10.2: Factorial representation - Probability of a woman to be pregnant

$P(d\|g)$	d_0	d_1
g_0	0.9	0.1
g_1	0.05	0.95

Table 10.3: Factorial representation - Probability of a doctor's pregnancy test result, given the pregnancy state of a woman. Most chances are that the test result corresponds to the real woman's state, but there is a minor room for mistakes.

Computing the original distribution $P(G, D)$ is done by multiplying the probabilities of the wanted events. For example, the joint probability of the event of a woman being not pregnant and event of the doctor's test result to be positive is 0.06, given in the second row of Table 10.1. It can be computed from the factorial representation:

$$P(G = g_0) \cdot P(D = d_1|G = g_0) = 0.6 \cdot 0.1 = 0.06$$

As mentioned before, the factorial representation is a compact representation of the same information, exploiting the known dependencies between the random variables. Let's look at how much we saved in the pregnancy example by switching to factorial representation. The original joint distribution table has four values in it, but since this is a distribution,

they values complement to 1, which means the fourth value can be computed using the three others ($0.38 = 1 - (0.02 + 0.06 + 0.54)$) so we used a total of three parameters.

Using the factorial representation we have 1 parameter in the distribution of $P(G)$ (again, 1 complement), and 2 parameters for the distribution of $P(d|g)$ (each line is 1 complement). Total of three parameters again. Adding a third binary random variable H, representing a home pregnancy test result, would reveal that the saving grows as the number of variables raises, using the independence assumption.

The pregnancy test example - three variables conditional probability distribution

The result of a home pregnancy test and that of a doctor's pregnancy test are dependent, but given the real pregnancy state of a woman this dependency is canceled - $I(H; D|G)$ - that is, given G, H is independent of D. By adding a third table of $P(H|G)$ to the factorial representation (Table 10.4), we can now calculate the joint distribution $P(D, H, G)$ by using the product rule and the independence assumption.

$\boldsymbol{P(h\|g)}$	$\boldsymbol{h_0}$	$\boldsymbol{h_1}$
$\boldsymbol{g_0}$	0.9	0.1
$\boldsymbol{g_1}$	0.2	0.8

Table 10.4: Factorial representation - Probability of a home pregnancy test result, given the pregnancy state of a woman. Now we can calculate $P(d, h, g) = P(d, h|g) \cdot P(g) = P(d|g) \cdot P(h|g) \cdot P(g)$. The first equality is a simple product rule, while the second stands because $I(H; D|G)$

.

While the number of parameters for the joint distribution of the three binary variables is $2^3 - 1 = 7$, the number of parameters using the factorial representation, exploiting the independence of H and D given G is now only $1 + 2 + 2 = 5$. The new representation is also very *natural*, breaking the events into "cases", and *modular*, meaning that a adding a new variable reuses the local probability models so far, and only adds a new local probability table for the new variable, while a joint distribution representation requires changing all the parameters.

The only addition required for the factorial representation to be complete is a dependency graph (or net), where the nodes are the random variables, and a directed edge is connecting node u and node v if u influences v (v depends on u). The simple dependency graph for our example is given in Figure 10.1.

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.

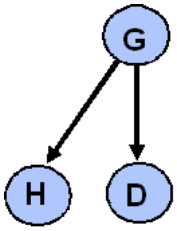

Figure 10.1: Dependency graph between the random variables in the pregnancy example. G denotes the actual pregnancy state of a woman, D is the doctor's pregnancy test result, and H is the home test result. D and H both depend on G but are independent of each other if G is known.

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 10.5.

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 10.5: Joint distribution representation of four fair tosses

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 10.6). 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 10.6: 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.

10.2 Bayesian Networks

10.2.1 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 $P(X_2)$ or $P(X_2 \mid X_3, X_5)$. Unfortunately, even when assuming for simplicity that the variables are binomial, the representation complexity of $P(X_1, \ldots, X_n)$ is, as we saw, 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 edges correspond to dependencies and their directions. This component is known 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.

10.2.2 The Markov assumption

The Markov assumption is that 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 the Bayesian network. Suppose w.l.o.g (without limitation of globality) $X_1, \ldots, X_n$ 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 10.8):

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

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:

$$P(X_1, \ldots, X_n) = \prod_{i=1}^{n} P(X_i \mid \mathbf{Pa}(X_i)), \tag{10.9}$$

Example 10.4: A Simple Bayesian Network with Five Variables

In Figure 10.2 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.

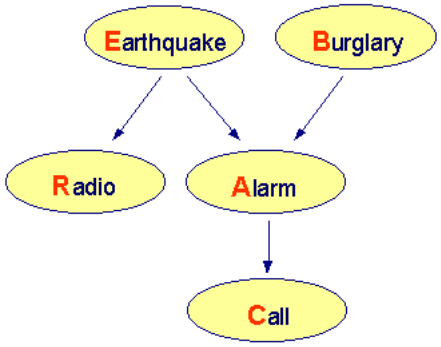

Figure 10.2: A simple Bayesian network with five variables

The alarm is set to detect earthquakes and attempts of burglary. 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 had not gone off). In addition the local radio 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 10.2).

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 10.8), 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 10.9), 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, $l \ll n$ and the representation complexity is much better than the $O(k^n)$ representation complexity of the joint distribution representation.

10.3 Inference in Bayesian networks

10.3.1 Introduction to inference

So far we saw 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 10.9). 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 10.7), 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.

10.3.2 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 10.3. Suppose we want to

Figure 10.3: 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 (10.7), 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 10.8), 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 have seen how to answer a query 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.

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

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

Next $P(X_1|X_k)$ is found, using the product rule (equation 10.2): $P(X_1 \mid x_k) = \frac{P(X_1, x_k)}{P(X_k = x_k)}$

Complexity Analysis

Let's observe the complexity of solving the queries presented. As mention in section 10.3.1, 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 following 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 (10.3), 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:

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

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 was not optimal. Choosing an efficient order of elimination can be done using a *clique tree algorithm*, which will not be discussed here.

To summarize, the complexity of inferring distributions from the Bayesian network representation obeys the following rules :

- The naive method of inference is exponential in the number of variables. In general, the inference problem is NP-hard.
- Using variable elimination method is exponential in the size of the largest summation factor and polynomial in the number of variables.
- Complexity of variable elimination depends on order of elimination.

10.4 Learning Bayesian networks

10.4.1 Introduction [8]

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.

10.4.2 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 10.4, or learning both the distributions and the graph's structure (the dependencies), as in Figure 10.5. 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 10.7.

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.

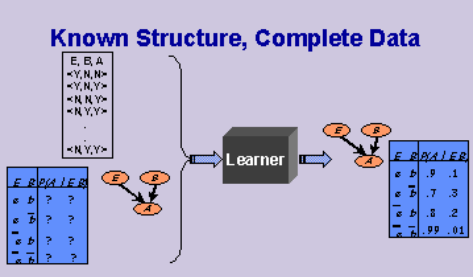

Figure 10.4: Learning from a known structure and complete data

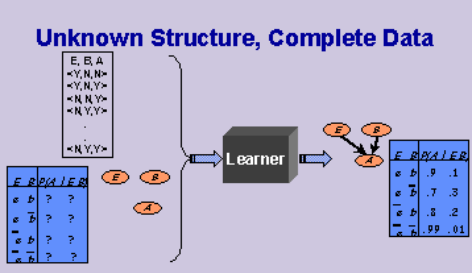

Figure 10.5: Learning from an unknown structure and complete data

10.4.3 Learning parameters

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 "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.

	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 10.7: Means of learning the network in different cases

Likelihood definitions

- **The likelihood function** [17, 13]
  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 receives 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.

10.4.4 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 $\frac{3}{5} = 0.6$.

MLE principle for multinomial data

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

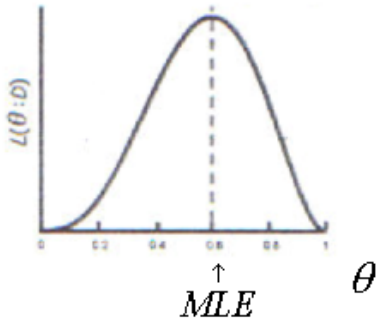

Figure 10.6: The maximum likelihood estimation

on a dice). 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

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

MLE principle for Bayesian networks

When learning parameters for Bayesian networks, the training data has a form as in figure 10.7.

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

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

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

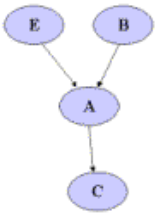

Figure 10.7: The training data, including the network structure

where (*) is due to switching to the factorial representation, by definition of the network in figure 10.7. Rewriting the terms (looking at the columns of the training data instead of the rows) we get:

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

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:

$$\begin{aligned} L(\theta : D) &= \prod_m P(x_1[m], \ldots, x_n[m] : \theta) \underbrace{=}_{\text{network factorization}} \\ &= \prod_i \prod_m P(x_i[m] \mid Pa_i[m] : \theta_i) \\ &= \prod_i L_i(\theta_i : D) \end{aligned}$$

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 10.8).

In Bayesian networks, when we assume that $P(x_i \mid pa_i)$ is multinomial, we get further decomposition. From equation 10.10 we get:

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

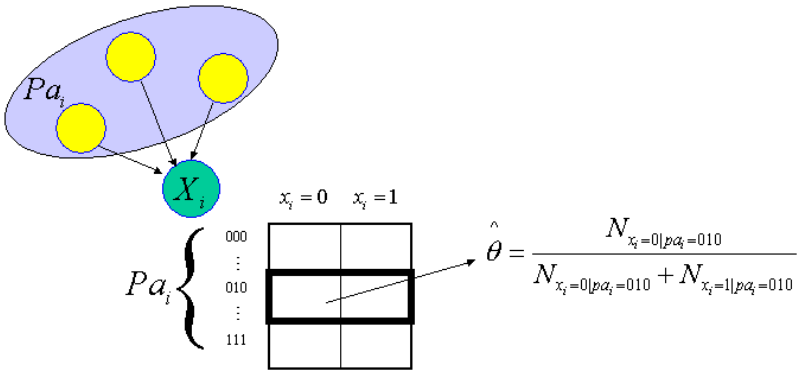

Figure 10.8: Demonstrating the decomposability principle - the variable X_i has three parents, which are the members of the set Pa_i. Each row of the variable's CPD is a different combination of the parents' values and the columns are the different values X_i can receive. Each row is X_i's distribution 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.

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 10.7, 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 10.8, but for the multinomial case.

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 10.9.

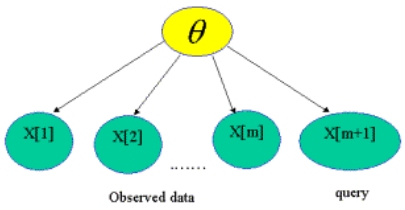

Figure 10.9: Representing Bayesian prediction as inference in Bayesian network

All the data are dependent on the certain θ that "created" them. As seen in the Markov 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:

$$= \int_0^1 P(x[m+1] \mid \theta) P(\theta \mid x[1], \ldots, x[m]) d\theta \tag{10.10}$$

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

$$\underbrace{P(\theta \mid x[1], \ldots, x[m])}_{\text{posterior}} \underbrace{=}_{\text{Bayes rule}} \frac{\overbrace{P(x[1], \ldots, x[m] \mid \theta)}^{\text{likelihood}} \overbrace{P(\theta)}^{\text{prior}}}{\underbrace{P(x[1], \ldots, x[m])}_{\text{probability of data}}} \tag{10.11}$$

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 10.11 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 10.10 and can be taken out of the integral there (see the example in the following section, 10.4.4).

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.

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 10.8: The differences between the MLE and Bayesian approaches

Summary: MLE vs. Bayesian Learning

The differences between the MLE approach and the Bayesian approach are summarized in table 10.8.

10.4.5 Dirichlet Priors

In example 10.4.4, 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**:

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

This is the density function of a k-dimensional random variable θ with hyperparameters $\alpha_1, \dots, \alpha_k$. If we know these hyperparameters, we will know the density function explicitly as a function of $\theta_1, \dots, \theta_k$. 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 10.11 we get:

$$P(\theta \mid D) \overset{(*)}{\propto} P(\theta)P(D \mid \theta) \propto \prod_{j=1}^{k} \theta_j{}^{\alpha_j - 1} \prod_{j=1}^{k} \theta_j{}^{N_j} = \prod_{j=1}^{k} \theta_j{}^{\alpha_j + N_j - 1} \tag{10.12}$$

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 10.11. 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 10.10 and (**) is the result of solving the integral with $P(\theta \mid D)$, which appears in equation 10.12. 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 learning schemes can be implemented in an on-line manner by accumulating counts.

10.4.6 Learning structure

As described in section 10.4.2, 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:

$$Score(G : D) = \sum_i Score(x_i \mid Pa_i^G : D) \tag{10.13}$$

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 - we start with a certain structure and perform random modifications to it. The number or size of modifications is reduced with time so we start with a search of the entire solution space and later use smaller modifications in order to achieve a local minima.

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 10.10 - 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.

An algorithm for learning Bayesian network's structure

An example for a structure inference algorithm is the algorithm by Pen and Ding [21]. Their algorithm uses local learning in order to generate parts of the Bayesian network and merge them into a valid network. This method is based on the locality of the network's score (Equation 10.13. The algorithm is composed of 3 steps:

- Generation of local structure. For each node, the optimal set of parents is found by starting with an optimal (i.e. likelihood maximizing) singleton set and adding nodes in order to improve the solution (i.e. increase the likelihood for that node)
- Elimination of cycles. After composing the local structures, found in the first step, into a Bayesian network the algorithm makes sure the network is valid. The network might contain cycles which have to be removed in order to create a valid network (Bayesian networks have to be DAGs[1]). The challenge is to remove the cycles with minimal damage to the network's score.
- Local structure perturbation. To asses and improve the stability of the network the algorithm creates local perturbations (e.g. eliminating a random edge and replacing it by other local edges) and tests for improvement of the network's score. Unstable edges whose replacement improves the total score are replaced.

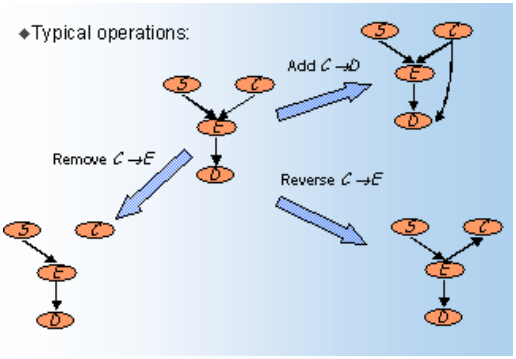

Figure 10.10: The typical operations performed in the heuristic search

10.5 Applying Bayesian networks to expression data

This section describes an approach for analyzing gene expression data using Bayesian network learning techniques. We model the expression level of each gene as a random variable. In

[1]Direct Acyclic Graph

addition, other attributes that affect the system can be modeled as random variables. These can include a variety of attributes of the sample, 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.

By learning a Bayesian network based on the statistical dependencies between these variables, we can 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?

We now describe how one can learn such a model from expression data. Many important issues arise when learning in this domain. These involve statistical aspects of interpreting the results, algorithmic complexity issues in learning from the data, and preprocessing the data.

Most of the difficulties in learning from expression data revolve around the following central point: Contrary to previous applications of learning Bayesian networks, expression data involves transcript levels of thousands of genes while current datasets contain at most a few dozen samples. This raises problems in computational complexity and the statistical significance of the resulting networks. On the positive side, genetic regulation networks are sparse, i.e., given a gene, it is assumed that no more than a few dozen genes directly affect its transcription. Bayesian networks are especially suited for learning in such sparse domains.

10.5.1 Network Features

When learning models with many variables, small datasets are not sufficiently informative to significantly determine that a single model is the "right" one. Instead, many different networks should be considered as a reasonable explanation of the given the data. From a Bayesian perspective, we say that the posterior probability over models is not dominated by a single model (or equivalence class of models). We would like to analyze this set of plausible (i.e., high-scoring) networks. Although this set can be very large, we might attempt to characterize features that are common to most of these networks, and focus on learning them.

Before we examine the issue of inferring such features, we briefly discuss two classes of features involving pairs of variables:

Markov relations

A relation of this type specifies if Y is in the Markov blanket of X, where the Markov blanket of X is the minimal set of variables that shield X from the rest of the variables in the model (see Figure 10.11 for an example). More precisely, X given its Markov blanket is independent from the remaining variables in the network. It is easy to check that this relation is symmetric: Y is in X's Markov blanket if and only if there is either an edge between them, or both are parents of another variable [19]. In the context of gene expression analysis, a Markov relation indicates that the two genes are related in some joint biological interaction or process. Note, that two variables in a Markov relation are directly linked in the sense that no variable in the model mediates the dependence between them. It remains possible that an unobserved variable (e.g., protein activation) is an intermediate in their interaction.

Order relations

An order relation specifies if X is an ancestor of Y in all the networks of a given equivalence class. That is, if the given PDAG[1] contains a directed path from X to Y . This type of relation does not involve only a close neighborhood, but rather captures a global property. Thus, we view such a relation as an indication, rather than evidence, that X might be a causal ancestor of Y.

While at this point we handle only pairwise features, it is clear that this analysis is not restricted to them, and we should examine also features that are more complex (see [20]).

10.5.2 Estimating statistical confidence in features

We now face the following problem: To what extent do the data support a given feature? More precisely, we want to estimate a measure of confidence in the features of the learned networks, where "confidence" approximates the likelihood that a given feature is actually true (i.e., is based on a genuine correlation and causation) (see Figure 8.16).

An effective, and relatively simple, approach for estimating confidence is the bootstrap method [6]. The main idea behind the bootstrap is simple. We generate "perturbed" versions of the original dataset, and learn from them. In this way we collect many networks, all of which are fairly reasonable models of the data. These networks show how small perturbations to the data can affect many of the features. In our context, we use the bootstrap as follows:

- For $i = 1...m$ (in the experiments, we set $m = 200$):

[1]Partially Directed Acyclic Graph - A Graph in which only a part of the edges are directed

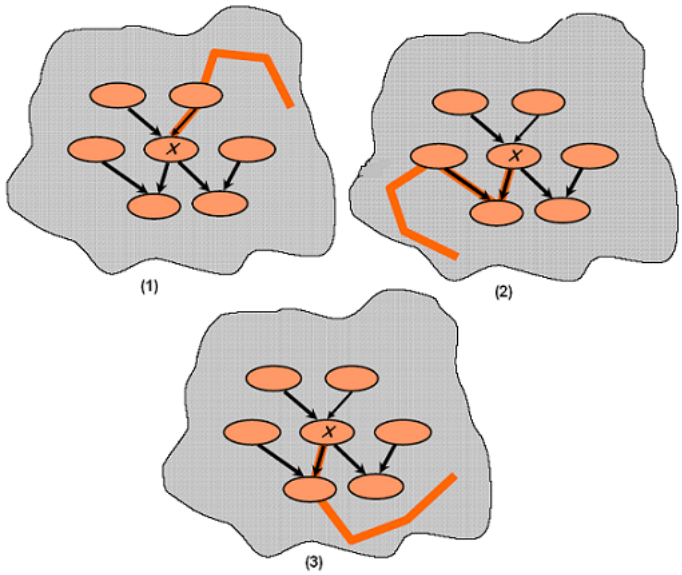

Figure 10.11: Source [27]. An example for a Markov blanket. This Markov blanket of X contains all paths from X to other nodes. There are three kinds of such paths as shown in the figure: (1) *Upward paths* the parents of X. (2) *Sideway paths* the spouses of X. (3) *Downward paths* the children of X.

  - Resample with replacement N instances from D. Denote by Di the resulting dataset.
  - Apply the learning procedure on D_i to deduce a network structure $\hat{G}_i$.

- For each feature f of interest calculate

$$conf(f) = \frac{1}{m}\sum_{i=1}^{m} f(\hat{G}_i) \tag{10.14}$$

  where $f(G)$ is 1 if f is a feature in G, and 0 otherwise.

10.5.3 Efficient learning algorithms

In section 11.4 we formulated learning Bayesian network structure as an optimization problem in the space of directed acyclic graphs. The number of such graphs is super-exponential in the number of variables. As we consider hundreds and thousands of variables, we must deal with an extremely large search space. Therefore, we need to use (and develop) efficient search algorithms.

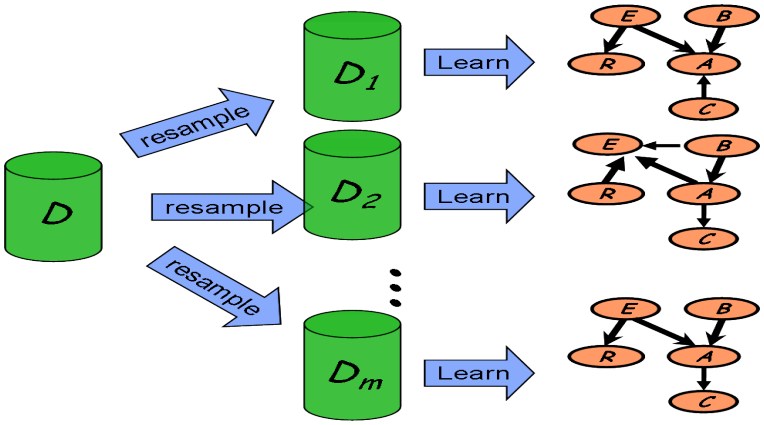

Figure 10.12: A Bootstrap method. Only features with high confidence will be accepted

To facilitate efficient learning, we need to be able to focus the attention of the search procedure on relevant regions of the search space, giving rise to the sparse candidate algorithm [12]. The main idea of this technique is that we can identify a relatively small number of candidate parents for each gene based on simple local statistics (such as correlation). We then restrict our search to networks in which only the candidate parents of a variable can be its parents, resulting in a much smaller search space in which we can hope to find a good structure quickly.

10.6 Experimental results

The Bayesian Networks approach was applied by Freidman et al. [9] to two datasets: the data of Spellman et al. [24] and the data of Hughes et al. [15]. From this point and on, we will refer only to the data of Spellman et al. We refer the reader to [20] for details about the results from the data of Hughes et al.

The data contains 79 gene expression measurements of the mRNA levels of 6177 S. cerevisiae ORFs. These experiments measure expression in fixed time intervals under different cell cycle synchronization methods. Spellman et al. identified 800 genes whose expression varied over the different cell-cycle stages. They clustered these 800 genes, based on the similarity of expression profiles, resulting 8 major clusters, which contained 250 genes in total. The variables of the learned networks were the expression level of each of these 800 genes. Some of the robustness analysis was performed only on the set of 250 genes that appear in the 8 major clusters.

Freidman et al. [9] used the Sparse Candidate algorithm with a 200-fold bootstrap in the learning process. The learned features show that intricate structure can be recovered even from such small data sets. It is important to note that this learning algorithm uses no prior

biological knowledge nor constraints. All learned networks and relations are based solely on the information conveyed in the measurements themselves. These results are available in [11]. Figure 10.15 illustrates the graphical display of results of this analysis.

10.6.1 Protein-signaling networks derived from multiparameter single cell-data

In a work by Sachs et al. ([22]), a signaling network of pathways in human immune system cells was reconstructed using Bayesian Network methods. A simultaneous measurement of proteins was done in thousands of human immune system cells. Perturbing the cells with different molecular intervenes yielded a predicted order of the connection between pathways in the cells, and Bayesian Network methods were used to predict specific signalling relationships between the proteins.

Figure 10.13 shows the conventional known signaling network, measured proteins and perturbations used to gather the data for reconstructing the network using Bayesian Network methods.

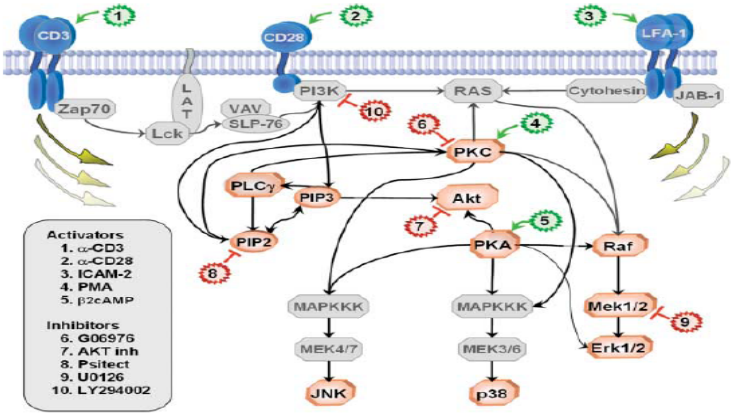

Figure 10.13: The conventional known network used by Sachs et al [22]. The gray line at the top is the cell membrane. Orange proteins levels were measured, while grey ones were not measured. Green perturbations marked 1-5 are activators (elevating the protein level) while red ones are inhibitors (blocking the protein, thus not allowing signals to reach it). Some perturbations (1-3) were done extracellular, while others were done inside the cell (Taken from [22])

Ten perturbations were used, and for each of them repeated measures of the level of 11 proteins within the immune cells were taken. Measurements were done using multi-coloring of the different proteins so the experiment could capture all the levels at a single timeframe simultaneously.

The Bayesian inference learning automatically excludes arcs that are based on dependencies already explained by the model. An example is shown in Figure 10.14 **B(c)** where $RAF \rightarrow MEK \rightarrow ERK$ path was learned, and not the direct $RAF \rightarrow ERK$ arc, although RAF and ERK are highly correlated in the data. Nevertheless, the Bayesian learning can only construct relations between measured proteins, so it can reveal a direct edge between two proteins which is actually indirect in the real world, just because the intermediate proteins in the path were not included in the measures. The $PKC \rightarrow JNK$ direct relation found, while it is really indirect, and passes through two $MAPK$ proteins in figure 10.14 **B(b)**.

A small disadvantage of the Bayesian Network modeling, is the inability to model feedback arcs, which are known to be present in signaling pathways, but cannot be modeled in a Bayesian Networks, due to the fact that they form cycles, which create mutual dependency which temper the derivation of parameter estimation.

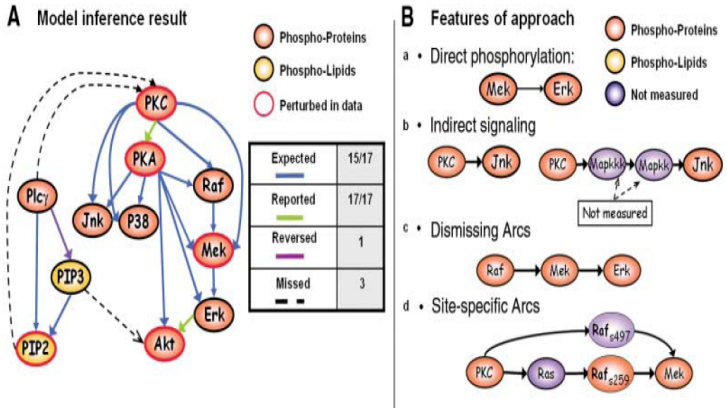

Figure 10.14: Results of Sachs et al (2005). The signaling network reconstructed is shown in **A**. A total of 17 arcs were revealed, 15 of them are well established (*Expected*), and 2 were reported before in very few sources (*Reported*). Three arcs were missing in the Bayesian Network (*Missed*), and one was found in a reverse direction. **B** shows different relations revealed - direct, indirect (due to lack of measurements) (Taken from [22])

Using simple protein level measurements, the Bayesian Network succeeded in reconstructing a signaling network constructed by biologists using classic biochemistry and genetic analysis over more than two decades, and also revealed new paths that were later experimentally validated.

10.6.2 Robustness analysis

Freidman et al. performed a number of tests to analyze the statistical significance and robustness of their procedure. They carried most of these tests on the smaller 250 gene data set

for computational reasons. To test the credibility of their confidence assessment, they created a random data set by randomly permuting the order of the experiments independently for each gene. Thus for each gene the order was random, but the composition of the series remained unchanged. In such a data set, genes are independent of each other, and thus we do not expect to find "real" features. As expected, both order and Markov relations in the random data set have significantly lower confidence. Clearly, the distribution of confidence estimates in the original data set have a longer and heavier tail in the high confidence region. The runs on the random data sets do not learn almost anything with a confidence level above 0.8, which can lead us to believe that most features that are learned in the original data set with such confidence levels originate in true signals in the data. Also, the confidence distribution for the real dataset is concentrated closer to zero than the random distribution. This suggests that the networks learned from the real data are sparser. Since the analysis was not performed on the whole S. cerevisiae genome, Freidman et al. also tested the robustness of their analysis to the addition of more genes, comparing the confidence of the learned features between the 250 and 800 gene datasets.

10.6.3 Biological Analysis

Freidman et al. believe that the results of this analysis can be indicative of biological phenomena in the data. This is confirmed by their ability to predict sensible relations between genes of known function. We now examine several consequences from this analysis. We consider, in turn, the order relations and Markov relations found.

Order Relations

The most striking feature of the high confidence order relations, is the existence of dominant genes. Out of all 800 genes only few seem to dominate the order (i.e., appear before many genes). The intuition is that these genes are indicative of potential causal sources of the cell-cycle process. A list of the highest scoring dominating genes appears in Table 10.9.

Inspection of the list of dominant genes reveals quite a few interesting features. Among the dominant genes are those directly involved in cell-cycle control and initiation. For example, CLN1, CLN2 and CDC5, whose functional relation has been established [4, 5]. Other genes, like MCD1 and RFA2, were found to be essential [14]. These are clearly key genes in basic cell functions, involved in chromosome dynamics and stability (MCD1) and in nucleotide excision repair (RFA2). Most of the dominant genes encode nuclear proteins, and some of the unknown genes are also potentially nuclear: (e.g., YLR183C contains a forkhead associated domain which is found almost entirely among nuclear proteins). Some of them are components of pre-replication complexes. Others (like RFA2,POL30 and MSH6) are

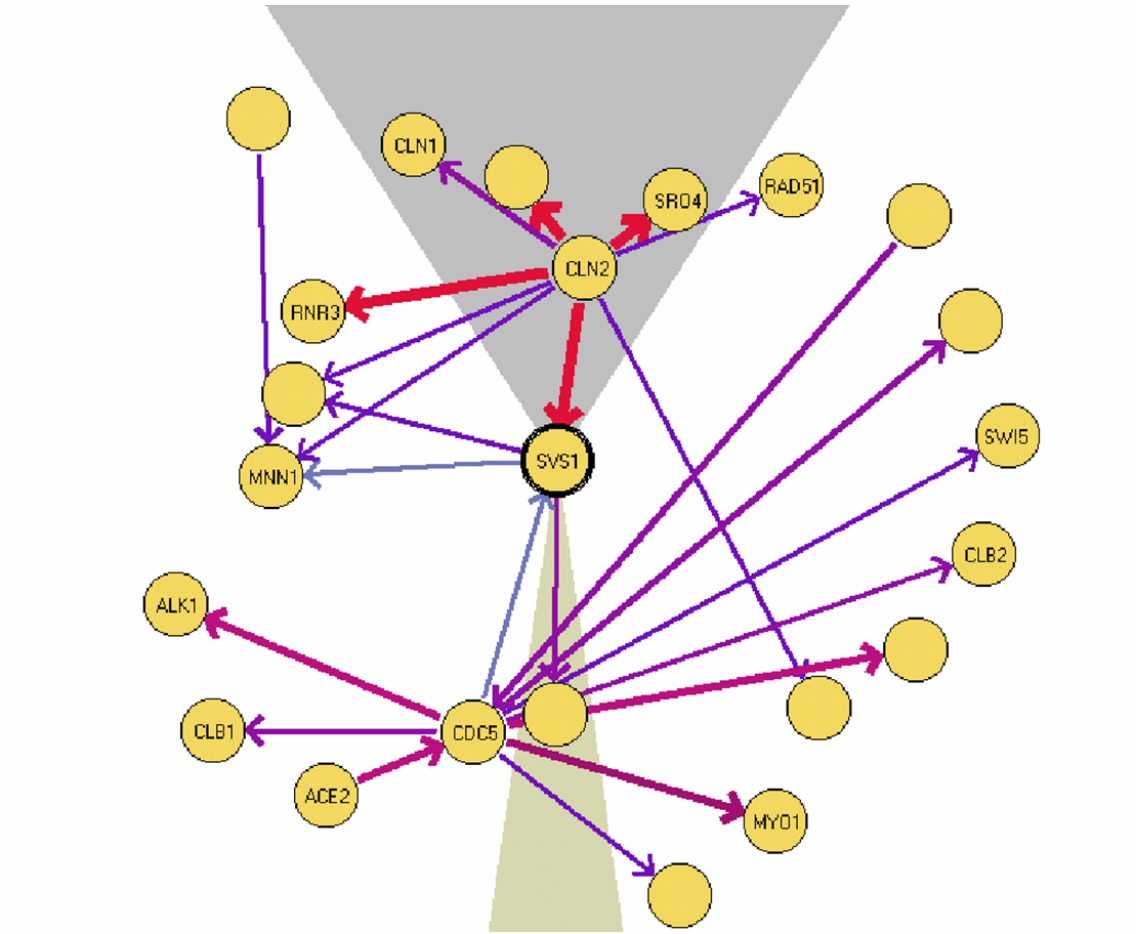

Figure 10.15: Source: [9]. An example of the graphical display of Markov features. This graph shows a "local map" for the gene SVS1. The width (and color) of edges corresponds to the computed confidence level. An edge is directed if there is a sufficiently high confidence in the order between the pair 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.

Gene/ORF	Dominance Score	♯ of desc. genes > .8	> .7	Notes
YLR183C	551	609	708	Contains forkheaded associated domain, thus possibly nuclear
MCD1	550	599	710	Mitotic chromosome determinant, null mutant is inviable
CLN2	497	495	654	Role in cell cycle START, null mutant exhibits G1 arrest
SRO4	463	405	639	Involved in cellular polarization during budding
RFA2	456	429	617	Involved in nucleotide excision repair, null mutant is inviable
YOL007C	444	367	624	
GAS1	433	382	586	Glycophospholipid surface protein, Null mutant is slow growing
YOX1	400	243	556	Homeodomain protein that binds leutRNA gene
YLR013W	398	309	531	
POL30	376	173	520	Required for DNA replication and repair, Null mutant is inviable
RSR1	352	140	461	GTPbinding protein of the ras family involved in bud site selection
CLN1	324	74	404	Role in cell cycle START, null mutant exhibits G1 arrest
YBR089W	298	29	333	
MSH6	284	7	325	Required for mismatch repair in mitosis and meiosis

Table 10.9: Source [9]. List of dominant genes in the ordering relations (top 14 out of 30). The first column specifies the name of the gene/ORF, the second column specifies the level of dominance score of the gene/ORF as appeared in the experiments results, the next column contains the number of descendent genes with a level of confidence higher than 0.8, the next column contains the number of descendent genes with a level of confidence higher than 0.7 and the last column supplies additional biological information about the gene/ORF.

involved in DNA repair. It is known that DNA repair is a prerequisite for transcription, and DNA areas which are more active in transcription, are also repaired more frequently [18, 26]. A few non nuclear dominant genes are localized in the cytoplasm membrane (SRO4 and RSR1). These are involved in the budding and sporulation process which have an important role in the cell-cycle. RSR1 belongs to the ras family of proteins, which are known as initiators of signal transduction cascades in the cell.

Markov Relations

Inspection of the top Markov relations reveals that most are functionally related. A list of the top scoring relations can be found in Table 10.10. Among these, all involving two known genes make sense biologically. When one of the ORFs is unknown careful searches using Psi-Blast [1], Pfam [23] and Protomap [28] can reveal firm homologies to proteins functionally related to the other gene in the pair (e.g. YHR143W, which is paired to the endochitinase CTS1, is related to EGT2 - a cell wall maintenance protein). Several of the unknown pairs are physically adjacent on the chromosome and, thus, are presumably regulated by the same mechanism (see [2]), although special care should be taken for pairs whose chromosomal location overlap on complementary strands, since in these cases we might see an artifact resulting from cross-hybridization. There are some interesting Markov relations found that are not discovered using clustering techniques. One such regulatory link is FAR1-ASH1: both proteins are known to participate in a mating type switch. The correlation of their expression patterns is low and [24] cluster them into different clusters. Among the high confidence markov relations, one can also find examples of conditional independence, i.e., a group of highly correlated genes whose correlation can be explained within the resulted network structure. One such example involves the genes: CLN2, RNR3, SVS1, SRO4 and RAD41. Their expression is correlated and in [24] all appear in the same cluster. In the resulting network CLN2 is with high confidence a parent of each of the other 4 genes, while no links are found between them. This suits biological knowledge: CLN2 is a central and early cell cycle control, while there is no clear biological relationship between the others.

Confidence	Gene 1	Gene 2	Notes
1.0	YKL163WPIR3	YKL164CPIR1	Close locality on chromosome
0.985	PRY2	YKR012C	No homolog found
0.985	MCD1	MSH6	Both bind to DNA during mitosis
0.98	PHO11	PHO12	Both nearly identical acid phosphatases
0.975	HHT1	HTB1	Both are Histones
0.97	HTB2	HTA1	Both are Histones
0.94	YNL057W	YNL058C	Close locality on chromosome
0.94	YHR143W	CTS1	Homolog to EGT2 cell wall control, both do cytokinesis
0.92	YOR263C	YOR264W	Close locality on chromosome
0.91	YGR086	SIC1	Homolog to mammalian nuclear ran protein, both involved in nuclear function
0.9	FAR1	ASH1	Both part of a mating type switch, expression uncorelated
0.89	CLN2	SVS1	Function of SVS1 unknown, possible regulation mediated through
SWI6	0.88	YDR033W	NCE2 Homolog to transmembrane proteins, suggesting both involved in protein secretion
0.86	STE2	MFA2	A mating factor and receptor
0.85	HHF1	HHF2	Both are Histones
0.85	MET10	ECM17	Both are sulfite reductases
0.85	CDC9	RAD27	Both participate in Okazaki fragment processing

Table 10.10: Source [9]. List of top Markov relations. The first column describes the level of confidence of the relation, the next two columns contain the names of the two genes of the relation and the last column supplies additional biological information referring the relation.

10.7 Using biological knowledge to increase robustness

An interesting method to increase the robustness of created Bayesian networks is based on the usage of prior biological knowledge regarding the structure of the Bayesian network. This knowledge is used to filter possible networks created in the Bootstrap phase. in [25] the Bayesian network is limited to a two level tree (see 10.16).

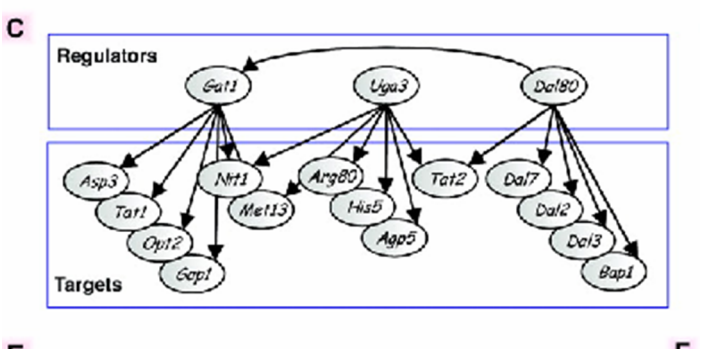

Figure 10.16: In [25] the structure of the Bayesian network is limited to a two level tree, based on the naive separation between regulators and targets

In [16] a more sophisticated structure is imposed, in which the network is composed of modules - sets of co-regulated parameters (i.e. immediate descendants of the same parameters) (see 10.17).

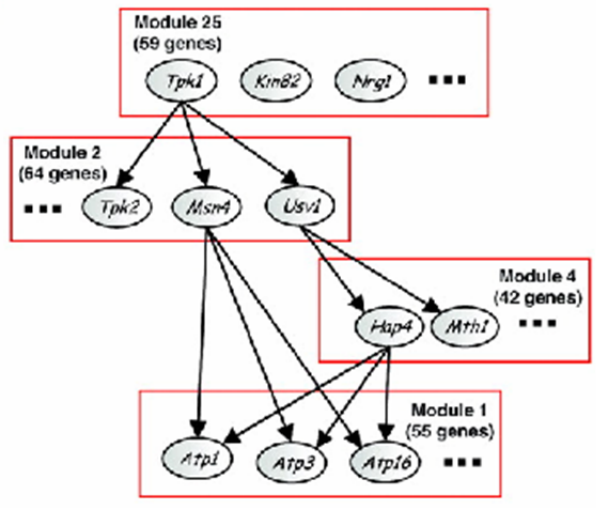

Figure 10.17: In [16] the structure of the Bayesian network is limited to a tree of modules

10.8 Improving The Framework

The framework we described here can be expanded in a number of promising directions:

- Correct handling of hidden variables (active proteins)
- Developing the theory for learning local probability models that are capable of dealing with the continuous nature of the data.
- Improving the theory and algorithms for estimating confidence levels.
- Incorporating biological knowledge (such as possible regulatory regions) as prior knowledge to the analysis.
- Improving the search heuristics.
- Applying Dynamic Bayesian Networks ([10]) to temporal expression data.
- Dealing with feedback loops which are common to metabolic pathways

Finally, one of the most exciting longer term prospects of this line of research is discovering causal patterns from gene expression data. We can build on and extend the theory for learning causal relations from data and apply it to gene expression. The theory of causal networks allows learning both from observational data and interventional data, where the experiment intervenes with some causal mechanisms of the observed system. In gene expression context, we can model knockout/overexpressed mutants as such interventions. Thus, we can design methods that deal with mixed forms of data in a principled manner (See [3] for a recent work in this direction). In addition, this theory can provide tools for experimental design, that is, understanding which interventions are deemed most informative to determining the causal structure in the underlying system. Friedman et al. have extended their framework in this direction (see [20]).

Bibliography

[1] S.F. Altschul, T.L. Madden, A.A. Schaffer, J. Zhang, Z. Zhang, W. Miller, and D.J. Lipman. Gapped blast and psi-blast: a new generation of protein database search programs. *Nucleic Acids Res.*, 25, 1998.

[2] T. Blumenthal. Gene clusters and polycistronic transcription in eukaryotes. *Bioessays*, 480-487, 1998.

[3] G. Cooper and C. Yoo. Causal discovery from a mixture of experimental and observational data. *Proc. Fifthteenth Conference on Uncertainty in Artificial Intelligence (UAI'99)*, pages 116–125, 1999.

[4] F. Cvrckova and K. Nasmyth. Yeast g1 cyclins cln1 and cln2 and a gap-like protein have a role in bud formation. *EMBO*, pages 12:5277–5286, 1993.

[5] M.A. Drebot, G.C. Johnston, J.D. Friesen, and R.A. Singer. An impaired rna polymerase ii activity in saccharomyces cerevisiae causes cell-cycle inhibition at start. *Mol Gen Genet*, pages 241:327–334, 1993.

[6] B. Efron and R.J. Tibshirani. *An Introduction to the Bootstrap*. Chapman and Hall, London, 1993.

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

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

[9] N. Friedman, M. Linial, I. Nachman, and D. Pe'er. Using bayesian networks to analyze expression data. *Journal of Computational Biology*, pages 7:601–620, 2000.

[10] N. Friedman, K. Murphy, and S. Russell. Learning the structure of dynamic probabilistic networks. *Proc. Fourteenth Conference on Uncertainty in Artificial Intelligence (UAI'98)*, pages 139–147, 1998.

[11] N. Friedman, I. Nachman, and D. Pe'er. http://www.cs.huji.ac.il/labs/compbio/expression/.

[12] N. Friedman, I. Nachman, and D. Pe'er. Bayesian network structure from massive datasets: The 'sparse candidate' algorithm. *Proc. Fifthteenth Conference on Uncertainty in Artificial Intelligence (UAI'99)*, pages 196–205, 1999.

[13] C.M. Grinstead and J.L. Snell. *Introduction to Probability.* American Mathematical Society, second edition, 1997.

[14] V. Guacci, D. Koshland, and A. Strunnikov. A direct link between sister chromatid cohesion and chromosome condensation revealed through the analysis of mcd1 in s. cerevisiae. *Cell*, pages 91:47–57, 1997.

[15] T.R. Hughes et al. Functional discovery via a compendium of expression profiles. *Cell*, pages 102:109–126, 2000.

[16] D. Koller, S.K. Kim, J.M. Stuart, and E. Segal. A gene-coexpression network for global discovery of conserved genetic modules. *Science*, pages 302:249–255, 2003.

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

[18] W. G. McGregor. Dna repair, dna replication, and uv mutagenesis. *Investig Dermatol Symp Proc*, pages 4:1–5, 1999.

[19] J. Pearl. *Probabilistic Reasoning in Intelligent Systems.* Morgan Kaufmann, San Francisco, 1988.

[20] D. Pe'er, A. Regev, G. Elidan, and N. Friedman. Inferring subnetworks from perturbed expression profiles. *Proc. Ninth International Conference on Intelligent Systems for Molecular Biology (ISMB '01)*, 2001.

[21] H. Peng and C. Ding. Structure search and stability enhancemen of bayesian networks. *Proceeding of the Third IEEE Conference on Data Mining*, 2003.

[22] Karen Sachs, Omar Perez, Dana Pe'er, Douglas A. Lauffenburger, and Garry P. Nolan. Causal Protein-Signaling Networks Derived from Multiparameter Single-Cell Data. *Science*, 308(5721):523–529, 2005.

[23] E.L. Sonnhammer, S.R. Eddy, E. Birney, A. Bateman, and R. Durbin. Pfam: multiple sequence alignments and hmm-profiles of protein domains (http://pfam.wustl.edu/). *Nucleic Acids Res*, pages 26:320–322, 1998.

[24] P.T. Spellman et al. Comprehensive identification of cell cycle-regulated genes of the yeast sacccharomyces cerevisiae by microarray hybridization. *Molecular Biology of the cell*, pages 9:3273–3297, 1998.

[25] A. Tanay, D. Pe'er, and A. Regev. A fast and robust method to infer and characterize an active regulator set for molecular pathways. *Bioinformatics*, 18(1):258–267, 2002.

[26] S. Tornaletti and P. C. Hanawalt. Effect of dna lesions on transcription elongation. *Biochimie*, pages 81:139–146, 1999.

[27] Y. Weiss. http://www.cs.huji.ac.il/∼pmai/tirguls/tirgul3.pdf.

[28] G. Yona, N. Linial, and M. Linial. Protomap - automated classification of all protein sequences: a hierarchy of protein families, and local maps of the protein space. *Proteins: Structure, Function, and Genetics*, pages 37:360–378, 1998.