Predicting protein-protein interactions in time-series of gene expression data

MASTER THESIS

C.F.H.A. Gilissen

June 5, 2006

University Medical Center Nijmegen
Department of Antropogenetica
Supervisor: Dr. Michael Egmont-Petersen

Radboud University Nijmegen
Information and Knowledge Systems
Supervisors: Dr. Peter Lucas
Dr. Perry Groot

Abstract

The function of all living cells is controlled by the process of *gene expression*; the translation of the genetic codes (genes) in the DNA into proteins that regulate the cell, as well as the expression of genes themselves [Str92, SR96]. This indirect interaction of genes composes a gene regulatory network regulating the cell [DLS00]. Due to the development of microarrays it is now possible to (indirectly) measure the expression of genes and thus to reverse engineer these gene regulatory networks. These can give insight into many genetically related diseases and disorders [WMSB98, ZHZ+04].

In this thesis we extend on a method from [EPdJS04]. For this we use the Spellman [SSZ+98] time-series gene expression data. Firstly we remove noise by applying a scale-space smoothing, after which the expression data is discretized based upon the local extrema in the expression pattern. A similarity function is applied to the discretized expression values of the genes, grouping them in a pairwise fashion. We remove chance findings by combining the findings of two datasets. We validate the results by a gold standard protein-protein interaction database, showing that local extrema are a significant feature for identifying gene relations.

We also experiment with learning dynamic Bayesian networks from the discretized expression data, for finding causal gene interactions.

Table of Contents

List of Figures

List of Tables

"Computer Science is no more about computers than astronomy is about telescopes."

E. W. Dijkstra

Chapter 1

Introduction

With the development of microarray technology [Dop02, HQA$^+$00, SMS03, WMSB98, ZHZ$^+$04], the unraveling of the functioning of DNA has taken an enormous step forward. Great amounts of gene expression data from various species are now available for analysis. However, the sheer amount of data alone poses a problem as manual analysis methods are no longer sufficient. The data analysis must be automated, which brings to light a new problem: the inherent noisiness of the data [FLNP00, Hus03b]. This is a far greater problem with automated analysis than with expert analysis. Another problem lies in the fact that biological processes are often very complex, and thus the underlying gene regulatory network is very complex [FLNP00, GK02, DLS00, vSWR01]. Common interactions for example include stimulation, inhibition, feedback loops, and activation [SR96].

In this thesis we expand on a method from [EPdJS04] for finding co-regulation and control regulation between genes, from time-series microarray expression data, with the ultimate goal of recovering gene regulatory networks that provide us with insight in the functioning of the DNA. We use a scale-space method [FtHRKV92, SM03] to smoothen and discretize the data and show that local extrema are indeed a significant feature of gene expression data, and can be used for finding gene interactions, and more specific, protein-protein interactions. We use the discretized data for experiments with learning Bayesian networks to model protein-protein interactions [FLNP00, Hus03b].

The thesis is organized as follows, the first chapter constitutes this introduction. The second chapter provides a background for the main topic, the analysis of microarray expression data. In Chapter 3 we treat the applied methods. The results are presented in the Chapter 4 and will be discussed in Chapter 5. In the final Chapter 6 we formulate our conclusions and consider future directions for research.

"Imagination is more important than knowledge ... "

Albert Einstein

Chapter 2

Background

2.1 Cell Biology

The biological information contained in an organism is encoded in its DNA (deoxyribonucleic acid). Human DNA is a double stranded helix composed of base pairs of different nucleotides: A (adenine), C (cytosine), T (thymine), and G (Guanine). The pairing of nucleotides follows the Watson-Crick rules: adenine binds only to thymine (A-T) and cytosine binds only to guanine (C-G), together forming a base pair. The two strands of DNA from the helix can be considered each others complement. The nuclear genome of a human being roughly consists of 1.3×10^9 of such base pairs [Str92, SR96].

The DNA which carries genetic information in biological cells is normally packaged in the form of one or more large macromolecules called chromosomes which reside in a cell's nucleus. If fully uncoiled the DNA content of a chromosome would be between 1.7 and 8.5 cm long. A normal human cell contains 46 chromosomes of which one is a sex specific chromosome.

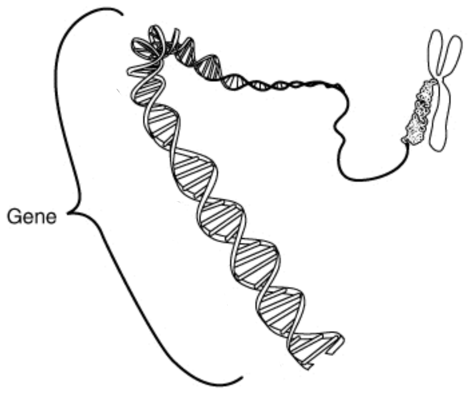


Figure 2.1: This stylistic schematic diagram shows a gene in relation to the double helix structure of DNA and to a chromosome (right).

Genes are regions of the DNA that encode information essential for the construction and regulation of proteins as well as other molecules that determine the growth and functioning of the organism. Only 2-3% of the human DNA is coding DNA; DNA that specifies a polypeptide or functional RNA product. The remaining DNA does not encode for any proteins, this could be a reservoir or a remainder of evolution. This DNA is sometimes referred to as "junk-DNA". Human DNA is estimated to contain about 22000 genes which code for even more proteins. More than 99% of these genes are exactly the same for any human being independent of their ethnic background [Str92, SR96].

Genes are templates for the protein construction within a cell. Protein synthesis takes place at the ribosomes (shown in Figure 2.2), and thus somehow information has to be carried from the DNA in the nucleus to the ribosomes in the cytoplasm. During *transcription*, a molecular complex named RNA polymerase II makes a copy of a gene from the DNA to messenger RNA (mRNA) as needed. The mRNA then travels to a ribosome which then translates the mRNA into a protein. This is called *translation*. Transcription and translation are graphically illustrated in Figure 2.3; this entire process is called *gene expression*. A ribosome can be thought of as a factory that builds a protein from a set of genetic instructions. These proteins then regulate the cell including gene transcription. In this way genes control the functioning of a cell, and thus the functioning of an organism. This is the fundamental dogma of molecular biology [SR96].

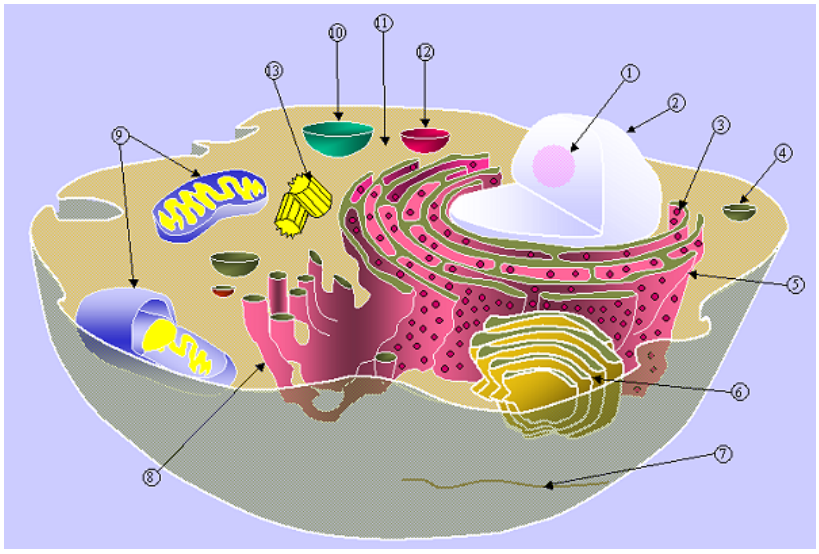


Figure 2.2: Schematic of typical animal cell, showing subcellular components. Organelles: (1) nucleolus (2) nucleus (3) ribosome (4) vesicle(5) rough endoplasmic reticulum (ER), (6) Golgi apparatus, (7) Cytoskeleton, (8) smooth ER, (9) mitochondria, (10) vacuole, (11) cytoplasm, (12) lysosome, (13) centrioles

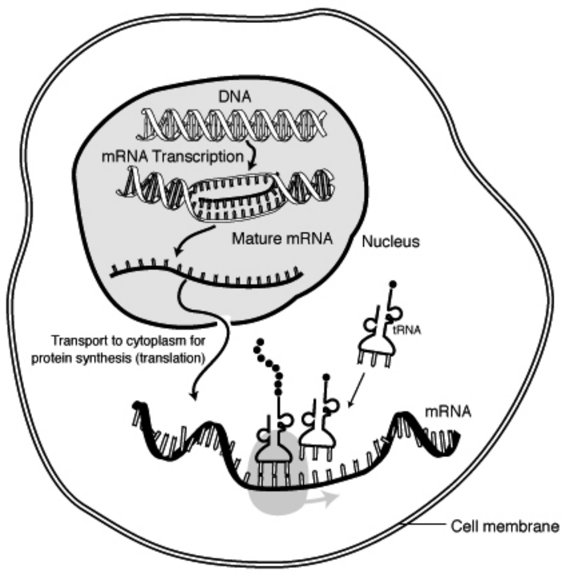


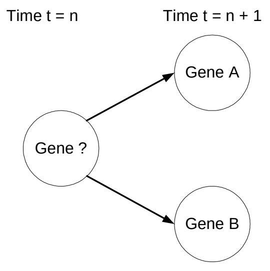
Figure 2.3: Schematic representation of transcription and translation of DNA.

2.2 Gene Regulatory Networks

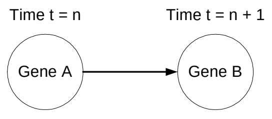
Most genes can be viewed as on-off switches regulating a cell's functions [Gen05]. The expression of genes is, like all other functions within a cell, controlled by proteins and thus indirectly by other genes. Groups of genes control the functioning of a cell by complex interactions. These interacting gene groups are called Gene Regulatory Networks (GRNs). Gene expression while in fact a continuous biological process, is often abstracted to *up-regulation* and *down-regulation*. Gene interactions can be divided into two major groups:

- *Co-regulation* (or co-expression), meaning that the expression of two genes is controlled by another gene, as shown in Figure 2.4(a). This does not, however, mean that two co-regulated genes show the same sign in relative expression pattern, but two co-regulated genes are expected to show significant changes in expression at the same times.

- *Control regulation* (or control expression), meaning one gene controls the expression of another gene at the transcriptional level, as shown in Figure 2.4(b). It is conjectures that gene that is control regulated will show changes in expression a constant time c later than the gene it is controlled by. (For $c = 0$ we have co-regulation, provided both are controlled by a common third gene).



(a) Co-regulation, some gene controls genes *A* and *B* (at the same time)



(b) Control-regulation, gene *A* controls gene *B*

Figure 2.4: Gene expression patterns

Additionally we distinguish two ways in which one gene can control another:

- *Stimulation*: The expression of one gene is stimulated by the other. The increase of expression of one gene triggers an increase of expression of the other, while a decrease of expression triggers a decrease of expression.
- *Inhibition*: The gene's expression is inhibited by the other. The increase of expression of one gene triggers a decrease of expression of the other, while a decrease of expression triggers an increase of expression.

Furthermore it is very common for biological systems to be controlled by *feedback loops* [Gen05, Hus03b]. For example, a gene *A* stimulates a gene *B* which in turn inhibits *A* as shown in Figure 2.5. The shortest path of such feedback loops can span any number of genes.

Some genes are regulated by only one other gene, but others are regulated by multiple genes. However, it is known that genes are commonly regulated by only a few other genes [FLNP00, Hus03b] (they have a low indegree), while on the other hand a gene can regulate many other genes (a high outdegree). Such highly connected genes dominating the network topology, are referred to as a *hub genes* [ZH05]. Many hub genes are known to regulate in a dose independent way which makes their detection relatively difficult.

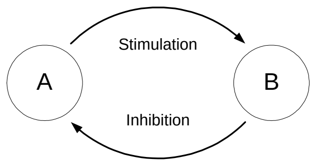


Figure 2.5: An example of a feedback loop

2.3 Microarrays

Because an mRNA strand is the compliment of the gene that was transcribed, measuring mRNA levels will give a measure of the gene transcription levels (the gene expression). This gene expression information is extremely valuable not only from the point of pure scientific interest, but also from the point of characterizing and curing numerous diseases. A recent technological breakthrough called *microarrays* now makes it possible to measure simultaneously the amount of thousands of different mRNA strands within a cell. The two main types of microarrays are cDNA microarrays (a.k.a. spotted microarrays) and oligonucleotide microarrays.

2.3.1 cDNA microarrays

With cDNA microarrays* single strands of DNA with known sequences (called probes) are immobilized on spots arranged as a grid (or array) on a (mostly glass or nylon) slide (the microarray) [Dop02, Got05, HQA+00, SMS03, WMSB98, ZHZ+04]. These strands correspond to genes (or just nucleotide sequences) of which the transcription levels are to be measured. mRNA is extracted from two cell types and reverse transcribed into more stable cDNA. One type is made up out of control cells which are used as a reference (control cDNA) for the other cells. The other cells are the cells under study (test cDNA). The cDNA is labeled with fluorescent dyes, commonly Cyanine 3 (red) and Cyanine 5 (green) (Cy3, Cy5) in order to identify reference and test DNA. Now the labeled cDNA is hybridized onto the microarray and undergoes competitive binding (hybridization). The cDNA that did not bind is washed away. A scanner can measure how much of each red and green labels is present for each spot on the microarray. This corresponds to the amount of bound cDNA. By comparing the amounts of control cDNA with the test cDNA on a certain spot on the microarray, relative transcription differences can be found between the control and test DNA for the specific strands of DNA that were on that spot. This is the most common cDNA microarray experiment: a comparative hybridization experiment. The entire process is illustrated in Figure 2.6.

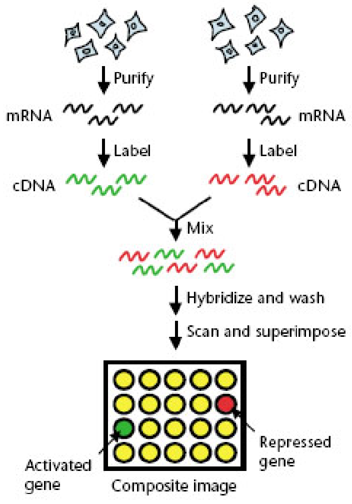
2.3.2 Oligonucleotide microarrays

With oligonucleotide microarrays (Affymetrix, Santa Clara, CA) the labels are directly synthesized on the microarray using a photolithographic process. For this reason a comparative hybridization experiment as described above is not possible. Oligonucleotide microarrays are mostly used for conducting measurements at different time intervals under changing conditions for the test DNA. For instance, the first measurement occurs at a temperature of 37 degrees Celsius while a second measurement is done at 39 degrees.

The data obtained from measuring the expression of test DNA at more than one point in time is referred to as time-series expression data. Time-series can be used for studying both gene co-regulation as well as control regulation.

Another well known method for measuring gene expression is PCR (Polymerase chain reaction). PCR is a more accurate method than microarrays and is relatively fast and simple to perform. It can however only be used for a relatively small number of genes and the sequences of the target genes must be known [SR96].

*Complementary DNA (cDNA) is DNA synthesized from a mature mRNA template.



(a) The cDNA Microarray process

(b) A DNA Microarray, the different colours indicate relative expression of different genes

Figure 2.6: The microarray process from [SMS03]

2.4 Analysis of microarray data

Microarrays have provided us with large quantities of data for finding gene functionality and gene regulatory pathways. However, extracting information from the data is not a straightforward process.

When comparing data, it must be normalized for different variances that can occur. There can be variances due to unequal quantities of starting RNA and differences in labeling or detection efficiencies. For all these different variances different normalization techniques exist, mostly based on statistical methods. For instance, analysis of variance (ANOVA) techniques are widely used [KC01, KMC00, Qua02] as well as *lowess* normalization [ZHZ+04].

When analyzing the (normalized) data for finding gene functionality there are some well-known problems [FLNP00, GK02, vSWR01]:

- The data contains a lot of noise which makes it difficult to analyze.
- The number of genes is very large (6000 (Saccharomyces cerevisiae) to 22000 (Homo sapiens)) while the number of samples is usually very small. We are thus performing multiple testing, which makes conventional statistics less reliable.
- Interest lies in gene functionality instead of mRNA levels. mRNA levels only give a partial view of gene functionality. mRNA is broken down relatively fast, while the protein it encoded for may be present for a far longer time, influencing the expression of other genes.

- Gene regulations are often complex chemical processes involving multiple components with different biochemical roles.

A lot of computational methods have been used to analyze gene expression data. Clustering methods [DLS00] were one of the first to succeed. These methods attempt to classify genes into groups with similar expression patterns as shown in Figure 2.7. Several different clustering methods like hierarchical clustering, K-nearest neighbor, K-means, voting, and Self-Optimizing Maps [BDSY99, CdAdCdS04, ESBB98] have been applied. These methods have been very useful for finding co-regulated genes. They are however less suitable for finding regulatory pathways [Hus03b]. Some of the other methods employed for analyzing gene expression are *differential equations* [dHIK+03], *linear splines* [dHIM02], *singular value decomposition* [ABB02], *wavelets*, *neural networks* [NKGT04], *support vector machines* [BGL+99], *Boolean networks* [Kau69] and *Bayesian networks* [FLNP00, Hus03a].

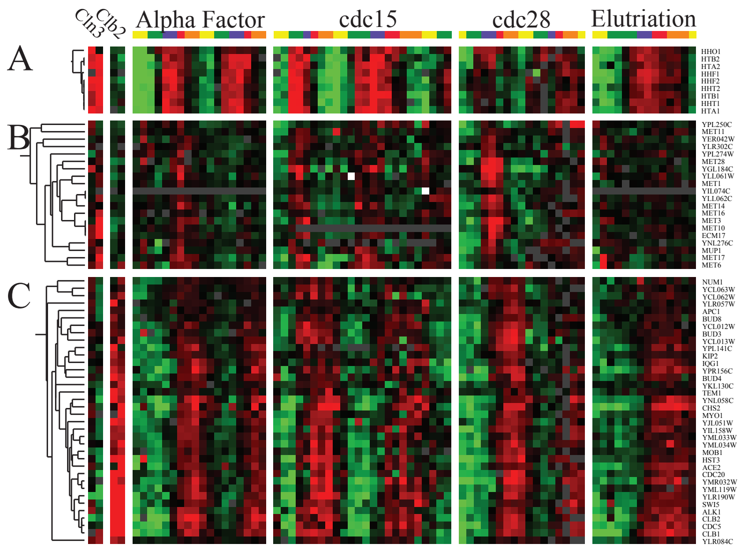


Figure 2.7: An example of clustering expression data from [SSZ+98].

2.4.1 Time-series analysis

Gene expression time-series representations can be categorized into four main groups according to [AO01, SM03]:

1. time-domain continuous representation: the time-series are not preprocessed and are represented as time-ordered continuous values.

2. transformation based representations: the time series are transformed to another domain and a point in that domain is used as a representation for the sequence (for example, a transformation to the frequency domain.

3. discretisation based representation: the continuous values are represented as discrete values.

4. generative models representation: a model is obtained that generates the time-series (for example Hidden Markov Models).

When an appropriate representation for the time-series is chosen, some kind of similarity measure is needed to group genes together. Based on the representation there are again four groups of similarity measures [AO01, MLCYW03]:

1. time-domain continuous. The most common similarity measure is the Euclidean distance where each time-series of length k is a point in a k-dimensional space.

2. transformation based. The transformed time-series (represented as a point) are commonly compared using the Euclidean distance.

3. discretisation based: Each point in a sequence is compared to the corresponding point in the other sequence (e.g., Hamming distance).

4. generative models: similarity is measured on the basis of how likely the time-series was generated by the model.

"The problem with wrong proofs to correct statements is that it is hard to give a counterexample."
Hendrik Lenstra, 1997

Chapter 3

Methods

We intend to find co-regulation genes and control regulated genes with a discrete representation of time-series expression data, based on an approach from [EPdJS04]. We use local extrema as the basis for discretization and use additional filters to improve our findings.

We specifically use time series data because because we intend to discover causal relationships, which are characterized by the fact that the cause always precedes the effect, in time [Els83]. Furthermore it has been said that time series analysis shows a significant improvement compared to non-time series data for reverse engineering genetic networks [Hus03b]. We choose to look for local extrema within the expression patterns of genes [EPdJS04]. Local extrema signify the turning points in gene expression, where expression goes from increase to decrease of vice versa. Thus we determine a discrete value for each sample, where a gene g can at any time t show a local maximum, local minimum, or change. Genes are then compared on the basis of their discrete pattern and some similarity function. Such a similarity function can incorporate a lag in time with which we can find co-regulations (lag 0) or control regulations (lag $1, 2, \ldots$). Genes that are hypothetically similar are grouped together in a pairwise manner and are called relations. We then use filters to reduce the number of relations that we find. The remaining relations are then validated with use of public protein-protein interactions from gold standards [SMS$^+$04, JYG$^+$03, GMK$^+$05] .

For the entire method we use the following sequential approach as illustrated in Figure 3.1:

1. Pre-processing by max-min filter and a scale-space method (Figure 3.1 b, c)
2. Finding discrete values by looking for local extrema (Figure 3.1 d)
3. Finding possible gene relations by comparing discrete patterns (Figure 3.1 e)
4. Filtering relations using gene features and/or ontology information (Figure 3.1 f)
5. Validation of the found relations using a protein-protein interaction database

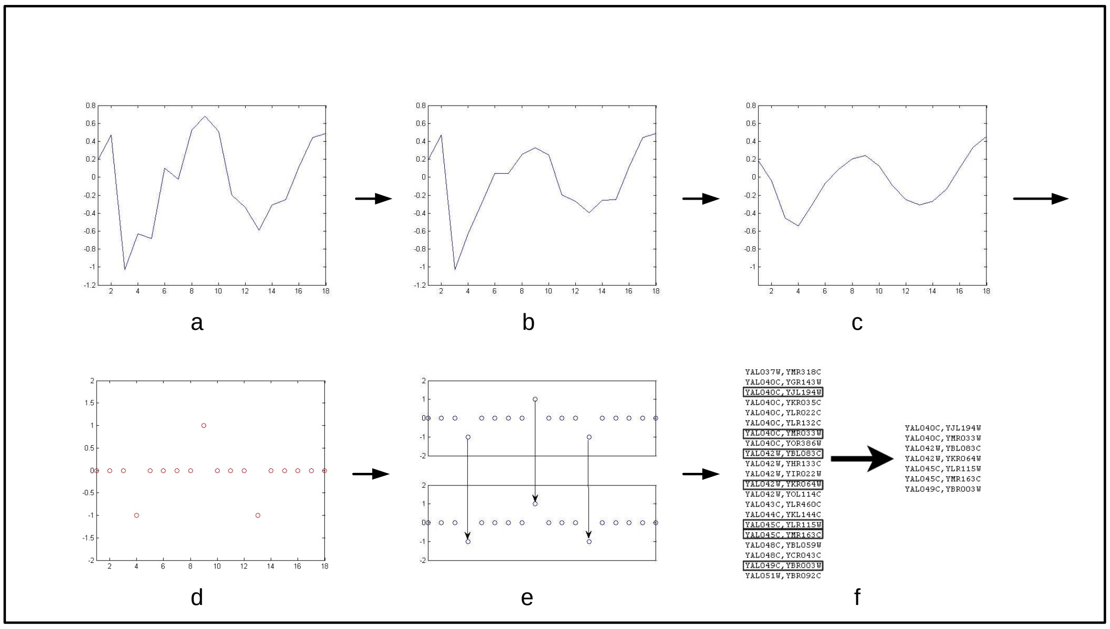


Figure 3.1: An overview of the applied method. Starting from the left top: a) original expression data; b) max-min filtered data; c) smoothed data; d) discretized data; e) gene comparison; f) relation filtering.

We also use the discretized expression data as a basis for experimenting with learning dynamic Bayesian networks with MCMC algorithms [Hus04, Mur01].

3.1 Data

Microarray data D can be viewed as a $n \times m$ data matrix where n is the number of probes and m is the number of samples. Each value can then be uniquely identified by the element D_{ij}, $1 \leq i \leq n, 1 \leq j \leq m$ for some i and j. Probes are strings of nucleotides and in most cases will directly correspond to a known gene. For this reason we will often refer to probes as genes and vice versa. Microarray data is usually analyzed on the log scale (usually log 2 [Got05]) because the effects of the data are believed to be multiplicative (and thus additive on the log scale) [Got05]. An expression value for a gene is thus typically a real number approximately between 2 and -2. In the case of time series expression data, the samples m represent consecutive measurements at different times.

$$D = \begin{pmatrix} D_{1,1} & D_{1,2} & \dots & D_{1,m} \\ D_{2,1} & D_{2,2} & \dots & D_{2,m} \\ \vdots & \vdots & \vdots & \vdots \\ D_{n,1} & D_{n,2} & \dots & D_{n,m} \end{pmatrix}$$

We will frequently refer to all or one specific sample of a probe. Let g denote a probe (or gene) then:

$$g = D_i \text{ for some } 1 \leq i \leq n \tag{3.1}$$

$$g_j = D_{i,j} \text{ for some } 1 \leq i \leq n \tag{3.2}$$

For our approach we need time-series data containing samples at different times. The Nyquist–Shannon sampling theorem [Sha49, Weib] states that the discrete samples are a complete representation of the signal if the bandwidth is less than half the sampling rate. Or more specific, we should sample often enough to be able to reconstruct the original signal. Unfortunately, there is not yet the biological knowledge to give a concrete answer as to what time intervals are too large for finding certain gene expression patterns. Ideally, these samples should be uniformly spaced, but algorithms for resolving problems related to unevenly spaced samples exist. As we will be comparing expression patterns, we need a sufficient amount of samples in order for the patterns to be unique enough.

When the sampling intervals are too large, different problems can occur. First of all the measurements will show little difference in expression, possibly only one or two peaks in expression, but nothing more. This will result in a lot of similar patterns, which results in finding a lot of relations. A second consideration is that when measuring expression for some biological process which takes time T, we (ideally) need time series data over $2T$ time, because we want to find at least one reoccurrence of every event. This way we can discriminate the biological process from other random processes.

Due to the relative high costs of time-series experiments, there are few public domain time-series available. Moreover, there are a lot of practical problems for conducting time-series experiments. As mice or yeast cells can be artificially grown it is possible to create large amounts of genetically identical cells. However with Homo sapiens, this is far more difficult. For this reason we started with yeast cells from the Spellman dataset, which is ideal for our approach.[*]

We downloaded three datasets of the Saccharomyces cerevisiae (baker's yeast) from the Gene Expression Omnibus [BST+05] which contains the original data from the Stanford Microarray Database [BAD+05]. All three datasets come from a comprehensive effort to identify cellcycle regulated genes [SSZ+98]. In each of the datasets a different method was used to synchronize the cells. The measurements in all sets were log converted.

1. The first dataset (GEO DataSet 38, GDS38) is a time series, where samples were taken every 7 minutes as the cells went through their cell cycle. This dataset is referred to as α-factor arrest because of the method that was used to synchronize the cells. The original dataset contained 7680 probes and 16 samples. If a probe was missing only one value, this value was averaged with neighboring samples. Probes with two or more missing values were left out, resulting in a set of 6493 samples.

[*]We also experimented with Homo sapiens data, but the results are not included here.

2. The second dataset (GDS39) contains 14 time samples taken every 30 minutes and 7680 probes. After removing probes with too many (more than one) missing values, 7074 probes remain. The original experiment was performed by a synchronization method called elutriation and thus we will refer to this dataset as the *elutriation* dataset.

3. The third dataset (GDS124) contains 19 samples taken every 10 minutes and 8832 probes. After removing probes with too many missing values, 6241 probes remain. The synchronization of the cells was achieved by blocking the gene CDC15; this is why the dataset is referred to as the *CDC15* dataset.† The original experiment contains more samples but not all were uniformly spaced in time. So we extracted only the samples with uniform intervals of 10 minutes.

3.2 Preprocessing

As the data is inherently noisy we need some way to filter out this noise; regularization is required. Secondly we need a method for efficiently detecting local maxima and minima within the expression pattern.

3.2.1 Max-min filter

We borrow the max-min filter technique [EPdJS04, VVvV88] from the field of signal processing, to remove insignificant extrema. Max-min filters attribute to each sample of a gene a new value equal to the maximum or minimum value in a neighborhood around that sample. A max-min filter employed on the expression data of gene g at time t will result in a new expression value according to:

$$g = \frac{\max\limits_{t' \in w(t)} \left(\min\limits_{t'' \in w(t')} (g_{t''}) \right) + \min\limits_{t' \in w(t)} \left(\max\limits_{t'' \in w(t')} (g_{t''}) \right)}{2} \tag{3.3}$$

Where $w(t)$ is the neighborhood window taken for time t. A max-min filter thus effectively removes small local extrema in the expression pattern, as shown in Figure 3.2.‡ By choosing a larger window, larger inflictions will be removed. However, with gene expression data there are typically too few samples for a larger window than the minimum of three.

†This dataset was used for the illustrations showing gene expressions.

‡Linear interpolation is used for connecting subsequent samples.

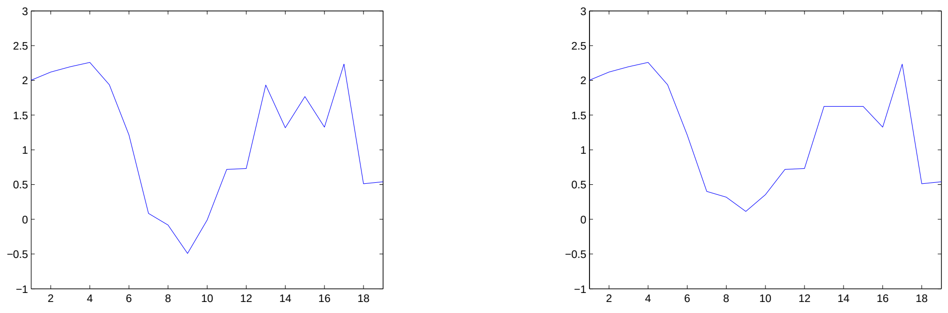


Figure 3.2: CLB1 gene expression data before and after the max-min filter where the window length $|w(t)| = 3$

3.2.2 Scale-space smoothing

Once we have removed insignificant local extrema, we smooth our data, capturing only the overall pattern of the data. We again use a technique from the field of signal processing, called convolution. A convolution is an integral that expresses the amount of overlap of one function g as it is shifted over another function f. It therefore "blends" one function with another. Let f, g be two functions with domain $\mathbb{Z}$ then the convolution of f and g denoted by $f * g$, is defined as follows:

$$(f * g)(m) = \sum_{k \in \mathbb{Z}} f(k)g(m-k) \tag{3.4}$$

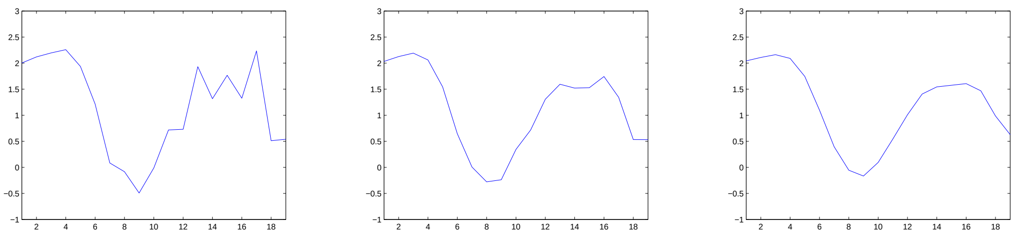
or for f, g with domain $\mathbb{R}$:

$$(f * g)(m) = \int_{-\infty}^{\infty} f(k)g(m-k)\,dk \tag{3.5}$$

By choosing g the Gaussian density function,

$$g(t; \mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(t-\mu)^2}{2\sigma^2}\right). \tag{3.6}$$

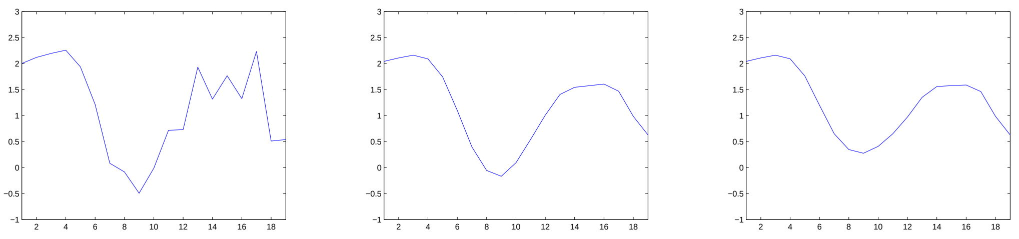
we create a linear scale-space representation of our original signal, effectively capturing the properties of the signal at a higher scale [FtHRKV92]. Scale space theory is a framework for multi-scale signal representation. It is a formal theory for handling image structures at different scales in such a way that fine-scale features can be successively suppressed and a scale parameter σ can be associated with each level in the scale-space representation. By varying σ, which corresponds to the variance of the Gaussian, we can vary the amount of smoothing (or the scale level) that occurs [EPdJS04, SM03], as illustrated in Figure 3.3. Because the Gaussian function is inherently a continuous function, we sample discrete values from it.



(a) Original CLB1 data (b) Smoothed CLB1 data, $\sigma = 0.5$ (c) Smoothed CLB1 data, $\sigma = 1$

Figure 3.3: CLB1 gene expression data with no smoothing, and after smoothing with $\mu = 0, \sigma = 0.5$, and $\mu = 0, \sigma = 1$

In combination with the max-min filter we are now able to effectively filter out noise and capture the overall pattern in the data as illustrated in Figure 3.4.[§] Note that by using this particular smoothing approach we are incorporating the temporal information of the expression data by averaging expression levels that are adjacent in time.



(a) Original CLB1 data (b) Smoothed CLB1 data (c) Max-min filtered and then smoothed CLB1 data

Figure 3.4: CLB1 gene expression data before and after the max-min filter $|w(t)| = 3$ and smoothing with $\mu = 0$ and $\sigma = 1$

3.2.3 Finding local extrema

Convolutions present us with another advantage because of the differentiation rule [FtHRKV92]:

$$D(f * g)(m) = Df * g(m) = f * Dg(m) \tag{3.7}$$

where Df denotes the derivative of f, or in the discrete case, the difference operator $\Delta f(m) = f(m+1) - f(m)$.

[§]With this data there is obviously not much need for an additional max-min filter, smoothing alone suffices. With other datasets (e.g., Homo sapiens) there is more noise and the use of a max-min filter will be useful.

We can verify the differentiation rule, but first we show communitativity. Let

$$h(m) = f * g(m) = \int_{-\infty}^{\infty} f(k)g(m-k)dk \tag{3.8}$$

Where f and g are now two functions with domain $\mathbb{R}$. By letting $k' = m - k$ we find:

$$-\int_{\infty}^{-\infty} f(m-k')g(k')dk' = \int_{-\infty}^{\infty} f(m-k')g(k')dk' \tag{3.9}$$

and because the integral ranges over infinity and we know multiplication to be commutative: $f * g(m) = g * f(m)$.

The derivative of h is:

$$\begin{aligned} \frac{dh(m)}{dm} &= \frac{d}{dm}\int_{-\infty}^{\infty} f(k)g(m-k)dk && (3.10)\\ &= \int_{-\infty}^{\infty} f(k)\frac{d}{dm}g(m-k)dk && (3.11)\\ &= f * D(g)(m) && (3.12) \end{aligned}$$

In 3.11 we used Leibniz integral rule [Weia]. Using commutaivity we can repeat the proof with f and g interchanged, thus prooving $D(f * g)(m) = Df * g(m) = f * Dg(m)$.

We use the differentiation rule by choosing g the first derivative of the Gaussian. This way we can simultaneously compute the derivative of the smoothed data f.

$$g(t; \mu, \sigma) = \frac{-\sqrt{2}}{2\sqrt{\pi}\sigma^3} \cdot (t-\mu) \cdot \exp\left(-\frac{(t-\mu)^2}{2\sigma^2}\right) \tag{3.13}$$

We discretize on the basis of local extrema for which we use the derivative data. Recall that, whenever we find a negative-positive transition the smoothed data will show a local minimum, while a positive-negative transition indicates a local maximum [EPdJS04, SM03] as illustrated in Figure 3.5.

This is a method preferable to determining more or less arbitrary thresholds for maxima and minima. A gene may have a high expression level at some point, but this does not necessarily mean it is also a local maximum. As mentioned in [MLCYW03]:

> “In microarray experiments, the intensity of gene expression is not relevant, instead, the relative change of intensity characterized by the shape of the expression profile is regarded as characteristic and informative.”

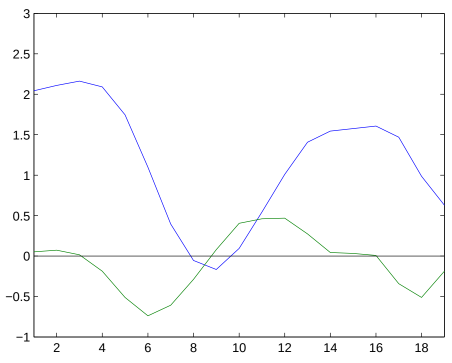


Figure 3.5: CLB1 smoothed data (blue) with first derivative data (red) showing how a zero passage corresponds to a local extrema.

We use the derivative data in choosing discrete values for the continuous data as illustrated in Figure 3.6. We encode

- a local maximum as 1
- a local minimum as −1
- all other values as 0

We encode the data in such a way for practical reasons. That is for optimization of various pattern matching procedures. This data is referred to as the *extrema data* or discrete data of a gene. Note that certain patterns of extrema data are impossible: two consecutive maxima or two consecutive minima are impossible and due to smoothing, a maximum next to a minimum is unlikely. As a result the extrema data will consist in majority of zero values.

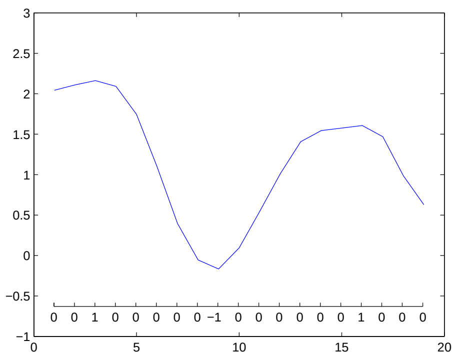


Figure 3.6: The expression of gene CLB1 with the corresponding discrete values

Detection limit

Our approach can not detect local extrema at the first ($t = 0$) and last sample ($t = m$), because the derivative data will never show a zero passage at these samples. We call this the *detection limit* of our method. As we already have few samples, we want to use the full range of samples available for creating a discrete pattern. We need an alternative way to determine local extrema at these samples. We use two different approaches.

In the first approach we determine a maximum- and minimum threshold for each gene g by:

$$th_{max} = ((\max(g) - \mu_g) \cdot c) + \mu_g \qquad (3.14)$$
$$th_{min} = ((\min(g) - \mu_g) \cdot c) + \mu_g \qquad (3.15)$$

where c is a constant for which $0 \leq c \leq 1$, $max(g)$ denotes the maximum sample value of probe g, and μ_g denotes the mean sample value of probe g. We then determine the extrema data E for gene i at $t = 0$ and $t = m$:

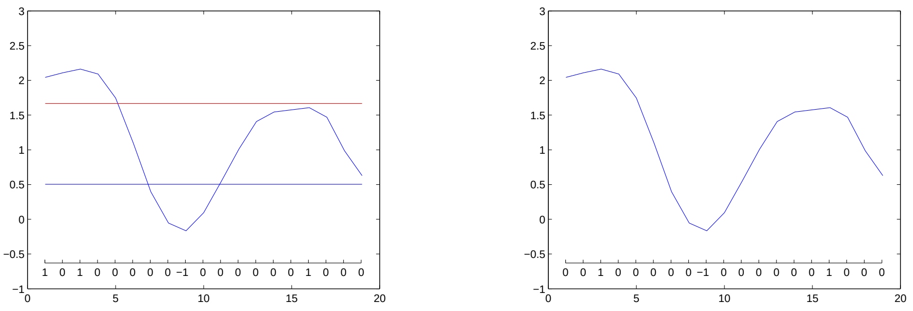
$$E_{g,0} = \begin{cases} -1 & g_0 < th_{min} \\ 1 & g_0 > th_{max} \\ 0 & else \end{cases} \qquad E_{g,m} = \begin{cases} -1 & g_m < th_{min} \\ 1 & g_m > th_{max} \\ 0 & else \end{cases}$$

This choice for th_{max} and th_{min} enables us to easily vary the threshold between the values of $[\max(g), \mu_g]$, and $[\min(g), \mu_g]$ respectively, by manipulating the value of c.

For the gene CLB1 with $\max(g) = 2.16$, $\min(g) = -0.17$, $\mu_g = 1.17$, $c = 0.5$ we calculate $th_{max} = ((2.16 - 1.17) \cdot 0.5) + 1.17 = 1.67$ and $th_{min} = ((-0.17 - 1.17) \cdot 0.5) + 1.17 = 0.50$ which is illustrated in Figure 3.7(a).

In the second approach we do not use the first and last samples in the comparisons. In practice this means we always determine the discrete value of the first and last sample to be zero. Or in terms of the first approach, we choose $c = 1$. Figure 3.7(b) illustrates this for gene CLB1.

With a third approach we mirrored the signal, and prepended and appended it to the original signal. This approach showed poor results. Using this approach we (almost) always find extrema at the beginning and end of the data, which is dubious.



(a) CLB1 extrema data with use of thresholds

(b) CLB1 extrema data with a zero value at $t = 0$ and $t = 19$

Figure 3.7: Extrema data using different approaches to the problem of the detection limit.

Plateaus

A problem with our method occurs with plateaus. A plateau constitutes two or more sequential samples for some probe g showing almost the same expression values:

$$\exists_i : 0 \leq i \leq n - 1\ [g_i \approx g_{i+1}] \tag{3.16}$$

With a plateau we cannot straightforwardly find a local extremum, because we cannot discriminate between either of the points. Determining whether two samples constitute a plateau relies on arbitrary criteria. A threshold needs to be fixed determining whether a certain change in expression from one sample to the next is small enough to be considered a plateau. Furthermore, we did not encounter any real problems with plateaus (partly because plateaus in the original data are smoothed away) and for these reasons we ignore them[¶] .

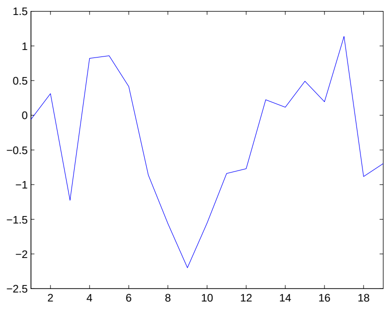


Figure 3.8: The expression of gene CLB2, showing a possible plateau at $t \in \{4 - 5\}$.

[¶]Furthermore we can circumvent the problem by using lagfunctions, see Section 3.3.1

3.3 Similarity function

We compare the pattern of the extrema data for each probe to the patterns of the extrema data of all other probes with a similarity function. In order to find relations between genes we thus use a discretization based similarity function [AO01, MLCYW03] as seen in Section 2.4. Patterns can be similar modulo a lag $l \in \mathbb{N}_0$, where l is constant.

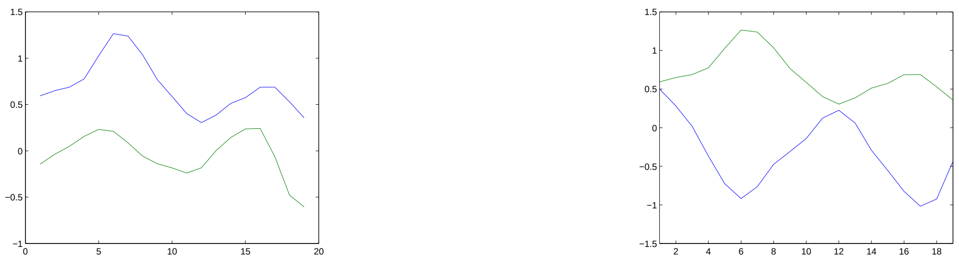
When comparing the discretized samples of two probes, g and h consisting of n samples with lag l we distinguish four cases.

1. $\forall_t : 0 \leq \; t \leq n - l \, [g_t = 1 \Leftrightarrow h_{t+l} = 1]$
2. $\forall_t : 0 \leq \; t \leq n - l \, [g_t = 1 \Leftrightarrow h_{t+l} = -1]$
3. $\forall_t : 0 \leq \; t \leq n - l \, [g_t = -1 \Leftrightarrow h_{t+l} = 1]$
4. $\forall_t : 0 \leq \; t \leq n - l \, [g_t = -1 \Leftrightarrow h_{t+l} = -1]$

We find a lot of these relations and combine the cases two by two resulting in only two similarity functions:

1. **Stimulation:** $\forall_t : 0 \leq t \leq n - l \, [g_t = h_{t+l}]$ (1+4)
(i.e. the maxima and minima of probe g correspond to the maxima and minima respectively of probe h modulo lag l; illustrated in Figure 3.9(a).)
2. **Inhibition:** $\forall_t : 0 \leq t \leq n - l \, [g_t = -(h_{t+l})]$ (2+3)
(i.e. the maxima and minima of probe g correspond to the minima and maxima respectively of probe h modulo lag l; illustrated in Figure 3.9(b).)

These similarity measures correspond to the two gene interaction groups from Section 2.2. We abbreviate this to $S_l(g, h)$ for a stimulation relation and $I_l(g, h)$ for an inhibition relation with lag l between a gene g and h. Furthermore, let $R_l(g, h) = S_l(g, h) \vee I_l(g, h)$.



(a) Stimulation with lag 1. The extrema correspond modulo lag 1.

(b) Inhibition with lag 0. The extrema are exactly opposite (modulo lag 0)

Figure 3.9: Stimulation and inhibition relations

Note that the greater the lag, the less samples included for comparison. If these extrema patterns are totally random, a greater lag will result in finding more relations. Furthermore the relations have some properties of interest:

$$S_0(g, h) \Leftrightarrow S_0(h, g) \text{ and } I_0(g, h) \Leftrightarrow I_0(h, g) \tag{3.17}$$

$$S_l(g, h) \wedge S_p(h, k) \Rightarrow S_{l+p}(g, k) \text{ and } I_l(g, h) \wedge I_p(h, k) \Rightarrow I_{l+p}(g, k) \tag{3.18}$$

provided $l + p < n$.

3.3.1 Different similarity functions

Besides the evident comparison of patterns that follows from our definition of stimulation and inhibition, we also tried two other methods.

The first method is an extension of stimulation and inhibition by use of a *lag-function*. Because the data is very noisy (or because of the smoothing) we anticipate extrema are slightly (one timepoint) shifted into the future. So instead of looking for a constant lag, we create a lag-function:

1. Stimulation: $\forall_t : 0 \leq t \leq n\ [g_t = h_{t+l} \vee g_t = h_{t+l+1}]$
2. Inhibition: $\forall_t : 0 \leq t \leq n\ [g_t = -(h_{t+l}) \vee g_t = -(h_{t+l+1})]$

Which simply means that for an extrema at time t of gene g we consider (when looking for lag 1 relations) not only an extrema at $t + 1$, but also at $t + 2$, for a gene h. Many variations are of course possible. One variation we use is the case in which we anticipate exactly one extrema to be shifted one time point into the future. Figure 3.10 is an example of two related genes showing similar patterns, where the maximum of CLN3 at $t = 17$ is shifted one sample into the future.

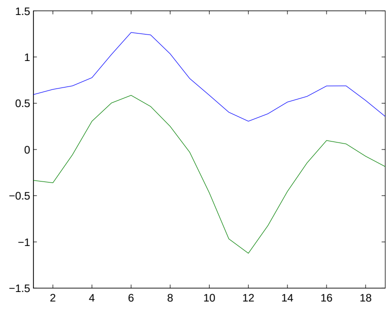


Figure 3.10: CLN3 (blue) and CDC54 (green) gene expression. Notice the maximum at $t = 16$ for CDC54 and at $t = 17$ for CLN3.

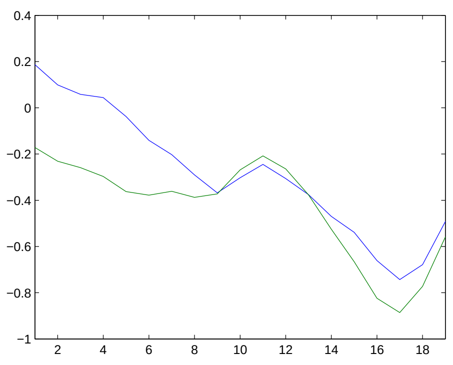


Figure 3.11: SEC7 (blue) and SEC21 (green) found by using derivative data only, with $c = 0.1$

With a second method we look at the derivative data. The pattern similarity we are looking for has two variables:

- The point in time. For example, a pattern may be shifted in time in comparison to another.
- The expression value. For example, a pattern may show higher or lower expression absolute values at extrema in comparison to another.

Thus if the derivative values for two genes are similar modulo a constant lag these genes intuitively have a higher chance of having a correlation. Because of the noise in the data the derivative data of two genes at a timepoint (modulo some lag) may differ by a constant value c. An example of two genes that are similar in this way are shown in Figure 3.11.

3.4 Filtering

As the steps sofar will lead to a lot of relations we use additional filtering to remove unlikely relations. For this we use three different approaches;

1. Spatial information from the Gene ontology [Con00]
2. Process information from the KEGG yeast cellcycle pathway [KGH+06]
3. Combining the results of different independent datasets

3.4.1 Ontology information

We can see that the relationships we are looking for are causal relations. With co-regulation two genes are both controlled by a third gene. And thus a co-regulation relationship consists of two control regulation relationships.

According to [Els83] a causal relationship is generally thought to obey three principles:

1. Determinism: any event has a cause.

2. Locality: a cause always acts on what is contiguous to it, in space and time.

3. Temporal asymmetry: the cause must precede its effect; or at least not succeed it.

It is clear that our approach already assumes principle 1, as we assume that the expression of a gene has a cause and is not governed by some random event (except for the fact that we have to account for noise). We also assume principle 3, because we only look for relations with lags greater than (or equal to) 0. Based upon the remaining principle 2 we decide to use spatial information from the gene ontology [Con00] as a filter.

The gene ontology (GO) provides gene information divided into three categories:

- Molecular function: describes activities, such as catalytic or binding activities, at the molecular level.

- Biological process: series of events accomplished by one or more ordered assemblies of molecular functions.

- Cellular component: a component of a cell but with the proviso that it is part of some larger object, which may be an anatomical structure or a gene product group.

All proteins are encoded by a specific gene. Two genes encoding for 2 proteins that are located at different parts of a cell, are (based upon principle 2) not likely to have a relation. We use the information from the cellular component category to filter out such relations. Unfortunately not all genes have a corresponding GO identifier. We choose to keep all relations where one of the genes has no corresponding GO identifier.

We downloaded the GO file for Saccharomyces cerevisiae. The (obo) file format describes multiple unrooted, directed graphs where an edge corresponds to a GO identifier and a vertex corresponds to a "is-part-of" or "is-a" relationship. A GO identifier is a unique identifier that corresponds to a molecular function, biological process, or cellular component. A gene is associated with multiple GO identifiers, based on current knowledge.

We also downloaded a file mapping gene identifiers to GO identifiers. Each gene g can have multiple GO identifiers. If one of the GO identifiers of one of the genes is within a path (from node to root) of the other, then they have a spatial relation. For instance, genes may not only be within the same cellular component of a cell, but it is also possible that one gene is within a component that contains the other gene. Or more simple, if the GO identifier of one gene is an ancestor of the GO identifier of another gene in any tree, we deem the genes spatially related. If two GO identifiers of two genes are within the same path from a leaf to the root of this tree, these genes have a spatial relation. Figure 3.12 is an example of a GO cellular component tree where for example GO:0005634 and GO:0042226 are spatially related.

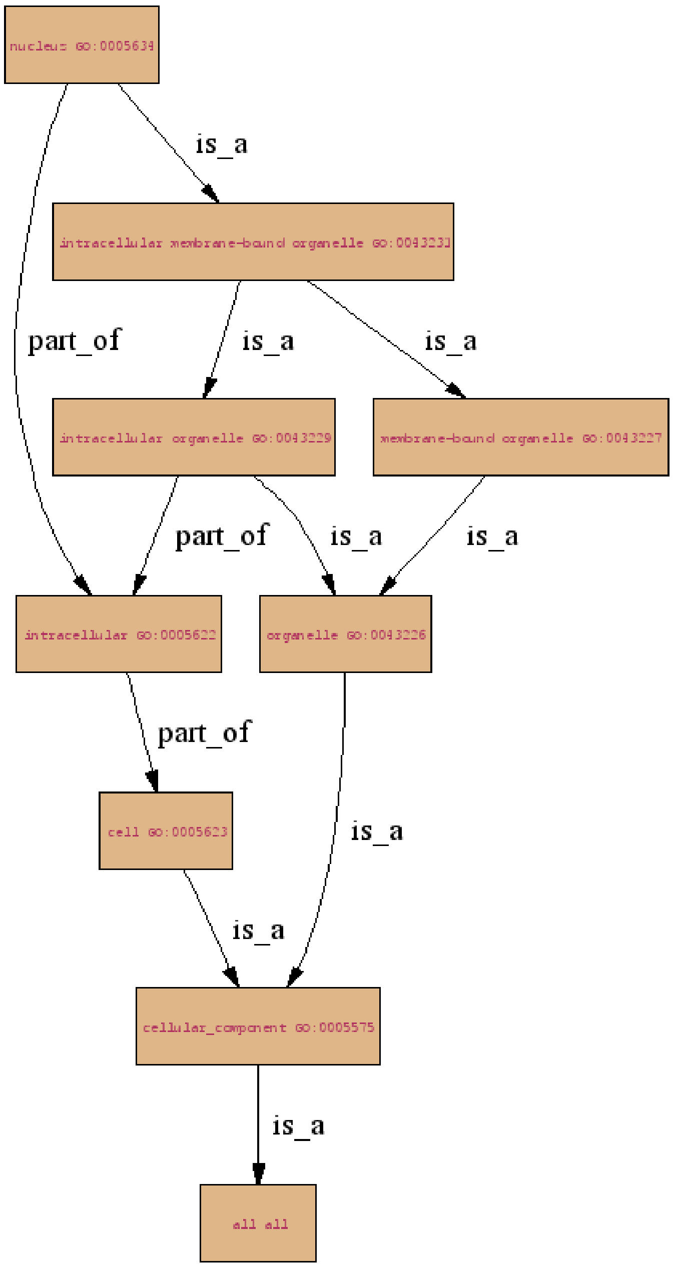

nucleus GO:0005634
is_a
intracellular membrane-bound organelle GO:0043231
part_of
is_a
is_a
intracellular organelle GO:0043229
membrane-bound organelle GO:0043227
part_of
is_a
is_a
intracellular GO:0005622
organelle GO:0043226
part_of
cell GO:0005623
is_a
is_a
cellular_component GO:0005575
is_a
all all

Figure 3.12: An example of a GO cellular component tree [Con00].

3.4.2 KEGG pathway information

The Kyoto Encyclopedia of Genes and Genomes (KEGG) [KGH+06] pathway database is a collection of graphical diagrams representing molecular interaction networks in various cellular processes, as illustrated in Figure 3.13.

As a second filtering method we extracted the genes from the KEGG yeast cell-cycle pathway and used these to filter relations that were found. Only relations where one of the genes occurs in the KEGG cellcycle pathway remain, all other relations are discarded.

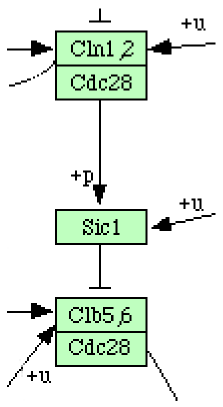


Figure 3.13: A part of the KEGG yeast cellcycle pathway.

3.4.3 Combining sets

With this filtering method we combine the relations that were found in two different datasets, using a logical AND. Only relationships that are in both sets remain in the combined set. Of course the combined sets need not be constructed in the same way.
Reasons for combining sets are:

1. The probabilities of chance findings are reduced significantly.
2. The resulting combination set is very small in comparison to the original sets. This allows us to be less strict in creating the original sets (because validation of large sets takes too much time).

3.5 Validation

Unfortunately there is little known about gene interaction and gene regulatory pathways. Validation of a potential gene interaction is usually done by an extensive literature research and/or many additional laboratory experiments. Extensive literature research does not only take a lot of time, but can only be performed by experts with a sound biological background, while a laboratory experiment requires additional expensive equipment [Hus03b].

An alternative way for validating gene interactions is done by the use of a *gold standard.* A gold standard for protein interactions is a database containing both confirmed protein-protein interactions (true positives) as well as confirmed non interactions (true negatives). At this time, some gold standards exist, but these are relatively small. For a yeast dataset containing 6,000 probes there are $\binom{6000}{2} = 17,997,000$ possible interactions. The gold standard of [JYG+03] consists of 8,617 true positive protein-protein interactions,[‖] and 2,705,844 true negative protein-protein interactions, which leaves 16,366,292 interactions unaccounted for. Notice that the amount of true positives versus true negatives illustrates how we are searching for the proverbial needle in

[‖]This is a subset of the MIPS protein-protein interaction database the haystack.

For our validation we will construct a gold standard using information from the following three different sources:

- The MIPS Comprehensive Yeast Genome Database (CYGD) [GMK+05]. The MIPS website contains a list of 15,454 confirmed protein-protein interactions relations.
- DIP true positive information [SMS+04]. The DIP website contains a protein-protein interaction database with 14,130 confirmed protein-protein interactions.
- A true negative database from [JYG+03], containing 2,705,844 entries. These negatives were synthesized from lists of proteins in separate subcellular compartments.

We combine the two true positive databases from [GMK+05] and [SMS+04] into one database containing 24,173 interactions. Together with the true negatives from [JYG+03] we now have a complete gold standard to validate our results. We do not use the true positive database from [JYG+03] because we deem it too small for our purposes.

As the time required for gene expression is unknown we anticipate finding indirect gene relations. For three genes g, h, k where it is known that $R(g, h)$ and $R(h, k)$ we will possibly only find a relation between g and k. Although these indirect relations are somewhat dubious, we decide to include them as they help to give a more accurate validation.

Our validation procedure is as follows: We define a subset K of all relations that are in the combined MIPS/DIP PPI database, and subset a M of all relations in the true negative database. A relation is commutative. We deem a relation a true positive if it is in the set T:

$$T = \{(g, h) \,|\, (g, h) \in K \text{ or } \exists f : [(g, f) \in K \wedge (f, h) \in K]\} \tag{3.19}$$

A relationship is a negative if it is in the set N:

$$N = \{(g, h) \,|\, (g, h) \in M \text{ and } (g, h) \notin T\} \tag{3.20}$$

Relations that are in neither of these sets are considered unknown.

3.6 Bayesian networks

We use the discretized expression data for experiments with learning Bayesian networks. A Bayesian network is a promising tool for analyzing expression data [FLNP00], especially when looking for gene regulatory pathways in time-series data [KIM04, LTW05]. Some known advantages of Bayesian networks are:

- The mathematical foundations for Bayesian networks are well understood
- Bayesian networks provide models of causal influence [HDR05, PR00]
- Bayesian networks allow for a useful graphical representation
- Bayesian networks are whitebox models (contrary to neural networks and SVMs) [Luc00]
- Bayesian networks are probabilistic models and thus capable of handling noise [Hus03b]

First we will give a short overview of Bayesian networks.

3.6.1 Background on Bayesian networks

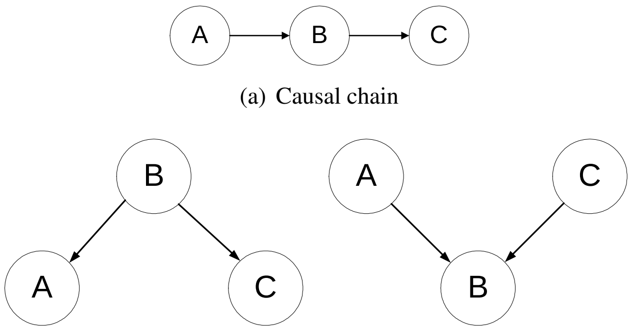
Bayesian networks (BNs) are graphical models for reasoning under uncertainty [HDR05, KN03, Nea03, PR98]. The nodes in the model represent variables, while arcs between these nodes represent direct dependencies between variables. The strength of the dependencies is quantified by a conditional probability distribution associated with each node. If the variables are discrete, the conditional probability distribution of each node can be specified as a conditional probability table (CPT).

Let V be a set of indices $\{A, \ldots, Z\}$, X_V denote a set of stochastic variables with indices V and let $X_i = x_i$ denote an instantiation of variable x_i, which we abbreviate to x_i. The joint probability distribution of these variables is represented by $P\,(X_A = x_A, X_B = x_B, \ldots, X_Z = x_Z)$. The specification of this joint probability distribution is exponential in the number of variables. However we can simplify the specification of the distribution by using the independence structure of the variables. Let $A, B, C \subseteq V$ and disjoint. We denote X_A is conditionally independent of X_C given X_B as

$$P(X_A \mid X_B, X_C) = P(X_A \mid X_C) \tag{3.21}$$

This is also denoted as

$$X_A \perp\!\!\!\perp_P X_B \mid X_C \tag{3.22}$$



(a) Causal chain

(b) Common cause (c) Common effect

Figure 3.14: Three basic Bayesian networks from a conditional independence view.

Conditional independence can be illustrated graphically by three basic types of networks shown in Figure 3.14 [HDR05, PR00, Kra98, Nea03, SAR04].

In Figure 3.14(a) we can see that given evidence on the value of *B*, additional evidence on the value of *A* will no longer influence the value of *C*, which has become entirely determined by *B*. We say *C* is d-separated from *A* given *B*.

$$A \perp\!\!\!\perp_G C \mid B \tag{3.23}$$

In Figure 3.14(b) we see a similar effect. For example, say node *B* represents having a cold and is a cause for both *A* (= sneezing) and *C* (= coughing). Given evidence that a person is sneezing, the chances of the person having a cold, and thus the chances of coughing will rise. But if we already know that someone has a cold, knowing that he is sneezing will not raise our belief that he might be coughing. We say that *A* is d-separated of *C* given *B*, and vice versa.

$$A \perp\!\!\!\perp_G C \mid B \tag{3.24}$$

In Figure 3.14(c) we see that given positive evidence on *A* will not change the chances on *C*. But given evidence on *B*, additional evidence on *A* will block information exchange with *C*. This is the opposite of conditional independence. We say *A* and *C* are d-separated and d-connected given *B* [FL04, Pea88].

$$A \perp\!\!\!\perp_G C \mid \varnothing \tag{3.25}$$

and

$$A \not\perp\!\!\!\perp_G C \mid B \tag{3.26}$$

We use the conditional independencies of the variables to factorize the joint probability distribution. If pa_i denotes the value of the parents of x_i, then $P(x_i \mid pa_i)$ denotes the conditional distribution of x_i, given its parents. By the *Markov* assumption (or local Markov property), the value of X_i depends only on the value of its parents. The *Markov Blanket* (or global Markov property) is the set of children, parents and other parents of the children for a given node. The Markov blanket shields a node from the remaining nodes in a network

[FL04, FLNP00, HDR05, Kra98, Nea03, SAR04].

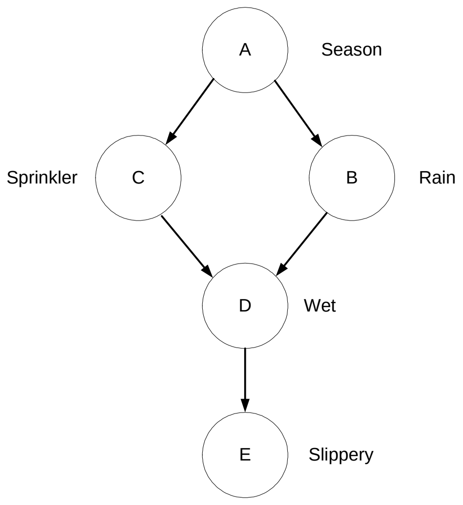


Figure 3.15: A Bayesian network representing causal influences among five variables, adopted from [PR00].

We factorize the joint probability distribution over the values with the *chain rule*:

$$P(x_1, x_2, \dots, x_n) = P(x_1) \cdot P(x_2 \mid x_1) \cdot \dots \cdot P(x_n \mid x_1, \dots, x_{n-1}) = \prod_{i=1}^{n} P(x_i \mid x_1, \dots, x_{i-1}) \quad (3.27)$$

Now we use the Markov assumption and we rewrite the joint probability distribution as:

$$P(x_1, x_2, \dots, x_n) = \prod_{i=1}^{n} P(x_i \mid pa_i) \quad (3.28)$$

Thus, the joint probability of 3.15 becomes:

$$P(A, B, C, D, E) = P(A)\, P(B \mid A)\, P(C \mid A)\, P(D \mid B, C)\, P(E \mid D) \quad (3.29)$$

We say that an acyclic directed graph G is an I-map of a probability distribution if

$$U \perp\!\!\!\perp_G W \mid Z \Rightarrow U \perp\!\!\!\perp_P W \mid Z \quad (3.30)$$

A Bayesian network $B = (G, \Theta)$ is a pair, where the graph G is an I-map of the joint probability Θ.

3.6.2 Learning Bayesian networks

Considering the advantages, Bayesian networks are not as frequently applied to modeling real world problems as one might think. The reason for this is the knowledge bottleneck [KN03].

The domain knowledge (i.e., the qualitative and quantitative part) must somehow be available. However for many problems there simply is no domain knowledge available and with other problems this knowledge comes from experts in the domain. Extracting this knowledge by means of interviews and refining the knowledge by testing is a time consuming business. This method of Bayesian network construction is called *manual construction* [LvdGAH04].

A promising alternative to knowledge acquisition is machine learning from domain data. With the help of datasets, an algorithm is able to construct (learn) a Bayesian network that fits the data. Learning basically consists of two different components: 1) learning the graphical structure G (model selection), 2) learning the conditional distributions Θ (CPTs; parameter estimation). Bayesian learning algorithms already exist; examples are the K2 algorithm (a search and scoring algorithm, which is one of the oldest algorithms from the beginning of the 1990's) and the more recent SEM algorithms [Fri98, LvdGAH04]. These algorithms all exploit the conditional independencies for learning efficiently.

Structure learning

As structure learning essentially entails looking for causal relations, we are mainly interested in learning the structure of Bayesian networks and not so much in learning the underlying parameters. For this reason we will focus on structure learning only [FMR98, HDR05, Kra98, Nea03, SAR04].

With a linear growth in variables (or nodes), the number of possible networks grows super exponentially in the number of variables n according to the following formula [Rob77]:

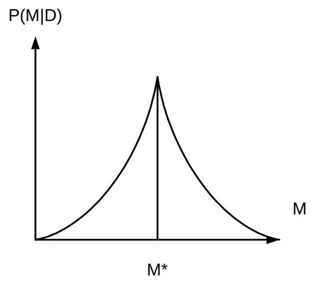
$$r(n) = \sum_{i=1}^{n} (-1)^{i+1} \binom{n}{i} 2^{i(n-i)} r(n-i) = n^{2^{O(n)}} \tag{3.31}$$

Thus, with an increasing number of variables, the algorithm must consider more possible networks when searching for the best one, i.e., best fitting the data. For this reason we cannot simply feed expression data directly into a learning algorithm. The problem is known to be NP-hard [Chi95, FLNP00] and the time needed for considering all possibilities when learning large (more than 100 nodes) networks would leave us waiting for centuries as illustrated by Table 3.1. As evaluting all networks is impossible one has to resort to heuristic optimization methods.

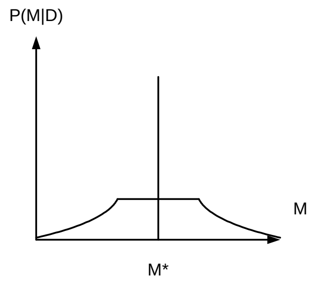
# nodes	2	4	6	8	10
# structures	3	543	3.7×10^6	7.8×10^{11}	4.2×10^{18}

Table 3.1: The number of BN topologies as a function of the number of nodes in a graph

In addition to the fact that there are so many possible network structures, gene expression data is mostly sparse which results in a lot of uncertainty. In practice this means that a lot of networks will have the same maximum posterior as illustrated in Figure 3.16 [HDR05]. For this reason



(a) Large dataset, the optimal network structure is well defined



(b) Small dataset, uncertainty about the optimal network structure

Figure 3.16: The posterior probability against the model with large datasets and small datasets from [HDR05].

network models are sampled from the posterior distribution leading to a collection of networks with high posterior probability [Hus03b]. The posterior probability $P(M \mid D)$ gives a measure of certainty that the model M denoted as $\langle G, \Theta \rangle$ consisting of the graphical structure G and the parameters Θ is correct given the data D.

We can calculate $P(M \mid D)$ with the help of Bayes' rule:

$$P(M \mid D) = \frac{P(D \mid M) P(M)}{P(D)} \tag{3.32}$$

where $P(M)$ is the *prior* probability and $P(D)$ is a normalizing constant when working with a fixed dataset:

$$P(M \mid D) \propto P(D \mid M) P(M) \tag{3.33}$$

$P(D \mid M)$ is called the *likelihood* which is the probability that a particular model explains the observed data. This is a measure of how plausible the model is in light of the observed data. The *Maximum likelihood* procedure attempts to find a model that maximizes the likelihood given the observed data [FLNP00, Hec96, HDR05, SAR04].

Because we are only interested in the structure independent of the parameters Θ, we need to average $P(M \mid D)$ over Θ to obtain $P(D \mid G)$. We do this by integrating over Θ [FLNP00, Hec96, HDR05, SAR04]:

$$P(D \mid G) = \int P(D \mid M) P(\Theta \mid G) \, d\Theta \tag{3.34}$$

$P(D \mid G)$ is called the marginal likelihood. The marginal likelihood is used for calculating the Bayes factor K of two graphs G_1 and G_2 given the data D [Hec96, SAR04]:

$$K = \frac{P(D \mid G_1)}{P(D \mid G_2)} \tag{3.35}$$

which is a way of measuring which graph fits best. The Bayes factor is a common way of scoring networks.

Priors

A problem in structure learning is that a more complex model will be likely to fit the data best, simply because with a more complex model there are more parameters. This is called *over-fitting*. A way of avoiding over-fitting is by manipulating the prior in such a way that complex models are penalized, which is called *regularization*. However, by penalizing complex networks, bias increases and variance decreases. This is called the *bias-variance trade off* [Hec96, HDR05, SAR04, SJ02].

The prior encompasses our belief of the model before we have seen any data. When choosing a prior the choice is between an informative prior or a non-informative prior. An informative (or subjective) prior explicitly encapsulates background knowledge of the structure of the Bayesian network. For example, if we know that two variables are completely independent no matter what the data might suggest we can specify this explicitly in the prior.

A non-informative (or objective) prior is used when no bias for a any model is wanted. By choosing the prior from a certain family of prior distributions, the posterior distribution will belong to the same family of distributions. Such priors are called *conjugate* priors and are used for reasons of computational efficiency. By using a conjugate prior the integral in 3.34 becomes analytically tractable. For multinomial distributions the *Dirichlet* distribution is the usual conjugate prior [Hec96, SAR04, SJ02].

Equivalence

Different networks can have the same joint probability. Such networks are called *equivalent*. In general, networks are equivalent if they have the same structure (ignoring the direction of the arrows), and the same *v-structures*. A v-structure is a configuration of two directed edges converging on the same node, without an edge between the parents as shown in Figure 3.17(d). Equivalent graphs are grouped into an equivalence class which can be represented by a unique chain graph called an essential graph. Because all graphs in an equivalence class have the same joint probability (which implies the same likelihood), learning Bayesian networks from data can only be done up to the part of learning equivalent graphs. Or in other words: it is not possible to distinguish between equivalent graphs on the basis of the data [FLNP00, Hec96, HDR05].

If we look at the factorizations of the networks in Figure 3.17 (from left to right):

1. $P(B \mid C)\, P(C \mid A)\, P(A) = P(B \mid C)\, P(A \mid C)\, P(C)$
2. $P(A \mid C)\, P(B \mid C)\, P(C)$

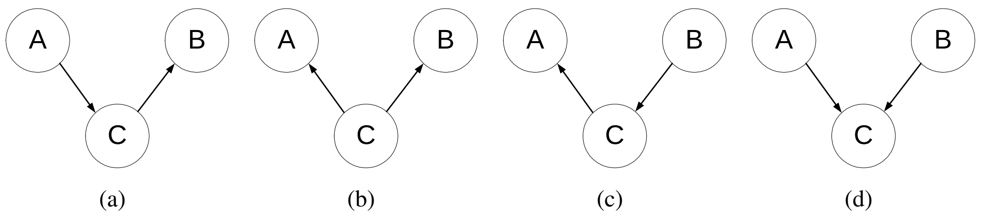


Figure 3.17: 4 Basic Bayesian networks. The three left networks are equivalent; the right most network is a v-structure.

3. $P(A \mid C)\, P(C \mid B)\, P(B) = P(A \mid C)\, P(B \mid C)\, P(C)$
4. $P(C \mid A, B)\, P(A)\, P(B))$

we can see that the first three networks have equivalent joint probabilities.

3.6.3 Dynamic Bayesian networks

A major disadvantage of using Bayesian networks for modeling gene interaction is the acyclicity constraint which prohibits feedback loops. Feedback loops are very common in biological processes [Gen05, Hus03b]. Dynamic Bayesian networks (DBN) [BvSMR03, FMR98, GK02, MM99] circumvent this problem.

A DBN can be used for modeling temporal processes. Assuming the variables $\mathbf{X} = \{X_1 \ldots X_N\}$ where $X_i(t)$ denotes the value of variable X_i at time t. We assume that the process being modeled is Markovian: $P(\mathbf{X}(t+1) \mid \mathbf{X}(0), \ldots, \mathbf{X}(t)) = P(\mathbf{X}(t+1) \mid \mathbf{X}(t))$ and that the process is stationary, meaning: $P(\mathbf{X}(t+1) \mid \mathbf{X}(t))$ is independent of t. A DBN then consists of two parts:

- a prior network, specifying distributions over the initial states $\mathbf{X}(0)$.
- a transition network specifying the probabilities $P(\mathbf{X}(t+1) \mid \mathbf{X}(t))$.

We can 'unroll' the prior network with the help of the transition network, moving forwards in time, as shown in Figure 3.18. Each network is called a slice; arrows within one slice are called *intra-slice* arrows, while arrows between two slices are called *inter-slice* arrows.

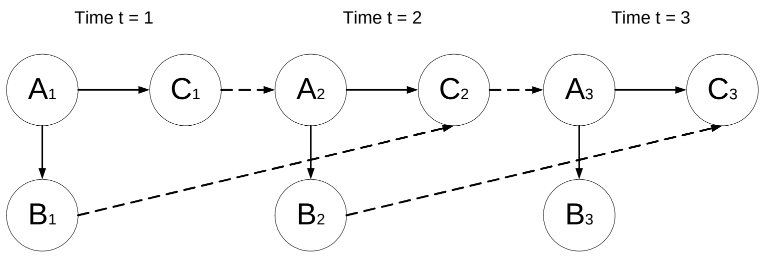
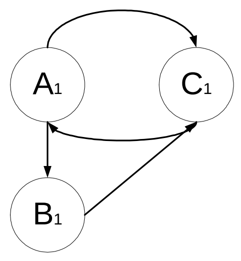
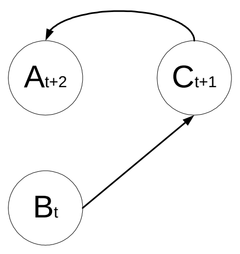


Figure 3.18: An example of a three node dynamic Bayesian network. Solid arrows indicate intra-slice dependencies; dashed arrows indicate inter-slice dependencies.



(a) Compact dynamic Bayesian network



(b) Even more compact dynamic Bayesian network, showing only inter-slice (temporal) arcs

Figure 3.19: Compact dynamic Bayesian networks

The network in Figure 3.18 can be represented in a more compact way by showing only the inter-slice (temporal) arrows.

Because we want to model causal relationships, a dynamic Bayesian network seems more appropriate due to its ability to explicitly model time.

3.6.4 Markov Chain Monte Carlo

Markov Chain Monte Carlo (MCMC) is a method used for sampling from a probability distribution P. Let us assume a state space S and

$$P(s) = \frac{h(s)}{\sum_{s \in S} h(s)}$$

with $s \in S$, and h completely specified. The normalizing constant $\sum_{s \in S} h(s)$ is not known, and the state space S is too large to calculate it directly [Bes00, Nea03].

For learning Bayesian networks P would be the posterior distribution $P(M \mid D)$, D the (gene expression) data and M would be a graphical model [HDR05]:

$$P(M \mid D) = \frac{P(D \mid M)P(M)}{P(D)} = \frac{P(D \mid M)P(M)}{\sum_{M'} P(D \mid M')P(M')} \tag{3.36}$$

Now the probability distribution P is so complicated that it is hard to draw independent and identically distributed (i.i.d.) samples from it. We can however construct a Markov chain that converges to P. A path $X_1, X_2, \dots$ of the chain is then simulated and we use X_i as an i.i.d. sample from P [Bes00, CG95, Nea03].

Markov Chains

With a Markov chain we define a set of states $e_1, e_2, \dots$. For each pair of states e_i, e_j there exists a transition probability t_{ij}. Furthermore we have a sequence of moves (or transitions)

$E^{(1)}, E^{(2)}, \ldots$ such that the outcome of each move is one of the states and

$$T\,(E^{(h+1)} = e_j \mid E^{(h)} = e_i) = t_{ij}$$

Thus each move within a Markov chain entirely depends on the current state and not on any previous states. Furthermore the transition probabilities do not change over time. We can arrange all transition probabilities in a transition matrix for the chain [AdFDJ03, Nea03]:

$$T = \begin{pmatrix} t_{11} & t_{12} & t_{13} & \cdots \\ t_{21} & t_{22} & t_{23} & \cdots \\ t_{31} & t_{32} & t_{33} & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{pmatrix}$$

A Markov chain will converge to a stationary distribution (or equilibrium distribution) invariant from the starting state if it obeys the following properties [AdFDJ03, Nea03]:

- **Irreducibility**: if for every i and j there exists an $n \geq 0$ such that $t_{ij}^{(n)} > 0$. Or in other words, from any state in the chain there is a positive probability of reaching any other state in a finite amount of steps.
- **Aperiodicity**: there does not exist a p such that for every e_i it is so that $t_{ii}^{(n)} > 0$ and $n = m \cdot p$ for some integer m. If such a p exists, it is called the period. Or in other words, the greatest common divisor of return times to any particular state e_i is 1 (the chain does not get trapped in cycles).

A chain that is irreducible and aperiodic is called *ergodic*. In an ergodic Markov chain the limits

$$p_j = \lim_{n \to \infty} t_{ij}^{(n)}$$

exists and are independent of the initial state. So P is the distribution we converge to.

Furthermore: $r_j = \sum_i r_i \cdot t_{ij}$, or in matrix form: $r^T = r^T \cdot T$, where T is the transition matrix and r^T is the transpose of column vector r. Which simply means that after moving (with T) from the stationary distribution we remain in the stationary distribution.

Example

The following example from [Dd] supposes a small town where there are three places to eat. A Chinese, a Mexican restaurant and a pizza place. Everyone in town eats dinner in one of these places or has dinner at home.

Furthermore suppose that from all the people that eat at home in certain week, the next week 25% of them will eat at home again, 20% will go to the Chinese, 25% will go to the Mexican and 30% will go to Pizza Place. In matrix notation this will look like:

$$\begin{array}{l c} & H(ome) \\ H(ome) & \left(\begin{array}{c} 0.25 \\ 0.20 \\ 0.25 \\ 0.30 \end{array}\right) \\ C(hinese) & \\ M(exican) & \\ P(izza) & \end{array}$$

Suppose the entire transition matrix would be:

$$A = \begin{array}{c} \\ H \\ C \\ M \\ P \end{array} \begin{array}{c} \begin{array}{cccc} H & C & M & P \end{array} \\ \begin{pmatrix} 0.25 & 0.20 & 0.25 & 0.30 \\ 0.20 & 0.30 & 0.25 & 0.30 \\ 0.25 & 0.30 & 0.40 & 0.10 \\ 0.30 & 0.20 & 0.10 & 0.30 \end{pmatrix} \end{array}$$

A state vector for a Markov chain with k states is a column vector

$$x^{(n)} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_k \end{pmatrix}$$

where x_i is the chance of being in the i^{th} state after n transitions.
Now we calculate the probability of being in the i^{th} state at the n^{th} transition. Suppose at first everyone eats at home:

$$x^{(0)} = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

After the end of the week the state vector will be:

$$x^{(1)} = Ax^{(0)} = \begin{pmatrix} 0.25 \\ 0.20 \\ 0.25 \\ 0.30 \end{pmatrix}$$

At the end of the second week:

$$x^{(2)} = Ax^{(1)} = \begin{pmatrix} 0.25 & 0.20 & 0.25 & 0.30 \\ 0.20 & 0.30 & 0.25 & 0.30 \\ 0.25 & 0.30 & 0.40 & 0.10 \\ 0.30 & 0.20 & 0.10 & 0.30 \end{pmatrix} \begin{pmatrix} 0.25 \\ 0.20 \\ 0.25 \\ 0.30 \end{pmatrix} = \begin{pmatrix} 0.2550 \\ 0.2625 \\ 0.2325 \\ 0.2500 \end{pmatrix}$$

And we see that $x^{(2)} = Ax^{(1)} = A(Ax^{(0)}) = A^2x^{(0)}$, thus at the end of the n^{th} week:

$$x^{(n)} = A^n x^{(0)}$$

For large values of n the state vectors become very similar: the state vector converges to the stationary distribution:

$$x^{(30)} = \begin{pmatrix} 0.2495 \\ 0.2634 \\ 0.2339 \\ 0.2532 \end{pmatrix} x^{(50)} = \begin{pmatrix} 0.2495 \\ 0.2634 \\ 0.2339 \\ 0.2532 \end{pmatrix}$$

Metropolis-Hastings algorithm

How does one create a Markov chain that converges to a particular distribution P? Surprisingly, this is not very hard. A good chain should converge fast to its stationary distribution from an arbitrary position (rapid mixing). The way in which the Markov chain is constructed and thus its convergence is the key point for algorithms implementing MCMC [HDR05].

Suppose we want to construct a Markov chain for distribution P for the state space S with values r^T. We need to find a transition matrix T on S which satisfies:

$$r^T = r^T \cdot T \tag{3.37}$$

which means:

$$\sum_{i \in S} r(i)\, t_{i,j} = r(j) \text{ for all } j \in S \tag{3.38}$$

This simply says that the chance of making a transition from any state i to any state j after n (with $\lim_{n\to\infty}$) previous transitions, is equal to the chance of ending up in state j after n transitions.

For Bayesian model selection the state space consists of all possible models, the stationary distribution r will be the posterior distribution $P(M \mid D)$ and 3.38 will be equivalent to:

$$P_{n+1}(M_i) = \sum_k T(M_i \mid M_k) P_n(M_k) \tag{3.39}$$

In order to avoid the summation over the state space S we use a more stringent condition for T, namely *detailed balance* [AdFDJ03, Bes00, CG95, Nea03]:

$$r(i)\, t_{ij} = r(j)\, t_{ji} \tag{3.40}$$

for all $i, j \in S$. This means that the chance after n (with $\lim_{n\to\infty}$) steps, of being in a state i and making a transition to state j is the same as the chance of after n steps of being in j and making a transition to state i. In other words the chain is time-reversible; given a chain we cannot tell whether it moves forwards or backwards in time [AdFDJ03, Bes00].

To construct a transition matrix T with this property of detailed balance, we first choose a simple transition matrix Q on S with transitions $q_{ij} > 1$ and $\sum_{j \in S} q_{ij} = 1$ for each $i \in S$, called the *proposal* distribution. This proposal distribution may have no relation with P, and not even have a stationary distribution (although a better choice of Q leads to faster convergence) [Nea03]. We then modify Q in such a way that its stationary distribution will approximate P.

Suppose we generated a sample (network structure) from the chain $X_n = i$. To compute a new sample X_{n+1} is as follows:

1. Choose a proposal $Y_{n+1} = j$ according to q_{ij} from Q.

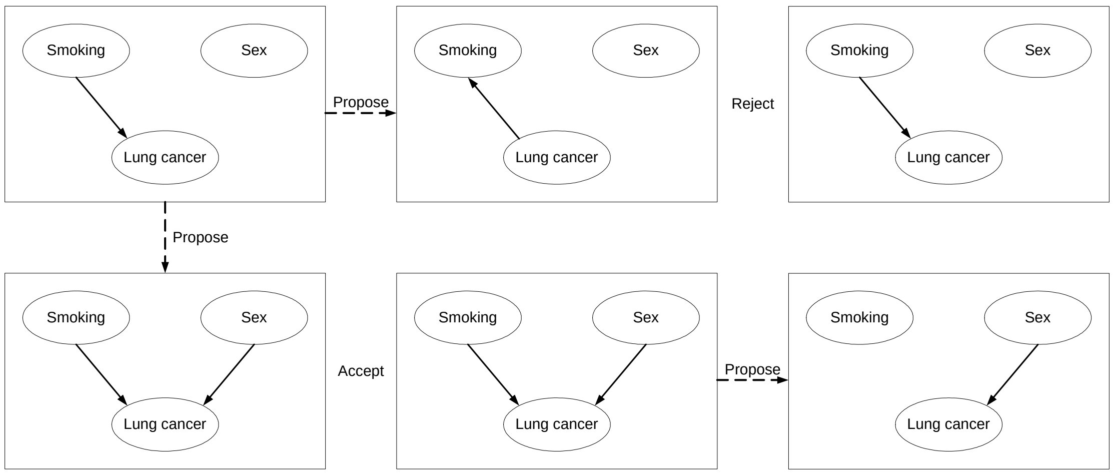


Figure 3.20: Starting at top left corner a new network (top middle) is proposed and accepted with probability α. Here the proposed network is rejected (top right). A new network (bottom left corner) is proposed and accepted. The new network becomes the initial network and a new network is proposed (right bottom corner). Illustration taken from [Hus04].

2. Set the acceptance probability

$$\alpha_{ij} = \min\left\{1, \frac{r(j)\, q_{ji}}{r(i)\, q_{ij}}\right\}$$

3.
   - Either the proposal $Y_{n+1} = j$ is accepted with probability α_{ij}. Thus $X_{n+1} = j$.
   - Or the proposal $Y_{n+1} = j$ is rejected with probability $1 - \alpha_{ij}$ and $X_{n+1} = X_n$.

Notice that $\frac{r(j)}{r(i)}$ is the likelihood ratio between the proposed sample and the previous sample and that the normalizing constant is canceled out by the division. This results in a Markov chain $X_1, X_2, \ldots,$ which has P as a stationary distribution. This way of constructing the Markov chain is called the Metropolis-Hastings method [AdFDJ03, CG95, HDR05, Nea03].

For structure learning the proposal is a new structure. This new structure is proposed from the previous one with the proposal distribution as illustrated in Figure 3.20. There are three elementary proposal moves for a Bayesian network structure: 1) deletion of an edge 2) reversal of an edge 3) creation of a new edge. Thus for structure learning the acceptance probability is [HDR05]:

$$\alpha_{ij} = \min\left\{1, \frac{P(D \mid M_j) P(M_j) Q(M_i \mid M_j)}{P(D \mid M_i) P(M_i) Q(M_j \mid M_i)}\right\} \tag{3.41}$$

where Q is the transition matrix specifying the chances of moving from one network structure to the other by one of the elementary MCMC moves shown in Figure 3.21.

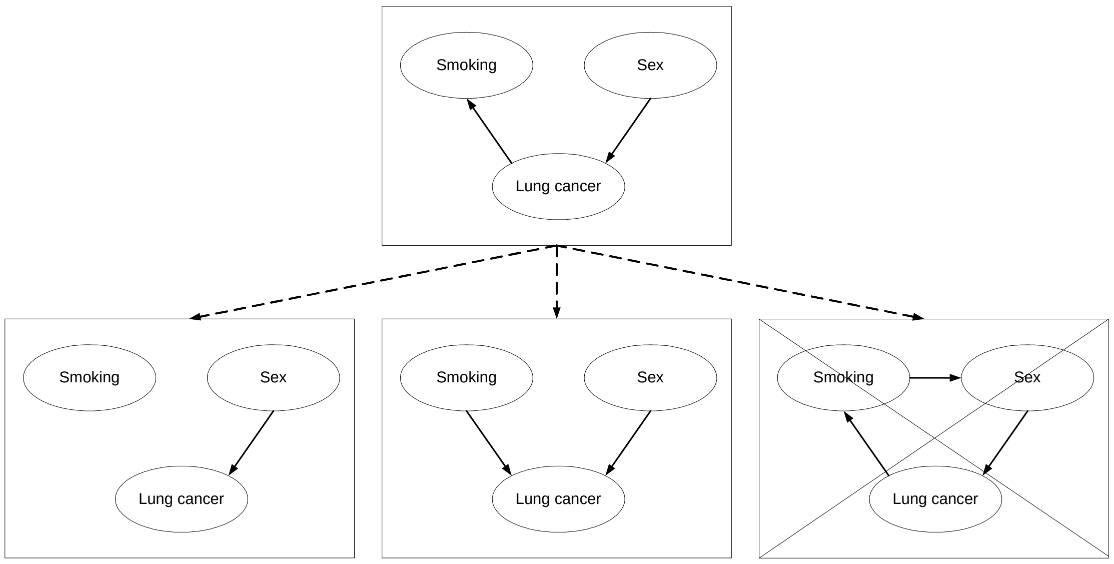


Figure 3.21: The three elementary MCMC moves [Hus04]. From bottom left to right: edge deletion, edge reversal, edge creation. The last two operations may violate the acyclicity constraint.

If we choose Q symmetric (e.g., $Q(M_i \mid M_j) = Q(M_j \mid M_i)$ for all i and j), step 2 simplifies to:

$$\alpha_{ij} = \min\left\{1, \frac{P(D \mid M_j)P(M_j)}{P(D \mid M_i)P(M_i)}\right\} \tag{3.42}$$

which is called the Metropolis algorithm [Nea03].

3.6.5 MCMC implementation

For learning Bayesian networks with the MCMC algorithm we use Dirk Husmeier's Dynamic Bayesian Networks MCMC tool (DBmcmc) [Hus04]. DBmcmc is build on top of Kevin Murphy's Bayes Net Toolbox (BNT) [Mur01]. BNT is an extensive set of Matlab functions for working with Bayesian networks. BNT supports among others:

- many types of conditional probability distributions.
- decision and utility nodes, as well as chance nodes, i.e., influence diagrams as well as Bayes nets.
- many different inference algorithms (with the possibility to add more).
- methods for parameter learning, regularization and structure learning.
- BNT is open-source and easy to extend and adapt.

For learning we use the following parameters:

- Burnin: 2,000
- Samples: 20,000
- Sample interval: 1
- Maximum Fan in: 3
- Maximum Fan out: 0 (no restriction)

We use a fan-in of 3 because it is known that the expression of a gene is influenced by only a small number of other genes. On the other hand regulatory hub genes can influence a great deal of other genes (fan-out). [FLNP00, Hus03b]. The advantage of restricting the fan-in is not only a considerable reduction in the number of possible networks, but also in the complexity of resulting networks. This improves the convergence and mixing of the Markov Chain [Hus03b]. For small networks we will also train without a maximum Fan in.

We calculate the resulting network from the 20,000 samples by averaging the arrows. We divide the number of times an arrow occurs in all the sampled networks by the number of total networks. Then we round the resulting values. Thus an arrow occurring in more than 60% of the networks will eventually be drawn.

Using BNT we adapted the learning algorithm slightly for learning *dynamic* Bayesian networks. The $n \times m$ datamatrix is converted to a $2n \times m-1$ datamatrix where the first n rows are the columns $1 \ldots m-1$ and the second n rows are the columns $2 \ldots m$. In this way we include the time aspect of the data. This way of simulating dynamic Bayesian networks for Bayesian learning algorithms is the same method as used in the DBmcmc toolkit.

As the learning algorithms are very sensitive to noise we adapted the prior (for the DBmcmc toolkit) manually, including a strong preference for networks containing only the most evident arcs in either direction.

"All truths are easy to understand once they are discovered; the point is to discover them."

Galileo Galilei

Chapter 4

Results

In this chapter we present only the most interesting results. Some of the results are not included here because they are either not very informative, or they were insignificant:

- Relations found by using a threshold for determining extrema at the detection limits.
- Relations found using only derivative data
- Inhibition relations
- Relations with a lag > 1
- Relations found by the lagfunction, where *each* sample may be shifted one sample into the future.

The reasons why these results were insignificant in comparison to the other results are discussed in Section 5.3.

4.1 Unfiltered sets

Applying our method to entire datasets yielded many relations. Due to the sheer amount of relations these results were not systematically validated. However, from unofficial validations we got a rough idea of the proportions of true positives, true negatives, and unknown relationships.

We construct set names based upon the amount of smoothing, the lag and the similarity function that was used. The amount of smoothing is expressed by the σ value that was used for the Gaussian. The similarity is one of two functions:

1. comparison (**c**): straightforward pattern comparison.
2. lagfunction-1 (**l**): straightforward pattern comparison, but *one* extreme may be shifted one sample into the future.

As an illustration the elutriation dataset is shown in Table 4.1 with the corresponding variables.

Set	Smoothing	Lag	Similarity
elutriation_sc0	0.5	0	comparison
elutriation_sc1	0.5	1	comparison
elutriation_sl0	0.5	0	lagfunction-1
elutriation_sl1	0.5	1	lagfunction-1
elutriation_Sc0	1	0	comparison
elutriation_Sc1	1	1	comparison
elutriation_Sl0	1	0	lagfunction-1
elutriation_Sl1	1	1	lagfunction-1

Table 4.1: Set names, depended on smoothing, lag and similarity.

For all tables we will use the following column headers:

- **Set**: The set that was validated using the gold standard.
- **# Relations**: The number of relations in the set.
- **%TPs 1**: True Positives, the percentage of correct protein-protein interactions in the set according to the gold standard.
- **%TPs 2**: True Positives, the percentage of correct *indirect* protein-protein interactions in the set according to the gold standard.
- **%TNs**: True Negatives, the percentage of incorrect protein-protein interactions in the set according to the gold standard.

4.2 KEGG cellcycle filtered sets

We will first treat the results from filtering with the KEGG cellcycle pathway (as explained in Section 3.4.2), because we will reuse the produced sets when filtering with the GO information. Using the unfiltered sets for GO filtering would yield sets that are to large to be validated within a reasonable time.

Set	# Relations	%TPs 1	%TPs 2	%TNs
elutriation_sc0	1078	0.56	8.35	13.73
elutriation_sc1	246	0.00	5.28	11.38
alpha_Sc0	1640	1.10	11.10	14.15
alpha_Sc1	1075	0.09	8.56	16.37
cdc15_Sc0	226	0.00	7.08	15.93
cdc15_Sc1	131	0.00	8.40	16.03

Table 4.2: Relations with KEGG cellcycle filtering

4.3 GO cellular component filtered sets

Filtering with the GO cellular component (as explained in Section 3.4.1)does not remove many of the relations. The filtering takes a lot of time and thus we will only illustrate our findings for a few small sets. We will use the datasets from Section 4.2.

Set	# Relations	%TPs 1	%TPs 2	%TNs
elutriation_sc0	626	0.64	8.63	10.86
alpha_Sc0	974	1.44	13.14	12.94
cdc15_Sc0	146	0.00	6.85	16.44

Table 4.3: Relations first filtered with KEGG cellcycle and then with GO

4.4 Combined sets

Here we present the most promising results achieved by combining datasets as explained in Section 3.4.3.

Set 1	Set 2	# Relations	%TPs 1	%TPs 2	%TNs
alpha_Sc0	cdc15_Sc0	66	15.15	15.15	9.09
alpha_Sc0	elutriation_sc0	150	5.33	13.33	9.33
cdc15_Sc0	elutriation_Sc0	38	5.26	21.05	21.05
alpha_Sc0	elutriation_Sc0	528	1.89	15.53	15.53
alpha_Sc1	cdc15_Sc1	38	0.00	5.26	0.00
alpha_Sc1	elutriation_sc1	24	0.00	0.00	25.00

Table 4.4: Combined sets

Because of the small amount of remaining relations we also combined relations found by the more flexible lagfunction from Section 3.3.1.

Set 1	Set 2	# Relations	%TPs 1	%TPs 2	%TNs
alpha_Sl0	cdc15_Sl0	764	2.62	9.57	9.44
alpha_Sl0	elutriation_sl0	2008	1.59	9.76	14.14
cdc15_Sl0	elutriation_sl0	290	1.38	4.14	11.72

Table 4.5: Combined sets with lagfunction

4.5 Random sets

For comparison we created three sets of random relations from each of the datasets (not all datasets contain exactly the same genes).

Set 1	# Relations	%TPs 1	%TPs 2	%TNs
random alpha	1000	0.30	2.00	12.80
random elutriation	1000	0.10	3.50	13.70
random cdc15	1000	0.20	2.30	14.20

Table 4.6: Randomly generated sets

4.6 Chi-squares

For a better interpretation we represented the results from the combined datasets in chi-squares.

	positive 1	positive 2	negative	neither	total
tested positive	10	10	6	40	66
tested negative	20.624	459990	2699994	14819326	17999934
total	20.634	460000	2700000	14819366	18000000

Table 4.7: alpha_Sc0 and cdc15_Sc0 chi-square

Sensitivity 1:	0.048 %
Sensitivity 2:	0.002 %
Specificity:	100%

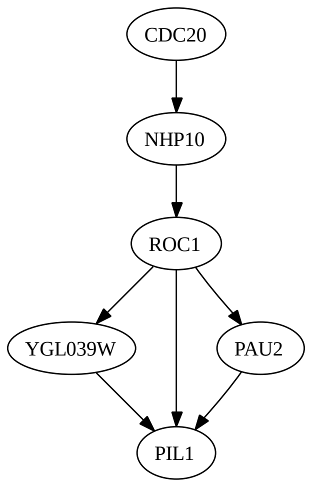
	positive 1	positive 2	negative	neither	total
tested positive	14	41	30	255	340
tested negative	20.620	459959	2699970	14819111	17999660
total	20.634	460000	2700000	14819366	18000000

Table 4.8: alpha_Sl0 and cdc15_Sl0 chi-square

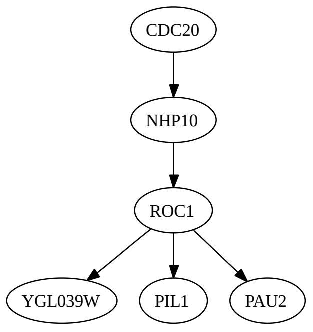
Sensitivity 1:	0.068 %
Sensitivity 2:	0.009 %
Specificity:	100%

4.7 Bayesian networks

Because the learning of Bayesian networks was a bit of a problem (see Section 5.7) we imposed a strong prior. The resulting networks are shown below in each of the following figures. The left network is the standard learning algorithm while the right network shows the adapted learning algorithm with a strong prior. Note that the genes are only related on the basis of their discrete patterns and may have no biological relation at all. For each of the networks we added a strong prior belief in a relation between NHP10 and CDC20, NHP10 and ROC1, and ROC1 and YGL039W. We did not specify the nature (direction) of the relation. Furthermore, the data supported a relation between ROC1 and PIL1 as well as ROC1 and PAU2. These relations however were not specified by our strong prior.

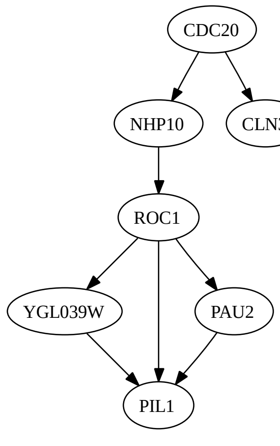


(a) normal learning of related genes

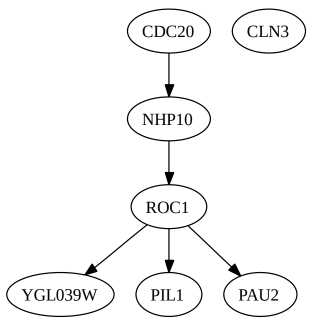


(b) learning with adapted prior

Figure 4.1: Two networks learned with DBmcmc. The left network is learned without prior while the right network has a strong prior for the most obvious relations.

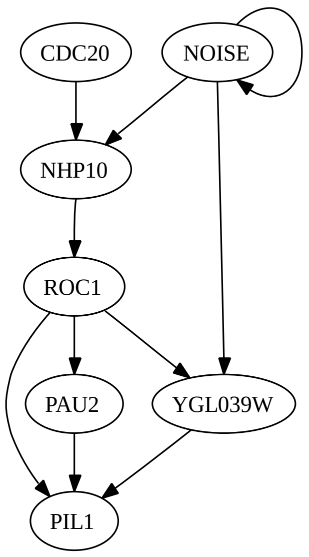


(a) normal learning of related genes, CLN3 is a noise gene

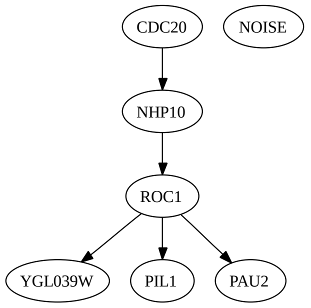


(b) learning with adapted prior, CLN3 is a noise gene

Figure 4.2: Two networks learned with DBmcmc. The left network is learned without prior while the right network has a strong prior for the most obvious relations; CLN3 is a noise gene.



(a) normal learning of related genes, NOISE is a noise gene, where every sample is a 1.



(b) learning with adapted prior, NOISE is a noise gene, where every sample is a 1.

Figure 4.3: Two networks learned with DBmcmc. The left network is learned without prior while the right network has a strong prior for the most obvious relations.

Using this strong prior we were also able to learn new relations that were reasonably supported by the data.

"There is no harm in doubt and scepticism, for it is through these that new discoveries are made."
Richard Feynman, 1985

Chapter 5

Discussion

This chapter will discuss the methods that were employed as well as address some issues with the interpretation of the results.

5.1 Data

Although the Spellman data is widely used, it is not without controversy. For instance, [FSZ02] claims that less than 20% of known regulatory pairs exhibited strong correlation in the data, while both [SC02] and [WFS04] come to the conclusion that the elutriation data contains only few cellcycle related genes and that periodically expressed genes in the other experiments are likely to be artifacts of the method used for synchronization of the cells.

5.2 Preprocessing

In the preprocessing phase a lot of choices were made that were of great influence to our later results.

5.2.1 Smoothing

The parameters that were used for smoothing the data (using the maxmin-filter, and the σ value for smoothing) have a great effect on the ultimate results. These values must be adjusted specifically for different datasets dependent on the amount of noise, the time between samples and the time of the biological process of interest. An improvement in our method would be to estimate the scale-space parameter σ automatically [YLZ05]. From the results we can furthermore see that with more samples we should use more smoothing.

5.2.2 Finding local extrema

Choosing discrete values for continuous data is not an obvious discission when the data is sparse. With this process information is lost, which seems unwanted as there is initially little information available. However choosing discrete values dependant on the local extrema is also a method of removing noise within the data; focusing only on the overall pattern in the expression data. Furthermore, most algorithms (including Bayesian learning algorithms) handle discrete data much more efficiently than continuous data (mainly because calculating sums is much easier than calculating integrals).

Instead of only three discrete values, a more detailed division could be achieved by using additional values for rising and falling edges. However, since the pattern of local extrema determines such values, such a division will not enhance on the chosen representation. An even more detailed discretisation might be done by using the derivative data.

Detection limit

From a purely theoretical point of view, determining an extrema at the first and last sample of the data is impossible. From the results of the experiments we concluded that determining such values is not beneficial.

Plateaus

One improvement on the current method could be to introduce one extra discrete value for a stable expression (no extrema, no rising or falling edge).However, completely stable expression is not likely to occur because of the noise in the data. Thus a more or less arbitrary threshold of some kind should be used to determine stable expression. Stable expression seems most relevant near local extrema where the exact timepoint of the local extrema is dubious. The use lagfunction might already circumvent this problem.

5.3 Similarity function

The first striking observation seems to be the sheer amount of relations that are found within individual datasets. We should however consider that, for example with yeast, finding 360,000 relations, constitutes only 2% of the total amount of 18,000,000 possible relations. On the other hand, there are some reasons why we find more relations than strictly necessary.

First of all, we find a lot of “duplicate” relations. This is because our similarity function is *symmetric*. This is most obvious with lag 0 relationships, where for m genes with a similar pattern we will find $m \cdot (m-1)$ relations because as seen in 3.17:

$$S_0(g,h) \Leftrightarrow S_0(h,g)$$

One way of dealing with this would be to group genes with similar patterns into equivalence clusters, and to deduct the relations from these groups. Note that the similarity lag-function is

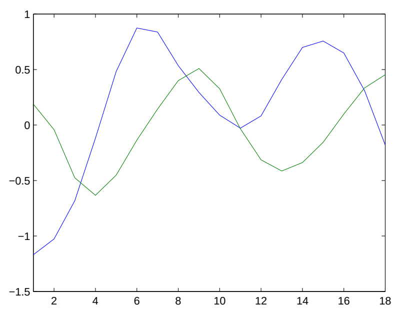


Figure 5.1: Gene 1 (blue) is possibly inhibited with lag 2 by gene 2, or gene 2 is possibly stimulated with lag 2 by gene 1.

not symmetric.

A second reason for "duplicate" relations is the fact that our similarity functions are *transitive* with respect to each other, when including the time aspect. For example a gene g which has a lag 1 relation with a gene h and h has a lag 1 relation with a gene k then g must have a lag 2 relation with k, as we saw in 3.18:

$$S_l(g,h) \wedge S_p(h,k) \Rightarrow S_{l+p}(g,k)$$

These relationships could be filtered out by Occam's razor, or more simple still, by not specifically looking for lags greater than 1, as these can easily be found indirectly.

A third reason is that when we look at lags greater than 1, relations that were previously found as lag 2 stimulation relationships are also discovered as potential lag 2 inhibition relationships as illustrated in Figure 5.1. This also occurs with different combinations of lag and stimulation/inhibition.

The above three arguments do not hold for finding relations with the lag-function, since this relationship does not have the symmetric or transient properties that the other similarity functions have.

A last reason why we find so many relations is a matter of chance. With a relatively small amount of samples, a large amount of genes, and only three possible discrete values, the probability that two genes will purely by chance have similar patterns is quite large. This plays a larger role with larger lags because we loose more samples. The same problem of chance also applies for biological processes. The probability that with one dataset some other biological process is running in parallel modulo with some lag to the process we are interested in, is relatively high. We believe this fact is illustrated by the results of our filtering method that combines two datasets.

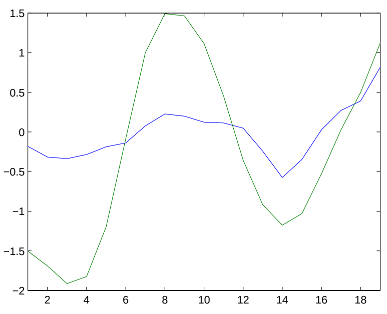


Figure 5.2: Possibly related genes not found by our similarity function based upon derivative data only.

The results obtained by only using derivative data were disappointing. This negative results can easily be explained; genes may have a very similar expression pattern, but if one gene has higher maximum expression values as well as lower minimum expression values, the edges in the rising and falling edges will be much more steep compared to the other gene (as illustrated in Figure 5.2). This mostly explains why this method will overlook important possible gene interactions.

5.4 Filtering

As can be seen from the results in Section 4.3, the use of GO spatial information does not improve our results very much. It is clear that by using GO we are also removing true positive relations. As an example of such a relation we look at the genes DBF2 (YGR092W) and SPO12 (YHR152W) which according to [PJ92] are directly related. However, if we look at the GO cellular component information of both, neither have a GO identifier as an ancestor of the other. And thus according to the GO cellular component information, these genes cannot be related.

As a small test we compared the expression patterns of 20 genes belonging to the same GO biological process, GO:0031497 (chromatin assembly). A lot of the genes show very similar patterns as shown in Figure 5.3.

It is interesting (though not very surprising) to note that genes that do not have a GO identifier never occur in our true positive database.

Furthermore we see that filtering with the KEGG cellcycle pathway, increases our results somewhat, but in our opinion not significantly compared to the results obtained by combining datasets.

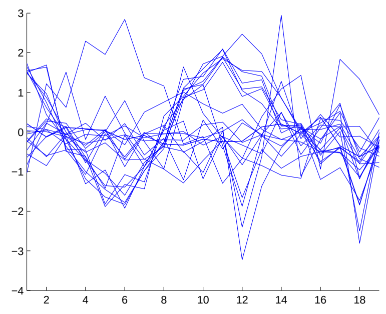


Figure 5.3: Expression of chromatin assembly genes (GO:0031497)

5.5 Validation

Here we will discuss why our methods do not find more correct gene interactions, and less incorrect interactions.

5.5.1 True positives

There are several reasons why we find a relative small portion of the interactions in the true positive database.

First of all, the measurements we used are indirect measurements. We are not measuring the proteins that the genes encoded for, but only the intermediate mRNA. While the mRNA may already have been broken down, the actual protein it encoded for may still remain. The opposite is also possible: while the mRNA remains, the protein might already have been transformed into something else. Thus some relationships are simply not supported by the data (see Figure 5.4), which corresponds with the findings of [FSZ02, SC02, WFS04].

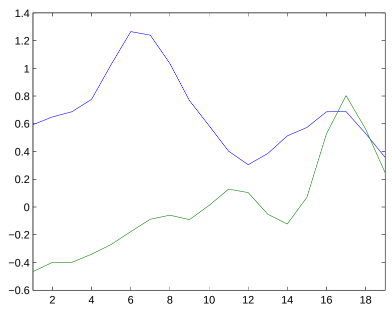


Figure 5.4: The expression of gene CLN3 (blue) and CDC53 (green) which according to the golden standard are related.

Furthermore some genes show very low expression values, which makes them more sensitive to noise. For example the expression of the gene CDC28 in the α-arrest experiment, varies only between the log expression values 0.4 and -0.4 while CLN1 log expression values vary between 1.5 and -1.5.

On the other hand, a lot of true relationships are not in this database, because they are either not real protein-protein interactions or because their correctness is still under discussion. For example we find a relationship between HTA1 and HTA2, which are indeed homologous [KCY$^+$06] but are not in our true positive database. We do however find them in our set of indirect true positive relations. The relationship HFF2 and HTA2 is in no database (direct true positive, indirect true positive, negative) at all . However HFF2 has a relationship with HTA1 which is homologous to HTA2 [FHSF94, GEJ$^+$94].

The gold standard we use is likely to contain mainly co-regulation interactions because the most successful methods for finding gene interactions (like clustering techniques) mostly find co-regulations. So relations with a lag > 0 will be less likely to be in the gold standard. Furthermore, there are a lot more true negative relations than true positive relations. Only by looking at the size of the true negative database we can see that the chance of finding a true negative by coincidence is a lot higher than finding a true positive.

The datasets we are using were created specifically for studying the cellcycle. It seems very logical that most relations we find are cellcycle related. The true positive database however does not only contain protein-protein interactions that are purely cellcycle related.

A last reason may be that the sampling times are too large and some direct relationships cannot be found. It is unknown what time is needed for gene expressions. Some gene interactions may be a matter of seconds, while others may take minutes or hours. Because of this it is very likely that we are missing crucial interactions that take place between two consecutive measurements. For this reason we also decided to look at indirect gene relations. Looking at the results this approach seems valid.

5.5.2 True negatives

As already mentioned, the true positive database we used (a combination of DIP and MIPS gold standards) has an overlap with the true negative database. This overlap constitutes of no less than 6440 gene relations, which is more than 30% of the 20634 relations. For indirect relations this overlap is even greater. As the true negative database is constructed, among others, on the basis of spatial information we give the true positive database the benefit of the doubt. This phenomenon of contradicting interaction databases is not unknown [JYG$^+$03].

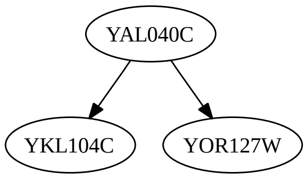


Figure 5.5: A simple Bayesian network, learned from three genes with DBmcmc and BNT.

5.6 Performance

Our method requires $\binom{m}{2}$ comparisons for m genes. Although this may seem very inefficient [JT04], in practice the comparison itself can be performed very efficiently, resulting in quite low computation times. In fact, the validation procedure takes a lot more time.

Some authors [FSZ02] note that a good expression analysis method should be able to handle scaling and shifting. Scaling and shifting can be caused by the microarray experiment or by the measured biological process itself. Scaling and shifting can be modeled as a linear function $y = qx + b$ where q is the scaling factor and b is the vertical shift. A horizontal shift is also possible. We can see that our method is invariant of vertical shifts and scaling differences. By using a different similarity function (e.g., the lagfunction) we are also able to cope with minor horizontal shifts.

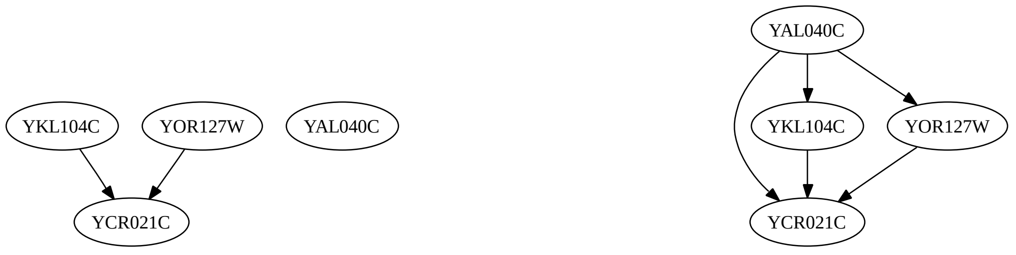
5.7 Bayesian networks

We stumbled on some unexpected problems in learning the Bayesian networks.
For a very small amount of genes (more than 7), where the discretized expression patterns match exactly with other patterns, the learned network is exactly what one would expect. The relations that were previously found by simple pattern matching are also learned by the Bayesian network. Unfortunately this also means that we have learned nothing new. On the other hand, if the network would show new relationships, these would most likely not be supported by the data, because we would have found them with the pattern matching procedure ourselves.

As an illustration we show a three gene network that was learned from data (α-arrest dataset filtered with $\sigma = 1$):

YAL040C	0 0 0 -1 0 0 0 0 1 0 0 0 -1 0 0 0 0 0
YKL104C	0 0 0 0 -1 0 0 0 0 1 0 0 0 -1 0 0 0 0
YOR127W	0 0 0 0 -1 0 0 0 0 1 0 0 0 -1 0 0 0 0

The network in Figure 5.5 was learned independently with BNT and DBmcmc. The arrows were found by calculating how many times each arrow occurs in the sampled networks, divided by the total amount of networks. The resulting values were then rounded: arrows occurring in more more than half of the sampled networks remain, other arrows are discarded. With BNT



(a) Bayesian network learned with BNT (b) Bayesian network learned with DBmcmc

Figure 5.6: Learned Bayesian networks, with 4 four nodes where YCR021C is added as noise.

this meant we had to manually increase the value with 0.05 for any arrows to remain.

For very small networks and a select set of genes we see that a very logical Bayesian network is constructed. However, this network does not contain any new information, e.g., information about relations that was not already known to us by matching genes manually.

We tested the robustness of the network by adding one or two random genes to the learning set. We evaluated the learned network on the basis of the original network that was learned without the random nodes. The random nodes that were added showed no clear correlation with the learning set and thus we would want the learned network not to have learned any additional arrows between the learning set and the noise nodes. However in practice some spurious relations were learned between the learning set and the noise nodes. Moreover, previously learned relations might be unlearned and additional relations within the training set would appear. Upon manual inspection, these new relations were only vaguely supported by the data.

As an illustration we learn the above network again with an additional noise node:

YCR021C 0 0 0 -1 0 1 0 0 -1 1 0 -1 0 0 1 0 0 0

As can be seen, in both networks the noise node has relationships with other nodes (although DBmcmc seems to learn a more consistent network). When using the strong prior, our results improved considerably. The learned networks corresponded better to the data, but were also able to find relationships not favored by the prior. We believe there are two reasons why learning without a strong prior produces negative results:

1. Bayesian networks are probabilistic models that are able to cope with noise. However, the noise was already removed from the data in our pre-processing phase. Thus the learning algorithm considers the data being noisy while in fact it is not.

2. The number of samples is too small. The data that we used contained only 18 sample, while [FLNP00] used 86 samples.

"If you're not living on the edge then you are taking up too much room."

No Fear

Chapter 6

Conclusions and future work

In this thesis we have applied signal processing techniques to extract local extrema from time-series expression data. We used these local extrema to discretize the expression data. A similarity function then grouped genes together in a pairwise fashion based on their discretized expression values. The results were validated by a gold standard protein-protein interaction database.

We achieved the best results by using:

1. no max-min filter;
2. the smoothing factor $\sigma = 1$;
3. no additional methods for the detection limit;
4. the stimulation similarity function with lag $l = 0$;
5. a filtering method where the relations found in two datasets were combined on the basis of a logical AND function.

Although there are a lot of factors to consider, we can conclude that our method discovers true gene relations from microarray expression data; and thus local extrema are a significant feature. Furthermore, we compared the results of our method with results obtained by discretization based upon different threshold values (which are not included in this thesis). These results confirmed our hypothesis that discretization based upon local extrema is superior to other discretization based upon threshold values.

Furthermore the applied method is:

- *Robust*: our method of comparison is invariant to vertical shifts and scaling differences. By using more advanced similarity functions the method can also cope with horizontal shifts [FSZ02].
- *Flexible*: by varying the smoothing factor σ we can adjust the procedure to different samplings, and the amount of noise. More samples will need more smoothing.

- *Specific*: with our method we obtained very high specificity; relatively few (with respect to the prior) found relations are false positives.
- *Extendible*: we can easily include additional filters based for example on, periodicy [WFS04], the amount of extrema, or expression values [CFS01]. Experiments have shown that such additional filtering will improve our results.
- *Efficient*: our method is computably efficient in that it can analyze entire datasets (6000 probes, 16 samples) within a reasonable amount of time (approx. 2 hours, depending on the hardware).

Our results show that some extensions to this method will yield negative results:

- GO Cellular component filtering: the use of GO cellular component information did not significantly increase our results, and validation using GO biological process information was not consistent with the true positive database.
- Determining discrete values at the detection limits. We saw that we find less confirmed gene relationships when we determine local extrema at the detection limits. We also noted that the samples at the beginning and end of the data are more sensitive to noise as the smoothing (based upon neighboring values, which are less at the beginning and end of the signal) has less impact there.
- Similarity function based upon derivative data only. Simply using derivative data as a comparison does not yield very good results. A more advanced method would be to use normalized derivative data, thus comparing genes on the basis of their relative change.

Although not included in this thesis, we also validated some of the relations with a larger true positive database (about 86.000 relations from [SBR^{+}06]), which resulted in a large increase of our true positive findings (for some sets more than 33%). For future validation of results we recommend this database.

We recommend extending the current method with known techniques for pre-filtering the data (possibly on the basis of absolute expression, change in expression, or the amount of local extrema within a subset of samples), and combining it with one of the many other existing methods. Another improvement would be to determine the smoothing factor σ automatically.

Finally we would recommend to compare the current method with other known analysis methods for gene expression time-series, and specifically with other (threshold based) discretization methods.

"Computers are to biology what mathematics is to physics."

Harold Morowitz

Acknowledgements

First of all I want to thank my supervisors. Michael Egmont-Petersen for the idea of this research, his original thinking, enthusiasm and support; Peter Lucas, for the assignment at the UMC St. Radboud, his support and his help with the more formal parts in my thesis and Perry Groot for his many detailed comments on my thesis.

From the UMC St. Radboud I want to thank Jayne Hehir-Kwa for correcting my poor English, and her pleasant company and Diederik Passchier for his interesting talks and discussions on many political, social and historical issues. Also my thanks to the department of Human Genetics, and particularly Joris Veltman and Ad Geurts van Kessel.

Furthermore I would like to thank everybody who in any way supported me, or contributed to this result, especially Maurice Samulski.

Bibliography

[ABB02] O. Alter, P. Brown, and D. Botstein. Singular value decomposition for genome-wide expression data processing and modeling. *Proceedings of the national academy of sciences in the united states of america*, 97(18):10101–10106, August 2002.

[AdFDJ03] C. Andrieu, N. de Freitas, A. Doucet, and M. Jordan. An introduction to MCMC for machine learning. *Machine Learning*, 50(1):5–43, 2003.

[AO01] C.M. Antunes and A.L. Oliveira. Temporal data mining: an overview. In G. Webb, editor, *Data Mining and Knowledge Discovery*, volume 10618, San Francisco, USA, August 2001. Springer US.

[BAD$^+$05] C. Ball, I. Awad, J. Demeter, J. Gollub, J. Hebert, T. Hernandez-Boussard, H. Jin, J. Matese, M. Nitzberg, F. Wymore, Z. Zachariah, P. Brown, and G. Sherlock. The Stanford Microarray Database accommodates additional microarray platforms and data formats. *Nucleic Acids Research*, 33 Database issue:580–582, January 2005.

[BDSY99] A. Ben-Dor, R. Shamir, and Z. Yakhini. Clustering gene expression patterns. *Journal of Computational Biology*, 6(3/4):281–297, 1999.

[Bes00] J. Besag. Markov Chain Monte Carlo for statistical inference, 2000. Available from: http://www.csss.washington.edu/Papers/wp9.pdf.

[BGL$^+$99] M. P. S. Brown, W. N. Grundy, D. Lin, N. Christianini, C. W. Sugnet, T. S. Furey, M. Ares, and D. Haussler. Support vector machine classification of microarray gene expression data. Technical report, June 1999. Available from: http://noble.gs.washington.edu/papers/genex.pdf.

[BST$^+$05] T. Barrett, T. Suzek, D. Troup, S. Wilhite, W. Ngau, P. Ledoux, D. Rudnev, A. Lash, W. Fujibuchi, and R. Edgar. NCBI GEO: mining millions of expression profiles. *Nucleic Acids Research*, 33 Database issue:562–566, January 2005.

[BvSMR03] R. van Berlo, E. van Someren M., and Reinders. Studying the conditions for learning dynamic Bayesian networks to discover genetic regulatory networks. *Simulation*, 79, December 2003.

[CdAdCdS04] I.G. Costa, F. de A.T. de Carvallho, and M.C.P. de Souto. Comparative analysis of clustering methods for gene expression time course data. *Genetics and Molecular Biology*, 27(4):623–631, August 2004.

[CFS01] T. Chen, V. Filkov, and S.S. Skiena. Identifying gene regulatory networks from experimental data. *Parallel Computing*, 27(1–2):141–162, 2001.

[CG95] S. Chib and E. Greenberg. Understanding the Metropolis-Hastings algorithm. *The American Statistician*, 49(4):327–335, November 1995.

[Chi95] D. Chickering. Learning Bayesian networks is NP-complete. In *Learning from Data: AI and Statistics V*, Florida, USA, January 1995. Springer-Verlag.

[CM98] G. Cooper and S. Moral, editors. *UAI '98: Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*, Madison, Wisconsin, USA, 1998. Morgan Kaufmann.

[Con00] The Gene Ontology Consortium. Gene Ontology: tool for the unification of biology. *Nature Genetics*, 25:25–29, 2000.

[Dd] A. Dad-del. Markov chains. Available from: http://www.math.ucdavis.edu/~daddel/linear_algebra_appl/Applications/MarkovChain/MarkovChain_9_18/MarkovChain_9_18.html.

[dHIK+03] M.J.L. de Hoon, S. Imoto, K. Kobayashi, N. Ogasawara, and S. Miyano. Inferring gene regulatory networks from time-ordered gene expression data of Bacillus subtilis using differential equations. In R. Altman, K. Dunker, L. Huter, T. Jung, and T. Klein, editors, *Biocomputing 2003*, volume 8, pages 17–28, Kauai, Hawaii, 2003. World Scientific.

[dHIM02] M. de Hoon, S. Imoto, and S. Miyano. Statistical analysis of a small set of time-ordered gene expression data using linear splines. *Bioinformatics*, 18(11):1477–1485, 2002.

[DLS00] P. D'haeseleer, S. Liang, and R. Somogyi. Genetic network inference: from co-expression clustering to reverse engineering. *Bioinformatics*, 16(8):707–726, 2000.

[Dop02] J. Dopazo. Microarray data processing and analysis. *Microarray data analysis II. Kluwer Academic. Publ.*, 2002.

[Els83] J. Elster. *Explaining Technical Change - A Case Study in the Philiosophy of Science*. Cambridge University Press, 1983.

[EPdJS04] M. Egmont-Petersen, W. de Jonge, and A. Siebes. Discovery of regulatory connections in microarray data. *Lecture Notes in Artificial Intelligence: Knowledge Discovery in Databases*, 3202:149–160, September 2004.

[ESBB98] M.B. Eisen, P.T. Spellman, P.O. Brown, and D. Botstein. Cluster analysis and display of genome-wide expression patterns. *Proceedings of National Academy of Science USA*, 95:14863–14867, December 1998.

[FHSF94] M. Fukuma, Y. Hiraoka, H. Sakurai, and T. Fukasawa. Purification of yeast histones competent for nucleosome assembly in vitro. *yeast*, 10(3):319–331, March 1994.

[FL04] I. Flesch and P. Lucas. Markov equivalance in Bayesian networks. Technical report, Radboud University Nijmegen, 2004.

[FLNP00] N. Friedman, M. Linial, I. Nachman, and D. Pe'er. Using Bayesian networks to analyze expression data. *Journal of Computational Biology*, 7:601–620, 2000.

[FMR98] N. Friedman, K. Murphy, and S. Russell. Learning the structure of dynamic probabilistic networks. In Cooper and Moral [CM98], pages 139–147.

[Fri98] N. Friedman. The Bayesian structural EM algorithm. In Cooper and Moral [CM98].

[FSZ02] V. Filkov, S. Skiena, and J. Zhi. Analysis techniques for microarray time-series data. *Journal of Computational Biology*, 9(2):317–330, 2002.

[FtHRKV92] L. Florack, M. ter Haar Romeny, J. Koenderink, and M. Viergever. Scale and differential structure of images. *Image and Vision Computing*, 10:376–388, July/August 1992.

[GEJ+94] P. Grant, A. Eberharter, S. John, R. Cook, B. Turner, and J. Workman. Expanded lysine acetylation specificity of gcn5 in native complexes. *journal of biological chemistry*, 274(9):5895–5900, February 1994.

[Gen05] DOE Genomics:gtl, systems biology for energy and environment, 2005. Available from: http://doegenomestolife.org/science/generegulatorynetwork.shtml.

[GK02] L. Göransson and T. Koski. Using a dynamic Bayesian network to learn genetic interactions. Technical report, Linköping University, Sweden, 2002. Available from: http://www.mai.liu.se/~tikos/dynbayesian.pdf.

[GMK+05] U. Güldener, M. Münsterkötter, G. Kastenmüller, N. Strack, J. van Helden, C. Lemer, J. Richelles, S. Wodak, J. Garcia-Martinez, J. Perez-Ortin, H. Michael, A. Kaps, E. Talla, B. Dujon, B. Andre, J. Souciet, J. De Montigny, E. Bon, C. Gaillardin, and H. Mewes. CYGD: the Comprehensive Yeast Genome Database. *Nucleic Acids Research*, 33 Database issue:364–368, January 2005.

[Got05] Raphael Gottardo. *Robust Bayesian Analysis of Gene Expression Microarray Data*. PhD thesis, University of Washington, 2005.

[HDR05] D. Husmeier, R. Dybowski, and S. Roberts. *Probabilistic Modeling in Bioinformatics and Medical Informatics*. Springer-Verlag, London, 2005.

[Hec96] D. Heckerman. A tutorial on learning with Bayesian networks. Technical Report MSR-TR-95-06, Microsoft Research, 1996. Available from: http://research.microsoft.com/research/pubs/view.aspx?msr_tr_id=MSR-TR-95-06.

[HQA+00] P. Hegde, R. Qi, K. Abernathy, C. Gay, S. Dharap, R. Gaspard, J. Hughes, E. Snesrud, N. Lee, and J. Quackenbush. A concise guide to cDNA microarray analysis. *BioTechniques*, 29(3):548–562, September 2000.

[Hus03a] D. Husmeier. Reverse engineering of genetic networks with Bayesian networks. *Biochemical Society Transactions*, 31:1516–1518, December 2003.

[Hus03b] D. Husmeier. Sensitivity and specificity of inferring genetic regulatory interactions from microarray experiments with dynamic Bayesian networks. *Bioinformatics*, 19:2271–2282, November 2003.

[Hus04] D. Husmeier. Inferring dynamic Bayesian networks with MCMC, 2004. Available from: http://www.bioss.sari.ac.uk/~dirk/software/DBmcmc/.

[JT04] L. Ji and K.-L. Tan. Identifying time-lagged gene clusters using gene expression data. *Bioinformatics*, 21(4):509–516, 2004.

[JYG+03] R. Jansen, H. Yu, D. Greenbaum, Y. Kluger, N. Krogan, S. Chung, A. Emili, M. Snyder, J. Greenblatt, and M. Gerstein. A Bayesian networks approach for predicting protein-protein interactions from genomic data. *Science*, 302(5644):449–453, October 2003.

[Kau69] S. Kauffman. Metabolic stability and epigenesis in randomly constructed genetic nets. *Theoretical Biology*, 22:437–467, 1969.

[KC01] M.K. Kerr and G.A. Churchill. Statistical design and the analysis of gene expression microarray data. *Genetical Research*, 77(2):123–128, April 2001.

[KCY+06] N. Krogan, G. Cagney, H. Yu, G. Zhong, X. Guo, A. Ignatchenko, J. Li, S. Pu, N. Datta, A. Tikuisis, T. Punna, J. Peregrin-Alvarez, M. Shales, X. Zhang, M. Davey, M. Robinson, A. Paccanaro, J. Bray, A. Sheung, B. Beattie, D. Richards, V. Canadien, A. Lalev, F. Mena, P. Wong, A. Starostine, M. Canete, J. Vlasblom, S. Wu, C. Orsi, S. Collins, S. Chandran, R. Haw, J. Rilstone, K. Gandi, N. Thompson, G. Musso, P. St Onge, S. Ghanny, M. Lam, G. Butland, A. Altaf-Ul, S. Kanaya, A. Shilatifard, E. O'Shea, J. Weissman, C. Ingles, T. Hughes, J. Parkinson, M. Gerstein, S. Wodak, A. Emili, and J. Greenblatt. Global landscape of protein complexes in the yeast Saccharomyces cerevisiae. *Nature*, 440(7084):637–643, March 2006.

[KGH+06] M. Kanehisa, S. Goto, M. Hattori, K. Aoki-Kinoshita, M. Itoh, S. Kawashima, T. Katayama, M. Araki, and M. Hirakawa. From genomics to chemical genomics: new developments in KEGG. *Nucleic Acids Research*, 34 Database issue:354–357, 2006.

[KIM04] S. Kim, S. Imoto, and S. Miyano. Dynamic Bayesian network and nonparametric regression for nonlinear modeling of gene networks from time series gene expression. *Biosystems*, 75(1-3):57–65, July 2004.

[KMC00] M.K. Kerr, M. Martin, and G.A. Churchill. Analysis of variance for gene expression microarray data. *Artificial Intelligence in Medicine*, 30(3):257–281, March 2000.

[KN03] K. Korb and A. Nicholson. *Bayesian Artifical Inteligence*. Chapmann & Hall/CRC Press UK, 2003.

[Kra98] P. Krause. Learning probabilistic networks. *Knowledge Engineering Review*, 13(4):321–351, 1998.

[LTW05] S.P. Li, J.J. Tseng, and S.C. Wang. Reconstructing gene regulatory networks from time-series microarray data. *Physica A: Statistical Mechanics and its Applications*, 350(1):63–69, May 2005.

[Luc00] P. Lucas. Bayesian analysis, pattern analysis, and data mining in health care. *Current Opinion in Critical Care*, 10:399–403, 2000.

[LvdGAH04] P. Lucas, L. van der Gaag, and A. Abu-Hanna. Bayesian networks in biomedicine and health-care. *Artificial Intelligence in Medicine*, 30:201–214, March 2004.

[MLCYW03] C.S. Möller-Levet, K.-H. Cho, H. Yin, and O. Wolkenhauer. Clustering of gene expression time-series data, 2003.

[MM99] K. Murphy and S. Mian. Modelling gene expression data using dynamic Bayesian networks. Technical report, 1999. Available from: http://www.cs.ubc.ca/~murphyk/papers.html.

[Mur01] K. Murphy. The Bayes Net Toolbox for matlab (BNT). *Computing Science and Statistics*, 33, 2001.

[Nea03] R. Neapolitan. *Learning Bayesian Networks*. Pearson Prentice Hall, Upper Saddle River, NJ, 2003.

[NKGT04] A. Narayanan, E. Keedwell, J. Gamalielsson, and S. Tatineni. Single-layer artificial neural networks for gene expression analysis. *Neurocomputing*, 61:217–240, Oktober 2004.

[Pea88] J. Pearl. *Probabilistic Reasoning in Intelligent Systems : Networks of Plausible Inference*. Morgan Kaufmann, September 1988.

[PJ92] V. Parkes and L. Johnston. SPO12 and SIT4 suppress mutations in DBF2, which encodes a cell cycle protein kinase that is periodically expressed. *Nucleic Acids Research*, 20(21):5617–5623, November 1992.

[PR98] J. Pearl and S. Russell. Bayesian networks. Technical Report R-216, 1998. Available from: http://ftp.cs.ucla.edu/pub/stat_ser/R216.pdf.

[PR00] J. Pearl and S. Russell. Bayesian networks. Technical Report R-277, November 2000. Available from: http://ftp.cs.ucla.edu/pub/stat_ser/R277.pdf.

[Qua02] J. Quackenbush. Microarray data normalization and transformation. *Nature Genetics*, 32:496–501, December 2002.

[Rob77] R. Robinson. Counting unlabeled acyclic digraphs. *Combinatorial Mathematics V*, 622:28–43, 1977.

[SAR04] P. Sebastiani, M. Abad, and M. Ramoni. Bayesian networks for genomic analysis. *Genomic Signal Processing and Statistics, EURASIP Book Series on Signal Processing and Communications*, pages 281–320, November 2004.

[SBR$^+$06] C. Stark, B. J. Breitkreutz, T. Reguly, L. Boucher, A. Breitkreutz, and M. Tyers. BioGRID: A General Repository for Interaction Datasets. *Nucleic Acids Research*, 34 database issue:535–539, January 2006.

[SC02] K. Shedden and S. Cooper. Analysis of cell-cycle gene expression in Saccharomyces cerevisiae using microarrays and multiple synchronization methods. *Nucleic Acids Research*, 30(13):2920–2929, 2002.

[Sha49] C. Shannon. Communication in the presence of noise. In *Proceedings of the Institute of Radio Engineers*, volume 37, pages 10–21, January 1949.

[SJ02] H. Steck and T. Jaakkola. On the dirichlet prior and bayesian regularization. In S. Becker, S. Thrun, and K. Obermayer, editors, *NIPS*, pages 697–704. MIT Press, 2002.

[SM03] T. Syeda-Mahmood. Clustering time-varying gene expression profiles using scale-space signals. In *CSB*, pages 48–56, Stanford, CA, USA, August 2003. IEEE Computer Society.

[SMS03] R. L. Stears, T. Martinsky, and M. Schena. Trends in microarray analysis. *Nature Medicine*, 9(1):140–145, January 2003.

[SMS$^+$04] L. Salwinski, C. Miller, A. Smith, F. Pettit, and J. Bowie D. Eisenberg. The Database of Interacting Proteins: 2004 update. *Nucleic Acids Research*, 32 Database issue:449–451, 2004.

[SR96] T. Strachan and A. Read. *Human Molecular Genetics*. Bios Scientific Publishers, Oxford, 1996.

[SSZ+98] P. Spellman, G. Sherlock, M. Zhang, V. Iyer, K. Anders, M. Eisen, P. Brown PO, D. Botstein D., and B. Futcher. Comprehensive identification of cell cycle-regulated genes of the yeast Saccharomyces cerevisiae by microarray hybridization. *Molecular Biology of the Cell*, 9(12):3273–3297, December 1998.

[Str92] T. Strachan. *The Human Genome*. Bios Scientific Publishers, Oxford, 1992.

[vSWR01] E. van Someren, L. Wessels, and M. Reinders. Genetic network models: A comparative study. In K. Iwata, editor, *MicroArrays: Optical Technologies and Informatics*, volume 4266, pages 236–247, San Jose, USA, January 2001.

[VVvV88] P. Verbeek, H. Vrooman, and L. van Vliet. Low-level image processing by max-min filters. *Signal Processing*, 15(3):249–258, October 1988.

[Weia] E. Weisstein. Leibniz integral rule. Available from: http://mathworld.wolfram.com/LeibnizIntegralRule.html.

[Weib] E. Weisstein. Sampling theorem. Available from: http://mathworld.wolfram.com/SamplingTheorem.html.

[WFS04] S. Wichert, K. Fokianos, and K. Strimmer. Identifying periodically expressed transcripts in microarray time series data. *Bioinformatics*, 20(1):5–20, 2004.

[WMSB98] A. Watson, A. Mazumder, M. Stewart, and S. Balasubramanian. Technology for microarray analysis of gene expression. *Current Opinion in Biotechnology*, 9(6):609–614, 1998.

[YLZ05] W. Yu, X. Li, and H. Zhao. Aligning peaks across multiple mass spectrometry data sets using a scale-space based approach. In *IEEE Computational Systems Bioinformatics Conference*, pages 126–127, Stanford University, 2005.

[ZH05] D. Zhu and A. Hero. Gene co-expression network discovery with controlled statistical and biological significance. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP'05)*, Philadelphia USA, March 2005.

[ZHZ+04] S. Zareparsi, A. Hero, D. J. Zack, R. W. Williams, and A. Swaroop. Seeing the unseen: Microarray-based gene expression profiling in vision. *Investigative Ophthalmology & Visual Science*, 45(8):2457–2462, August 2004.