On the Use of Learning Bayesian Networks to Analyze Gene Expression Data: Classification and Gene Network Reconstruction

Emiel Ver Loren van Themaat

MASTER THESIS

Universty of Amsterdam

Artificial Intelligence

June 2005

Supervisors:
Frans Voorbraak
Ben Krose
Han Rauwerda

Pieter Emiel Ver Loren van Themaat
Email: P.E.VerLorenvanThemaat@amc.uva.nl
Artificial Intelligence, Multimodal Intelligent Systems
Informatics Institute, Faculty of Science
University of Amsterdam
Ter verkrijging van de graad van Master of Science

Abstract

Recent technological developments in molecular biology have made it possible to measure the expression level of thousands of genes simultaneously. These measurements provide a "snapshot" of what is happening at the level of genes within a cell. The massive amount of these datasets make automated analysis nessecary. It is becoming evident that statistical and computational issues will largely determine what scientific questions can be answered.

This thesis deals with the application of a probabilistic framework called learning Bayesian networks for the analysis of gene expression datasets. Bayesian networks have already proven to be useful in this field. Our approach shows the usability of implementations of standard learning Bayesian network algorithms to analyze microarray data. We use gene expression measurements made on patients with mutiple myeloma and breast cancer.

Our setup results in answers about the kind of biological information that can be found with such experiments, and which algorithms should be used to learn Bayesian networks. We will show that Bayesian networks can achieve good classification accuracies and can find interesting subsets of disease related genes. Problems we encountered are the difficulties to find *one* network that best represents the data, estimating the classification accuracies and finding optimal settings of the parameters used by the algorithms.

Contents

Chapter 1

Introduction

1.1 Motivation

Ever since the discovery of the cell, the study of the function, components, and interaction of cells has been a major part of biology. Genes are among the most important components of a cell, and the last few years a revolution has taken place in monitoring the behavior of genes in a cell. Traditional time-consuming experiments are automated in miniature experiments carried out by robots. Thanks to these so-called "high-throughput experiments" experimental scientists can perform thousands of experiments at once or at an unprecedented rate. For example, a molecular biologist can measure the expression of all the known genes in yeast (several thousand) simultaneously by using the microarray technique. Massive amounts of data are available.

The experimental scientists have to analyze the experimental data. A molecular biologist may ask which genes are important for certain cell functions and in what way the genes are related to each other. These kinds of questions require a lot more than statistical tests like a t-test, for example [Gla01]. The development of statistical and computational procedures to address the scientific questions asked by these experimenters is developing rapidly. Also it is becoming evident that statistical and computational issues will as much as experimental methods or technologies determine what scientific questions can be answered and what breakthroughs will be made.

This thesis applies a probabilistic framework called *learning Bayesian networks* for the analysis of DNA microarray data. Bayesian networks have been developed within artificial intelligence to construct expert systems. In the 1980s Pearl [Pea88] showed that with Bayesian networks, experts can construct networks which are able to give advice, by reasoning with probabilities, on the basis of observations from the real world. In that time the networks were constructed by experts.

Techniques to automatically *learn* Bayesian networks from a dataset

have improved enormously over the past decade. Learning Bayesian networks have been applied in all different kinds of fields including weather forecasting, image processing, and medical systems [Neo04]. Learning networks with microarray data have several potential advantages for the analysis of experimental results, including a graphical representation of the result. Some studies have already proven that robust and interesting networks can be learned from microarray datasets [FLNP00, DWFS99].

Section 1.2 provides a short introduction into the biological background of microarrays. In Section 1.3 the analysis of microarrays is shortly discussed and in Section 1.4 the layout of this thesis is presented.

1.2 Biological introduction

Basic knowledge about biology is needed to understand the methods used and the context of this thesis. As some of the readers might lack biological background we decided to explain biological terms and background in this chapter. Terms like genes and DNA are explained in Section 1.2.1 and the subsequent section describes the DNA microarray techniques.

1.2.1 Central dogma

In 1953 Watson and Crick [WC53] discovered the structure of DNA (deoxyribonucleic acid). Human DNA is a double-stranded helix of nucleotide sequences which carries the genetic information. Each strand is constructed by 4 nucleotides; A (adenine) ,C (cytosine),T (thymine) and G (guanine). The two strands are connected by hydrogen bonds between A and T, and between C and G (these pairings are called complementarities; A is the complement of T).

Only a few years later Crick introduced a model of protein synthesis [Cri58]. The model is known as the central dogma of molecular biology. The model is not fully accepted, and had to be updated with reverse transcriptase in 1970, but gives a basic idea about the information flow in a cell, see Figure 1.1. In the DNA sequences all the information is stored to construct proteins that carry out most cell functions. RNA is a single strand of nucleotides, just like DNA, only T is substituted by nucleotide U (uracil), and U and A are each others complement.

The complete DNA sequence of an organism is called the genome of that organism. In molecular terms, a *gene* is a portion of DNA sequence that codes for a functional protein or RNA. A large proportion of DNA in the human genome does not encode for any protein or RNA; the genes make up merely 3% of our genome. Biologists believe the other part, which they sometimes call "junk DNA", could be a result of evolution or serve as a reservoir for making new genes. The function of this junk DNA is not known yet.

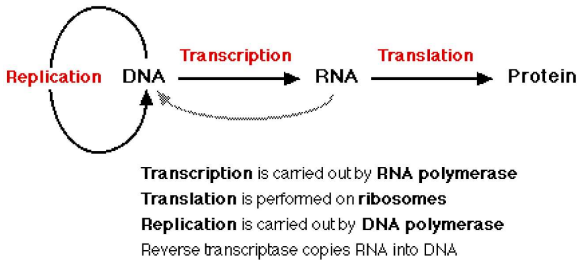

Figure 1.1: Central dogma of molecular biology.

Though no two humans have the same set of genes (except for identical twins), experts estimate that about 99.9% of any one person's genes are exactly the same as any one other person's DNA. In April 2003 the human genome project finished the identification of almost all the common DNA sequences in our entire genome[1]. The next job is to annotate this map by allocating the position of all our *genes* and understanding their role and function. Humans are thought to have at least 20,000 genes.

Nearly every cell in a multi-cellular organism contains its complete genome. However, the *expression* (activity) of genes in cells with different functions within a multi-cellular organism is normally not the same. The function and differentiation of a cell could be explained by the expression levels of the genes. The expression level of a gene in a cell at a certain point in time is the amount of transcribed RNA encoded by the gene at that timepoint. Monitoring the expression levels of all the genes in the genome of an organism is exactly what microarrays can do. Microarrays measure the amount of RNA in a sample and therefore we have an indication of the amount of the corresponding protein in the sample, see Figure 1.1.

1.2.2 Microarray technology

While it is very hard to detect the quantities of proteins in a cell, methods for detecting RNA can take advantage of the sequence complementarity of RNA or DNA. This means that if DNA or RNA occurs as a single strand of a certain sequence it really likes to pair with another single strand DNA or RNA of complementary sequence. A single strand of DNA with the sequence ACTTACG likes to pair with a strand of exact complementary sequence, thus TGAATGC. This pairing of two complementary single strands of DNA or RNA is called hybridization.

[1] Further information on the human genome project can be found on the NHGRI website, http://www.nhgri.nih.gov.

Essentially, a microarray is a carefully constructed set of the complements of known gene sequences arrayed on a special impermeable rigid surface (glass for example). A sample of unknown RNA sequences is poured over the surface to pair with the known sequences. If a RNA sequence has paired with its complement on the glass this is measured by special scanning techniques. Using a rigid surface has many practical advantages, such as the help of robots to create the arrays on microscopic scales.

Oligonucleotide versus spotted arrays Two basic methods to create microarrays are widely used; spotted arrays and oligonucleotide arrays [LDB+96]. In the *spotted arrays* a large number of DNA strands, complementary to known mRNA[2] sequences, are prepared from cDNA libraries and spotted onto a glass slide by a robot. Because the DNA is complementary to the mRNA, it is called cDNA. In this type of experiment cDNA can vary in length from 100 to 1,000 bases. Each spot on the slide contains several copies of a particular cDNA probe.

The key difference in the *oligonucleotide arrays* is that the probes are constructed nucleotide by nucleotide using photolithography or inkjet technologies. In hundreds of thousands of positions on the glass array, probes of length 20 or more are synthesized. For most genes, a probe containing 20 nucleotides can be created which is unique for that gene. Usually a set of probes is synthesized for each gene.

Single channel versus two channel arrays To estimate the expression of a gene in a sample two methods can be used. In the single channel method we elute only one sample over the array. RNA of the sample is colored and will hybridize to it's complement on the array. The expression rate of each gene is measured by scanning the intensity of the color on the array.

By eluting two samples over the same array, one sample is colored red and the other green, we can estimate the relative expression rate for each gene. On one array we can measure the expression of a healthy sample versus a tumour sample for example.

The resulting arrays can be viewed in Figure 1.2. It's not clear yet which method is best and to what extend experiments from different platforms are comparable.

1.3 Analyzing microarray data and using Bayesian networks

Turning scans into expression values The scans of a microarray have to be transformed into values representing the gene expression rates in or-

[2]RNA that codes for proteins is also referred to as messenger RNA, or mRNA.

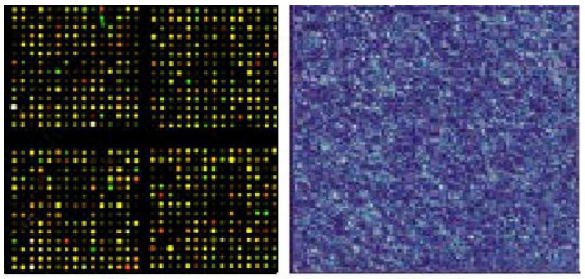

Figure 1.2: Two microarray techniques: On the left side a segment of a two cannel cDNA chip is shown and on the right side a single channel oligonucleotide chip (both taken from yeast cells).

der to make quantitative analysis possible. The scans usually contain a lot of noise and specialized image processing methods are used to reduce this. Background estimation and finding the optimal spot regions are a few examples. The resulting intensity values are transformed into well distributed gene expression values, by logarithmic ($\log_2$) transformation for example, making statistical analysis easier. At last, the data has to be normalized to correct for systematical differences between the conditions in which the microarrays are hybridized[3]. The result is a data matrix representing the (relative) gene expression values.

Machine learning challenges Analysts of microarray data make use of statistics, pattern recognition and machine learning techniques to extract interesting and relevant biological information. The information that can be found depends on the techniques that are used and on the experimental setup of the microarray experiment. Hierarchical clustering techniques, for example, can find groups of genes or persons that have similar expression patterns. Genes that show similar expression patterns are believed to be related in biological function. The biological functions, and how the functions are interrelated, are described in so-called biological pathways. Discovering the structure of these pathways is one of the ultimate goals in molecular biology. Gene network learning algorithms can discover relationships between genes and provide information on possible pathways.

The challenge for researchers in artificial intelligence in analyzing microarray experiments is the fact that the number of samples is usually a few dozen while the dimensionality, the number of genes whose expression is measured, is very large, usually several thousand. The low number of samples is mainly caused by the high costs of microarray experiments and

[3] A lot of conditions influence the results. Even the amount of ozon in the laboratorium influences the results (because the behaviour of the fluorescence depends on it)!

the difficulty of acquiring samples, in case of tumours for example. Another challenge, and problem, are the unstable conditions in which probes are labelled and arrays are hybridized. The data contains a lot of noise. In order to let a computer 'learn' or extract meaningful information from such a noisy, high dimensional, dataset a lot of samples are needed. But that is just what is missing. Basically, the problem is that the dataset does not reveal the 'real' world extensive enough and the danger of biased, data specific, results increase.

Several studies have proven in an empirical way that interesting information *can* be found by using machine learning techniques [Val02]. Some of these methods reduce the dimensionality of the data by selecting relevant genes, based on mathematical criteria. They make the assumption that in the context of an experimental setup, only some mechanisms (and genes) are (in)active in the sample. A lot of genes behave independently from the experimental setup and are therefore less interesting. If in an experiment two types of cancer are compared, correlation measures (or PCA) can be used to find genes associated to each cancer type. Machine learning techniques can be used to learn how to distinguish these cancer types and diagnose new patients.

Bayesian networks All kinds of machine learning techniques have been used and each one has its own advantages and flaws. In this thesis we will use techniques to learn Bayesian networks. One of the advantages of learning Bayesian networks is they have well understood statistical foundations. Clear methodologies are incorporated enabling it to learn from the noisy conditions of gene-expression data. A second advantage of Bayesian networks is the appealing graphical representation of the result.

Analyzing microarray data with learning Bayesian networks can potentially give several kinds of biologically interesting results:

- classification of samples into biological classes,
- find disease related genes,
- show gene to gene relations,
- general overview of relations and processes in which genes are involved.

In short: "a Bayesian network represents the joint probability distributions of a set of variables and captures the dependencies and conditional independencies between variables in a graphical manner" [FLNP00].

With gene expression data, modeling of dependencies and conditional independencies between variables (genes) can be done as follows. Suppose we have gene A, B and C. Let's say gene B is up regulated by A and C is up regulated by B. We can represent these relationships in a Bayesian network where A is the parent of B and B is the parent of C :

$$A \rightarrow B \rightarrow C.$$

We say that B is dependent on A.

Suppose we measure that B is highly expressed. We can reason with the network that C is more probable to be highly expressed as well. Our network is also designed such that knowing that A is highly expressed doesn't matter anymore for the probability that C is highly expressed if B is given. In this case C is conditionally independent of A given the expression value of B.

Algorithms have been designed to automatically construct a Bayesian network that represents dependencies and (conditional) independencies within a dataset. These algorithms are known as Bayesian network learners.

1.4 Goals and thesis overview

In this thesis we use learning Bayesian network algorithms to analyze microarray data. In most of the existing projects applying Bayesian network learners to microarray data, large groups of people from very different science fields are working together to obtain interesting networks. The methods used are complex, specialized and therefore not usable on a wide-scale for microarray analysts and research groups. Though several implementations of standard learning Bayesian network algorithms are publicly available, nobody, as far as we know, has fully explored using these methods for the analysis of microarray data.

We will explore the possibilities of applying well-known, and relatively easy to use, implementations of learning Bayesian networks to the field of microarrays. Learning Bayesian network algorithms implemented in Bayesware Discoverer [Ram99] and the BNsoft package made by Cheng [CBL97a] shall be used. These algorithms are subdivided in classification network learners and general network learners.

Of course, a complete answer is not realizable within our time-scope. We have focussed on a selection of features of the algorithms — such as classification, gene selection and discretization — and used them to analyze publicly available and as large as possible cancer microarray datasets. A breast cancer dataset [VDH+02] and a multiple myeloma dataset [PCW+02] will be used. The multiple myeloma set was already used to learn a special type of Bayesian networks, but we will extend this experiment. For both experiments we try to obtain interesting networks, small subsets of interesting genes and high classifcation accuracies.

Overview First of all, in Chapter 2, a selection of publications related to our study is discussed. An overview of machine learning techniques used for microarray data analysis is followed by the discussion of methods used to construct networks from microarray data. The last part of Chapter 2

presents a collection of publications on learning Bayesian networks (LBN) applied to the analysis of microarray data.

The third chapter is used to explain the LBN techniques and Bayesian theory in general. Examples of Bayesian networks are given to explain the concept of Bayesian networks. In Section 3.3 the algorithms used in this thesis are discussed in detail.

The microarray datasets used to learn networks are described in the fourth chapter as well as the biological questions that we hope to answer. In that chapter we will also discuss the pre-processing techniques which are used to select a subset of genes and to discretize the data.

The 5th chapter presents and analyzes the resulting networks and classification accuracies. In the last chapter the conclusions of the results are formulated as well as some ideas for future research.

Chapter 2

Scientific background

The analysis of gene expression data obtained in microarray experiments has been of great interest in the research areas of pattern recognition, machine learning and statistics. Researchers all over the world are attracted to the problem of discovering biologically interesting information in the expression data of so many genes.

As mentioned in Section 1.3 the main problem — a problem for every analysis method in the microarray world — is the ratio between the enormous amount of genes measured per sample and the small number of available samples. All kinds of data mining and machine learning methods have been applied and extended for the purpose of analyzing microarray data.

In this chapter the methods and results of several AI techniques in microarray context are discussed. The biological information found by these methods is different for each method. Section 2.2 deals with learning gene networks from microarray data. Section 2.3 analyzes the few studies published about learning Bayesian networks in a microarray environment. First, we discuss standard machine learning terms and algorithms.

2.1 Machine learning methods to analyze microarray data

Before we go on we will distinguish the machine learning methods used by the biological information they can find. First of all we have classification techniques that find potentially interesting genes related to a certain class and use the expression patterns of these genes to classify new arrays. Such techniques are called *supervised* learning because information on the biological class of the sample is given to the learning method. Not surprisingly, the other methods are called unsupervised learning. These methods can cluster genes or persons that have similar expression patterns.

Before the data is given to us, AI-scientists, a lot of work and manipulation of the data has already been done. Image analysis and data preprocessing steps to transform the microarray scans into a clean normalized set, representing the expression levels of the individual genes, will not be discussed here. However, a large part of the current microarray research is involved in these processes and are therefore worth mentioning.

2.1.1 Supervised classification methods

Suppose we have a typical microarray experiment in which the expression profiles of $20,000$ genes from 50 prostate cancer patients and 50 healthy persons is measured. Genes related to prostate cancer are hoped to be found, and classification techniques are used to try to predict that a patient suffers from prostate cancer. Classification methods are used to distinguish between disease subtypes but also for treatment response and disease prognosis [SBC+02].

To check if a sample taken from a patient displays healthy or prostate cancer patterns we could use techniques as (k-) nearest-neighbour, voting, decision trees, support vector machines (SVM), neural networks and a lot more. Central problems of classification methods in microarray context are the reduction of the dimensionality of the problem space (reduction of the number of genes), coping with missing data and estimating the classification accuracies. These issues are discussed next, and in the latter part of this section several examples of supervised learning methods are given.

Central problems In machine learning and pattern recognition we often represent the gene expression measurements made in an experiment as a m dimensional space (genes) with n data points (samples) in it. In the prostate cancer case we get $m = 20,000$ and $n = 100$. Using gene selection methods we can reduce the dimensionality m of the so-called problem space, which is necessary, for computational reasons, for some of the supervised and unsupervised learning methods.

Most gene selection methods, in the supervised case, are based on statistical measurements to find the genes that differ most between, as in our example, the prostate cancer and healthy samples. The n-fold approach is a popular method to tackle this problem. In the n-fold method genes are selected that differ in their mean value between classes at least n times. Variations of this method are also often used, see [SKR03]. In [DFS00] genes are selected by the ratio of their between-class to within-class sums of squares. Golub *et al.* [GST+99], and many others, select informative genes by the correlation of their expression pattern with the class distinction. A pre-set threshold, based on the significance of the correlation, is used to find the potentially interesting genes. Other methods to select genes that are differentially expressed across the different classes can be found in [SBC+02],

but the authors mention that a search for better ways of measuring differences in the expression of a given gene in two tissues is far from complete exploration.

The problem of missing data in gene expression measurements can be solved by using nearest neighbour or expectation maximization (EM) techniques (also used for cluster techniques). Troyanskaya *et al.* [TCS+01] conclude that the nearest neighbour technique can compute robust and accurate estimates of the missing value of a gene.

The issue not discussed so far is, once we have chosen and learned a classifier, how the performance of a classifier can be estimated. If we learn a classifier we usually test it on samples the algorithm hasn't seen before. Basically, the gene expression dataset is divided into a test set and a training set. A classifier is constructed on the basis of the training set. This classifier is then used to classify the samples in the test set. The percentage of correctly classified test samples is an estimation of the true classification accuracy of the constructed classifier. The test set should be independent from the training data and may not be used in the learning procedure.

Because the number of samples is usually very low in the microarray world, leave-one-out cross-validation is popular. In this method a classifier is repeatedly constructed on, for our example, $100 - 1 = 99$ samples and tested on the sample left over. The mean error (equal to the percentage of misclassified samples) is the estimation of the true error of the classifier learned on 100 samples. The variation leave-N-out is also often used and seems more reliable [Val02].

Some of the machine learning methods create classifiers that seem to be "overfitted" to the training data and have a very poor classification performance on the test data [BSB+03]. This means that the underlying structure of the data is poorly modelled but only the specific characteristics of the training set are learned. Complex models tend to perfectly classify training data while making a lot of errors on the test set. The challenge for machine learning techniques is to generalize the training data such that unseen data can be classified correctly.

Examples of supervised machine learning algorithms Now we have, after treatment with the methods discussed above, a complete data matrix with a reduced dimension p ($p < m$). With this dataset we can use supervised classification techniques. A short description of some of the algorithms will give an idea about the current situation of research.

The nearest neighbour techniques use distance measures to find a sample (with a known class) that is closest to the, unknown, sample in the problem space. The unknown sample inherits the class of its nearest neighbour. In k-nearest neighbour the unknown class of the new sample is predicted by the most common class among its k nearest neighbours. Nearest neighbour

techniques are popular and suited for classification of microarrays because they are easy to implement and understand, and the method seems to be able to compete in performance with other, advanced methods such as neural networks and SVM [BSB+03].

Because nearest neighbour techniques calculate distances between two points in space, the choice of the distance measure has significant consequences for the nearest neighbour that is found. Special scaling of the axes and trying several kind of distance measures (such as the Euclidian and Manhattan distances) give different results [Qua01]. Many other supervised and unsupervised methods also require the specification of some distance measure.

Another relatively simple approach is to use a voting algorithm. Each gene votes for a certain class and the class that receives the majority vote is selected as being the class of the unknown sample. For example, if a gene is highly expressed within prostate samples and has a low expression in healthy patients, then the gene will vote for prostate cancer if the gene has a high expression in an unknown sample. Golub *et al.* [GST+99] and Page *et al.* [PCW+02] use slightly adjusted voting algorithms that can predict the state of patients with high accuracy, ranging from 90% to 100%.

Studies on using support vector machines (SVM) to classify the class of a sample based on its gene expression pattern report promising results. SVMs solve the problem by mapping the gene-expression patterns in the problem space non-linearly into a higher-dimensional space, in which kernel functions are used to measure distance such that the data can be separated into two classes by a hyperplane with optimal margins [Qua01]. Non-linear robust classifiers can be learned with high predictive accuracies. A serious drawback is that the choice of kernel function affects the learned hyperplane and classification accuracies [SBC+02].

Comparative studies of machine learning methods to classify biological samples on basis of their gene expression pattern show no real winner, see [Val02], [PCW+02] and [BSB+03]. Some prefer SVM for its robustness, while others prefer nearest neighbour for its easy interpretation.

2.1.2 Unsupervised cluster methods

Reasons why a medicine does not work can be that a disease is subdivided into several, unknown, classes with different types of behavior. Expression patterns of genes have proven to reveal subtypes of diseases [SEBB98].

An example of such an experiment is that researchers take 100 prostate cancer samples and look for subsets (clusters) of genes that exhibit similar expression patterns across samples. A cluster of similar genes can represent biologically related genes —which may therefore be functionally related [SEBB98]— and can reveal biologically related samples. Clustering techniques are used to find clusters of genes or samples that show similar-

ity in their expression pattern. Time-course experiments, where expression profiles of an organism are measured at several points in time, are also analyzed with clustering methods to find genes with similar expression patterns over time. A collection of hierarchical clustering methods, k-means clustering and self-organizing maps [Val02] are often used in these experimental scenarios.

K-means clustering divides the data points into k clusters. The clusters are created by moving a data point from one cluster to the other with the purpose of minimizing the *within* class distances and maximizing the *between* class distances. An obvious disadvantage is that the number of clusters must be specified in advance while usually this is not known. Another difficulty is to visualize and interpret the resulting clusters, as this is dependent on the number of clusters [TSM+99].

Hierarchical clustering resolves the problem of interpreting the clusters because the clusters can easily be visualized. In contrast to K-means clustering, hierarchical clusters can show how the clusters are related with each other and how the clusters are build. See Figure 2.1 for an example of the visual advantage. By calculating the similarity between each cluster, the clusters that are most similar are merged. The algorithm starts with single genes as a cluster and ends with one cluster; a hierarchy of clusters is created[1]. A demonstration of the possibilities of hierarchical clustering is given in [SEBB98]. The problems, however, are the choice of a similarity measure and the meaning of a cluster at each level in the dendrogram [Qua01].

Finally we mention a popular clustering method called the self-organizing map (SOM) [Koh95]. The method is somewhat complex with respect to the other methods, so we shall not go into detail. A variety of studies, including [GST+99] and [TSM+99], have found interesting yeast cell cycle clusters and disease subtypes respectively.

With SOMs, k-means clustering and hierarchical clustering the problem is to make a choice on the number and size of the clusters in the problem space. Although some people have tried to tackle the problem no conclusive solution has been proposed. D'Heaseleer *et al* [DWFS99] state that "Each clustering method imposes some underlying structure on the data and therefore no objective score of a cluster can be given without mentioning this assumption". The visual results of a cluster can be misleading if the underlying structure is not taken into account. We must also note that, especially with the noisy microarrays, the resulting clusters might look totally different when the expression data is slightly modified. The robustness of the resulting clusters are important when the clusters are analyzed.

As knowledge about the function of genes grows the integration of this

[1] Other algorithms for hierarchical clustering exist. The divisive method for hierarchical clustering works the other way around and starts with all the genes and then splits it into clusters which do not look alike.

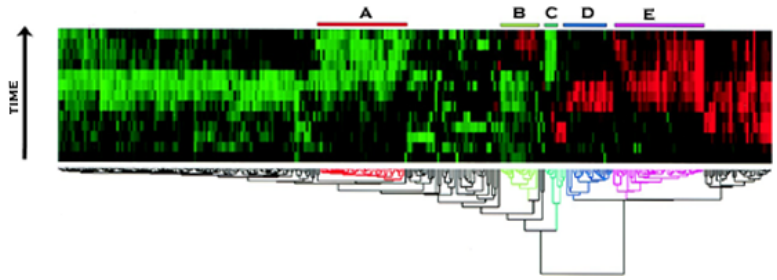

Figure 2.1: Clustered display of data from time course of serum stimulation of primary human fibroblasts. Experimental details are described in [SEBB98]. Note that similar expression profiles are grouped together. The height of the branches in the dendrogram represent the similarity between clusters (short disctance means very similar). The genes in the subcluster annotated by A are also biologically related because they are involved in cholesterol synthesis. Cluster B contains cell cycle genes and E wound healing genes.

knowledge with cluster visualization becomes important. Partially supervised methods, with a priori knowledge about the function of genes integrated with the clustering method, are recommended by Quackenbush *et al.* [Qua01]. We could, for example, start with clusters of genes which are known to be related in a certain process and then check the similarity of the expression of this process with other processes.

2.2 Gene networks

The ultimate goal microarray researchers have in mind is to decipher the precise connections of the genetic network: for each gene, they want to know which other genes and functions it influences, and in what way. The problem is how to infer gene networks from the small gene-expression datasets. Clustering techniques already show some information about the regulation of groups of genes but novel functional relations of genes between, and within these groups are hard to discover.

Because of computational reasons and the small availability of samples, simplifications and assumptions must be made to infer networks of gene relations from our microarray datasets. The resulting networks can reflect only a subset of types of (mathematical) relationships and model only a part of the genome.

The choice of model, and simplification of biological processes, is best illustrated by comparing the random Boolean network of Kauffman [Kau93] against the biochemically realistic network constructed by Arkin and Adams [MA97]. The random Boolean network, in which genes can be either ON or

OFF, can represent some biologically interesting gene interactions such as self-organization. Such networks are mathematically well formulated, and allow examination of thousands of genes. In a realistic approach, where genes are continuous and full biochemical interactions with stochastic kinetics[2] can be modelled, only a few genes are allowed in the model because of computational limitations. The relationships in the resulting networks can be very complex and hard to analyze.

A method to infer (or learn) gene networks from microarray data (often referred to as reverse engineering) worth mentioning uses differential equations. Mjolness [MSR91] showed differential equations able to simulate developmental processes in the fruit fly. In differential expressions, the expression of a gene is calculated by the sum of weighted expression levels of other genes. To find the best weight parameters optimization procedures are used, such as least squares. A small known part of the gene network of a rat is reconstructed from gene expression data in [WH00]. The determination of the parameters, using genetic algorithms, show great accuracy. Usually a set of differential expressions is used, motivated by biological processes. Due to the complexity of these functions, a lot of parameters have to be estimated and only a small part (6 genes for example) can be modelled.

To infer a network from a microarray dataset a choice has to be made on which type of, mathematical, model should be inferred *and* what kind of experimental setup should be used. Any attempt at predictive data analysis and model building critically depends on the scope and quality of the input data [DLS00]. We can use time-course experiments, comparative experiments and gene-knockout experiments. Time-course experiments generally contain less information than measurements under dissimilar environmental conditions, but can reveal information on the dynamics of a process. In gene-knockout datasets a gene is externally put to sleep in each experiment. Effects of this sleeping gene on gene expression can be identified and modelled [Wag02].

If we want to put more genes in the inferred networks more networks are theoretically possible and more parameters have to be learned or estimated. In consequence we need more data as evidence to estimate all these parameters and find the optimal network [Dut99]. Narrowing the range of possible models by extra constraints and simplifying the networks can make the search for the model faster. Using clustering techniques to find subset of genes is a possibility to narrow down the search space. In [DLS00], estimates on the amount of data needed to construct Boolean and continuous networks are made. The authors stress that comparing data from different experiments should be made possible. In this respect standardization of

[2]The time intervals in genetic processes can vary widely across otherwise identical cells, as a result of stochastic processes. Mechanisms as the need of collision between reagents to start a process explain this stochasticity. Therefore the inclusion of stochasticity seems to be necessary.

microarray experimentation is important[3].

A balance between the mathematical complexity of a network, the amount genes in the network and the data available is needed in order to infer a meaningful network. For now, gene networks constructed from microarray data mainly lead to *hypothetical* relationships interesting for further research.

2.3 Learning Bayesian networks

In the previous sections we have discussed techniques that enable us to answer the following biological questions:

- Which specific genes are related to the class of a sample?
- Can we diagnose a new sample on the basis of its gene expression pattern?
- Are there groups of functionally related genes?
- Can we construct a network that represents the relationships between genes from microarray data?

Learning Bayesian networks (LBN) could have been placed in each section because variants exist to answer these questions. Classification networks can be used to answer the first two questions and general Bayesian networks can help to find answers to the last two questions.

Not so many studies have been published yet on Bayesian networks to analyze microarray data, but the number of studies is growing rapidly. In the first part of this section we discuss studies on Bayesian network classifiers. These networks are especially learned to predict the value of a certain variable, such as the disease-state variable, and are usually supervised. The second part presents studies on unsupervised learning Bayesian networks that reveal functionally related groups of genes and how genes interact within these groups.

Several LBN concepts might still be abracadabra. A more detailed discussion on LBN follows in the next chapter. A short description of a Bayesian network, as given in Section 1.3, might be helpful to understand some of the terms.

2.3.1 Bayesian network classifiers

To learn a Bayesian network classifier (BNC), the pre-processed gene expression dataset, including information about the class of samples, is given and a network that can predict the class of a new sample is the result. This

[3]The issue of standardization is recognized by the introduction of a standard microarray object model and data-format, MAGE-OM and MAGE-ML.

classifier has the advantage of giving a graphical representation of the result and potentially providing insights into the disease by showing relationships between genes.

As with other network reconstruction techniques, the number of genes (nodes) in the network can't be very large. The total number of theoretically possible networks increases super exponentially with the number of nodes in the network [Neo04]. Searching for the best network is a difficult task. Learning BNC methods can already reduce the search space because they only need to consider genes that are related to the resulting class of a sample. Even with this reduction of the number of possible networks, the number of genes can't be very large and ranges roughly between 10 and a 1000. Supervised gene selection methods, discussed in Section 2.1.1, are often used to achieve the reduction from the thousands of genes measured in a microarray experiment to the maximum of genes allowed by the BNC learner.

Zhang *et al.* [ZH03] learn BNCs with a selection of 50 and 30 genes. Before the gene expression data was given to the BNC learning algorithm the continuous dataset needed to be discretized because of the nature of their BNC learners (probability tables are used). Discretization methods based on mean or information values assign bins to the, originally, continuous data. Of course discretization implies loss of information, and this is a drawback of BNCs.

The results obtained in [ZH03] are good. A comparison was made between other state-of-the-art techniques; the Bayesian classifiers could achieve competitive prediction accuracy. The leukemia set[4], the same as in [GST+99], and a colon cancer set[5] were used. Classification accuracies, in the leave-one-out cross-validation was 97.22% for the leukemia set and 85.48% for the Colon cancer set. The authors constructed a variation on the BNC learning algorithm to make the classifier more robust. Instead of a single best network they used an ensemble of networks. Ensembles consisting of 5, 7, 10, 15 and 20 networks were constructed from the cancer datasets and the best ensemble was selected. The authors mention that, besides a high classification accuracy, the resulting networks also generated relationships, potentially interesting for further research.

Helman *et al.* [HVAW02] propose a Bayesian network classification methodology for microarrays. They propose to select a variable number of plausible networks and try to blend them in such a way that classification accuracy is optimized. Special methods to search through the networks, select genes and to select and blend the resulting networks are implemented. Their methodologies were tested on the same leukemia and colon cancer datasets

[4]The leukemia set consisted of 72 samples; 25 acute myeloid leukemia and 47 acute lymphoblastic leukemia samples, see http://www.genome.wi.mit.edu/MPR

[5]The colon cancer consisted out of 62 colon tissues; 22 healthy and 40 tumor tissues, see http://microarray.Princeton.edu/oncology/affydata

as mentioned before. In contrast with the previous study, Helman subdivided the dataset into a training (about 70%) and test set. The authors tested with different sets of networks and with different type of gene selection methods. The best performing networks occur when gene selection is based on the informative power of a single gene and a large number of networks (about 300) are merged. The authors selected the best classifiers based on leave-one-out cross validation error on the training set. The colon cancer test set was classified with 92% accuracy (training set had 84%), and the accuracy on the leukemia set was 95% on both the test and training set. One important conclusion is that the networks have generalized very well; test results are as good as the results on the training data. The authors do not go into details of the structure of the constructed networks and concentrate solely on the classification performance of the BNC algorithms.

We conclude that BNC learners show some promising results. However, the methods developed in these studies are not publicly available in an easy to use program. In chapter 4 another article on Bayesian network classifiers is discussed and will be partially reproduced to get a better understanding of the possibilities of standard BNC learners.

2.3.2 General Bayesian networks

In the microarray world the best-known study on using Bayesian networks to analyze microarray data is done by Friedman *et al.* [FLNP00]. His group constructed a network that shows interactions between genes and groups of genes. The yeast cell cycle dataset created by Spellman *et al.* [SSM+98] was used to learn a network. The dataset had measured the expression of 6177 genes during its cell cycle at different points in time. One of the advantages of using Bayesian networks to reconstruct networks from microarray data, according to Friedman *et al.*, is that they are able to describe complex stochastic processes and provide clear methodology for learning from (noisy) observations.

In learning general Bayesian networks (GBN), in contrast to BNCs, every relationship between two variables (genes) is relevant. The total possible networks is immensely large for a relatively small set of genes, see Section 3.2. Friedman *et al.* managed to learn a network from the expression profile of 800 genes (out of the 6177 originally measured). These 800 genes were selected by unsupervised clustering, done by Spellman *et al.* in [SSM+98]. The Bayesian network learning method is described below.

Two-hundred Bayesian networks are learned and only two features of each gene in each network are stored for the final network. Each network is learned on a slightly pertubated dataset by a special LBN algorithm they call the Sparse Candidate Algorithm (SCA). The set of possible parent relations (candidates) for a node are preselected by SCA on the basis of simple statistics, such that the search space of all possible networks is reduced. For

each gene in each learned network the set of genes on which the expression level of the gene *directly* depends is selected as the first feature. The second feature concerns some type of relationships, where *intermediate* genes relate two genes. Both relationships are stored for the final network. This final network can have relationships between genes in which the direction of the relation is undecided, because not enough data supports either direction. The level of confidence in a feature is decisive for the resulting final network.

With the Bayesian method, Friedman *et al.* obtained gene clusters resembling the clusters found by Spellman *et al.* using hierarchical clustering. But Friedman *et al.* were able to reveal more features in and structure of the dataset[6]. Several dominant genes were found: genes that have a lot of direct relationships within a gene cluster. A large part of the gene pairs that had a high-confidence relationship were believed to be biologically related in function, based on literature.

A few other articles propose to integrate prior knowledge into the Bayesian networks (about experimental conditions or gene function) or to use dynamic Bayesian networks to analyze time-course data [OPG02]. The possibility of integrating non-expression knowledge is an advantage of the Bayesian approach. Experts could make an a priori Bayesian model of a certain process [TDO+03] and update this network with experimental microarray data; patient data (as age and gender) could be included as well. Dynamic Bayesian networks have the advantage of being able to model feedback loops because a gene is modelled separately at each time point. Remember that a Bayesian network is by definition an acyclic graph. Difficulties on the complexity of learning dynamic Bayesian network (DBN) algorithms and scarcity of time-course data in each time point make DBN not very useful yet.

2.4 Concluding remarks

We have seen results obtained by learning Bayesian network methods are to be taken seriously, with respect to other analysis methods. Many microarray analysts are not aware of the capabilities of LBNs for several reasons. First of all the standard analysis tools (commercial and non commercial) have not included the LBN algorithms; they focus on clustering techniques, statistical hypothesis testing and gene ontology integration. The second reason is the biological background of most of the analysts. They lack time and knowledge to learn, implement or apply the specialized methods proposed in the literature on LBNs for microarray data. This thesis will therefore use relatively simple, *accessible*, and publicly available implementations of LBNs to explore their usability for microarray analysis.

[6]The resulting, quite complex, network can be viewed on http://www.cs.huiji.ac.il/~nirf/GeneExpression/top800.

Chapter 3

Learning Bayesian networks

Nearly 250 years ago, in 1764, Thomas Bayes wrote a theorem about chance and probability. In artificial intelligence, Bayesian thinking has played an important role in modelling, learning and reasoning with uncertainties.

Next we discuss Bayesian calculus and Bayesian networks [Pea88] in general. In Section 3.2 several issues about learning a Bayesian network are discussed and the last section presents the algorithms used in this thesis.

3.1 Bayesian networks

Basic concepts in Bayesian theory must be explained before we move on to Bayesian networks. Although there are several ways to explain Bayesian calculus we will focus on the concepts of (in)dependence and conditional (in)dependence throughout this section and chapter.

Conditional probabilities Conditional probability is formally notated as: $p(A|B) = x$. In words this means that given the event B, - and supposed that everything else known is irrelevant - the probability of event A is equal to x. An event could be a variable (gene) taking on a certain value (high expression). The remaining part of this chapter only deals with variables instead of events.

Two variables X and Y are independent if $p(X|Y) = p(X)$ for all values of X and Y, otherwise they are dependent. This means that the probability distribution of X is not dependent on Y. In a probability distribution the chances of X taking on a certain value are given. Imagine that the distribution of gene X is .2 for high expression and .8 for low expression. If gene X and gene Y act independently, then for all values of Y the probability distribution of X will stay the same.

If X is dependent on Y ($p(X|Y) \neq P(X)$), the probability distribution of X is different for each value in Y. Suppose we found that the expression of gene X is up regulated by the expression of gene Y. So our belief that

gene X has an high expression depends on the expression of gene Y. If we measure an high expression rate for Y, we believe that X is probably highly expressed as well. In this case X is dependent on Y.

Conditional independence describes situations in which variables X and Z are independent *given* the value of variable Y. Conditional independence holds when for all values of X, Y and Z we have: $p(X|Y) = p(X|Y, Z)$. Let's extend our previous example with gene Z which up regulates gene Y but does not regulate X directly. Suppose it's given that Y is highly expressed in a sample. We have the distribution for the probabilities of the values in X given Y. If this distribution of X doesn't change when we come to know the value of Z, we say X and Z are independent *given* the value of variable Y. But we can also reason that if we don't know Y but do know that Z is highly expressed, the probability distribution of Y changes (Z up regulates Y, so Y is more probable to be highly expressed)). In turn, because X is up regulated by Y, the distribution of X also changes. So X is dependent on Z and at the same time conditionally independent of Z when the value of Y is given. Conditional independence is an important principle in Bayesian networks, because knowing the conditional independencies in a dataset can greatly reduce the complexity of networks.

At last we have conditional dependence which describes dependence between X and Z given Y (formally we have $p(X|Y) \neq p(X|Y, Z)$ for all or some values of X, Y and Z). Correlation measures, often used in bioinformatics, between two variables can imply dependence, although correlation is not a necessary condition [FLNP00].

Bayesian networks Now we move on to Bayesian networks and see how they can reason with the concepts stated above. Bayesian networks are graph-based models of joint multivariate probability distributions that capture properties of conditional independence between variables [Jen96]. They are also used to reason with probabilities. A Bayesian network consists of the following

- A set of variables (nodes).
- A set of directed edges between variables representing dependencies between variables. A variable (node) X is called a parent of Y if there is a directed edge going from X to Y. These directed edges must form, together with the variables, a directed acyclic graph (no directed path from a variable that leads to itself).
- To each variable X with parents $Y_1, ..., Y_n$ there is attached a conditional probability distribution $p(X|Y_1, ..., Y_n)$. If a variable X has no parents, the probability distribution $p(X)$ is attached.

We can view the lack of representing cycles as a restriction, especially in our biological domain where feedback cycles are thought to happen. For

time course microarray experiments dynamic Bayesian networks could be a solution because a network can be made for each time point. The directed edges between the variables in a network can sometimes be seen as causal influences [SGS93], but causality in Bayesian networks (BNs) is still under discussion and not generally accepted.

BNs can cope with variables that have continuous values but we use methods that can only cope with discrete data. When BNs contain discrete variables, conditional probability tables (CPT) are used instead of conditional probability distributions (CPD). The main motivation for using discrete values is that incorporating continuous data in methods to learn Bayesian networks is still somewhat difficult and can lead to biased learning methods [FLNP00]. Discretization of continuous (gene expression) data is reasonably unbiased. Also, most of the standard Bayesian network learners can not use continuous data.

A simple example of a Bayesian network is given below. This example illustrates how Bayesian networks capture conditional independencies and how we can reason with the networks.

Example 1 Suppose we have three genes X, Y and Z, which can take on mutually exclusive values h (high expression) or l (low expression). In figure 3.1 the network is shown (qualitative part), including probability tables (quantitative part).

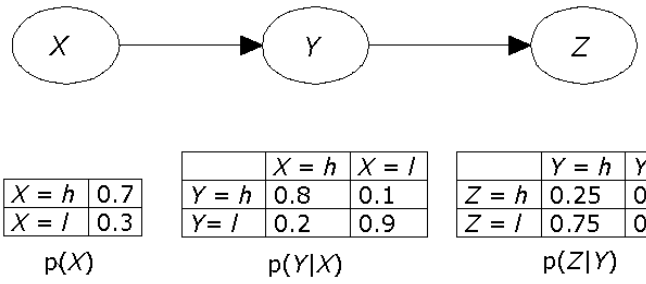

Figure 3.1: A simple Bayesian network structure containing 3 variables. Conditional probability tables are also shown.

The graph is constructed such that gene X can influence gene Y directly, and gene Z indirectly through Y. Gene X has no parents and the probability table simply consists of our prior belief on the probability distribution of gene X (also called prior probability, $p(X)$). Because X is a parent of Y we have a conditional probability table $p(Y|X)$. The CPT consists of probabilities for the expression values of gene Y given the expression value of gene X. If we add the values in a column of a CPT we always get 1

($p(Y = h|X = h) + p(Y = l|X = h) = 1$ because Y is certain to be either h or l). The prior probabilities $p(Y)$ and $p(Z)$ are also given by the CPT tables, although indirectly (by marginalizing Y out of $p(Y, X)$ which in turn can be calculated by the *fundamental rule*: $p(Y, X) = p(X)p(Y|X)$).

If evidence on the value of a gene is given, we can update the conditional probability tables and prior probabilities of the other genes using the fundamental rule and Bayes' laws of probability. Thomas Bayes introduced his theorem in 1764. The equation is as follows:

$$p(A|B) = \frac{p(A)p(B|A)}{p(B)}.$$

Suppose we only know that the expression of Y is h, so $p(Y = h) = 1$ and $p(Y = l) = 0$. In this case we can update the probabilities of both X and Z. For Z we get $p(Z|Y = h) = (0.25, 0.75)$ (because now $p(Z, Y = h) = p(Z|Y = h) = p(Z)$). To update the probabilities of X we use Bayes theorem' to calculate $p(X|Y = h) = (0.95, 0.05)$:

$$p(X = h|Y = h) = \frac{p(X=h)p(Y=h|X=h)}{p(Y=h)}$$
$$\Rightarrow$$
$$p(X = h|Y = h) = \frac{.7*.8}{.8*.7+.1*.3} = .95$$

We see that the probability of $X = h$ is increased (from 0.8 to 0.95). The increased probability of $p(X = h)$ can be viewed as an 'explanation' for the high expression value of Y.

Once we know Y, extra knowledge on the value of X can not change anything for the probabilities of Z because gene Y blocks the information from X to Z. The conditional independence $p(Z|Y) = p(Z|Y, X)$ is modelled in this network and we say that X and Z are conditionally independent given Y.

Example 2 The Bayesian network in Figure 3.2 models the factors and symptoms involved in the diseases pneumonia and influenza. Artificial data was used by Abbas *et al.* [AMRM04] to estimate the CPT tables of the network. With the network they reason, for example, that the chance of influenza increases given a persons income or ethnicity.

Bayesian networks can be a lot more complex than the ones described above. We showed that the information flow in all BNs (updating probabilities) is blocked by conditional independencies or passed on by (conditional) dependencies. The conditional independencies in complex networks are identified by a few rules called d-separation. The resulting conditional independencies are not as obvious as in the previous example. Fast methods to update CPTs in larger BNs have been developed [Jen96].

We have discussed some principles of Bayesian networks and Bayesian calculus. Readers should by aware we are giving only a very short introduction to BNs and not every aspect of BNs can be treated. Important

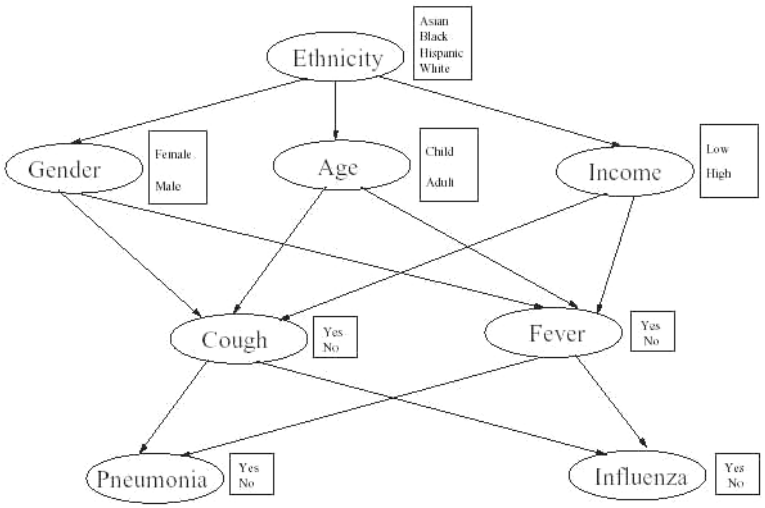

Figure 3.2: A Bayesian network representing disease factors and symptoms for influenza and pneumonia.

and intersting subjects such as hidden nodes and fast inference-methods are skipped. Jensen [Jen96] wrote an easy to read introduction to Bayesian networks. The important message here is that BNs can be used to model (in)dependencies between variables in such a manner that we can reason with their probabilities given new knowledge and using Bayesian calculus.

3.2 Learning Bayesian networks

Originally Bayesian networks were constructed by experts. Prior probabilities and CPT values were chosen on the basis of the knowledge of experts. The last decade or so, methods to automatically learn Bayesian network structures from a given dataset have improved. The increased power of computers has been a major support.

Learning Bayesian network (LBN) methods need to find the set of edges between variables (structure) and the values in the (conditional) probability tables that best represents a dataset. One way to do this is to give each possible network a score such that the network with the highest score matches the data optimally. This score can be designed in several ways and can, for example, prefer simple networks. A fully connected network can represent any probability distribution (and can therefore optimally represent the data), but there are many reasons for not using a fully connected model. A fully connected model requires more memory and computations, is more sensitive to noise (overfitting) and does not find underlying independence structures.

Unfortunately, the number of possible network structures is super exponential in the number of variables, which makes it practically impossible to score each possible network. Ten variables already produce 4.2×10^{18} possible directed acyclic graphs (DAGs) [CG99]. The LBN algorithms have special ways to make the search for the best network computationally feasible. Direct restrictions on the structure of the resulting network (maximum number of parents), using search strategies (greedy search) and looking at local structures (finding optimal set of parents for each node separately) are examples of reducing the computational complexity.

The prior probabilities are usually based on the frequency with which values of variables occur in the dataset and can be designed to further reduce the computational complexity [RS01].

Depending on the purpose of the network, important structural restrictions can be made. As we mentioned in Section 2.3, BNs can be learned especially for classification. In this case, only those edges have to be found which can influence the probability distribution of the target variable, given that all the values of all the genes in the network are known (so conditional independence is important here). In the case of general networks, less structural restrictions can be made.

3.3 Methods used in our research

Many BN programs are available on the internet, and among them are several Bayesian network learners, both commercial and non-commercial[1]. We wanted to learn both general and classification networks to fully exploit the possibilities of analyzing microarray data with the existing BN learning programs. We selected two programs for reasons discussed next. The rest of this section will discuss the algorithms implemented in these programs.

By Page *et al.* [PCW+02] and by winning the KDD cup 2001[2], interesting results were obtained with the use of a LBN package developed by Cheng [CBL97a, CBL97b] called *PowerPredictor* and *PowerConstructor*. In the next chapter more information on the results of [PCW+02] *et al.* is given. The algorithms implemented are the Markov blanket classification network learner and a general network learner called CBL. The programs can be freely downloaded from Cheng's website[3].

Bayesware Discoverer, developed by Ramoni [Ram99], seemed a good choice as our second program to show the possibilities of using LBN implementations to analyze gene-expression data for several reasons. First of all it provides methods to cope with missing data. Secondly, Bayesware has two

[1] Kevin Murphy made a comparison of all the available BN software, which can be viewed on: http//:www.ai.mit.edu/~murphyk/Software/BNT/bnsoft.html.

[2] The KDD cup is an Knowledge Discovery and Data mining competition. More information can be found on http://www.cswisc.edu/~dpage/kddcup2001

[3] http://www.cs.ualberta.ca/~jcheng.

types of BNC (Bayesian network classifier) learners, the naive Bayesian and the CTan algorithm. A GBN (general Bayesian network) learner, which we shall call K2 is also available. A third reason is the nice graphical interface that helps to understand and interact with the networks. And finally, a version of Bayesware is free for non-commercial purposes. Bayesware (and *PowerConstructor* and *PowerPredictor*) can learn from datasets with approximately 200 variables and 700 samples (in the non-commercial version).

Other programs are available, but we think our selection of programs, and their implemented algorithms, to be a good representation of easy to use LBN programs. Bayesian network packages implemented in Matlab by Murphy[4] and Java are also available. The packages contain some LBN algorithms but tend to focus more on algorithms for constructing and reasoning with Bayesian networks. Also, these packages are more difficult to use, especially when the user is not familiar with programming environments. On the other hand they do have the advantages of open source software.

Next, we will discuss each algorithm separately, starting with the simple algorithms (naive Bayesian) and ending with the complex GBN learners. Each algorithm has its own way to cope with the problems described in 3.2 and have made different restrictions and assumptions.

3.3.1 Naive Bayesian classifier

The structure of a naive Bayesian classifier (NBC) is fixed and is shown in figure 3.3. The root node C represents the target variable that needs to be predicted (type of cancer for example). All the child variables A_i represent the attributes (genes in our case). The structure implies that all the attributes are conditionally independent of each other, given the value of the target variable (formally notated as: $p(A_i|C) = p(A_i|A_j, C)$ and $i \neq j$).

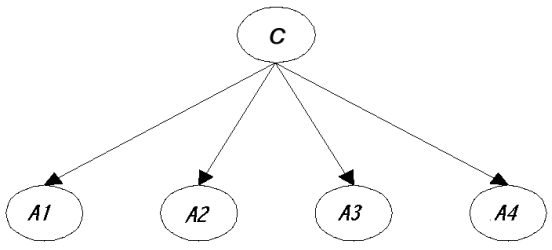

Figure 3.3: An example of the structure of a naive Bayesian classifier.

[4]Kevin Murphy implemented BN algorithms, and some LBN algorithms, in Matlab. This package can be downloaded from http://www.ai.mit.edu/~murphy/software/BNT/bnt.html.

No structure has to be learned. A NBC only needs to learn the values in the conditional probability tables (CPT) of the attributes ($p(A_i|C)$) and the prior probabilities of the target variable ($p(C)$).

The estimation process of the CPTs becomes relatively easy. Remember that in the supervised learning procedure we have a training set of attribute values for which we know the values of the target variable. Because conditional independence is assumed between the attributes, given the target variable, the NBC can estimate each row of each CPT independently. Frequencies of the values of the variables in the given dataset are used in this process.

Once all the entries in the CPTs are estimated by the NBC, it can be used to predict the most probable value (or class) of the target variable, given the values of the attributes (gene expression values of a patient for example). $p(C)$ is adjusted for this set of values of the attributes and the value (class) for C with the highest probability is chosen as the best option.

The implementation in Bayesware Discoverer also makes use of a missing value algorithm called *bound-and-collapse* [RS98, RS01]. The estimation of the probability tables is modified and results in probability *intervals*. These intervals are based on the instances in the dataset that are given and that are missing. This results in using scoring functions for the probability intervals for $p(C)$ making classification possible. As a consequence the value of the target variable remains sometimes undecided.

3.3.2 CTan classifier

The strong independence assumptions made by the naive Bayesian is often somewhat unrealistic. The CTan tries to avoid these assumptions, maintain computational simplicity and outperform the NBC in classification accuracies.

Friedman *et al.* argue in [FGG97] that "the performance of a BNC may improve if the learning procedure takes into account the special status of the class variable". They propose to ensure this by setting an edge from the target variable to each attribute, just like the NBC. However, the BNC constructed by CTan is augmented with edges among the attributes. Figure 3.4 shows an example.

The, intractable, problem how to find the best set of edges is made computationally feasible by imposing some restrictions on the structure. The most important restriction is that each attribute is allowed to have at most one other attribute pointing to it. This restriction allows us to calculate an optimal tree-shaped Bayesian network for the attributes only, based on the Chow and Liu [CL68] maximum spanning tree algorithm.

Edges between two attributes, given class in the tree are scored by the conditional mutual information principle described in [FGG97]:

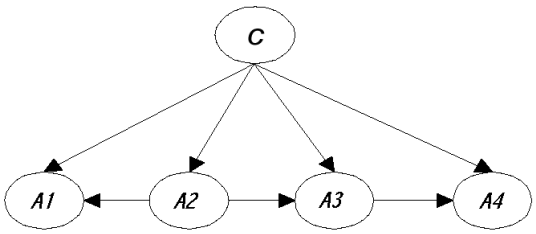

Figure 3.4: A simple TAN structure. Note that the attributes $A1, A2, A3$ and $A4$ form a tree.

$$I_P(A_i; A_j|C) = \sum_{A_i,A_j,C} P(a_i, a_j, c) \log \frac{P(a_i,a_j|c)}{P(a_i|c)P(a_j|z)}$$

In the next steps the maximum weighted spanning tree is build with the attributes; first a complete undirected graph is build and then the subset of edges forming a (undirected) tree structure which maximizes the sum of the scores is found. The directions of the edges between the attributes are chosen merely to obtain a tree structured graph. One attribute in the undirected graph constructed by the Chow-Liu algorithm is chosen as the root attribute. The directions of all edges are set outward from it. The directed tree is transformed into the final BNC by adding the naive structure. So in the end we have constructed a tree-augmented naive Bayesian network (TAN)[5].

Learning the parameters (probability values in the CPTs) of this TAN is done slightly different than in the NBC, mainly because we need to learn more values (more edges cause CPTs to grow). A smoothing parameter is introduced to cope with those entries in the CPTs for which little evidence is given[6] in the dataset. The structure is not affected by this parameter.

3.3.3 Markov Blanket classifier

While the CTan algorithm still imposed some structural restrictions on the resulting BNC, the MB (Markov blanket) based BNC is an unrestricted BN. The target node is not treated differently in the structure learning process.

To construct a Markov blanket (MB) classifier, the structure of a general Bayesian network (GBN) must be learned. How this GBN is learned is discussed in Section 3.3.4. For now we assume we have a GBN matching the given dataset and discuss the MB principle shortly.

[5] CTan stands for Construct-TAN.

[6] Suppose we have an edge from gene A to gene B. If in the dataset only one out of 100 examples exists where both gene A and B are highly expressed. In this case there is only little evidence for $P(B = h|A = h)$.

The target node in the GBN is, given the values of all the attributes, only dependent on those attributes that belong to the MB of the target node. The MB of variable C is the minimal set of variables that shield C from the rest of the variables in the model. Only attributes related directly ($A_i \leftarrow C$ and $A_i \rightarrow C$) or indirectly through conditional dependence ($A_i \rightarrow A_j \leftarrow C$) are selected from the GBN. For these attributes, the CPT entries are estimated.

We can view this selection of attributes as a kind of feature selection. For microarray analysis feature selection methods are very interesting (finding "marker" genes) and a lot of research is done in this field. Friedman *et al.* [FLNP00] state that a Markov relation between two genes indicates that both genes may be involved in the same biological interaction or process.

Because no restriction is made on the structure that can be learned, the search space is very large and overfitting is possible. Overfitting means that the learned model fits the given dataset very well but classifies new data poorly. A fully connected graph, for example, always represents a dataset optimally because each example is completely represented in the CPTs. The problem is that these networks can not classify new samples with high accuracy because they are overfitted to the original data.

In *PowerPredictor*, the MB algorithm is implemented with a special threshold, representing a statistical condition that must be met to add or remove edges when learning the GBN. The optimal value for this "confidence" threshold can be found automatically, or can be set manually. In [CG98], Cheng and Greiner mention that setting this threshold to a higher value for small datasets can improve the classification accuracies and, in general, will prevent overfitting. Automatic detection of this threshold is done by comparing the classification performance of the Bayesian networks with different confidence thresholds, within the training.

3.3.4 The CBL algorithm

In *PowerConstructor* a method to learn a GBN from a dataset is implemented. As mentioned above, the Markov blanket classifier makes use of this algorithm.

A GBN learner has a different purpose as compared to the BNC learners. It does not focus on the variable to be classified. The algorithms of the GBNs are designed to find the model that best describes the conditional independencies between *all* the variables. By doing this it makes an efficient model to calculate the joint probabilities.

The CBL algorithm, designed by Cheng [CBL97a, CBL97b], makes use of the suggestion that BN structures can be learned by identifying the conditional independence relationships among the nodes. Using mutual information tests:

$$I_P(X;Y) = \sum_{X,Y} P(x,y) \log \frac{P(x,y)}{P(x)P(y)}$$

they make a list of possible dependencies between variables. Conditional mutual information (see Section 3.3.2) is used to find the conditional independencies among the variables and these independencies are used as constraints to construct a BN. These kind of algorithms are referred to as conditional independency based algorithms [CG98]. The CBL method algorithm uses three different phases.

At first, a basic BN tree is constructed, based on the Chow-Liu [CL68] algorithm (same as in the CTan algorithm but without special treatment for the target node). In the second phase edges are added on the basis of mutual information and conditional independence. In the last phase, unnecessary edges are removed. To add or remove an edge, a certain amount of confidence must be obtained. The level of confidence that must be reached can vary and is represented by the confidence threshold (see Section 3.3.3). In the CBL algorithm the optimal threshold can not be automatically detected and has to be set manually. If insufficient confidence is obtained to choose a direction of an edge, the CBL algorithm leaves the edge undirected. Therefore, a partially directed acyclic graph (PDAG) is returned.

In the CBL1 algorithm a node ordering is necessary to reduce the number of calculations that is made. This order is based on the order in which the variables occur in the database, and can be changed by the user. A node ordering assigns a value to each node; no node can be an ancestor of a node that has a higher value. The CBL2 algorithm doesn't use node ordering but needs to do a lot more computations. *PowerConstructor* can do both and we chose for the last option, because less restrictions are made. The Markov blanket classifier uses CBL1 and ordering of the variables is based on when they appear in the database. The target variable is set as the first node in the ordering.

3.3.5 The K2 algorithm

The GBN learner implemented in Bayesware Discoverer makes use of a scoring-based algorithm. To each DAG a score is assigned and the network with the highest score is selected. Unfortunately, the total number of possible DAGs doesn't allow us to score each network.

The problem is solved by choosing a special model for the prior probabilities, factorizing the model and imposing an order to the variables [RS99]. Finding the best model is reduced to a sequence of locally exhaustive searches, picking the best set of parents for each node independently. Because the local models are often still too large and exhaustive search is not possible, the greedy search algorithm called K2 is used to find the local model.

Node ordering implies a structural restriction such as in the CBL1 algorithm (no node can be an ancestor of a node that appears earlier in the

ordering). K2 starts with the local model of a certain node and calculates the score of the model with the node having no parents. The parent which maximally increases the score of the local model is added. Until no parent is found that can increase the score of the local model, this process is repeated. This search for parents is done for every node. The K2 search strategy can end up in a local maximum and does not always find the set of parents with the best score.

The learning procedure makes use of the missing value algorithm described in Section 3.3.1 when values are missing. Also, a threshold called prior precision is added. This threshold adjusts the prior probability tables used to score the models. The threshold represents the belief in the amount of evidence that is needed to be confident about adding an edge.

3.4 Summary

Dependencies and conditional independencies can be used to construct and understand Bayesian networks. We can use data to automatically learn the structure and parameters of these networks. As explained in Section 3.3, we use a set of LBN algorithms which can be divided in general Bayesian network learners and Bayesian network classifiers. These algorithms are implemented in two publicly available programs.

Chapter 4

Microarray datasets: Experiments and data pre-processing

In this chapter we present two datasets which we will analyze with the help of the LBN programs in the next chapter. Both datasets are the result of a comparative clinical experimental setup. The goal of these experiments is to gain new insights into the diseases and identify new therapeutic targets, based on gene expression profiles. The experimenters already tried to achieve this goal by searching for accurate predictors and applying supervised machine learning methods.

The first experiment compares multiple myeloma patients with healthy persons. The second set contains samples of patients with breast cancer that develop metastases within five years and of breast cancer patients that remain free of metastases for five years. We will show the data and classification methods of both experiments in the first two parts of this chapter. The last section concerns the pre-processing of the datasets, discretization and gene selection, which is necessary to learn Bayesian networks.

4.1 Multiple myeloma

4.1.1 The microarray data and backgrounds

A large amount of microarray data is publicly available. Organized databases can be explored to view the data of microarray experiments[1]. Most of these experiments are not very large and range from 6 to 30 microarrays and

[1] A list of these organized databases is given on different websites. On http://ihome.cuhk.edu.hk/%7Eb400559/array.htm an almost complete overview can be explored. We also made a small list: http://www.science.uva.nl/~vthemaat containing some other databases as well.

from potatoes to yeast to human samples. Often, the experimental setup is designed in such a manner that a lot of different conditions are compared.

Because the study by Page *et al.* [PCW+02] has been one of the triggers to perform our research, we have chosen to use the same dataset used in their study. The gene expression profiles of multiple myeloma patients are compared to those of healthy persons and the data is publicly available at http://lambertlab.uams.edu/publicdata.htm. The dataset is relatively large; 74 newly-diagnosed multiple myeloma patients and 31 healthy persons.

Multiple myeloma is a cancer of antibody secreting cells that grow in the bone marrow. The disease is incurable and usually progresses rapidly after diagnosis. In healthy patients, bone marrow plasma cell samples show a great variety of different types of plasma cells. The samples taken from multiple myeloma only contain one type of plasma cell that has taken over the bone marrow.

From samples of the plasma cells, cRNA is derived. By hybridizing the cRNA to single channel Affymetrix microarrays (see Section 1.2.2), containing about 7,000 different genes, gene expression values could be measured. The resulting dataset contains two values per gene per sample: a continuous value (called Average difference, or AD in short) and a discrete value (Absolute call, AC). The AC value can be A (absent), P (present) or M (marginal), and is calculated by Affymetrix standards (see http://www.affymetrix.com).

4.1.2 A study on data mining the multiple myeloma data

Page *et al.* [PCW+02] tried to find "an accurate predictor of multiple myeloma that will provide insight into the disease". They test the performance of several supervised machine learning techniques as well as the comprehensibility of the result. Five publicly available techniques were used: Decision trees, boosted trees, support vector machines (SVM), voting and Bayes nets.

SVM was already shortly described in Section 2.1.1. For (boosted) decision trees C5.0 was used (we refer to http://www.rulequest.org for more information). The voting algorithm works in three steps: first all the genes are scored separately according to the entropy-based information gain score (we will explain this later in Section 4.3), then the top scoring 1% of the genes is selected (70 in our case), and finally a majority vote among these genes is taken to classify new samples. Information gain is also used to select 30 genes for the Bayes net algorithm that is based on *PowerPredictors'* Markov blanket algorithm. Because *PowerPredictor* can only learn networks with discrete values, the AD values were discretized based on the split value that gave optimal information gain (see Section 4.3).

Genes associated with immune function that had extremely low expression rates in the patients were called "trivially-accurate" genes —because cells producing these genes were 'eaten' by the cancer—, and were removed from the database. Remember that new insights into the disease is one of

the goals of Page *et al.*, and these genes provide no new insights. This set is quite large and in correspondence with the authors we also removed the genes in our further research.

The classification results are shown in Table 4.1. Leave-one-out cross-

Method	AC Only	AC+AD
Trees	90.5	98.1
Boosted Trees	96.2	99.0
SVMs	95.2	93.3
Vote	94.0	100.0
Bayes Nets	95.2	100.0

Table 4.1: Leave-one-out cross-validation results for myeloma patients vs. normal persons as they were obtained by Page [PCW+02]

validation was used to estimate the classification accuracies. This means that the learning algorithms (including the gene selection procedure) are run 105 times, and at each run they are learned with 104 samples and classify the sample that is left over. The resulting accuracies show only a few significant differences between the methods (SVMs are outperformed using AC+AD, trees when using AC only). However, the authors argue that voting and Bayes nets provide greater direct insight.

The Bayes nets for AC+AD resulted in a naive network but used only 19 features of the 30 provided (average over the cross validation runs). Remember that the Markov blanket algorithm is used, which finds a subset of genes which aren't conditionally independent given the target variable. They suspect the high correlation of each gene with the class value to be the reason for the lack of dependencies among the genes. The Bayes net learned from AC only contained a large number of dependencies among the genes and only used about 20 genes out of 30 to classify new samples.

The voting algorithm shows the ranking of the top voters directly and clearly. Both voting and Bayes nets present their results in an understandable form which can be an advantage in analysis and finding potentially interesting genes.

The decision trees were less informative, according to the authors, because most of the trees constructed in the cross-validation runs contained only two or three genes.

This experiment is partly reproduced by us (the Bayes net part) and we extended the data mining research with more LBN techniques, more discretization methods and other ways for the estimation of the classification error. The next chapter will show these results but first the second dataset is presented.

4.2 Breast cancer and prognosis of metastases

4.2.1 The microarray data and backgrounds

The well-known breast cancer experiment published by van 't Veer *et al.* [VDH+02] in *Nature* is our second dataset . This set contains samples of 46 patients whose breast cancer results in metastases within five years and of 51 breast cancer patients which remain free of metastases for five years. The relatively large set is publicly available at http://www.rii.com/publications/2002/vantveer.

Breast cancer patients with apparently the same stage of disease behave differently to treatment and in their overall outcome. Development of metastases— cancer moving to other parts in the body— in breast cancer patients without tumourous cells in the local lymph nodes[2] can not reliably be predicted yet. Development of metastases is dangerous and fatal. About $70-80\%$ of the patients receiving chemotherapy or hormonal therapy to reduce the risk of metastases by one-third would have survived without these physically and financially costly therapies. Patient-tailored therapy based on gene expression profiles is the general purpose of the experiment.

From each breast cancer patient (all patients were younger than 55), breast cancer samples were used to derive cRNA. A reference cRNA pool was made by pooling equal amounts of cRNA of each sample. Two hybridizations per tumour were carried out on microarrays containing $25,000$ genes by using the inkjet oligonucleotide technique (see Section 1.2.2). After scanning, normalization, etc., the intensity ratio between a gene in the sample and the gene in the reference pool represented the transcript abundance of the gene with respect to the reference pool.

4.2.2 Predicting the clinical outcome

The breast cancer dataset was used to develop a gene expression signature that would make prediction of metastases possible. A 3-step supervised classification method was trained with 78 samples, containing 34 patients who developed metastases within 5 years and 44 patients who continued to be disease-free over a period of at least 5 years. The remaining 19 samples (12 metastases, 7 non-metastases) were used as an independent test set.

The first step in the supervised method selected $5,000$ genes out of the $25,000$ genes, because they were significantly regulated in more than 3 tumours out of 78. The correlation coefficient of the expression for each gene with the disease outcome was calculated and 231 genes were found to be significantly associated with disease outcome (correlation coefficient < -0.3

[2] A lymph node is a small bean shaped organ connected to the lymph system by lymph channels. In a lymph node lymphocytes are created. Lymphocytes play an important role in detecting antigenes and removing bacterial cells and viruses. The lymph system is a network of channels through the whole body.

or > 0.3). In the second step, these 231 genes were rank-ordered on the basis of the magnitude of the correlation coefficient. Some of the genes known to be related to breast cancer processes such as cell cycle, signal transduction and angiogenesis occurred in the set of 231 disease correlated genes.

The third step optimizes the number of genes in the 'prognosis-classifier' by sequentially adding subsets of 5 genes from the top of this rank ordered list and evaluating its power for correct classification using leave-one-out cross-validation. Classification was made on the basis of the correlations of the expression profile of the leave-one-out sample with the mean expression levels of the samples from the metastases and non-metastases group respectively.

The accuracy of the prognosis classifier improved until the classifier contained 70 genes. When more genes were added the accuracy dropped. The classifier predicted correctly the actual outcome of disease for 65 out of 78 patients for the training set (83%). The performance on the test set was, surprisingly, even higher and predicted 17 out of 19 correctly (89%). An artificial threshold was set by van 't Veer *et al.* to decrease the number of patients falsely classified as non-metastases from 5 to 3. The number of patients falsely classified as metastases increased a little (from 8 to 12) with this artificial threshold, but these misclassifications are less fatal.

The gene expression profile outperforms all currently used clinical parameters in predicting disease outcome, according to van 't Veer *et al.*. Using current treatment guidelines, up to 90% of the patients used in this experiment are selected to be treated with chemotherapy, while 70-80% would have survived without it. The findings by van 't Veer *et al.* can provide a strategy to select patients who would benefit from a therapy other than chemotherapy.

Predicting the clinical outcome of breast cancer patients seems possible. Because the breast cancer set compares patients with apparently the same stage of disease, and the multiple myeloma experiment compares healthy persons versus multiple myeloma patients, learning a classifier (and BNs) for the breast cancer set looks more challenging. Before we can use the breast cancer dataset however, we must discretize the data.

Comments on classification method The breast cancer study by van 't Veer *et al.* has been widely discussed in literature (and media). Especially on the classification method comments have been made.

The classification accuracies obtained by van 't Veer *et al.* for the training set are too optimistic for two reasons. First, they used all 78 training samples to select the 231 genes. They included samples which where used as test set during the cross validation runs. Second, the ensemble size for 70 genes is based on cross validation performances using the whole training set, including those which are used as test set later on. These two "errors"

resulted in what we could call "information leakage" and the estimated accuracies on the training set are probably too optimistic [MWPS]. Recognizing this issue, van 't Veer *et al.* modified their procedure and the accuracy dropped from 83% to 73%[3].

It is remarkable that the classification accuracies on the unseen test data are better than the accuracies obtained on the training data. This is probably a coincidence caused by the small amount of test data and it stresses the difficulties of handling thousands of variables with only a hundred samples.

To obtain a better estimate of the true classification error, the study has been extended by van de Vijver *et al.* [VHV+02][4]. They included the 78 samples of van 't Veer *(*et al.), but added 67 patients with the same symptoms. van de Vijver *et al.* used the same prognosis profile of the 70 genes found by van 't Veer *et al.*. The classification accuracies for these 67 patients dropped to 65%. However, only 1 of the 12 patients in the important group with metastases was misclassified.

4.3 Pre-processing the data

All ingredients to explore the applicability of Bayesian networks for the analysis of microarray data have been described. But the LBN programs we presented can not be used with all the 25,000 genes. The data from the microarrays must be filtered such that the number of genes is reduced (from 25,000 to at most 200) and the continuous gene expression values are discretized.

In Sections 4.1.2 and 4.2 methods to select genes and discretize the continuous data were proposed. We will make use of these methods as well, because comparison of the results of the LBN (and BNC especially) programs against the results shown in Sections 4.1.2 and 4.2 is desirable. Next, we will discuss these gene selection and discretization algorithms in greater detail. Two other discretization methods which we shall use are presented as well.

4.3.1 Gene selection: Information gain

A lot of gene selection methods exist and quite an amount of research is involved in finding subsets of interesting genes. The correlation-based and entropy-based methods presented previously are standard methods and used in other microarray or machine learning studies. The correlation-based method was partly explained in the three-step supervised learning algorithm in Section 4.2 and will be discussed here in greater detail.

[3]The modified results are given in the supplementary information on http://www.rii.com/publications/2002/vantveer.

[4]The dataset was not available to us at time of writing.

Entropy In the entropy-based method genes are selected on the expected amount of information on the target variable provided by the gene. Entropy, as a concept in the field of information theory, can be mathematically defined as:

$$H(X) = Entropy[p(x), ..., p(x_n)] = \sum_{i=1}^{n} -p(x_i) \log_2 p(x_i)$$

The probabilities $p(x_i)$ are the probabilities of, for example, a gene X taking on the value x_i. The formulas calculate the average information content (bits) of the various events ($-log_2 p$) weighted by the probabilities of the events [RN95].

Example 1 Suppose we have a gene A taking on values h ($high$ expression) with probability $p(h) = 0.5$ and l (low expression) with $p(l) = 0.5$. We can use the formula to calculate the entropy of gene A and get:

$$Entropy(.5, .5) = -.5 \log_2 .5 - .5 \log_2 .5 = 1$$

When we have a dataset where gene A has value h in 99 out of 100 cases, and thus $p(h) = 99/100$, we get

$$Entropy(99/100, 1/100) = (99/100) \log_2 (99/100) - (1/100) \log_2 (1/100) = .08$$

The entropy of the gene decreases when the probabilities are not equally distributed and the outcome of a certain value is more probable (and thus less informative).

Information gain To find genes that provide information on the target variable C we move on to the principle of information gain. Information gain for gene A that takes on v values is calculated as follows:

$$IGain(C; A) = H(C) - H(C|A)$$

if C takes on w values we have:

$$H(C|A) = \sum_{i=1}^{v} p(A = a_i) \sum_{j=1}^{w} Entropy(p(C = c_j | A = a_i))$$

If C can take on two values, positive or negative, we can rewrite the equation:

$$IGain(C; A) =$$
$$Entropy(\tfrac{pos}{pos+neg}, \tfrac{neg}{pos+neg}) - \sum_{i=1}^{v} \tfrac{pos_i+neg_i}{pos+neg} Entropy(\tfrac{pos_i}{pos_i+neg_i}, \tfrac{neg_i}{pos_i+neg_i})$$

where *pos* and *neg* are respectively the number of positive (healthy for example) and negative (multiple myeloma) samples. The value of pos_i is equal to the number of positive samples in the subset of the data where gene A has taken on value i. Calculating neg_i is done in the same way, only now the number of negative samples in the subset is counted. Information gain subtracts the weighted entropy of the subdivided datasets from the entropy of the target variable in the entire dataset. Mutual information, described in Section 3.3, actually measures the same as information gain[5] and is symmetric.

Example 2 Let us recall Example 1 where gene A could be h or l, and $p(h) = 50/100$ and $p(l)$ is also $50/100$. Suppose the number of patients having multiple myeloma (neg) is 70 and the number of healthy samples (pos) is 30. We can divide the data in two groups, one group G_h containing all samples where gene A has a high expression and a group G_l where gene A has a low (l) expression. Suppose the number of positive samples (pos_h) in G_h is 10 (so neg_h is 40). It follows that in G_l we have $pos_l = 20$ and $neg_l = 30$. Now we can calculate information gain which will be equal to: $Entropy(\frac{30}{100}, \frac{70}{100}) - \frac{50}{100}Entropy(\frac{20}{50}, \frac{30}{50}) - \frac{50}{100}Entropy(\frac{10}{50}, \frac{40}{50}) = 0.0349$. But when we have $neg_h = 50$ the information gain increases to 0.396 (we assume that $0/50 \log_2 0/50 = 0$). When $neg_h = 50$ gene A can provide more information on the class value when it has a high expression.

The gene selection method selects a set of genes with highest information gain (where either a threshold is needed or a maximum number of genes). In contrast to correlation, information gain based gene selection can be seen as a consistency based gene selection method; how the differences between the two classes are distributed does not matter as long as the two classes are consistently separated by the expression value of the genes. This means that minimal differences within the gene expression of a gene between two groups can result in high information gain scores.

4.3.2 Gene selection: Correlation

We will shortly present the correlation based gene-selection method as it was used by van 't Veer *et al.*. In general correlation values determines the extent to which values of two variables are "proportional" to each other. Several methods to estimate the correlation between to variables exist, but the most widely-used Pearson correlation is also used by van 't Veer *et al.* to select 231 genes.

The Pearson correlation is formulated as:

[5] The information gain formula can be rewritten into the mutual information formula, but that goes beyond the scope of this thesis. Interested readers can view this on http://cgm.cs.mcgill.ca/~soss/cs644/projects/simon/Entropy.html.

$$C(X,Y) = \frac{\sum_{i=1}^{n}(X_i - \overline{X})(Y_i - \overline{Y})}{(\sum_{i=1}^{n}(X_i - \overline{X})^2 \sum_{i=1}^{n}(Y_i - \overline{Y})^2)^{1/2}}$$

where $\overline{X}$ and $\overline{Y}$ are the averages for X and Y respectively. We can re-write the function with the standard deviations for X and Y:

$$C(X,Y) = \frac{\sum_{i=1}^{n}(X_i - \overline{X})(Y_i - \overline{Y})}{(n-1)S_x S_Y}$$

The Pearson correlation calculates the distances between all data points and a linear regression line. The linear regression line is estimated such that the sum squared distances between all data points and the linear line are minimal. Scale does not matter. The correlation (linear relationship) between two variables is strong when the Pearson correlation is close to 1 or -1. If the Pearson correlation is close to zero the linear relationship is weak.

In our case we calculate the correlation between a gene X and the corresponding class values. The class values are either 0 or 1. The correlation is close to 1 or -1 if the distance between the means for the two classes is very large and/or the standard deviation within the classes is very small. In Figure 4.1 two examples are given.

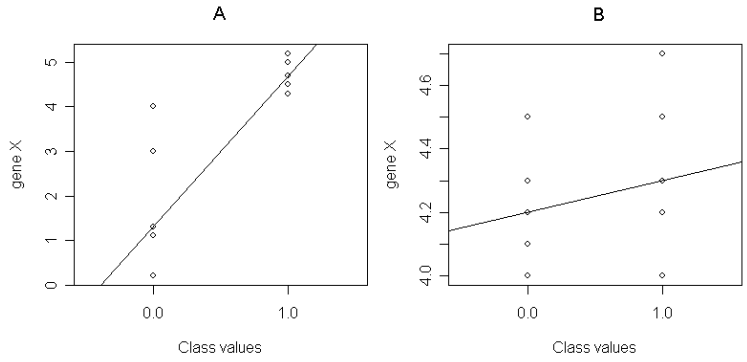

Figure 4.1: In plot A the distribution for the 2 classes are quite different. The corresponding Pearson correlation between gene X and the class is 0.82. The Pearson correlation for B is 0.27. The regression lines are also plotted.

It is important to note that this gene selection method, and the information gain based method, are actually already a part of the learning process. The fact that the selection methods already use the target class value is sometimes forgotten by the analysts, as we already mentioned in Section 4.2.2.

4.3.3 Discretization

Three methods to discretize continuous data are available in a program that comes with *PowerPredictor* and *PowerConstructor*: a frequency-based method, a range-based method, and a method based on information gain will be discussed.

The frequency-based methods defines split values in such a way that each interval contains an equal amount of examples. In the range-based method, the intervals have an equal distance in the continuous space.

In the third discretization method, information gain is used to find the optimal split values . The optimal split value is the value that gives the highest information gain, given a set of continuous values and their corresponding class labels. The set of continuous data, and their corresponding class labels, are ranked by their magnitude. For each neighboring pair of values the information gain is calculated. The average of these two values is used as the split level, that divides the data into two groups. All continuous values greater than the split level become equal to one, the other values become zero.

The frequency-based and range-based method are unsupervised discretization methods but the entropy-based method is a supervised discretization method.

Chapter 5

Results and discussion

In the previous chapters we presented the Bayesian network learning methods and the gene expression datasets we want to use to learn Bayesian networks. The results are shown in this chapter.

The first section of this chapter shows the results obtained by applying the LBN algorithms to the multiple myeloma dataset as described in Section 4.1. Because we want to distinguish the learning algorithms for general networks and classification networks we divided the presentation of the results for the myeloma dataset accordingly. Section 5.2 will be subdivided similarly and presents the results obtained for the breast cancer dataset. A short summary of the results to further analyze and compare the results is given in Section 5.3.

5.1 Results on the multiple myeloma dataset

Details of the multiple myeloma dataset were discussed in Section 4.1, but will be repeated here briefly. The multiple myeloma experiment was carried out to compare the gene-expression of healthy human beings against multiple myeloma patients. In total 105 bone marrow samples were measured containing 34 healthy samples and 71 multiple myeloma samples. Affymetrix oligonucleotide arrays were used and resulted in a set of absolute values (Absolute Call, AC) and continuous values (Average Difference, AD) both representing gene expression values.

One of the goals of the experiment was to check how well classification techniques could distinguish between the gene expression values of a multiple myeloma patient and a normal healthy person. Section (5.1.1) will show to what extent our Bayesian network classifiers (BNC) learned on the multiple myeloma dataset are capable of doing this. We will compare the results to those obtained by Page *et al.* [PCW+02].

The multiple myeloma set was also used to construct general Bayesian networks. We used two general Bayesian network learning algorithms (K2,

	CTan Classifier	Naive Network	Markov Blanket
AC Only	100 ± 0.00	99.05 ± 0.94	99.05
AC+AD	100 ± 0.00	100 ± 0.00	100

Table 5.1: Classification accuracies of BNCs learned on gene expression values of 30 genes from the multiple myeloma dataset. The entropy-based gene selection method is used to select 30 genes. Discretization of the AD values is also based on the entropy. Default settings of the algorithms were used.

CBL) as described in Section 3.3 to see if they can extract interesting networks from the data. The resulting networks can be found in Section 5.1.2.

5.1.1 Classification networks

To compare our classification accuracies against the accuracies obtained by Page *et al.* [PCW+02], we used exactly the same 30 genes they selected by using the top information gain genes for both AC (discrete values) and AC+AD (discrete and continuous values). The set of trivial genes was removed from the dataset. The resulting classification accuracies, using default settings of the algorithms[1], are shown in Table 5.1.

We used leave-one-out cross validation to estimate the classification errors shown in Table 5.1. The classification accuracies are the percentages of correctly classified samples. In the implementation of the Markov blanket (MB) algorithm, leave-one-out cross validation was not possible. Instead, the Markov blanket was trained and tested on all the 105 samples, which is a too optimistic way to estimate the true classification error.

Another remark is about the selection of genes. The genes were selected using all the 105 samples, including the leave-one-out samples. Remember that entropy-based gene selection was used and that this is already part of the learning process. The leave-one-out sample should be excluded from the total learning procedure. The "true" classification error is expected to be somewhat higher. We'll return to this issue later on in this section.

In [PCW+02] other machine learning techniques were used to learn to classify new bone marrow samples, of which the results are shown in Table 5.2 (see previous chapter for more information).

Almost all classifiers (both our BNCs and the classifiers in Table 5.2) achieve high classification accuracies. For the AD+AC values, accuracies are for all BNCs 100%. Classifying a bone marrow sample as healthy or multiple myeloma, almost seems as a trivial task, even after removing the

[1] In the default settings the prior precision is equal to one for the CTan and Naive BNCs, and the confidence level threshold for the Markov blanket is automatically detected.

Method	AC Only	AC+AD
Trees	90.5	98.1
Boosted Trees	96.2	99.0
SVMs	95.2	93.3
Vote	94.0	100.0
Bayes Nets	95.2	100.0

Table 5.2: Leave-one-out cross-validation results for multiple myeloma patients vs. normal persons as they were obtained by Page *et al.* [PCW+02]

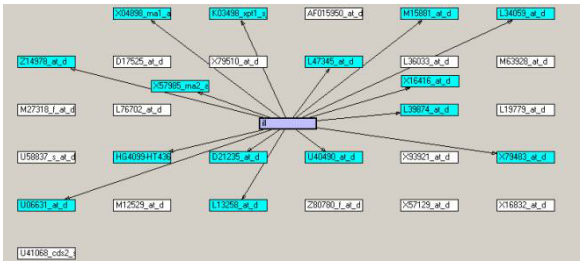

Figure 5.1: BNC learned with PowerConstructors' default Markov blanket algorithm. The AC+AD gene expression values of the top 30 entropy genes of the multiple myeloma experiment were used. The middle node represents the multiple myeloma variable.

known trivial genes. The BNC learning algorithms in general perform well in comparison to the other techniques.

In Figures 5.1 and 5.2 two of the six generated networks are shown. The networks were all learned on the total set of samples. Comparing all the generated networks, it seemed that the naive Bayesian network was the most common among the networks based on AC+AD gene expression values, see for example Figure 5.1. Using only AC (discrete) values to learn classification networks, the resulting networks became less naive, see Figure 5.2. The Markov blanket shows that only half of the 30 genes are needed to obtain high classification accuracies, see Figure 5.1.

Using an independent test set We already noted before that the estimation of the classification accuracies of the learned networks was not ideal in our leave-one-out strategy. We selected one subset of 30 genes based on all the data, including the leave-one-out samples. Selecting genes based on only the training set—in each cross-validation run— should make the leave-one-out strategy a better estimator of the true classification error. But because gene selection is not automatically possible in the programs we use, the genes should be selected manually in each cross validation run. So we

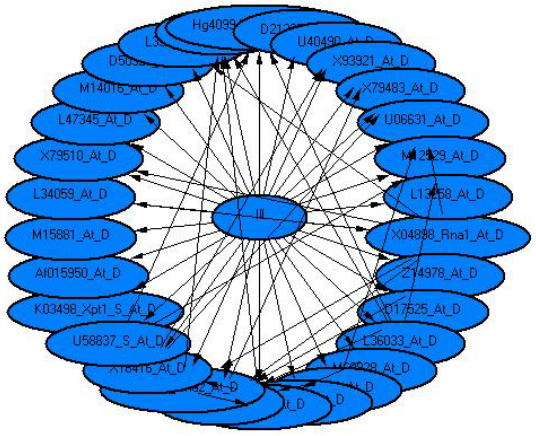

Figure 5.2: Bayesian classification network learned with CTan algorithm. The prior precision threshold was set to 1 (default value). The AC gene expression values of the top 30 entropy genes of the multiple myeloma experiment were used. The middle node represents the multiple myeloma variable.

would need to select 30 genes based on our own algorithms, learn a BNC, and classify the left-out sample 105 times in a row.

Estimation of the classification error based on one large independent test set, with gene selection based on the training set only, can give a less biased estimate of the true classification error as compared to the previous method. Therefore we carried out another classification error estimation procedure, more similar to van 't Veer *et al.* [VDH$^+$02]. We randomly selected 20 samples from the 105 samples as an independent test set. A set of 30 genes was selected, based on the information gain of individual genes within the training set (the remaining 85 samples). Networks were learned with the gene expression values of these 30 genes in the training set. The networks were tested on the independent test set, see Table 5.3.

Accuracies are fairly high although they decreased a little as compared to the accuracies in Table 5.1. Notice that the classification accuracy for the test set is 100% (all test samples were correctly classified) and for the training set only 94.9%, in case of AC Only and the CTan BNC.

5.1.2 General networks

To test the possibilities of the LBN programs used in our approach it is also important to use general Bayesian networks to analyze the gene expression data of the multiple myeloma experiment. The GBN algorithms we used are K2 and CBL as described in Chapter 3 and are especially suited to learn structures. The algorithms construct a network that best fits and

	CTan Classifier		Naive Network		Markov Blanket	
	training	test	training	test	training	test
AC Only	94.9 ± 2.4	100	94.1 ± 2.5	95.0	97.7	90
AC+AD	97.7 ± 2	95	96.2 ± 2	100	97.7	95

Table 5.3: Classification accuracies for multiple myeloma patients vs. normal persons. The test set contained 20 samples, the training set contained 85 samples. Entropy-based gene selection was used, based on the training set. The continuous AD values were discretized using bins based on information gain. Default settings of the algorithms were used.

represents the values of the given variables (genes and disease state). The K2 algorithm learns a directed acyclic graph (DAG) and the CBL algorithm leaves connections undirected when not enough evidence is given to choose one of the directions. Therefore CBL learns a partially directed acyclic graph (PDAG).

Tuning the parameters of the algorithms, such as the confidence level threshold of a dependency in the CBL, and the prior precision parameter in K2, resulted in different networks. The GBNs learned by K2 and CBL1 depend on the order in which the genes appear in the database, see Chapter 3.

Figures 5.3, 5.4 and 5.5 depict some of the resulting networks. We used the default parameters of the algorithms (see Figures 5.4 and 5.3) but also modified the parameters (see Figure 5.5 for example).

The disease-state variable was also included but because the network is not especially constructed for the prediction of this variable, we can view the GBN learning strategy more as a kind of unsupervised learning method[2].

Most of the networks (not all shown) seemed to look similar to a naive Bayesian network with the multiple myeloma state variable being the parent of the gene nodes. Setting the thresholds to a value such that more confidence is needed, the networks changed in structure; see Figure 5.4 and Figure 5.5. Other settings could also be changed (node ordering, discretization). We have constructed many networks by changing these settings For practical reasons not all these networks can be shown here. On our website other networks can be viewed, see http://www.science.uva.nl/~vthemaat.

Changing the settings of the algorithms always resulted in GBNs with different structures. A comparison of the structure of two, or more, networks is hard. Some basic methods exist to compare, or combine, networks (such as common link, complexity measures and methods mentioned in 2.3) but these are not implemented in the software we used, so the networks can only

[2]We note that the genes were selected by a supervised method, therefore the complete GBN learning procedure is supervised.

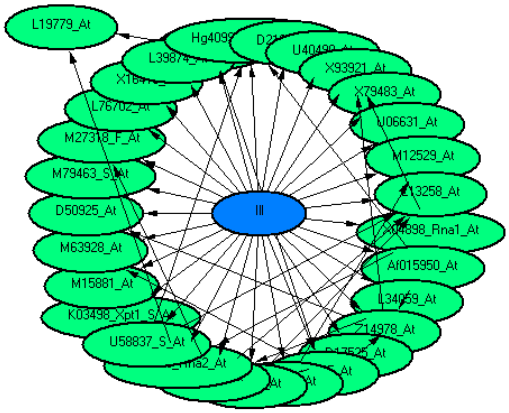

Figure 5.3: General Bayesian network learned with K2 Bayesware default settings. The AC+AD gene expression values of the top 30 entropy genes of the multiple myeloma experiment were used. The middle node represents the multiple myeloma variable.

be compared manually.

We also learned networks (not shown here) with a random set of genes which did not have to be disease-related. In this case, the resulting networks sometimes contained almost no dependencies between genes and sometimes a lot of dependencies between genes. The networks looked a lot different as compared to the naive networks learned with the information gain based gene selection. The disease state variable was never a dominating node in the networks with randomly selected genes (which seems plausible because the random genes are not specifically related to the disease).

Quantitative analysis of the resulting networks is not very easy. The classification accuracy might give an indication how reasonable the network fits the data. With the networks learned by K2 we can classify samples. Because CBL2 results in partially directed networks, classification is not possible. The classification accuracies are very high; 99.1% using only AC values and 100% using AC+AD values. These accuracies are almost the same as the BNCs (using only AC values, the performance is again slightly worse). Table 5.4 also shows these results.

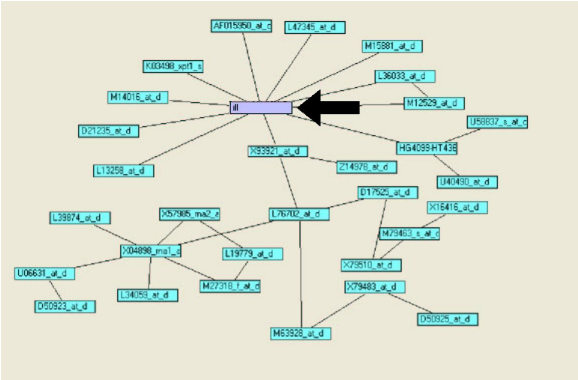

Figure 5.4: General Bayesian network learned with Powerconstructors' CBL2 algorithm. The confidence level threshold was set to the default value. The AC+AD gene expression values of the top 30 entropy genes of the multiple myeloma experiment were used. The node indicated by the thick arrow represents the multiple myeloma variable.

	K2
AC Only	99.1
AC+AD	100

Table 5.4: Classification results of GBNs learned by K2 on the multiple myeloma gene expression data.

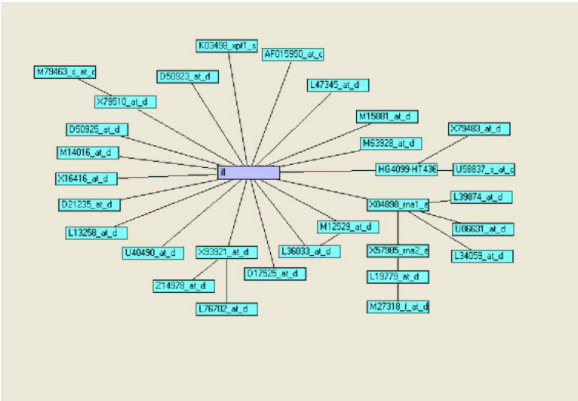

Figure 5.5: General Bayesian network learned with Powerconstructors' CBL2 algorithm. The confidence level threshold was set to 8. The AC+AD gene expression values of the top 30 entropy genes of the multiple myeloma experiment were used. The middle node represents the multiple myeloma variable.

5.2 Results on the breast cancer dataset

The breast cancer experiment is carried out with a slightly different purpose as compared to the multiple myeloma experiment. Predicting the future development of breast cancer and finding developmental marker genes are the main purposes in the breast cancer experiment. The spotted array technique was used to spot 25,000 genes of 51 metastases breast cancer tissues and 46 non-metastases samples against a reference pool.

In order to check the usability of the LBN programs on the breast cancer set we learned classification and general networks. The results will be shown below.

5.2.1 Classification networks

The classification networks were learned under several conditions we mentioned before. We used different types of discretization methods, different types of gene selection methods and a variety of LBN methods. We start with showing the classification accuracies obtained by using the same gene selection methods as used in the breast cancer article [VDH+02]. The 70 genes they found can be extracted from their publication in *Nature*[3]. The genes were selected on the basis of correlation and predictive powers on a training set (78 samples), see Section 4.2 for more details. Learning our classification networks on 78 samples (34 metastases and 44 non-metastases) and testing the networks on 19 independent samples (12 non-metastases en 7 metastases), resulted in classification accuracies as shown in Table 5.5.

	CTan Classifier		Naive Network		Markov Blanket	
Discretization	training	test	training	test	training	test
Entropy	99	84	87	94	94	68
Equal width	88	84	88	84	52	57
Frequency	90	78	78	89	85	57

Table 5.5: Breast cancer classification accuracies in percentages using the genes selected by van 't Veer *et al.* in [VDH+02]. Several discretization methods and BNC learners were used.

The training set was slightly modified for the Markov blanket algorithm. Because one of the samples contained a lot of missing values and the Markov blanket algorithm cannot cope with that, this sample was not included. One might argue this sample should be included as misclassified, but we leave that up to the reader. In case one wants to include the sample as misclassified, the accuracies should be adjusted with -1.3%.

[3] See http://www.nature.com.

In Table 5.6 the results obtained by van 't Veer *et al.* are presented. Our classification accuracies can compete with their results.

Results by van 't Veer *et al.*		Adjusted Markov Blanket	
training	test	training	test
83	88	94	88

Table 5.6: Breast cancer classification accuracies in percentages using the genes selected by van 't Veer *et al.* in [VDH+02]. The adjusted Markov blanket is learned with entropy discretization but now the confidence level is set to 8.

A second remark on the Markov blanket is about the confidence level threshold. Cheng [CG98] recommends to manually increase the threshold if the number of samples is much lower than the number of variables. If this value is increased, the accuracies on the test set seem to increase as well. The result when the threshold was set to 8 times the default value (we just manually picked a number as suggested by Cheng) is represented by the "Adjusted Markov Blanket" column, see Table 5.6.

It looks as if the BNCs are overfitted to the training data if the threshold is set to default. The problem however, is that a good learning strategy does not allow us to adjust the threshold based on the classification accuracies on the independent test set, which makes it hard to find the good threshold when automatic threshold selection fails to do so.

Just like van 't Veer *et al.*, we selected the genes on the basis of the complete training set. We also learned the BNCs on the complete training set. Because of these facts the resulting accuracies on the training set may be too optimistic. The accuracies on the independent test set give a better indication of the true error of the learned BNCs. The problem with the test set accuracies is the small amount of samples. This might explain why sometimes the accuracy is higher on the test set than on the training set.

Figures 5.6 and 5.7 show a Markov Blanket and CTan network respectively. Most networks contained more dependencies between genes (see Figure 5.6 for example), and looked more interesting, as compared to the BNCs learned with the multiple myeloma dataset. But this is not always the case (see Figure 5.7).

Information gain based gene selection In addition to the results obtained using the correlation based gene selection method (based on correlation) used by van 't Veer *et al.* we also used the information gain based gene selection used in the multiple myeloma experiment. We selected 70 genes from 25,000 genes based on their individual predictive power represented by the information gain value. We did not select a larger set of genes because

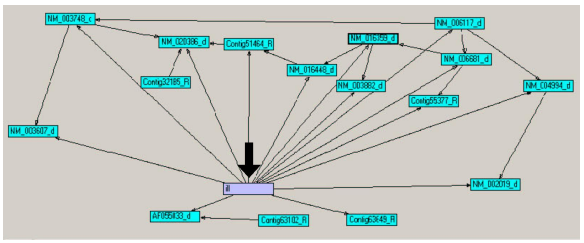

Figure 5.6: Bayesian network classifier learned with the breast cancer gene expression values. Selection of the genes was based on correlation with the breast cancer state. The algorithm to learn a Markov blanket, as implemented in Powerpredictor, was used. Discretization of the gene expression values was based on equal widths. The node indicated by the thick arrow represents the metastases variable.

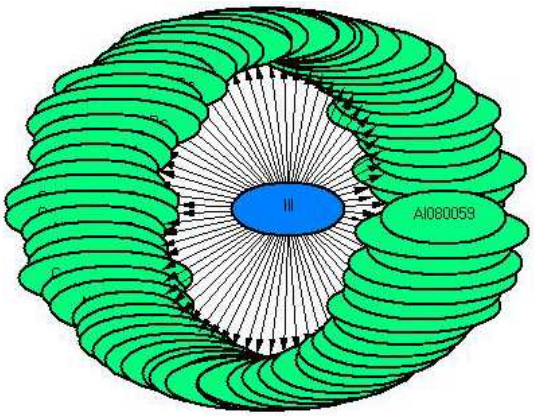

Figure 5.7: Bayesian network classifier learned with the breast cancer gene expression values. Selection of the genes was based on correlation with the breast cancer state. CTan algorithm was used with default settings. Discretization of the gene expression values was based on equal widths. The middle node represents the metastases variable.

we wanted to use the same amount of genes as van 't Veer [VDH+02]. The learning strategy is also similar. The classification accuracies we obtained are given in Table 5.7. The classification accuracies on the test set in our entropy based selection method are overall lower in comparison with the scores of the BNCs learned on the genes selected by van 't Veer *et al.*. On the other hand, the results on the training set seemed to improve a little. These results indicate that the BNCs are overfitted to the training data, so we conclude that, in this case, entropy based gene selection performs worse compared to correlation based selection.

	CTan Classifier		Naive Network		Markov Blanket	
Discretization	training	test	training	test	training	test
Entropy	*Failed*	*Failed*	96.1	73	100	68
Equal width	98	68	97	73	80	47
Frequency	96	78	98	68	97	42

Table 5.7: Breast cancer classification accuracies using the genes selected by information gain. Several discretization methods and BNC learners were used. Default settings of the thresholds were used. *Failed* means that the program crashed.

5.2.2 General networks

We have learned general Bayesian networks (GBN) under the same conditions as we used above. Only 78 samples were used. The resulting networks looked interesting and showed a lot of gene-gene dependencies. But again, under different circumstances and by tuning parameters different networks were learned by the algorithms. Showing all these networks is practically impossible and of no real use. In Figures 5.8, 5.9, 5.10 and 5.11 a small selection of networks is shown.

At first sight, the networks look quite complex. In contrast to the GBNs from the multiple myeloma experiment (see Section 5.1.2), these GBNs show structures where the disease node is not the parent of all gene nodes.

By varying the threshold values of the GBN learning algorithms, the structure of the networks always change quite a lot. In Figures 5.9, 5.10 and 5.11, the prior precision parameter was set to 1,4 and 78 respectively. By increasing this threshold the evidence needed for a dependency between two genes decreases. While Figure 5.9 contains not so many dependencies, the GBN in Figure 5.11 contains a very large amount of dependency relationships (and the program even crashed before it could finish).

The GBNs learned by the K2 algorithm could be tested for their classification result on the metastases variable, the results are given in Table 5.8. As expected, classification accuracies on the test set are much lower compared to the BNCs.

Comparison of GBNs Comparing the resulting networks cannot be done by the programs. Because we wanted to know to what extent the resulting GBNs resembled each other we compared some of them by hand. This resulted in a list of genes and the frequency of their direct or indirect dependency relationship with the metastases variable[4]. Although the networks

[4]We compared five networks using correlation-based gene selection: CBL2 range discretized, K2 entropy, range and frequency discretized, and K2 range with other threshold settings.

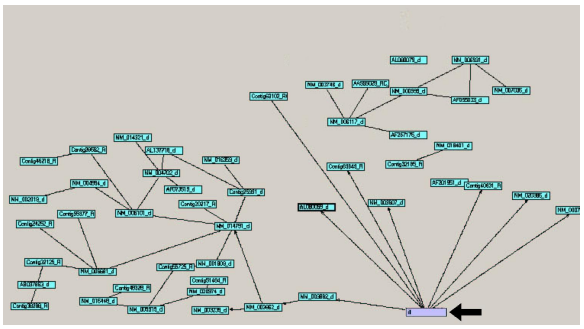

Figure 5.8: General Bayesian network learned with the breast cancer gene expression values. Selection of the genes was based on correlation with the breast cancer state. Powerconstructor (default settings) was used to learn the network. The node indicated by the thich arrow represents the metastases variable.

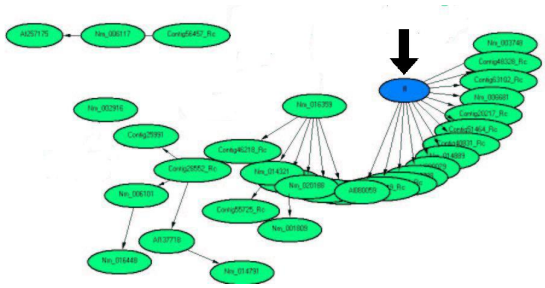

Figure 5.9: General Bayesian network learned with the breast cancer gene expression values. Selection of the genes was based on correlation with the breast cancer state. The K2 algorithm was used with default settings (prior equal to one). The node indicated by the thick arrow represents the metastases variable.

	Correlation based selection		Entropy based selection	
Discretization	training	test	training	test
Entropy	*Failed*	*Failed*	100	36
Equal width	90	68	*Failed*	*Failed*
Frequency	87	73	97.4	47

Table 5.8: Breast cancer state prediction results of the GBNs created by the K2 algorithm using different gene selection and discretization methods.

were quite different from each other, four genes were consistently directly related to the metastases variable throughout all the networks we checked.

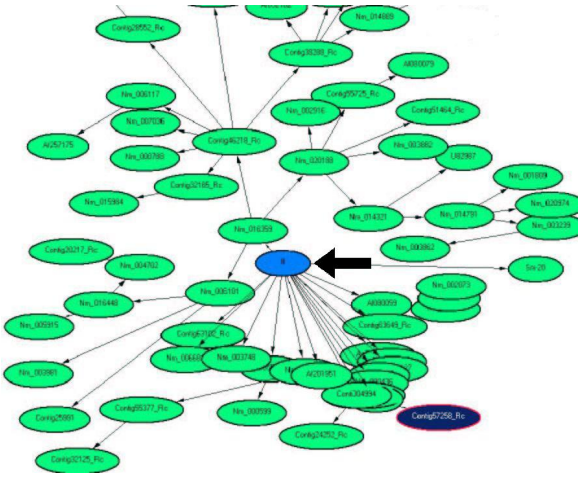

Figure 5.10: General Bayesian network learned with the breast cancer gene expression values. Selection of the genes was based on correlation with the breast cancer state. The K2 algorithm was used with prior set to 4. The node indicated by the thick arrow represents the metastases variable.

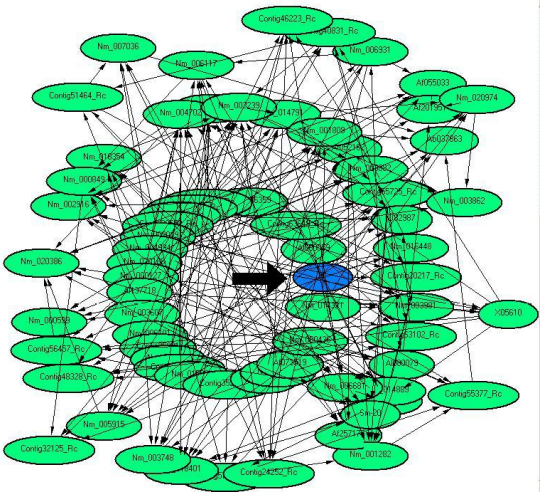

Figure 5.11: General Bayesian network learned with the breast cancer gene expression values. Selection of the genes was based on correlation with the breast cancer state. The K2 algorithm was used with prior set to 78. Learning the network crashed after several minutes. The node indicated by the thick arrow represents the metastases variable.

A CTan BNC was learned with the expression values of these four genes in the 78 training samples and this resulted in classification accuracies of 80% on the training set and 84% on the test set. These accuracies are quite

4 genes selection	
training	test
80	84

Table 5.9: Breast cancer development prediction based on gene expression values of 4 genes in 78 samples (training set). The test set contained 17 samples. The CTan Bayesian network classifier was used.

good compared to the results when 70 genes are used.

The four genes are: NM_003748, NM_006681, AL080059 and contig63102_RC. Contig63102_RC is not mentioned in other literature, but AL080059, NM_003748 and NM_006681 have been linked to cancer. Kulaeva *et al.* [KDL+03] report that AL080059 is involved in cellular immortality, which could be an important process in early development of cancer. The gene-database provided by NCBI[5] links AL080059 to a brain protein (protein KIAA1750). Gene NM_003748 represents ALDH4 protein and is involved in several processes ranging from cancer to cellular stress. Alevizos *et al.* suggest in [AMZ+01] "the need for further study of the role of NMU in carcinogenesis". NMU is represented by the gene NM_006681. Alevizos *et al.* report that the poorly understood NMU seems to be strongly related to oral cancer and refer to other studies on the role of NMU in cancer.

So, even while it is hard to compare the networks, interesting information can be found.

Exploring relations between genes Finally we like to remark the possibility of exploring the properties of dependencies between genes. In the K2 version implemented in Bayesware this is easily possible because Bayesware shows the conditional probability table (CPT), as described in Chapter 3, in a 3-dimensional output. Of course, it is standard to be able to visusalize the CPTs because these are part of the Bayesian model, but nevertheless it can be useful to researchers.

Figure 5.12 shows such an example. In this figure, contig57258_Rc is dependent on X05610. The gene values were discretized into 4 bins. We can see, for example, that if X05610 has a continuous value of 0.2 (which represents a relative high expression), contig57258_Rc is most likely to have a relatively high expression (between 0.09 and 0.43) as well. For analyzing certain gene expression values and relationships, such a graphical display can be very helpful.

[5]For more information look at: HTTP://www.ncbi/nlm/nih/gov/entrez.

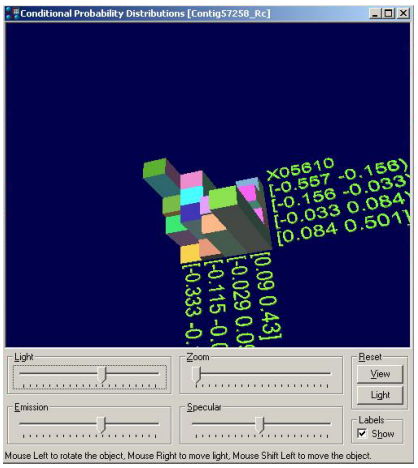

Figure 5.12: A conditional probability table as showed in Bayesware.

5.3 Summary and comparison of the results

In the previous sections the results of different experiments were presented. In this section a summary of the results is given, enabling us to compare the Bayesian networks as learned under different conditions.

The classification results of the BNCs and GBNs are graphically displayed in Figure 5.13.

Note the difference between the classification accuracies of the multiple myeloma and the breast cancer BNCs. This is in accordance with the classification results shown in [VDH+02] and [PCW+02].

We can also notice the low accuracies obtained by the general networks and the Markov blankets on the test sets in the breast cancer sample. Apparently these methods are less capable of avoiding overfitting and generalizing the training data. For the case of the general networks, it is easy to understand, because they are not especially learned to predict the state of one variable. The Markov blanket overfitting the training data is a little more surprising. The fact that the threshold might not be set optimally (see Table 5.5, here the result of a MB with a different threshold is shown) could be an explanation for this result.

Another interesting result is the ability of the BNCs to compete, in classification accuracies, with the methods used in [VDH+02] and [PCW+02].

Using information gain to select genes from the breast cancer gene expression database also gives an interesting result. This selection causes the Bayesian networks to classify the training data very well while the classification accuracies on the test patients are relatively low. When we compare these accuracies against the correlation based gene selection results it seems

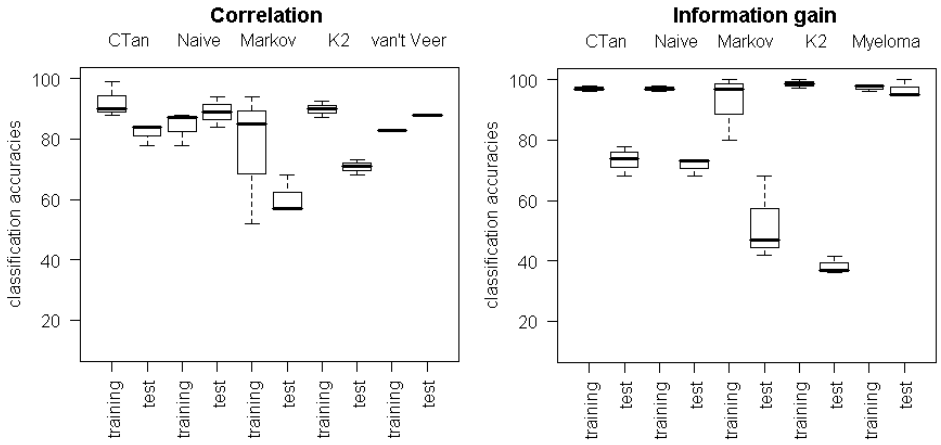

Figure 5.13: Boxplots of classification accuracies obtained by Bayesian networks under different circumstances for correlation (left) and information gain (right) based gene selection. All plots represent accuracies on the breast cancer dataset, except for the most right boxplot, which represents the accuracies for the multiple myeloma set.

that information gain causes overfitting.

The high accuracies on the training data throughout these experiments could be a too optimistic because the selection of genes was based on the whole training set.

A last remark on the results shown in Figure 5.13 is about Occam's razor stating that the simplest solution is probably the best. The naive Bayesian network, which has a very simple structure, obtains high classification accuracies on both training and test sets. The naive BNC is also very fast in learning and probably favorable for classification purposes. A disadvantage is that the naive BNC can not learn interesting networks or find marker genes.

A comparison between the structures of the Bayesian networks learned under the different circumstances is much more difficult. As we already noted earlier, there was a difference between the networks learned with the multiple myeloma set and the breast cancer set. The Bayesian networks based on the multiple myeloma set resulted in networks where the disease state variable was the parent of all gene nodes. In the breast cancer experiment more complex structures were created.

A reason for this difference in network structure, and for the differences in classification accuracies, between the multiple myeloma and breast cancer experiment, could be explained by the information gain values of the individual genes used in the networks. Figure 5.14 shows that the information gain values for the breast cancer genes are much lower than the information

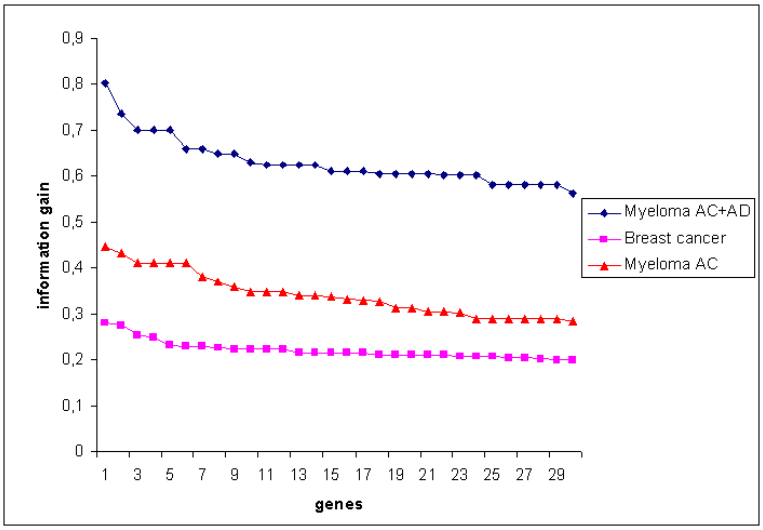

Figure 5.14: Top 30 information gain scores of genes from different gene expression sets

gain scores for the multiple myeloma genes.

Figures 5.9, 5.10 and 5.11 show that the structures of the constructed networks depend on the values of the thresholds used by the variables. A slight change of these thresholds can cause great differences in network structure. Because it is not possible to automatically compare the structures of the networks it is very hard to find the most representative network for a given dataset.

Chapter 6

Conclusions and future work

In this thesis we considered the analysis of microarray data with Bayesian networks. We described how the Bayesian techniques can be applied to analyze gene expression data. Our approach was based on the demands posed by the domain of microarray analysis. The central question stated in Chapter 1 was whether easy accessible and understandable learning Bayesian network (LBN) methods can be used for the analysis of microarray data. The results we obtained lead to conclusions on the possibilities of using LBN programs for *classification* purposes and on the difficulties of using *general* Bayesian network learners for gene expression analysis.

In Section 6.1 we present our general conclusions as well as our conclusions for classification and general Bayesian network learners specifically. We conclude in Section 6.2 with ideas for future work and improvements to the used programs.

6.1 Conclusions

In the experimental setup of this thesis we looked at LBN algorithms able to learn classification networks or general networks. To show the possibilities of these algorithms we applied them under different conditions, including:

- Two programs with different implementations of general and classification network learners
- Two different microarray experiments
- Different gene selection methods
- Different discretization methods

Our results show that the LBN methods we used can be very useful to answer relevant biological questions and to extract biologically interesting information. We have seen that for the purpose of diagnosis and prognosis

the Bayesian network classifier (BNC) learners achieve high classification accuracies and can compete with other classification methods. The LBN programs can also provide insights into gene relationships (by means of the graphical output and possibilities of relationship analysis). With the help of both general and classification LBN methods we found small sets of marker genes, ranging from 4 to 15 genes, which may be of interest for molecular biology.

However, the LBN algorithms we used did not always give good or easily interpretable results. The BNC learners poorly classified the test sets under some conditions and robust estimation of classification accuracies is difficult with our high dimensional microarray data. The networks learned with GBN learners varied a lot under slightly different settings (changing parameter values), which make the resulting networks not very robust and hard to interpretate. In the next parts we will discuss the conclusions for the classification and general networks separately.

6.1.1 Classification networks

By comparing the results from the two different experiments we see that the differences in the resulting network structures and prediction accuracies are significant. When we are comparing a multiple myeloma patient versus a healthy person, the classification task seems almost trivial with accuracies of 100%. In most of the resulting classification networks the multiple myeloma state variable is the parent of all other gene variables. When comparing two developmental *subtypes* of breast cancer the classification accuracies are far less accurate in comparison to the multiple myeloma case and the networks become less simple. The experimental design is of importance for the expected accuracies and networks.

The information gain scores of individual genes seem to give an indication for the classification accuracies and the resulting networks. The information gain scores were high for the genes in the multiple myeloma set but relatively low for the genes in the breast cancer set.

For the BNC learners the most important issues concerning the classification of high dimensional gene expression patterns are: overfitting to training data (see Section 2.1.1) and the estimation of classification error.

Overfitting The results we obtained show that simple Bayesian classification techniques, such as the naive and CTan Bayesian network classifiers, can generalize over the training data. We can see this in the high classification accuracies for the training data as well as for the independent test set. Classification networks which learn more complex networks, such as Powerpredictor's Markov blanket BNC, have more difficulties to generalize over the training data and perform less on the test data (especially when the classification task is less trivial as in the breast cancer experiment).

The gene selection methods used to select a subset of genes (about 1% of the total set) affects the ability of the BNC learners to generalize over the training data. The classification accuracies for the test set of the breast cancer dataset was lower with entropy based gene selection than with correlation based selection. If the entropy based gene selection is used, the chance of overfitting seems to increase.

By tuning the parameters of the algorithms the classification accuracies on the training set can be optimized. By increasing the confidence level of the Markov blanket BNC, the network can become less sensitive to overfitting and perform better on the test set. However, this kind of manually adjusting the parameters should only be tested within the training set. If the test set is used to optimize parameters the test is not independent anymore and we would insert a bias which can make the results too optimistic. A procedure to select a level of confidence is to increase the level of confidence until the accuracy on the training set drops below a certain level. We must note that, in most cases, the default settings of the algorithms can be used to obtain high accuracies.

Estimation of error The limited availability of the number of samples and the large amount of variables are a serious problem for the estimation of the classification error. Since we want to use a test set to estimate the classification accuracies, only a part of the data can be used as training data. Our estimations, based on an independent test set, sometimes resulted in higher classification accuracies for the test set than for the training set, which is remarkable. Leave-N-out cross-validation can't be easily used in the programs we tested when supervised gene selection methods are used, but is almost necessary to obtain a robust estimation of the classification accuracies in these small datasets. It should be noted that the sets we used may be relatively large in microarray terms, but are in fact quite small in machine learning terms.

Leave-N-out cross-validation can't be easily used because supervised gene selection methods are not integrated into the programs. When using leave-N-out cross-validation for error estimation, a supervised gene selection must be made for each training set at each cross-validation run separately. As this is not automatically possible, we are forced to select genes with our own gene selection methods at each cross-validation run, and we must manually insert the expression values of these selected genes into our BNC programs. This is practically impossible when we use leave-one-out cross validation with our datasets; we should manually learn 100 classifiers if the dataset contains 100 samples.

Learned structures The structure of the resulting networks learned by the BNC algorithms can be interesting for two reasons. We have stated

in Chapter 2 that the Markov blanket of a target variable might represent a collection of genes possibly involved in the same biological process. The Markov blanket of the metastases variable learned by Powerpredictor for example, possesses gene-to-gene relationships which can be interesting for further investigation. The second interesting feature is the selection of only several genes for classification. This set contains the genes that must be known to predict the value of the target variable and can be viewed as "marker" genes.

Summary We conclude that the BNC learners used in this thesis are a useful tool for classification and finding marker genes. However, ways to reliably estimate the classification error and the integration of gene expression specific tools, such as gene selection methods, are missing.

6.1.2 General networks

The purpose of learning general Bayesian networks (GBN) with gene expression data is to represent gene interactions by modelling conditional independencies between genes. Clusters of co-regulated genes, dominant genes and relationships between genes and the disease state of a sample are hoped to be discovered by learning GBNs.

As can be seen in the results presented in the previous chapter, the structure of the networks with the same set of genes differ considerably with varying discretization methods, learning methods and parameter settings. It seems that for our datasets several optimal networks can be created. This leads to the hypothesis that the number of Bayesian networks that reasonably represent the data is quite large (other studies report the same, see Friedman *et al.* [FLNP00] for example). The problem of having so many reasonable networks cannot easily be tackled. Automated ways to compare and combine a set of reasonable networks, as well as automated threshold optimizers are not available within the GBN programs.

The GBNs seem to get less naive and more interesting when the experiment at hand is less trivial. We saw differences in structure between the networks learned with the multiple myeloma dataset and the networks learned with the breast cancer dataset. The networks learned with the breast cancer dataset looked more complex and interesting. An explanation for these results could be given by the information gain and correlation scores of the genes that we selected In the multiple myeloma set the genes are far more correlated with disease state. These strong correlations with the disease state make all the genes almost conditionally independent of each other if the disease state is known, which results in naive networks.

Analysts can use the GBN methods to analyze their gene expression data for two reasons. As we have shown, marker genes can be found by comparing

the learned GBNs. Ideally, the programs are able to compare the networks automatically, but since this is not the fact we compared them by hand.

Another reason to use the GBN learners is their graphical output; direct overview of the results, gene relationships can be explored and the networks can be manipulated by experts.

The use of the GBN learners for the purpose of analyzing microarray data is restricted to finding marker genes and visualizing possibly interesting gene relationships. The methods we used cannot really cope with small amount of evidence and the large amount of variables; one "perfect" network cannot be expected as a result.

6.2 Future work

The results and conclusions of this thesis led to ideas how to improve the usability of the LBN programs and how we could make better use of them for analyzing gene expression data.

Improvements The most important improvement for the BNC learners is the integration of automated gene selection methods making leave-N-out cross-validation possible. Estimating the classification error is of great importance when we are dealing with patients. If we want to use the BNCs we need a robust estimation of the error and integrating gene-selection methods can help to achieve this.

A second improvement of the LBN programs would be to have a dynamic Bayesian network learner specialized to learn from time-course experiments. These learners can provide more insight into the dynamics of biological mechanisms. A lot of time-course experiments are already available.

At last, we would like to be able to automatically compare and combine the GBNs we learned. By making a comparison between networks, we were able to find possibly interesting marker genes, which is one of the purposes of doing microarray experiments. By automatically merging and combining networks, see Section 2.3 for examples, the networks might become more robust and give a better representation of the underlying structure.

Hopefully, methods to learn Bayesian networks will keep improving the coming years, just as they did the last few years. We would like to be able to include more genes in our networks and include continuous data. As computer powers keep increasing exponentially over time, hope exists, although the number of networks increases super-exponentially with the number of nodes in the network.

Experiments In general, the number of samples in microarray experiments must grow. We used relatively large datasets compared to common

microarray experiments with about 5 to 20 samples. For these experiments the robustness and overfitting problems are even more important. A transformation in the way of thinking about experiments is needed. Unfortunately, microarray experiments are very costly: only one array costs about 500 Euro's. If we add costs of extracting RNA etc., the experiment becomes very expensive. Luckily, experiments which use a lot of samples get more common.

Literature based gene selection An idea to make better use of LBNs to analyze microarray data, is to select genes based on prior belief that the used genes are somehow related. In our case we selected genes by their correlation with some class. These genes do not necessarily have to be involved in the same biological mechanism. To learn Bayesian networks we could select gene expression patterns of genes known, or thought, to be related. Mootha *et al.* [MLE+03] propose a gene selection method based on this idea, called Gene Set Enrichment Analysis. Basically, they make sets of genes known to be related[1] and tested the correlation of these *sets* of genes with the target classes. The gene group with the highest correlation could represent a disease specific mechanism and learning a network from these genes could give some extra insights into the underlying mechanism.

Our gene markers The set of four genes found by comparing several GBNs could be interesting for future research. First, the way in which the genes are involved in the development of metastases is not yet known and, as our results indicate, could be very important. Second, a diagnostic tool based on only four genes could be designed. The study by van 't Veer *et al.* was used to design a prognostic microarray containing the 70 genes [VHV+02]. The prognostic array performs better than currently used methods to predict metastases. A prognostic tool using only four genes instead of 70 genes could be an improvement for practical reasons, including costs.

Concluding remarks The field of microarrays will have impact in medicine, molecular biology and bio-technology. Artificial intelligence can greatly help and speed up knowledge discovery from microarrays. Our main contribution to microarray data analysis is that we have shown that interesting biological information can be found and good classification accuracies can be obtained using easy to use LBN programs. However, the programs need to be extended for microarray research to make full use of their possibilities. Hopefully, our results, conclusions and ideas for future work will inspire others to join this rapidly evolving field.

[1] Large databases exist with information on which genes belong to a certain biological mechanism. See also http://www.affymetrix.com/analysis/index.affx for example.

Acknowledgements As we have reached the end, I would like to thank a few persons who have been helping me along the way.

First, I would like to thank the MicroArray Department for creating a very pleasant atmosphere to work in. It was exiting to see the department expanding so rapidly and becoming an important part of life sciences. Han, and Marco, have been helpful in microarray issues, as well as statistics and by inspiring me with scientific articles.

Han, Sjaak and my Dad were kind enough to comment my thesis, which has been of great help (although it might have caused some delay :), thanks.

Finally, I would like to thank Frans. Starting from the day 1 when I contacted him to do research in biology and AI, he has been supporting me, giving me tips and hints on writing, explaining the Bayesian part, keeping me on track (which was nessecary considering some wild, probably unrealisable, ideas that I proposed) by making sure we would meet once in every two weeks and helping me to get a job at the AMC.

Bibliography

[AMRM04] K. Abbas, A. Mikler, A. Ramezani, and S. Menezes. Computational epidemiology: Bayesian disease surveillance. In *Proc. of the International Conference on Bioinformatics and its Applications (ICBA '04)*, 2004.

[AMZ+01] I. Alevizos, M. Mahadevappa, X. Zhang, H.Ohyama, and Y. Kohno. Oral cancer in vivo gene expression profiling assisted by laser capture microdissection and microarray analysis. *Oncogene*, 20(43) no.6:196–204, September 2001.

[BSB+03] D. P. Berrar, B. Sturgeon, I. Bradbury, C. S. Downes, and W. Dubitzky. Microarray Data Integration and Machine Learning Techniques for Lung Cancer Survival Prediction. *Critical Assessment of Techniques for Microarray Data Analysis Techniques*, 2003.

[CBL97a] J. Cheng, D. A. Bell, and W. Liu. An algorithm for Bayesian belief network construction from data. In *Proceedings of AI and STAT 1997*, pages 83–90, 1997.

[CBL97b] J. Cheng, D. A. Bell, and W. Liu. Learning belief networks from data: An information theory based approach. In *Proceedings of ACM CIKM 1997*, 1997.

[CG98] J. Cheng and R. Greiner. Comparing Bayesian Network Classifiers. University of Alberta, 1998.

[CG99] Cooper and Glymour. *Computation, Causation and Search.* AAAIPress/The MITPress, 1999.

[CL68] C.K. Chow and C.N. Liu. Approximating discrete probability distributions with dependence trees. *IEEE Trans. on Information Theory*, 14:462–467, 1968.

[Cri58] F.H.C. Crick. On protein synthesis. In *Symposium of the society for experimental biology XII*, page 153, 1958.

[DFS00] S. Dudoit, J. Fridlyand, and T. P. Speed. Comparison of Discrimination Methods for the Classification of Tumors Using Gene Expression Data. Technical Report #576, UC Berkely, Baltimore, Maryland, June 2000.

[DLS00] P. D'Haeseleer, S. Liang, and R. Somogyi. Genetic Network Inference: from Co-expression Clustering to Reverse Engineering. *Bioinformatics*, 16 no. 8:707–726, 2000.

[Dut99] Bas Dutilh. Analysis of data from microarray expperiments, the state of the art in gene network reconstruction, 1999.

[DWFS99] P. D'haeseleer, X. Wen, S. Fuhrman, and R. Somogyi. Linear modeling of mrna expression levels during cns development and injury. In *Pacific Symposium on Biocomputing*, pages 41–52, 1999.

[FGG97] N. Friedman, D. Geiger, and M. Goldszmidt. Bayesian Network Classifiers. *Machine Learning*, 29:131–161, 1997.

[FLNP00] Nir Friedman, Michal Linial, Iftach Nachman, and Dana Pe'er. Using Bayesian networks to analyze expression data. In *Proceedings of the fourth annual international conference on Computational molecular biology*, pages 127–135. ACM Press, 2000.

[Gla01] Stanton A. Glantz. *Primer of Biostatistics.* McGraw-Hill Professional, fifth edition, 2001.

[GST+99] T. R. Golub, D. K. Slonim, P. Tamayo, C. Huard, M. Gaasenbeek, J.P. Mesirov, H. Coller, M. L. Loh, J. R. Downing, M. A. Caligiuri, C. D. Bloomfield, and E. S. Lander. Molecular classification of cancer: Class discovery and class prediction by gene expression monitoring. *Science*, 286:531–537, 1999.

[HVAW02] Paul Helman, Robert Veroff, Susan R. Atlas, and Cheryl Willman. A Bayesian Network Classification Methodology for Gene Expression Data. Technical Report TR-CS-2002-18, Computer Science Department, University of New Mexico, 2002.

[Jen96] Finn V. Jensen. *Introduction to Bayesian networks.* UCL Press, 1996.

[Kau93] S. A. Kauffman. *Origins of Order : Self-Organization and Selection in Evolution.* Oxford University Press, Oxford, 1993.

[KDL+03] O. Kulaeva, S. Draghici, L.Tang, J.M. Kraniak, S.J. Land, and H.A. Tainsky. Epigenetic silencing of multiple inferon pathway genes after cellular immortalization. *Oncogene*, 22:4118–4127, 2003.

[Koh95] Teuvo Kohonen. *Self-Organizing Maps.* Springer, Berlin, first edition, 1995.

[LDB+96] D.J. Lockhart, H. Dong, M.C. Byrne, M.T. Follettie, M.V. Gallo, M.S. Chee, M. Mittmann, C. Wang, M. Kobayashi, H. Horton, and E.L. Brown. Expression monitoring by hybridization to high-density oligonucleotide arrays. *Nature Biotechnology*, 14:1675–1680, 1996.

[MA97] H.H. McAdams and A. Arkin. Stochastic mechanisms in gene expression. *Proc. Natl. Acad. Sci (PNAS)*, 94:814–819, 1997.

[MLE+03] V.K. Mootha, C.M Lindgren, K.F. Eriksson, A. Subramanian, and S. Sihag. PGC-1α-reponsive genes involved in oxidative phosphorylation are coordinately downregulated in human diabetes. *Nature Genetics*, 34(3):267–273, 2003.

[MSR91] E. Mjolness, D. H. Sharp, and J. Reinitz. A connectionist model of development. *J. theor. Biol.*, 152:429–456, 1991.

[MWPS] M. Molla, M Waddel, D. Page, and J Shavlik. Using machine learning to design and interpret gene-expression microarrays. *AI magazine, Special issue on bioinfomatics*, To appear.

[Neo04] Richard E. Neopalitan. *Learning Bayesian Networks.* Pearson Education, first edition, 2004.

[OPG02] I. Ong, D. Page, and J.D. Glasner. Modelling regulatory pathways in *E.Coli* from time-series expression profiles. *Bioinformatics*, 17:215–224, 2002.

[PCW+02] D. Page, F. Zhan J. Cussens, M. Waddell, J. Hardin, B. Barlogie, and J. Shaughnessy. Comparative Data Mining for Microarrays: A Case Study Based on Multiple Myeloma. Technical Report 1453, Computer Sciences Department, University of Wisconsin, 2002.

[Pea88] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: networks of plausible inference.* 1988.

[Qua01] John Quackenbush. Computational analysis of microarray analysis. *Nature Reviews Genetics*, 2(6):418–427, June 2001.

[Ram99] M. Ramoni. Bayesware discoverer 1.0, http://www.bayesware.com, 1999.

[RN95] Stuart J. Russel and Peter Norvig. *Artificial Intelligence: A Modern Approach.* Prentice-Hall International, 1995.

[RS98] M. Ramoni and P. Sebastiani. Parameter estimation in Bayesian networks from incomplete databases. *Intelligent Data Analysis*, 2(1), 1998.

[RS99] M. Ramoni and P. Sebastiani. *Intelligent Data Analysis: An Introduction*, chapter 4: Bayesian methods. Springer-Verlag, 1999.

[RS01] M. Ramoni and P. Sebastiani. Robust Bayes Classifiers. *Artificial Intelligence*, 125:209–226, 2001.

[SBC+02] A. Szabo, K. Boucher, W.L. Carrol, L.B. Klebanov, A.D. Tsodikov, and A.Y. Yakovlev. Variable selection and pattern recognition with gene expression data generated by the microarray technology. *Mathematical Biosciences*, 176:71–98, 2002.

[SEBB98] Paul T. Spellman, Michael B. Eisen, Patrick O. Brown, and David Botstein. Cluster analysis and display of genome-wide expression patterns. *Proceedings of the National Academy of Sciences (PNAS)*, 95(25):14863–14868, December 1998.

[SGS93] P. Spirtes, C. Glymour, and R. Scheines. *Causation, Prediction, and Search*. Springer Verlag, first edition, 1993.

[SKR03] Paola Sebastiani, Isaac S. Kohane, and Marco F. Ramoni. Machine learning in the genomics era editorial: Methods in functional genomics. *Machine Learning*, 52:5–9, 2003.

[SSM+98] P.T. Spellman, G. Sherlock, M.Q., M. Zhang, V. Iyer, K. Anders, M. Eisen, P. Brown, and D. Botstein. Comprehensive identification of cell cycle regulated genes of yeast by microarray hybridization. *Molecular Biology of the Cell*, 9:3273–3297, 1998.

[TCS+01] O. Troyanskaya, M. Cantor, G. Sherlock, P. Brown, T. Hastie, R. Tibshirani, D. Botstein, and R. Altman. Missing value estimation methods for DNA microarrays. *Bioinformatics*, 17 no 6.:520–525, 2001.

[TDO+03] O. Troyanskaya, K. Dolinsky, A.B Owen, R.B. Altman, and D.Botstein. A Bayesian framework for combining heterogeneous data sources for gene function prediction. *Proceedings of the National Academy of Sciences (PNAS)*, pages 8348–8353, 2003.

[TSM+99] Pablo Tamayo, Donna Slonim, Jill Mesirov, Qing Zhu, Sutisak Kitareewan, Ethan Dmitrovsky, Eric S. Lander, and Todd R. Golub. Interpreting patterns of gene expression with self-organizing maps: Methods and application to hematopoietic

differentiation. *Proceedings of the National Academy of Sciences (PNAS)*, 96(6):2907–2912, March 1999.

[Val02] F. Valafar. Invited article: Pattern Recognition Techniques in Microarray Data Analysis: A Survey. *Techniques in Bioinformatics and Medical Informatics*, 980:41–64, 2002.

[VDH+02] L.J. van 't Veer, H. Dai, Y.D. He, M. J. van de Vijver, A.A.M. Hart, and M. Mao. Gene expression profiling predicts clinical outcome of breast cancer. *Nature*, 415:530–535, January 2002.

[VHV+02] M.J. van de Vijver, Y.D. He, L.J. van 't Veer, H. Dai, and A.A.M Hart *et al.* A Gene-Expression Signature as a Predictor of Survival in Breast Cancer. *The New England Journal of Medicine*, 347:25:1999–2009, 2002.

[Wag02] A. Wagner. Estimating coarse gene network structure from large-scale gene perturbation data. *Genome Research*, 12:309–315, 2002.

[WC53] J. D. Watson and F. H. C. Crick. Molecular Structure of Nucleic Acids: A Structure for Deoxyribose Nucleic Acid. *Nature*, 171:737–738, 1953.

[WH00] M. Wahde and J. Hertz. Coarse-grained reversed engineering of genetic regulatory networks. *Biosystems*, 55:129 – 139, 2000.

[ZH03] B.-T. Zhang and K.-B. Hwang. Bayesian network classifiers for gene expression analysis. In D.P. Berrar, W. Dubitzky, and M. Granzow, editors, *A Practical Approach to Microarray Data Analysis*, pages 150–165. Kluwer Academic Publishers, 2003.