Expression Networks
Revision 0.42

Utz J. Pape
Max-Planck-Institute for Molecular Genetics
Computational Molecular Biology
Ihniestr. 63-73
14195 Berlin

February 14, 2008
# Contents

1 Introduction 5

2 Methods 7
  2.1 cDNA Microarrays 7
    2.1.1 Technical Background 7
    2.1.2 Normalization 7
  2.2 Oligo Microarrays 10
    2.2.1 Preprocessing 10
    2.2.2 Quality Check 17
    2.2.3 Discretization with k-means clustering 18
  2.3 Correspondence Analysis 19
    2.3.1 The Proof 19
    2.3.2 Interpretation 20
  2.4 Data Simulation 21
    2.4.1 Simulation Model 21
    2.4.2 The Simulation Network 24
  2.5 Bayesian Networks 24
    2.5.1 Univariate Case 25
    2.5.2 Multivariate Case 27
    2.5.3 Learning 30
  2.6 Monte-Carlo-Markov-Chain 33
    2.6.1 Motivation 33
    2.6.2 The Algorithm 33
    2.6.3 Proof of Convergence 35
  2.7 Method Validation - ROC curves 36

3 Appendix 39
  3.1 Quantile Normalization 39
  3.2 Some Maths for VSN 40
  3.3 Linear Algebra 41
    3.3.1 Transpose of Matrices 41
    3.3.2 Symmetric Matrices 42
    3.3.3 Orthogonality of Eigenvectors of a Symmetric Matrix 42
  3.4 Prior Selection for Bayesian Networks 43
    3.4.1 Conjugate Prior for Binomial Distribution 43
    3.4.2 Conjugate Prior for Multinomial Distribution 45

Bibliography 47
Chapter 1

Introduction

Understanding the relations between the structure and functions of organisms, is one of the goals of biological research. Besides the value of this knowledge for getting deeper insights into Nature’s design, the knowledge can be used to provide perspectives on both the diagnosis and eradication of diseases. Many diseases are caused by changes in the genetic information and the concomitant phenotypic expression of cells. “Chromosome theory of inheritance” found at the beginning of the last century that genetic information is located on chromosomes organized in genes. In principle, all cells in an organism contain the same genetic information. Yet, cells from higher organisms of different tissues differ in shape, structure, and function. The specialized research field “Molecular Genetics” tries to elucidate the molecular processes leading to these differences.

The main difference at the molecular level is the amount of synthetized and accumulated RNAs and proteins. This is caused by variable expression of genes coordinated by a complex regulatory system (see [Britten & Davidson, 1969, Britten & Davidson, 1971]). To understand this complex regulatory system, one has to analyze the functional relations of genes and gene products, which is called “Functional Genomics”. So far, there are several regulation methods that are used by eukaryotes (see [Alberts et al., 2002]):

2. RNA processing control: E.g. control of RNA splicing.
3. RNA transport control: Export of RNAs to the cytoplasm.
4. Translational control: E.g. efficiency of translation by ribosomes.
6. Protein activation and stability control: Activation and inactivation of proteins, e.g. by phosphorylation, and degradation of proteins.

According to [Davidson, 2001], transcriptional control is the most important control strategy. The importance is apparent as otherwise the cell would synthetize unnecessary intermediates, which would be wasted resources. How does the transcriptional control work? Each gene has a site where transcription starts. In the upstream region, there are motifs to which proteins can bind. These proteins are called transcription factors (TFs). Besides this control, there are other factors that influence the transcription rate, for example the extent to which the DNA is folded. However, TFs might have the strongest influence to gene expression. Binding of TFs can affect the transcription rate in two different ways. An activator increases the rate while a repressor decreases it. One gene normally has several binding sites for TFs. TFs can act together in different ways: accumulation, complementation, or competition. In principle, TFs are transcribed products of the DNA itself. Thus it is obvious that we can model regulation by a genetic network, the connections of genes being regulatory interactions: a so called genetic regulatory network.
CHAPTER 1. INTRODUCTION

Many diseases are the result of dysfunctional groups of cells. As described above, the function of cells depends on their regulatory network. Thus one should find modifications in the regulatory network of a dysfunctional cell compared to a normal cell of the same type. Knowing what the modifications are means understanding a fundamental aspect of the disease. This knowledge gives the possibility to search for substances, which undo the modifications in the regulatory network, or which inhibit the outcome of the changes. Therefore it is a major goal to predict the genetic regulatory networks.

In the past, biologists discovered small subsets of regulatory networks by analyzing the results of knock-out experiments. This method is limited by two constraints. First, the preparation of knock-outs is very time consuming. Second, only a few genes can be observed so that the resulting structure of regulatory systems are small and incomplete. Two inventions resolve these restrictions. First, it is possible to silence genes using RNAi which saves long preparation time (see [Sharp, 1999]). Second, a snapshot of the expression of thousands of genes at the same time can be done by microarrays (see [Brown & Botstein, 1999]). This reduces the practical restrictions but shifts the problem to data assessment (see [Palsson, 2000]) and generates a new interdisciplinary research field “Computational Biology”.

“Computational Biology” tries to find mathematical models to apply to the data. To deal with the expression of thousands of genes to reconstruct genetic regulatory networks, several models were developed and applied, reviewed by [D’haeseleer et al., 2000, de Jong, 2002]. The most intuitive approach is to model the network at the molecular level like RNA degradation, TF binding, and diffusion, as it was done by [Chen et al., 1999]. As this approach needs many parameters (reaction constants, diffusion constants, etc.), it is limited to small systems. Nevertheless, one can use this approach to simulate a genetic regulatory network as a reference network (see [Zak et al., 2003]) for a reverse engineering method (see [Husmeier, 2003]): After defining the network structure and the parameters, one can generate data by this simulation. Then, this data can be used for reconstructing the network with another method. Finally, we can compare the reconstructed network with the reference network to assess the predictive power of the reconstruction method.

[Kauffman, 1969] applied boolean networks to predict network structure. This approach is intuitive because the structure of the problem, which is the network, is inherent in the method. As biological processes as well as the measurements by microarrays are noisy, [Friedman et al., 2000] explored Bayesian Networks for reverse engineering. The stochastic nature of Bayesian Networks handles noisy data automatically. Therefore, Bayesian Networks seem to be a promising approach.
Chapter 2

Methods

2.1 cDNA Microarrays

2.1.1 Technical Background

cDNA microarrays, invented by [Schena et al., 1995], are based on cDNAs isolated from cells and then printed on glass or membrane slides. As the printing is based on the cDNA library, different spots have different amounts of cDNA. Thus, inter-microarray and inter-gene comparisons are difficult to perform as every spot might be on a different scale. Therefore, a reference is also hybridized to the microarray labelled by another color. The preprocessing uses this information to normalize the data such that every measurement is on the same scale.

2.1.2 Normalization

In the analysis of microarrays, one assumes that the measured intensity is linearly correlated with the amount of mRNAs in the sample. Anyway, there are effects like fluorescence quenching which introduce non-linear effects into the measurement. One can use a lowess transformation to correct against it, see next section. In addition, there are other linear effects from dye, sample, production batch, etc. influencing the signal. We introduce ANOVA to remove such effects.

Lowess

In this section, we introduce the fluorescence quenching effect, it’s manifestation and explain the lowess transformation to correct this effect.

Quenching Effect

The mRNAs in the sample are labelled by fluorescent markers. After hybridization, the intensity is measured by activating fluorescence by a laser and measuring the emitted photons. Quenching occurs at high concentrations of fluorescent material: The fluorophores emit photons while nearby fluorophores absorb them such that the measured photons are less than the emitted ones, see [Kubista et al., 1994]. The non-linear influence on the measurement is shown by [Ramdas et al., 2001].

MvA Plots

This quenching effect can be observed by MvA plots. MvA stands for 'Minus versus Add' because the y-axis contains $\log(R/G) = \log R - \log G$ and the x-axis $\log(R \cdot G) = \log R + \log G$ where $R$ stands for red color measured intensity and $G$ for green. Sometimes these plots are also called R-I plots for ratio-intensity plots. An example is shown in figure 2.1.

The raw data in figure 2.1 shows a so called 'banana' shape. Low expressed genes seem to be upregulated in the red channel while moderately expressed genes seem to be upregulated in the green channel. As there is no biological explanation for that, it might be an artefact. The quenching effect can explain this observation because different dyes have the quenching effect at
different levels/number of atoms. Here, the quenching effect for the green dye takes place for a lower number of atoms than for the red dye. Thus, for genes in this intensity interval, the green intensities are underestimated leading to an overestimation of the red intensities. As the red intensity is in the nominator of the values plotted at the y-axis, these values are going up. In the intensity interval where also the red quenching effect starts to occur, we measure less red intensity resulting in an overestimation of the green intensities. Thus, the denominator of the $M$ value decreases the value of $M$.

**Lowess Transformation** [Yang et al., 2001] proposes a lowess transformation to correct for the quenching effect. Lowess is an abbreviation for LOcally Weighted linear RegrESSion. The procedure is quite simple: Define a window width (or smooth span) and run with this window over the x-axis. For each window, estimate a regression line considering only the points within the window. These local regression lines are the colored lines in figure 2.1A. Then, subtract for each gene the value of the corresponding regression line from the $M$ value

$$M^* = M - c(R, G)$$

where $M^*$ is the transformed $M = \log(R/G)$ value. $c(R, G)$ is the value of the fitted regression line. One can compute the regression line by a least square approach: Considering only the points within the current window, define the error of the fit by:

$$e(R, G) = \sum_i (c(R_i, G_i) - \log(R_i/G_i))^2$$  \hspace{1cm} (2.1)

Minimizing (2.1) with respect to the parameter $a$ and $b$ of the function $c(x, y) = a \log(x \cdot y) + b$ gives us the parameter $a$ and $b$. Notice that we have to do this for every window. Then we can easily compute $M^*$ by subtracting $c(R, G)$ from $M$. 

---

**Figure 2.1:** An example MvA plot for (A) raw data with local regression lines (with different smooth spans) and (B) lowess transformed data with smooth span equal to 0.05.
2.1. CDNA MICROARRAYS

ANOVA

Applying the lowess transformation to the data gives us a data set where the intensities correlate with the mRNA amount. Anyway, other linear effects can still bias the data. I.e., if we would like to compare different microarrays while the experiment is based on two batches of microarrays (two different production cycles), this two batches might have different properties. Also the dyes can bias the results if the dyes behave differently. We call these influences effects. Thus, we can formulate a model for the data where the right hand side of the formula explains the (measured) value of the left hand side. The following model is based on [Kerr et al., 2000]:

\[ Y_{ijkg} = \mu + A_i + D_j + AD_{ij} + G_g + VG_{kg} + DG_{jg} + AG_{ig} + \epsilon_{ijkg} \] (2.2)

Here, the value \( Y_{ijkg} \) is the measured intensity on the \( i \)th array of the \( j \)th color (normally red or green), the \( g \)th gene and the \( k \)th variety. A variety is a sample, i.e., if we compare a wildtype with a mutant, we have the two varieties 'wildtype' and 'mutant'. The model states that the measured intensity is the sum of a baseline measurement \( \mu \), which is the same for all measurements, an array effect \( A_i \), a dye effect \( D_j \), an array-dye effect \( AD_{ij} \), a gene effect \( G_g \), a variety-gene effect \( VG_{kg} \), a dye-gene effect \( DG_{jg} \), a array-gene effect \( AG_{ig} \), and an error \( \epsilon_{ijkg} \). The goal is to estimate all these effects by minimizing the error. If we are interested in the differentially expressed genes between the varieties (often 'wildtype' and 'mutant'), we can analyze the variety-gene effect \( VG_{kg} \) effect because these effect describes the difference between the varieties for each gene.

There are several points to discuss regarding this model. First of all, we describe how to determine differentially expressed genes leading to the multiple hypothesis testing problem. This is followed by a discussion of the theoretical framework. We conclude with some comments about the experiment design.

Differentially Expressed Genes The model in (2.2) is a linear model. As we have already mentioned, normally one might be interested in differentially expressed genes. Thus, we would like to test whether the gene-variety effect is zero or not. This is a typical Analysis of Variances (ANOVA). If we assume the error to be normally and i.i.d. distributed then every effect follows an F-distribution. Hence, we can apply an F-test to check whether the variety-gene effect, is unequal to zero or not. In this case, the null hypothesis is \( H_0 : VG_{gk} = 0 \) under which \( VG_{gk} \) follows an F-distribution. If \( P_{H_0}(VG_{gk}) < \alpha \) then the gene is significantly differentially expressed on a significance level of \( \alpha \).

Multiple Hypothesis Testing If we check for each gene whether it is differentially expressed or not, we have to run thousands of tests. Let’s denote the number of genes/tests by \( n \). Testing each gene on a level of \( \alpha \) means that for each significant gene we have a probability of \( \alpha \) that it’s not differentially expressed. This is called a false positive. Assuming all gene to be not differentially expressed (complete \( H_0 \) hypotheses), leads to \( n \cdot \alpha \) expected false positives. Normally, a reasonable significance level is 5%. Having 10,000 genes on the array and testing each of them for differential expression, leads to 500 expected false positives. Now assume that we would find 750 differentially expressed genes. If a biologist would check this by single experiments he should expect that only every third experiment succeeds.

Obviously, this is not the intention of the prediction. Therefore, one should correct for the multiple tests. The simplest method is the Bonferroni correction where each test is performed on a significance level of \( \alpha/n \). Unfortunately, this often results in a no differentially expressed genes. Thus, there exist also not as conservative methods as the Bonferroni correction. See [Dudoit et al., 2004] for a detailed discussion.

Theoretical Framework As stated above, one performs an ANOVA to test whether an effect is zero or not. If one is interested in performing other analysis on the normalized data, one can simply take the estimated variety-gene effect as the expression value. Estimating the effects (instead of testing) is normally called a regression.
CHAPTER 2. METHODS

not yet completed

Experiment Design  not yet completed

2.2 Oligo Microarrays

In this section, we deal with data generated by oligonucleotide microarrays. The main difference to cDNA microarrays is the fact that for oligo microarrays, the number of oligonucleotides per spot is the equal for every spot. Thus, it is much easier to compare different arrays. Hence, the arrays are hybridized only once without a reference sample. We assume the use of Affymetrix microarrays, [Lipshutz et al., 1999]. Affymetrix microarrays are organized as follows: For each gene there are 14 to 20 probe pairs in one probe set on one chip. Each probe pair has a Perfect Match (PM) oligonucleotide and a Mismatch (MM) signal which is the same oligonucleotide with one mismatch. After sample RNA binds to the probes, it is labelled with fluorescence. The intensity of fluorescence which is assumed to be proportional to the amount of RNA in the sample is measured. These measurements contain systematic and stochastic errors, which we have to minimize. This is the goal of preprocessing. It retrieves for each gene one intensity which is comparable between different arrays. After preprocessing we discuss several methods for verifying the quality of the data. In the last part of this section, we look at the problem of discretization.

2.2.1 Preprocessing

Preprocessing comprises background correction, normalization, and expression summarization. There exist several preprocessing methods like MAS 5.0 from Affymetrix (see [Affymetrix, 2002]), Robust Multiarray Analysis (RMA) introduced by [Irizarry et al., 2003b], and Variance Stabilizing Normalization (VSN) described in [Huber et al., 2003].

Background Correction  Background correction is done for each microarray separately. It eliminates the effects of non-specific binding and of auto fluorescence of the array. In VSN, background correction is automatically done by normalization.

Normalization  Normalization is necessary because of systematic differences in the amount of RNA between arrays caused by:

- RNA extraction
- Reverse transcription
- Labelling
- Photodetection

There are two different approaches to normalize data:

- Retrieving normalization parameters by experiment
- Estimating normalization parameters

An experimental solution uses the intensities of house keeping genes. The amount of RNA of these genes is known from the experimental design so that all intensities can be adjusted. If no house keeping genes are available, the normalization parameters are estimated by the measured values assuming that most of the genes are not differentially expressed. A standard method is quantile normalization (see [Bolstad et al., 2003]). Improved methods like VSN are based on error models. These methods also consider stochastic effects like
Several error models have been proposed:

- **MAS 5.0:** This version takes as error model:
  \[
  \log(PM_{ij} - CT_{ij}) = \log(\theta_i) + \epsilon_{ij}
  \]
  Here \( PM_{ij} \) denotes the PM intensities and \( CT_{ij} \) is for avoiding taking the log of negative numbers and is equal to \( MM_{ij} \) when \( MM_{ij} < PM_{ij} \) and is adjusted to be less then \( PM_{ij} \) when \( MM_{ij} \geq PM_{ij} \).

- **LiWong:** [Schadt et al., 2001] proposes the error model:
  \[
  MM_{ij} = \nu_j + \theta_i \cdot \alpha_j + \epsilon_{ij}
  \]
  \[
  PM_{ij} = \nu_j + \theta_i \cdot \alpha_j + \theta_i \cdot \phi_j + \epsilon_{ij}
  \]
  There \( MM_{ij} \) and \( PM_{ij} \) is the MM and PM for the \( i \)th array and the \( j \)th probe. \( \nu_j \) is the baseline response due to nonspecific hybridization and \( \alpha_j \) is the rate of increase of the MM response. \( \phi_j \) is the additional rate of increase in the corresponding PM response. \( \epsilon_{ij} \) is the random error. The probe response is yielded by the PM-MM difference:
  \[
  y_{ij} = PM_{ij} - MM_{ij} = \theta_i \cdot \phi_j + \epsilon_{ij}
  \]
  With some constraints on the parameters, one can estimate the parameters iteratively with least squares estimates.

- **RMA:** In [Irizarry et al., 2003a] another error model is proposed:
  \[
  T(PM_{ij}) = e_i + a_j + \epsilon_{ij}
  \]
  There \( T \) represents the transformation for background correction, normalization and logs the PM intensities. The normalization method is called quantile normalization (see appendix on page 39). \( e_i \) is the \( \log_2 \) scale value found on the arrays, \( a_j \) represents the log scale affinity effects and \( \epsilon_{ij} \) the error. The advantage of this model in comparison to the LiWong model above is that here the log transformation avoids the very strong variance-mean dependence.

- **VSN:** This method is based the standard error model. In this model, one can show that the variance of measured intensity \( Y_k \) of gene \( k \) has a quadratic dependence of the mean of \( Y_k \). As shown in [Huber et al., 2003] and [Huber et al., 2002] one can transform the data to make the variance independent of the mean. Let \( X_u \) be a family of random variables with \( EX_u = u, Var(X_u) = v(u) \). Define the transformation
  \[
  h(x) = \int_x^\infty \frac{1}{\sqrt{v(u)}} du.
  \]
  Then \( Var(h(x)) \) is independent of \( u \). The variance stabilizing method is implemented in the library vsn of Bioconductor (www.bioconductor.org).

We think that VSN is the most advanced one based on a reasonable statistical framework. Therefore, we explain it in more detail: A microarray experiment comprises several microarray samples \( i = 1, \ldots, d \). For each sample, we have \( k = 1, \ldots, n \) different probes. Now, we can decompose the measured value \( y_{ki} \) into a subsumed unspecific signal contribution \( \alpha_{ki} \) and a proportionality factor \( \beta_{ki} \) of the real transcript abundance \( x_{ki} \):
\[ y_{ki} = \alpha_{ki} + \beta_{ki} \cdot x_{ki} \]  

(2.9)

The unspecific signal contribution consists of non-specific hybridization, cross-hybridization, and unspecific background signals. The technology of microarrays leads to a relation of the unspecific signal contribution \( \alpha_{ki} \) resp. the proportionality factor \( \beta_{ki} \) between different samples \( i \) and also between different probes \( k \):

\[ \beta_{ki} = \beta_i \cdot \gamma_k \cdot \exp(\eta_{ki}) \]  

(2.10)

\[ \alpha_{ki} = a_i + \bar{\nu}_{ki} \]  

(2.11)

\[ \sum \eta_{ki} = \sum \bar{\nu}_{ki} = 0 \]  

(2.12)

The proportionality factor is decomposed into a sample normalization factor \( \beta_i \) and a probe affinity factor \( \gamma_k \). The remainder is subsumed in \( \exp(\eta_{ki}) \). The unspecific signal contribution has two components: The per sample offset \( a_i \) and a remainder \( \bar{\nu}_{ki} \). The parameters can be chosen such that the remainders sum to zero. We can go on reducing the number of parameters by three modeling steps:

1. \( \gamma_k \) is not estimated as we consider the measure of transcript abundance in probe-specific units: \( m_{ki} = \gamma_k \cdot x_{ki} \).
2. Take \( \eta_{ki} \) and \( \bar{\nu}_{ki} \) as noise term, independently and identically distributed (iid).
3. Estimate the remaining parameters from the data.

Plugging in these assumptions and decompositions, and considering \( y_{ki} \) as realizations of a random variable \( Y_{ki} \) (because of the two stochastic noise terms), we retrieve a stochastic model

\[ Y_{ki} = \alpha_{ki} + \beta_{ki} \cdot x_{ki} \]  

(2.13)

\[ \frac{Y_{ki} - a_i}{\beta_i} = m_{ki} \cdot \exp(\eta_{ki}) + \nu_{ki} \]  

(2.14)

with \( \nu_{ki} = \bar{\nu}_{ki}/\beta_i \). We could also write (2.14) by ignoring the index \( i \) and comprising some variables such that we get

\[ Y_k = \alpha + \beta_k \cdot e^\eta + \nu \]  

(2.16)

This the standard error model introduced by [Rocke & Durbin, 2001]. It states that the measured intensity \( Y_k \) of gene \( k \) is composed of an offset \( \alpha \), the real intensity \( \beta_k \), and a multiplicative resp. additive error term \( \eta_k \) respectively \( \nu_k \). Anyway, let’s go on with (2.15): If we assume normally or approximately normally distributed noise terms \( \eta_{ki} \sim N(0, \sigma^2_\eta) \) and \( \nu_{ki} \sim N(0, \sigma^2_\nu) \), we can compute the expectation and variance of \( Y_{ki} \) as

\[ E(Y_{ki}) = a_i + \beta_i \cdot m_{ki} \cdot E(\exp(\eta_{ki})) \]  

(2.17)

\[ \text{Var}(Y_{ki}) = \text{Var}(\nu_{ki} \cdot \beta_i) + \text{Var}(\beta_i \cdot m_{ki} \cdot \exp(\eta_{ki})) \]  

(2.18)

\[ = \epsilon_n^2 \cdot \beta_i^2 \cdot m_{ki}^2 + \beta_i^2 \cdot \sigma_\nu^2 \]  

with \( \epsilon_n^2 = \text{Var}(\exp(\eta_{ki})) \). See page 40 for some preliminary maths. When computing the variance, we first of all substitute \( \bar{\nu}_{ki} \) by \( \nu_{ki} / \beta_i \). As \( \eta_{ki} \) and \( \nu_{ki} \) are iid, we can split the variance. Then we change the order and shift constant values out of the variance by squaring. Solving (2.17) for \( \beta_i \), plugging into (2.18) for the first \( \beta_i \) shows the dependence of the variance on the expectation
2.2. OLIGO MICROARRAYS

\[
\text{Var}(Y_{ki}) = c_i^2 \cdot \beta_i^2 \cdot m_{ki}^2 + \beta_i^2 \cdot \sigma_v^2
\]
\[
= c_i^2 \cdot \left( \frac{E(Y_{ki} - a_i)}{m_{ki} \cdot E(\exp(\eta_{ki}))} \right)^2 \cdot m_{ki}^2 + \beta_i^2 \cdot \sigma_v^2
\]
\[
= c_i^2 \cdot (E(Y_{ki}) - a_i)^2 + \beta_i^2 \cdot \sigma_v^2
\]

with \( c_i^2 = c_\eta^2 / E^2(\exp(\eta_{ki})) \). For probes with a high transcript abundance, the variance is dominated by the first quadratic term and is therefore depending on \( \sigma_v^2 \). Probes of unexpressed genes have a variance of approximately \( \beta_i^2 \cdot \sigma_v^2 \). This is the background noise level for the \( i \)-th sample. As we can see, the variance contains the expectation value, meaning that the variance depends on the expectation value. We can rewrite (2.19) more generally as

\[
v(u) = c^2 \cdot (u - a)^2 + b^2
\]

The next step is to derive a transformation in general for this variance, in which the variance becomes independent of the expectation value. Before doing so, we look at an example: Figure 2.2 shows an sdM-plot for raw and normalized data. An sdM plot has the ranked mean per gene on the x-axis and the standard deviation per gene on the y-axis. We can clearly see, that for low transcript levels, the variance (as it’s simply the squared standard deviation) is constant. The reason is that (2.20) is dominated by the additive error \( b^2 \). In contrast, genes with higher transcript levels having a higher mean have a higher and increasing variance. This is due to \( c^2 \) in (2.20) because the expectation value \( u \) dominated the variance. As such a dependence makes analysis difficult, we transform the data to get a constant variance.

**Transformation Derivation** Let \( X \) denote a random variable with \( E(X) = u \) and \( h \) a function which is defined over the range of \( X \) and differentiable. We can make a first-order Taylor approximation at the mean (also known as delta method)

\[
h(X) = h(u) + h'(u) \cdot (X - u) + r(X) \cdot (X - u)
\]

with \( r(X) \) as a continuous function with \( r(u) = 0 \). The variance can be computed as:

\[
\text{Var}(h(X)) = \text{Var}(h(u) + h'(u) \cdot (X - u) + r(X) \cdot (X - u))
\]
\[
= \text{Var}(h'(u) \cdot (X - u)) + 2 \cdot \text{Cov}(h'(u) \cdot (X - u), r(X) \cdot (X - u))
\]
\[
+ \text{Var}(r(X) \cdot (X - u))
\]
\[
= h'(u)^2 \cdot \text{Var}(X) + 2 \cdot [E[h'(u) \cdot (X - u) \cdot r(X) \cdot (X - u)]
\]
\[
- E[h'(u) \cdot (X - u)] \cdot E[r(X) \cdot (X - u)]] + \text{Var}(r(X) \cdot (X - u))
\]
\[
= h'(u)^2 \cdot \text{Var}(X) + \text{Var}(r(X) \cdot (X - u))
\]
\[
+ 2 \cdot h'(u) \cdot E[r(X) \cdot (X - u)^2]
\]

After using (3.2), we apply (3.7) to the covariance term. As \( E(X - u) = E(x) - u = u - u = 0 \) the second term of the decomposed covariance is zero. We get the first term because \( \text{Var}(X - u) = \text{Var}(X) \). If \( h \) is approximately linear, the term \( r(X) \) is small and therefore the terms involving \( r(X) \) are negligible. For a family of random variables \( Y_u \) with \( E(Y_u) = u \) and \( \text{Var}(Y_u) = v(u) \), we can write:

\[
\text{Var}[h(Y_u)] \approx h'(u)^2 \cdot v(u)
\]

The variance stabilizing transformation should make the right hand side constant. Setting \( h'(u) = v^{-1/2}(u) \) leads to a constant on the right hand side of the equation. By integrating, we obtain:
CHAPTER 2. METHODS

Figure 2.2: Rank vs. Mean Plot to show the need for a variance stabilizing transformation.
2.2. OLIGO MICROARRAYS

\[ h(y) = \int_{y}^{y} \frac{1}{\sqrt{v(u)}} \cdot du \]  
(2.24)

If we assume \( v(u) = c^2 \cdot (u - a)^2 + b^2 \), we can compute the integral of (2.24). The derivation is shown in the end of this document.

\[ h(y) = \int_{y}^{y} \frac{1}{\sqrt{c^2 \cdot (u - a)^2 + b^2}} \cdot du \]

\[ = \frac{1}{c} \cdot \text{arsinh} \left( \frac{c \cdot u}{b} - \frac{a \cdot c}{b} \right) \]  
(2.25)

With this equation we can make the variance independent of the expectation value. Now, we have to find the transformation for our model in (2.15).

**Transformation of the Model for Microarray Intensities**

It is easily seen that we can substitute the appropriate terms in 2.19 by \( a, b, \) and \( c \), such that we get the transformation

\[ h(Y_{ki}) = \frac{1}{c^2 \eta} \cdot \text{arsinh} \left( \frac{Y_{ki} - a_i}{b_i} \right) \]  
(2.27)

with \( b_i = \beta_i \cdot \sigma_v / c^2 \eta \). The factor is negligible because it is an overall scaling factor. We obtain a better interpretation of (2.27) if we substitute the term of the inverse hyperbolic sine according to (2.15):

\[ \text{arsinh} \left( \frac{Y_{ki} - a_i}{b_i} \right) = \text{arsinh} \left( \frac{Y_{ki} - a_i}{\beta_i \cdot \sigma_v / c^2 \eta} \right) \]

\[ = \text{arsinh} \left[ \frac{c^2 \eta}{\sigma_v} \cdot (m_{ki} \cdot \exp(\eta_{ki}) + \nu_{ki}) \right] \]

\[ = \tilde{m}_{ki} \]

\[ = E \text{arsinh} \left[ \frac{c^2 \eta}{\sigma_v} \cdot (m_{ki} \cdot \exp(\eta_{ki}) + \nu_{ki}) \right] + \epsilon_{ki} \]

\[ = \mu_{ki} + \epsilon_{ki} \]

where \( \tilde{m}_{ki} \) is a random variable. This random variable can be decomposed into its expectation \( \mu_{ki} \) which is the true abundance on the transformed scale and an error term \( \epsilon_{ki} \). It is obvious that the mean of \( \epsilon_{ki} \) is zero and its variance is independent of the expression value as we have constructed it. Thus, we have achieved two goals: First, we have transformed the data such that the mean of the error is zero. Second, the variance is independent of the expression value.

The use of \( \text{arsinh}(\cdot) \) as a transformation becomes more intuitive by comparing to the logarithmic transformation which is normally used. The \( \text{arsinh}(\cdot) \) function is similar to the logarithm with the difference that it has no singularity at zero. It continues to be smooth and real valued for small and negative intensities. Figure 2.3 shows the logarithm and the inverse hyperbolic sine function.

We can also look at the analytical relationship between \( \text{arsinh}(\cdot) \) and \( \log(\cdot) \):

\[ \text{arsinh}(x) = \log(x + \sqrt{x^2 + 1}) \]  
(2.29)

\[ \lim_{x \to \infty} (\text{arsinh}(x) - \log(x) - \log(2)) = 0 \]  
(2.30)

Thus, if we compare two expression values: \( \Delta h_{ki,j} = h_i(y_{ki}) - h_j(y_{kj}) \) we get
\[ \Delta h_{kij} = \text{arsinh}(z_{ki}) - \text{arsinh}(z_{kj}) \] (2.31)

\[ \Delta h_{kij} = \log \frac{z_{ki} + \sqrt{z_{ki}^2 + 1}}{z_{kj} + \sqrt{z_{kj}^2 + 1}} \] (2.32)

with \( z_{ki} = (y_{ki} - a_i)/b_i \) and \( z_{kj} = (y_{kj} - a_j)/b_j \). For large intensities, the squared value in the root dominates and we can ignore the plus one. This yields to a normal log-ratio, whereby we can state that the difference of two transformed genes with high intensities is similar to the log-ratio. On the other side, if the intensities are small (< 1), we can ignore the squared term in the root and get the log-ratio of a fraction where the nominator and the denominator are nearly one. As the logarithm of one is zero, the logarithm of a number near to one is the number itself. This results in the difference of the transformed values \( z_{ki} - z_{kj} \) for small intensities.

Parameter Estimation We have introduced several parameters in the model. These parameters are estimated by robust maximum likelihood. We cannot use the normal maximum likelihood approach because the model is valid for non-differentially expressed genes only. If we use normal maximum likelihood for parameter estimation, the differentially expressed genes would bias the parameters. Nevertheless, we introduce the normal maximum likelihood and modify it afterwards to make it more robust.

The maximum likelihood approach chooses the model \( M \) (our parameter set) with the highest probability given the data \( D \):

\[ \hat{M} = \arg \max_M P(M|D) \] (2.33)

We can substitute the posterior \( P(M|D) \) according to Bayes:

\[ P(M|D) = \frac{P(D|M) \cdot P(M)}{P(D)} \] (2.34)

As \( M \) is the set of parameters \( M = \{ a_i, b_i, c_i, \mu_k \} \) and we do not know anything more about them, we assume that they are uniformly distributed. This makes the maximization of (2.34) independent of \( P(M) \). It is also independent of \( P(D) \). Hence, to maximize \( P(M|D) \), we
2.2. OLIGO MICROARRAYS

can maximize $P(D|M)$. As we assume a standard normal distribution with density $\phi$, we get the probability of observing a value $y_{ki}$ in the interval $[y_{ki}^\kappa, y_{ki}^\tau]$: 

$$ P(D|M) = P(y_{ki} \in [y_{ki}^\kappa, y_{ki}^\tau]) $$

$$ = \int_{y_{ki}^\kappa}^{y_{ki}^\tau} \phi \left( \frac{h_i(y_{ki}) - \mu_k}{c} \right) \cdot dh_i(y_{ki}) $$

$$ = \int_{y_{ki}^\kappa}^{y_{ki}^\tau} \phi \left( \frac{h_i(y_{ki}) - \mu_k}{c} \right) \cdot h'_i(y_{ki}) \cdot dy_{ki} $$

$$(2.35)$$

$$(2.36)$$

$$(2.37)$$

The last step is done according to (3.17). Thus, we can choose the parameters by maximizing the likelihood:

$$ \prod_{k=1}^{n} \prod_{i=1}^{d} \phi \left( \frac{h_i(y_{ki}) - \mu_k}{c} \right) \cdot h'_i(y_{ki}) $$

$$(2.38)$$

Now we can estimate the parameters of $h_i$ numerically with help of the profile likelihood according to [Murphy & van der Vaart, 2000]. To make the estimator robust, we use only a subset of probes in the maximization of (2.38). This is done by least trimmed sum of squared regression (see [Rousseeuw & Leroy, 1987]), where the subset is chosen by avoiding probes, which have a large residual.

With this procedure, we can calibrate automatically the data by the estimated parameters, which in addition makes the variance independent of the expectation. Now, we can conclude that genes with higher variance are supposed to be differentially expressed. Thus, we can use the variance as a measure whether the gene is differentially expressed or not.

Expression Summarization After normalizing the data, we retrieve expression values for each probe in a probe set. These probes are summarized to an intensity of a gene in the last step called expression summarization. VSN uses median polish for this task (see [Tukey, 1971]). Median polish assumes an additive model. This means that the measured gene expression value is explained by a sum of an overall factor, a probe factor, a chip factor, and an error term. The factors are estimated by an iterative procedure which uses the median over the probes/chips. The predicted gene expression value is the sum of the overall factor and the chip factor. Therefore, we get a gene expression value for each chip where the probes are summarized as the probe factors are ignored.

2.2.2 Quality Check

In microarray experiments there are sometimes crude errors, which make a chip unusable. It is important to find and exclude such chips because otherwise these errors would affect the whole analysis. On the other hand normalization affects the data strongly. Therefore, we need methods to check the result of normalization. For these purposes we use two methods which are presented next.

Standard-Deviation vs. Mean Plot

The Standard-Deviation vs. Mean (SDM) plot takes as first coordinate the mean or the ranked mean of a gene over all chips. The second coordinate the standard deviation of the gene. This plot is made especially for the analysis of the effects of normalization methods. As we have described above, the variance, which is the squared standard deviation, is dependent on the mean. Thus, the SDM of unnormalized data will show an increasing gradient while the SDM of normalized data should have a nearly constant function.
Median Absolute Deviation

Median Absolute Deviation (MAD) is used to find waste chips. It is based on a distance matrix, visualized by color. Let \( x_{ki} \) denote the expression value of genes \( k = 1 \ldots n \) on chip \( i = 1 \ldots d \). We denote \( x_i \) the vector of all genes on chip \( i \). Then we get a distance matrix \( D = (d_{ij})_{i=1\ldots d, j=1\ldots d} \) by computing \( d_{ij} = \text{dist}(x_i, x_j) \). The function \( \text{dist}(\cdot, \cdot) \) computes the distance between the arguments.

Of course, for MAD we use the median of the absolute deviation \( \text{dist} := \text{mad}(\cdot, \cdot) \) defined as follows:

\[
\text{mad}(x_i, x_j) := \text{median}(|x_{1i} - x_{1j}|, \ldots, |x_{ni} - x_{nj}|) \tag{2.39}
\]

Apart from this measure, one can also use correlation coefficients or other standard distance measures. The distance matrix can be visualized easily by taking \( i \) and \( j \) as coordinates and \( d_{ij} \) as the intensity of color at this point. We use dark blue for small \( d_{ij} \) and yellow for large \( d_{ij} \). The transition from small to large values is marked by a continuous course of color. This results in a dark blue diagonal because the distance between a chip and itself is zero. The distance measure is symmetric as we take the absolute value of differences. Therefore, we have \( d_{ij} = d_{ji} \) and it is regardless whether we look at the rows or columns. We prefer columns.

Waste chips will have a high distance to all other chips. This results in a nearly complete yellow column except at the diagonal point. Normal chips should have similarities to the other chips especially to their replicates. Apparently, the visualization depends on the ordering. If we order all replicates next to each other, this should result in a matrix where all columns are homogeneous at least as broad as the number of replicates are. If we additionally order the chips corresponding to their biological similarity, we will retrieve many blue squares with the center on the diagonal.

To detect extreme outliers on the level of chips more easily, one can take the median of each column and plot this value vs. the chip number. Peaks in this plot are created by chips which have a large distance to all other chips.

2.2.3 Discretization with \( k \)-means clustering

The measurements of mRNA in a cell return real values. This makes it difficult to define when the amount of mRNA has not changed because we cannot expect to have the same measured value in this case. Therefore it is recommended to discretize the data.

Discretizing is a critical task. First of all, we have to decide whether we discretize all genes to the same levels or if we discretize each gene individually. A compromise is to assume for all genes the same number of levels but to compute the levels for each gene individually. We use the clustering algorithm \( k \)-means, due to [Hartigan & Wong, 1979], to cluster each gene separately. The discretized value of an intensity is the number of its cluster.

For the description of \( k \)-means clustering we ignore the gene index of the variables as we cluster each gene separately. Given the intensities of \( d \) chips by \( x_1, \ldots, x_d \), we set \( k \) equal to the number of discretizing levels retrieving discretized intensities \( x'_1, \ldots, x'_d \in \{1, \ldots, k\} \). The clustering algorithm is simple:

1. Initialize \( k \) centroids \( z_1, \ldots, z_k \) randomly.
2. Assign each \( x_i \) to its nearest centroids \( z_{j(i)} \):
   \[
   c(i) = \arg\min_{j=1\ldots k} \text{dist}(x_i, z_j) \quad \forall i
   \]
3. Move centroids to the center of its cluster:
   \[
   z_j = \frac{1}{|\{i : c(i) = j\}|} \sum_{i : c(i) = j} x_i \quad \forall j
   \]
4. Go to step 2 until the assignment keeps stable.
5. To use this clustering algorithm for discretization, we have to fulfill two additional steps:
   a. Order the index set \( \{1, \ldots, k\} \) according to the values of \( z_j \). We denote the ordered indices with \( j' \).
Set the discretized values to its cluster numbers:

\[ x'_i = j' \]

The one dimensional euclidean distance \( d_E(x, y) = \sqrt{(x - y)^2} = |x - y| \) is used as distance measure. It is apparent that k-means clustering group the data into \( k \) groups according to their values. The groups are represented by the centroids. As the centroids are attracted by the nearest data points exclusively, they spread over the complete interval of data values. Finally, each data point has a centroid in its near neighborhood, and all data points in this neighborhood have a similar value. The advantage of k-means clustering as discretization algorithm is the fact that for each gene only the relative distances between the data points are considered.

2.3 Correspondence Analysis

As one experiment comprises many microarrays and each microarray contains thousand of genes, we have to deal with very high dimensional data. If one has high dimensional data, it is difficult to look at the underlying structure of the data, as we cannot look at data in a space higher than two or three dimensions. Thus, it is proximate to reduce the dimensions of the data to two dimensions where we can visualize it easily. Of course, we want to see the two dimensions which contain most information. This is done by correspondence analysis.

First, we will transform the data by basis transformation. We do this so that the first two components of each basis contain most information. Then we ignore the remaining components and visualize the data according to the first two components. The transformation is done by singular value decomposition. This transforms a matrix \( A \) to a product of an orthogonal matrix \( U \), a diagonal matrix \( \Lambda^{1/2} \), and another orthogonal matrix \( V^T \):

\[
A = U \cdot \Lambda^{1/2} \cdot V^T \quad (2.40)
\]

\( U \) contains the eigenvectors of the column space while \( V \) contains the eigenvectors of the row space of \( A \). \( \Lambda^{1/2} \) contains the eigenvalues \( \sqrt{\lambda_i} \) of \( A \). We will show, that we can do such a decomposition, and that the eigenvectors to the highest eigenvalues correspond to the direction in the space containing most information. This means, that we can plot the rows of the data by transforming to the basis contained in \( V \) and the columns of the data by transforming to the basis in \( U \).

2.3.1 The Proof

As \( U \) and \( V \) are orthogonal matrices, they hold

\[
U^T \cdot U = V^T \cdot V = I \quad (2.41)
\]

where \( I \) is the identity matrix. Now, we can multiply \( A^T \) to (2.40):

\[
\begin{align*}
A &= U \cdot \Lambda^{1/2} \cdot V^T \\
A^T \cdot A &= (U \cdot \Lambda^{1/2} \cdot V^T)^T \cdot (U \cdot \Lambda^{1/2} \cdot V^T) \\
&= (U \cdot \Lambda^{1/2} \cdot V^T)^T \cdot (U \cdot \Lambda^{1/2} \cdot V^T) \\
&= V \cdot \Lambda^{1/2} \cdot U^T \cdot U \cdot \Lambda^{1/2} \cdot V^T \\
&= V \cdot \Lambda \cdot V^T
\end{align*}
\]

(2.42) \quad (2.43) \quad (2.44) \quad (2.45) \quad (2.46)

After multiplying with \( A^T \) from the left side, we substitute the \( A^T \) on the right side of the equation by (2.40). Then we apply the transposition as shown in the Maths section 3.3.1, and use the identity of \( U^T \cdot U \) as written in (2.41). We obtain the singular value decomposition of a symmetric, and therefore, quadratic matrix. The symmetry of \( A^T \cdot A \) is shown in (3.24) in the Math section 3.3.2. Now, we can compute the eigenvalues and eigenvectors of \( A^T \cdot A \). We show in
the Math section 3.3.3 that the eigenvectors of a symmetric matrix are always orthogonal. How do we retrieve $U$? This can be done by the transformation $A \cdot v_i = \sqrt{\lambda_i} \cdot u_i$ which we get by multiplying $v_i^T$ and $A$ to the eigenvalue equation. First, we multiply the equation with $v_i^T$:

$$A^T \cdot A \cdot v_i = \lambda_i \cdot v_i$$  \hspace{1cm} (2.47)
$$v_i^T \cdot A^T \cdot A \cdot v_i = \lambda_i \cdot v_i^T \cdot v_i$$  \hspace{1cm} (2.48)
$$\langle A \cdot v_i, A \cdot v_i \rangle = \lambda_i$$  \hspace{1cm} (2.49)
$$\|A \cdot v_i\| = \sqrt{\lambda_i}$$  \hspace{1cm} (2.50)

As $\lambda_i$ is a scalar, we can shift it. We use the definition of the scalar product $\langle x, x \rangle = x^T \cdot x$ where it does not matter whether $x$ is multiplied by a matrix or not. On the right hand side of the equation we apply the orthogonality of the vectors of $v_i$. Then we take the squared root on both sides yielding the norm which is defined as $\|x\|^2 = \langle x, x \rangle$. The next step is to multiply $A$ to the eigen equation:

$$A^T \cdot A \cdot v_i = \lambda_i \cdot v_i$$  \hspace{1cm} (2.51)
$$A \cdot A^T \cdot A \cdot v_i = \lambda_i \cdot A \cdot v_i$$  \hspace{1cm} (2.52)

On the left side we can set the brackets so that we get $(A \cdot A^T) \cdot A \cdot v_i$. We can therefore see, that $A \cdot v_i$ is a eigenvector to $A \cdot A^T$. We can get the unit eigenvector by dividing by the length of it which we have obtained in (2.50). We get:

$$\frac{A \cdot v_i}{\|A \cdot v_i\|} = \frac{A \cdot v_i}{\sqrt{\lambda_i}} = u_i$$  \hspace{1cm} (2.53)

We retrieve the equation $A \cdot v_i = \sqrt{\lambda_i} \cdot u_i$ which means that $A$ is diagonalized by the basis $U$ and $V$ because we can write the equation for all components as $A \cdot V = \Lambda^{1/2} \cdot U$. As $\Lambda^{1/2}$ is a diagonal matrix and $V$ a orthogonal matrix, we get the equation $A = U \cdot \Lambda^{1/2} \cdot V^T$. Thus, we have demonstrated how to decompose a matrix in its singular values.

In the beginning we said, that the components of the highest eigenvalues in the transformed space contain the most information. This can be shown simply, because so far we have seen that the eigenvalues of $A^T \cdot A$ are the squared eigenvalues of $A$. Thus, the ordering of the eigenvalues is not effected by taking $A^T \cdot A$ instead of $A$. As we know that the covariance matrix of centered data $A$ is given by $A^T \cdot A$ with a scalar factor (inverse of the number of samples), we diagonalize in the singular value decomposition the covariance matrix of the data. As the variances are on the diagonal and the covariances are zero in the transformed space, the highest eigenvalues correspond to the components with the highest variance. This makes sense because a component without variance would be constant which means containing no information. On the other side, components with high variance contain more information. We can add the two highest eigenvalues and divide by the sum of all eigenvalues to retrieve the percentage of information which is visualized by the two dimensional plot. This holds only if the data is centered, hence we have to center the data before decomposing it.

### 2.3.2 Interpretation

There is also a graphical interpretation of the singular value decomposition as shown in figure 2.4. The matrices $U$ and $V^T$ perform rotations and mirroring while $\Lambda^{1/2}$ dilates the data: $A$ is a linear map. It maps the canonical basis which one can imagine as a unit circle in two dimensions and a unit sphere in three dimensions into an ellipse respectively an ellipsoid which do not need to be parallel to the axes. We can express this map by rotating the unit circle so that the basis vectors are on the axes which is done by $V^T$. Then we dilate the circle to an ellipse with $\Lambda^{1/2}$ and rotate the ellipse by $U$. As we assume that $A$ has the canonical basis, we can visualize the data simply by plotting the two components corresponding to the two highest eigenvalues.
2.4. DATA SIMULATION

Data generated by microarrays is very noisy as the measurement process is complex and contains many different sources of errors. It is not reasonable to use only such data to assess whether a new method works in general or not. Therefore, we also use simulated data. As mentioned in the introduction, the simulation of the biochemical reactions in a cell leads to a very realistic model. We cannot use it for reverse engineering as there are too many parameters to be estimated. Nevertheless, we can determine the parameters and take the outcome of the system as measured data. Subsequently, we can apply our method to the generated data with the goal to reconstruct the known artificial network. After describing the mathematical model, we review the basic components of a genetic regulatory network. We then apply the model to these basic components. Finally, we illustrate the simulation network.

2.4.1 Simulation Model

The activation, transcription, and translation of genes are chemical reactions. Therefore, we need a mathematical model for chemical reactions. Subsequently, we apply this model to genetic regulatory networks. We simulate the temporal behaviour of chemical reactions with differential
equations under the assumption of no further interactions between the environment and the modeled molecules. We begin with the most basic reaction, and we finish with the complete reaction system of the network.

**Concept of Molecular Dynamics** We demonstrate the concept following the illustration of [Deuflhard & Bornemann, 2002]. Let $A$ and $B$ denote substances which follow the monomolecular reaction $A \rightarrow B$. We assume both elements to be aeriform, therefore we can apply the kinetic gas theory of Boltzmann. The fundamental of this theory is the assumption that the number of collisions between two molecules is constant per volume unit and time unit assuming constant pressure, volume, and temperature. Furthermore, we assume that the reaction $A \rightarrow B$ is fulfilled with a constant probability when a collision occurs. If we denote the number of molecules of $A$ with $n_A$, the number of molecules of $B$ with $n_B$, the difference in the number of molecules $\Delta n_A$ and $\Delta n_B$ in a small interval of time $\Delta t$ follows the proportionality $\Delta n_A \propto -n_A \Delta t$. This proportionality is apparent as $A$ is on the left side of the equation and therefore transformed to $B$ resulting in a smaller number of molecules $A$. The law of conservation of mass gives us an additional equation $\Delta n_B = -\Delta n_A$. For the time interval $\Delta t \rightarrow 0$ we get differential equations with $k$ as reaction rate coefficient, the speed of the reaction:

\[
\frac{dn_A(t)}{dt} = \dot{n}_A = -k \cdot n_A \quad (2.54) \\
\frac{dn_B(t)}{dt} = \dot{n}_B = k \cdot n_A \quad (2.55)
\]

We ignore the discrete nature of the number of molecules as we normally are interested in systems with a large number of molecules. We can also express the differential equations according to the concentrations $c_A(t) = n_A(t)/V$ and $c_B(t) = n_B(t)/V$ in a constant volume $V$. Substituting these terms into (2.54) and (2.55) results in differential equations:

\[
\dot{c}_A = -k \cdot c_A \quad (2.56) \\
\dot{c}_B = k \cdot c_A \quad (2.57)
\]

These equations can be solved easily for the initial values $c_A(0) = 1$, and $c_B(0) = 0$:

\[
c_A(t) = \exp(-k \cdot t) \quad (2.58) \\
c_B(t) = 1 - \exp(-k \cdot t) \quad (2.59)
\]

This scheme can be extended to bimolecular reactions $A + B \rightleftharpoons C + D$. In addition to the reaction rate coefficient from left to right $k_1$, we get the reaction rate coefficient from right to left $k_2$. This leads to differential equations:

\[
\dot{c}_A = \dot{c}_B = -k_1 \cdot c_A \cdot c_B + k_2 \cdot c_C \cdot c_D \quad (2.60) \\
\dot{c}_C = \dot{c}_D = k_1 \cdot c_A \cdot c_B - k_2 \cdot c_C \cdot c_D \quad (2.61)
\]

The interpretation is straightforward: In a small interval of time the change of concentration for molecules $A$ is composed of two parts: First, it is the negative amount of molecules $A$ transformed to other molecules (C, D). Second, the concentration is increased by the amount of molecules of other types (C, D) transformed to molecules $A$. The more reactans and reaction we consider, the more complex the system becomes. In general, we cannot solve the system of ordinary differential equations (ODEs) analytically. Therefore, one uses a numerical integrator.
Equations for the Transcriptional Regulatory Module  This concept of chemical reactions can be transferred to the simulation of genetic regulatory networks. Due to [Barkai & Leibler, 2000], the basic components of a genetic regulatory networks are transcriptional regulatory modules. Figure 2.5 shows a module.

Genes are modelled by their promotors. $P_{1J}$ denotes the promotor for gene $I$ which can be activated by a dimerized protein from gene $J$. Activation is modelled as a reaction between a dimerized protein from gene $J$ yielding $J_2$ with the promotor $P_{1J}$:

$$P_{1J} + J_2 \xrightarrow{k_1} J_2P_{1J} \quad (2.62)$$

The transcript $MI$ of gene $I$ is the result of transcription. This is modelled by a reaction from the (activated) promoter yielding $MI$:

$$MI \xrightarrow{k_3} I \quad (2.63)$$

The protein $I$ can dimerize to $I_2$.

$$2 \cdot I \xrightarrow{k_4} I_2 \quad (2.64)$$

The promotor $I_2$ is also affected by all its binding and unbinding reactions, which we denote as $\sum PB$ and $\sum PU$. With the degradation rate coefficient denoted as $kd_4$, it yields the following differential equation:

$$[\dot{I}_2] = k_4 \cdot [I]^2 - k_5 \cdot [I_2] - kd_4 \cdot [I] \quad (2.65)$$

For the equation of mRNA $MI$ based on the reactions (2.65) and (2.66), we also consider a degradation rate coefficient $k_{d3}$:
\[ [M_I] = k_6 \cdot [P_{I,j}] + k_7 \cdot [J_2 P_{I,j}] - k_{d_3} \cdot [M_I] \]  \hspace{1cm} (2.70)

If the gene is regulated by more promoters, the reaction rate coefficients of the additional promoters have to be added equally.

### 2.4.2 The Simulation Network

[Zak et al., 2003] proposes a gene regulatory network built of modules of transcriptional regulation. These modules are arranged to regulatory motifs. The fundamental mechanisms of this network are taken from biological literature. There are four different regulatory motifs:

- **Cascade**: A cascade is a unidirectional flow.
- **Mutual repression**: Two genes which mutually repress each other.
- **Auto-activation and sequestration**: The dimerization partner of a TF determines its regulation.
- **Agonist-induced receptor down-regulation**: A ligand binds to its receptor and activates transcription. The transcribed mRNA acts as a repressor of the receptor.

The constructed network is dependent on the presence of a stimulus \( Q \). The network in absence of the stimulus is shown in figure 2.6, in presence of the stimulus in figure 2.7. The network contains cascades (i.e. C - G - H, C - K - J), a mutual repression motif (C - D), auto-activation (A - B), and an agonist-induced receptor down-regulation (E - F - D).

![Figure 2.6](image1.png)

**Figure 2.6**: In silico genetic regulatory network in absence of the stimulus taken from [Zak et al., 2003].

![Figure 2.7](image2.png)

**Figure 2.7**: In silico genetic regulatory network in presence of the stimulus taken from [Zak et al., 2003].

### 2.5 Bayesian Networks

We want to retrieve the regulatory network structure by expression values of genes. We have to state that we do not know the exact expression of a gene because it is perturbed by the environment and the measurement as we have seen above. Thus it is obvious to consider the genes as random variables denoting \( X_i \) for gene \( i \). The observed expression values of the genes are realizations of the random variables. This leads to a statistical framework with a set of random variables
2.5. BAYESIAN NETWORKS

\( \mathbf{X} = \{X_1, \ldots, X_n\} \). The measured expression values on a microarray chip is the set of realizations of the random variables of the set \( \mathbf{X} \). As we are looking for the regulatory structure of the genes, we have to define what “regulate” means: A gene \( i \) regulates gene \( j \) if the expression value of gene \( i \) directly influences the expression value of gene \( j \). This influence of gene \( i \) on gene \( j \) is a stochastic dependence of random variable \( X_j \) on random variable \( X_i \). The joint distribution \( P(\mathbf{X}) \) is influenced by the dependencies, and as we measure realizations of the joint distribution, it is intuitive to take the joint distribution as a starting point.

Analytic distributions are defined by sets of parameters. We are interested in the parameters which define the dependencies between the random variables. We can visualize the dependencies by considering each random variable as a node drawing arrows from the given random variable to the dependent one. We denote these dependencies / parameters by \( \lambda \) and call it structure. Since \( \lambda \) does not describe the whole joint distribution, we need some additional parameters denoted by \( \theta = (\theta_i)_{i=1,\ldots,n} \). Each component of this vector describes the local distribution for one random variable / node. Altogether, the model \( M = \{\lambda, \theta\} \) defines the joint distribution. It describes how the expression values/realization denoted with data \( \mathbf{x} \) have been generated. Actually, we want to learn the structure \( \lambda \) from \( \mathbf{x} \). So we are interested in the posterior \( P(\lambda = \lambda | \mathbf{X} = \mathbf{x}) \), which is the probability of the structure given the data. We abbreviate terms like this generally by ignoring the random variables if they are clear from context giving \( P(\lambda | \mathbf{x}) \). One fundamental problem is that for the greatest part it is not possible to compute this probability directly. Therefore we use Bayes’ theorem\(^1\) to compute the posterior:

\[
P(\lambda | \mathbf{x}) = \frac{P(\mathbf{x} | \lambda) \cdot P(\lambda)}{P(\mathbf{x})} \tag{2.71}
\]

If we want to obtain the best model, we maximize in (2.71) over all structures \( \lambda \). However, we cannot compute the likelihood \( P(\mathbf{x} | \lambda) \) without the remaining parameters \( \theta \). Hence we integrate over all \( \theta \):

\[
P(\lambda | \mathbf{x}) = \frac{P(\mathbf{x} | \lambda) \cdot P(\lambda)}{P(\mathbf{x})} = \int_\Theta P(\mathbf{x} | \theta, \lambda) \cdot P(\theta | \lambda) \cdot P(\lambda) \cdot d\theta \tag{2.72}
\]

As the prior of the data \( P(\mathbf{x}) \) is independent of \( \lambda \), we do not need to compute it. Then, the result is not a probability anymore but a score for the structure \( \lambda \). Nevertheless, we have to compute the integral over the parameter space which is not trivial.

To clarify the meaning of these parameters, and to show how to solve the integral, we introduce the concepts with a toy example based on one random variable. Before we can generalize this concept for multivariate random variables, we analyze dependencies between random variables as this is the mathematical analogue of genetic regulations. We complete this section with the explanation of the approximative algorithm Monte-Carlo-Markov-Chain (MCMC) to learn the dependencies of a set of random variables given their realizations.

2.5.1 Univariate Case

In this subsection, we introduce some preliminary principles for Bayesian Networks. To keep it simple, we use for this purpose a univariate distribution instead of multivariate one. We show how to estimate the parameters, which is called parameter learning. A univariate distribution does not have a structure \( \lambda \). The model \( M \) only consists of \( \theta \), giving \( M = \theta \). Therefore, we learn the model \( M \) or parameters \( \theta \) instead of \( \lambda \). We clarify this with an example:

\(^1\)This is the reason for “Bayesian” Networks.
Binomial Model - Coin Toss Example  We consider a Bernoulli-experiment with a coin but do not know whether it is a fair coin or not. Then we have \( \Omega = \{\text{head}, \text{tail}\} \), which is the set of possible outcomes of the experiment (head and tail), and \( \Theta = [0,1] \) as the set of possible parameters. We define the parameter \( \theta \in \Theta \) with \( P_{\theta}(\{\text{head}\}) := \theta \). Now we want to estimate \( \theta \), which we can interpret as the probability of head, based on some observed data \( x = \Omega^N \) where \( N \) is the number of experiments. First of all, we have to compute the posterior, and then we can estimate \( \theta \) with \( \hat{\theta} = E_{P(\theta|x)} \theta \). According to (2.71), the posterior is:

\[
P(\theta|x) = \frac{P(x|\theta) \cdot P(\theta)}{P(x)} \tag{2.73}
\]

The data prior \( P(x) \) has the function of a normalizing constant. It is computed by integrating over all parameters \( \theta \):

\[
P(x) = \int_\Theta P(x|\theta) \cdot P(\theta) \cdot d\theta \tag{2.74}
\]

As we assume a Bernoulli-experiment, we can calculate the likelihood \( P(x|\theta) \) with \( h \) as the number of heads and \( t = N - h \) as the number of tails in a sequence of experiments:

\[
P(x|\theta) = \binom{N}{h} \cdot \theta^h \cdot (1 - \theta)^t \tag{2.75}
\]

\( h \) and \( t \) construct a sufficient statistic for a Bernoulli-experiment, as can be proven. So, we do not need to remember the order of events like \{tail, tail, head, tail, head, \ldots \}.

Now we have to assume a prior for the model. A prior is the knowledge we already have about different models. If we know that one model \( \theta^{(1)} \) is more probable than another model \( \theta^{(2)} \), we should take a prior \( P(\theta^{(1)}) > P(\theta^{(2)}) \). If we do not know anything about the models, one uses a uninformative prior, for example, the uniform distribution. For a further discussion of prior selection we refer to the appendix 3.4.1. There, we show that one can use the conjugate distribution of the likelihood in order to ensure that the integral over the parameters can be solved analytically. The conjugate of the Bernoulli-distribution is the \( \beta \)-distribution. Using it as uninformative prior in our example, we can solve (2.73) analytically. It results in the \( \beta \)-distribution with parameters \( \alpha_h \), and \( \alpha_t \):\(^2\)

\[
P(\theta|x) = \beta(\theta|\alpha_h + h, \alpha_t + t) \tag{2.76}
\]

The Bernoulli-model is a simple model as its parameter \( \theta \) consists of only one value. A standard model with two parameters is the normal distribution \( \mathcal{N}(\mu, \sigma^2) \) with the parameters \( \mu \) and \( \sigma^2 \). Of course, there also exist a lot of more sophisticated models with multiple parameters.

Multinomial Model  The binomial model would be sufficient if we discretize the gene expression values to zero and one. As we have stated earlier that we want to use more than two states for a gene, we have to use the multinomial model. In this case a random variable can have \( r \) different values. Let \( X \) be a random variable with \( X = x \in \{x_1, \ldots, x_r\} \). We define \( P_{\theta}(X = x_k) := \theta_k \) with \( \theta = (\theta_2, \ldots, \theta_r) \in \Theta = [0,1]^{r-1} \) and \( \theta_1 = 1 - \sum_{k=2}^r \theta_k \). With \( N_k \) denoting the number of events where \( X = x_k \), we can express the multinomial distribution as

\[
P(x|\theta) = \frac{(\sum_{k=1}^r N_k)!}{\prod_{k=1}^r N_k!} \prod_{k=1}^r \theta_k^{N_k} \tag{2.77}
\]

Using a multinomial distribution, the conjugate distribution is given by the Dirichlet-distribution \( \mathcal{D}(\alpha_1, \ldots, \alpha_r) \) which is shown in the appendix 3.4.2. Using this prior we can calculate the posterior analytically. We get the Dirichlet distribution with parameters \( \alpha_k \) and \( N_k \) as counts of \( X = x_k \):\(^3\)

\(^2\)See appendix 3.4.1 for a derivation.

\(^3\)See appendix 3.4.2 for a derivation.
\[ P(\theta|x) = \mathcal{D}(\theta|\alpha_1 + N_1, \ldots, \alpha_r + N_r) \] (2.78)

Again, we can easily compute the integral analytically as we have used the conjugate prior.

### 2.5.2 Multivariate Case

So far, we have considered only one random variable. As a genetic regulatory network consists of more than one gene, we have to extend our framework to more than one random variable. We have seen that we want to maximize \( P(\lambda|x) \). It is apparent that we can get the maximum of this term by computing \( P(\lambda|x) \) for each structure \( \lambda \). Then we choose structure \( \lambda^* \) with

\[ \lambda^* = \arg \max_{\lambda} P(\lambda|x) \] (2.79)

as best structure. Thus, \( \lambda^* \) contains the dependencies between the random variables, which are the regulatory relations between the genes. To retrieve \( \lambda^* \), we have to show how to compute \( P(\lambda|x) \). We have already shown that this can be computed via Bayes’ theorem with integrating over the parameters \( \theta \):

\[ P(\lambda|x) = \int_{\Theta} P(x|\theta, \lambda) \cdot P(\theta|\lambda) \cdot d\theta \cdot P(\lambda) P(x) \] (2.80)

We can compute \( P(x|\theta, \lambda) \) by the joint distribution of \( X \) as \( x \) contains realizations of \( X \).

**Factorization** Firstly, we show a normal representation of a joint distribution. Then we develop the representation of the factorized joint distribution. As an example, we take three binary random variables \( X, Y, \) and \( Z \). To represent the joint probability distribution \( P(X, Y, Z) \) without any independencies we need \( n_\theta = 2^3 - 1 = 7 \) parameters \( \theta = \{\theta_1, \ldots, \theta_7\} \) as displayed in table 2.1. This is due to the fact that the network has 8 different states \( \{000, 001, \ldots, 111\} \). One triple corresponds to the value of \( xyz \). The probability of the last state can be obtained from the others:

\[ \theta_8 = 1 - \sum_{i=1}^{7} \theta_i \]

In general the formula for the number of parameters is \( 2^n - 1 \) and asymptotically \( O(2^n) \) for \( n \) binary random variables.

<table>
<thead>
<tr>
<th>( X = x, Y = y, Z = z )</th>
<th>( y = 0, z = 0 )</th>
<th>( y = 0, z = 1 )</th>
<th>( y = 1, z = 0 )</th>
<th>( y = 1, z = 1 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( x = 0 )</td>
<td>( \theta_1 )</td>
<td>( \theta_2 )</td>
<td>( \theta_3 )</td>
<td>( \theta_4 )</td>
</tr>
<tr>
<td>( x = 1 )</td>
<td>( \theta_5 )</td>
<td>( \theta_6 )</td>
<td>( \theta_7 )</td>
<td>( 1 - \sum_{i=1}^{7} \theta_i )</td>
</tr>
</tbody>
</table>

**Table 2.1:** Representation of the joint distribution of three binary random variables \( X, Y, \) and \( Z \).

The number of parameters grows exponentially. This makes computations for many random variables intractable. A solution is to factorize the joint probability distribution according to Bayes’ theorem and to apply known independencies afterwards. One factorization for \( X, Y, \) and \( Z \) is:

\[ P(X, Y, Z) = P(Z|X, Y) \cdot P(X, Y) = P(Z|X, Y) \cdot P(Y|X) \cdot P(X) \] (2.81)

The factorization is done simply by using Bayes’ theorem multiple times. The result depends on the ordering of variables. Each term in the factorization can be represented as a table where we have a parameter for each state of all variables of the term. Anyhow, some of these parameters can be calculated from the others. How many parameters do we need by representing the joint
distribution factorized? We answer this question by an example with two variables \( X, Y \). The factorization is one application of Bayes’ theorem:

\[
P(X, Y) = P(X) \cdot P(Y|X)
\]

We know that we need \( n_\theta = 2^2 - 1 = 3 \) parameters for the direct representation. We expect to need the same number of parameters for the factorized form as we have not made any further assumptions. Thus the intuitive tables in table 2.2 seem to be over-determined because we retrieve four different parameters. To investigate this, we create the joint distribution table on the basis of the factorization by using (2.82) which is shown in table 2.3. As we know that the sum of the table has to be one, we can also calculate the last cell of the joint distribution table by subtracting the values of all other cells from 1. Thus we have two different terms for the last cell with the same value such that we can equate them. Now we can solve for \( \theta_4 \) so that we retrieve:

\[
(1 - \theta_1) \cdot (1 - [\theta_2 + \theta_3 + \theta_4]) = 1 - [\theta_1 \cdot \theta_2 + \theta_3 \cdot (1 - \theta_1) + \theta_1 \cdot \theta_4]
\]

\[
\theta_4 = \frac{2 \cdot \theta_1 \cdot \theta_2 - \theta_1 - \theta_2}{1 - 2 \cdot \theta_1}
\]

Table 2.2: Intuitive representation of the factorized joint distribution

<table>
<thead>
<tr>
<th>( P(X) )</th>
<th>( x = 0 )</th>
<th>( x = 1 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \theta_1 )</td>
<td>1 - ( \theta_1 )</td>
<td></td>
</tr>
<tr>
<td>( P(Y</td>
<td>X) )</td>
<td>( x = 0 )</td>
</tr>
<tr>
<td>( y = 0 )</td>
<td>( \theta_2 )</td>
<td>( \theta_4 )</td>
</tr>
<tr>
<td>( y = 1 )</td>
<td>( \theta_3 \cdot (1 - \theta_1) )</td>
<td>( 1 - \sum_{i=2}^{4} \theta_i )</td>
</tr>
</tbody>
</table>

Table 2.3: Joint Distribution from factorized parameterization

After this transformation we only need three parameters for representing the joint distribution. Thus, factorization does not change the number of parameters we need. As previously mentioned, if we have some independencies, the number of parameters is reduced. This is shown next. Preliminarily, we offer a formula for the computation of the number of parameters \( n_\theta \) of the joint distribution in factorized form. We have to sum over the exponentials to the basis of two of the number of variables in each term minus one. As said before, we retrieve in the above example with three random variables:

\[
n_\theta = 2^3 - 1 + 2^2 - 1 + 2^1 - 1 = \sum_{i=0}^{2} 2^i = 2^3 - 1 = 7
\]

**Conditional Independence** Now we assume an independency structure for the three random variables \( X, Y, \) and \( Z \) as given in figure 2.8. As we see, the random variable \( Y \) depends on \( X \), and \( Z \) depends on \( Y \). As \( Z \) does not depend directly on \( X \), we can state that it is not dependent on \( X \) if \( Y \) is given. We denote this \( Z \perp X|Y \) and call it that \( Z \) is conditionally independent of \( X \) given \( Y \). With the mathematical definition of conditional independence \( P(Z|X, Y) = P(Z|Y) \), we can substitute the first term in (2.81) and retrieve:
2.5. BAYESIAN NETWORKS

\[ P(X, Y, Z) = P(Z|Y) \cdot P(Y|X) \cdot P(X) \]  \hspace{1cm} (2.85)

Figure 2.8: Independency structure of three random variables.

With the formula given above, we can compute \( n_{\theta} \):

\[ n_{\theta} = 2^{2-1} + 2^{2-1} + 2^{1-1} = 5 \]  \hspace{1cm} (2.86)

In this example with only a few random variables we need two parameters less than we would without an independency structure. The reduction is more dramatic if we have a larger number of random variables. It can be shown that we need only \( O(n \cdot 2^k) \) parameters with \( k \) equal to the maximum indegree of a node instead of \( O(2^n) \) parameters. Normally we have \( n \gg k \) because no node is connected directly to a large part of the other nodes. This linearity in the number of nodes makes the computations tractable.

**Equivalence Classes** Intuitively, it makes sense to represent the independency structure in a graph as shown in figure 2.8. This gives us the definition of a Bayesian Network. A Bayesian Network is a tuple of a joint probability distribution of \( n \) random variables and a directed acyclic graph (DAG) with \( n \) nodes. Each node corresponds to one random variable. This formalization allows us to write the joint probability of a subset of variables \( U \subseteq X = \{X_1, \ldots, X_n\} \) in general as

\[ P(U) = \prod_{X \in U} P(X|\pi(X)) \]  \hspace{1cm} (2.87)

using the conditional independencies where \( \pi(X) \) denotes the parents of node \( X \). This transforms global problems considering the whole joint distribution to local problems considering only one random variable and its parents. This is an essential feature of Bayesian networks.

As mentioned above, the factorization of the joint distribution is not unique. This leads to equivalence classes, which are sets of DAGs that have the same factorization of the joint distribution. This affects learning in such a way that we cannot differentiate between DAGs of one equivalence class. To make this clearer, four equivalence classes are shown in figure 2.9. Now we compute the factorization of the joint probability distribution for each DAG from left to right and top to bottom:

(i)

\[ P(X, Y, Z) = P(Y|X) \cdot P(X) \cdot P(Z|Y) \]  \hspace{1cm} (2.88)

(ii)

\[ P(X, Y, Z) = P(X|Y) \cdot P(Y) \cdot P(Z|Y) = P(Y|X) \cdot P(X) \cdot P(Z|Y) \]  \hspace{1cm} (2.89)

(iii)

\[ P(X, Y, Z) = P(X|Y) \cdot P(Y|Z) \cdot P(Z) = P(Y|X) \cdot P(X) \cdot P(Z|Y) \]  \hspace{1cm} (2.90)

(iv)

\[ P(X, Y, Z) = P(Y|X, Z) \cdot P(X) \cdot P(Z) \]  \hspace{1cm} (2.91)

(v)

\[ P(X, Y, Z) = P(Z|X, Y) \cdot P(X) \cdot P(Y) \]  \hspace{1cm} (2.92)
### Reference Equations

\[(vi)\]

\[
P(X, Y, Z) = P(X | Z, Y) \cdot P(Z) \cdot P(Y)
\]  

(2.93)

The last step in (2.89) is done by applying \( P(X | Y) \cdot P(Y) = P(Y | X) \cdot P(X) \) and similarly in (2.90) with \( X \) being replaced by \( Z \). Thus we see that the first three equations (2.88) to (2.90) are equal and therefore their DAGs are within one equivalence class. The remaining equations cannot be transformed to each other so that each of the DAGs has its own equivalence class. One can identify equivalence classes graphically by comparing v-structures. A v-structure consists of three nodes where two of the nodes, which have to be unrelated, have a directed edge from itself to the third node. All DAGs in one equivalence class have the same v-structures and the same underlying skeleton (see [Verma & Pearl, 1990]). The skeleton is the graph with the directed edges replaced by undirected edges.

#### Interventions

We have stated that we cannot differentiate the structures within equivalence classes given realizations of the joint distribution. Amongst other things we cannot differentiate between \( X \rightarrow Y \) and \( Y \rightarrow X \). As \( X \) and \( Y \) are genes in our framework, we can easily imagine that in principle it is possible to knock-out \( X \). If the knock-out influences the expression of \( Y \), it is obvious that \( X \) regulates \( Y \) and not the other way around. This is equivalent to the directed edge from \( X \) to \( Y \): \( X \rightarrow Y \). Thus, knock-outs or more generally data generated by intervention experiments can be used to differentiate within equivalence classes (see [Tian & Pearl, 2001]).

How does this fit in our framework? The implementation of interventions in our statistical model is straightforward as shown by [Yoo & Cooper, 2003]. As the realization of an intervened variable is determined, we set the probability to one. Additionally, we have to remove all incoming edges because they do not have any influence on the realization of the intervened variable as it is determined. This results in a different factorization of the joint distribution, and therefore this joint distribution is no longer member of the former equivalence class. If we would be able to perform experiments with interventions for each subset of genes, we could eliminate all equivalence classes. Of course, in practise this is not possible. Nevertheless, including intervention data results in smaller equivalence classes. This achieves a better prediction of the genetic regulatory network.

#### 2.5.3 Learning

So far, we have seen how to present a joint probability and got familiar with the conditional independence as the basic network property. We have to compute \( P(x | \theta, \lambda) \) to estimate the best structure \( \lambda^* \). Firstly, we assume a given structure \( \lambda \) and show parameter learning with complete data. Learning with incomplete data is a harder problem (see [Pearl, 1995]) but as we do not have missing data there is no need to introduce it here. Finally, we learn the structure \( \lambda^* \) using the results from parameter learning.
2.5. BAYESIAN NETWORKS

We have given some data \( x \) with observations of the joint probability. Let \( \lambda \) denote a DAG with \( n \) nodes where edges indicate dependence relations between the corresponding random variables of the joint probability \( P(X = \{X_1, \ldots, X_n\}) \). This time we consider multinomial random variables with \( X_i = x_k \in \{x_1, \ldots, x_{r_i}\} \), which means that \( X_i \) has \( r_i \) possible values. As seen above and stated in (2.87), each random variable \( X_i \) depends only on its parents \( \pi(X_i) \)

\[
P(X|\theta, \lambda) = \prod_{i=1}^{n} P(X_i|\pi(X_i), \theta_i, \lambda)
\]

where \( \theta = (\theta_1, \ldots, \theta_n) \) is the vector of all parameters and \( \theta_i \) denotes the parameters of the distribution \( P(X_i|\pi(X_i), \theta_i, \lambda) \). The parameter \( \theta_i \) is a matrix \( \theta_i = (\theta_{ijk})_{j=1}^{q_i} \) with \( q_i = \prod_{X_j \in \pi(X_i)} r_j \). Thus, \( q_i \) is the number of different configurations of the parents of node \( X_i \). For each of these configurations we need a parameter describing the probability that \( X_i = x \) which leads to \( r_i \) parameters times \( q_i \). As we can express one parameter by 1 minus the sum of the others, we need only \( q_i \cdot (r_i - 1) \) parameters, which we store in the matrix \( \theta_i \). If we denote that the parents \( \pi(X_i) \) of \( X_i \) are in configuration \( j \) with \( \pi_j(X_i) \), we get:

\[
P(X_i = x_k|\pi_j(X_i), \theta_i, \lambda) = \theta_{ijk}
\]

For convenience, we define vector of parameters \( \theta_{ij} = (\theta_{ij1}, \ldots, \theta_{ijr_i}) \). Before we go over to the learning task, we need to make the assumption of parameter independence introduced by [Spiegelhalter & Lauritzen, 1990]:

\[
P(\theta|\lambda) = \prod_{i=1}^{n} \prod_{j=1}^{q_i} P(\theta_{ij}|\lambda)
\]

Parameter Learning

Assuming that data \( x \) is complete, we can write the posterior applying (2.96) as:

\[
P(\theta|x, \lambda) = \prod_{i=1}^{n} \prod_{j=1}^{q_i} P(\theta_{ij}|x, \lambda)
\]

The advantage of parameter independence is that we can update each parameter separately. We assume as prior for \( \theta_{ij} \) the Dirichlet distribution, and therefore we obtain analogously to the one dimensional case as posterior a product of Dirichlet distributions with parameters \( N_{ijk} \) denoting the number of events where \( X_i = x_k \) and the parents \( \pi(X_i) \) of \( X_i \) are in the configuration \( \pi_j(X_i) \).

\[
P(\theta|x, \lambda) = \prod_{i=1}^{n} \prod_{j=1}^{q_i} P(\theta_{ij}|x, \lambda)
\]

\[
= \prod_{i=1}^{n} \prod_{j=1}^{q_i} \mathcal{D}(\theta_{ij}|\alpha_{ij1} + N_{ij1}, \ldots, \alpha_{ijr_i} + N_{ijr_i})
\]

Now we can easily estimate \( \theta_{ijk} \) with the expectation value with respect to the posterior distribution:

\[
\hat{\theta}_{ijk} = \frac{E_{P(\theta_{ij}|x, \lambda)} \theta_{ijk}}{\alpha_{ij} + N_{ij}}
\]
In (2.99) we use the fact of parameter independence from (2.96). The last step is performed analogously to (3.61).

**Structural Learning**

In the beginning of this section we have already stated that we want to retrieve the independency structure of the random variables. In (2.79) we have defined the estimator of the best structure as \( \lambda^* = \arg \max_{\lambda} P(\lambda|x) \). Therefore, we have to iterate over all \( \lambda \in \{\lambda_1, \ldots, \lambda_m\} \) to compute the posterior

\[
P(\lambda|x) = \frac{P(x|\lambda) \cdot P(\lambda)}{P(x)} \propto P(x|\lambda) \cdot P(\lambda)
\]

(2.100)

Since \( P(x) \) is independent of the chosen structure \( \lambda \) we can use the proportionality. Without prior knowledge of the structure, we can use the uniform distribution. Therefore, we can neglect \( P(\lambda) \). Using the Dirichlet distribution as parameter prior, we can analytically solve \( P(x|\lambda) \) in (2.100):

\[
P(x|\lambda) = \int P(x|\theta, \lambda) \cdot P(\theta) \, d\theta \tag{2.101}
\]

(2.101)

\[
= \int \frac{n!}{\prod_{i=1}^{r_i} N_i!} \prod_{i=1}^{r_i} \theta_{ij}^{N_{ijk}} \cdot \frac{\Gamma \left( \sum_{k=1}^{r_i} \alpha_{ijk} \right)}{\prod_{k=1}^{r_i} \Gamma (\alpha_{ijk})} \prod_{k=1}^{r_i} \theta_{ijk}^{\alpha_{ijk}-1} \, d\theta_{ij}
\]

After expanding with \( \theta \), we reduce the global problem to local ones. We compute the probability for each node \( (i = 1, \ldots, n) \), for each configuration of the parent nodes \( (j = 1, \ldots, q_i) \), and for each state of the actual node \( (k = 1, \ldots, r_i) \) individually. Next, we shift some terms out of the integral. Using (3.59) in the appendix 3.4.2 on page 46, we can solve the integral. Finally, we reorganize the equation.

Applying (2.101) to each \( \lambda \in \{\lambda_1, \ldots, \lambda_m\} \), we can estimate the best structure. As \( \{\lambda_1, \ldots, \lambda_m\} \) contains all DAGs with the same number of nodes as the network, this set is huge and make the computations intractable. After clarifying this statement, we present MCMC as approximation algorithm.

**Number of Parameters**

We have introduced the concept of conditional independencies. This shrinks the number of parameters dramatically from exponential to linear with respect to the number of nodes. Actually, we do not reduce the number of parameters but we divide the set of parameters into distributional parameters \( \theta \in \Theta \) and the independency relations \( s \in S \) such that the dimension of \( \Theta \) becomes small. This is done by making the dimension of \( \Theta \) linear in the number of nodes \( \dim(\Theta) = O(n \cdot r^k) \) with \( r \) as the number of states of a node. This holds only under the assumption that the indegree \( k \) of a node (the number of dependency relations) is much smaller than the number of nodes \( n \): \( n \gg k \). In return, the set of different independency relations \( S \) is exponential in the number of nodes. [Robinson, 1973] has invented a formula which computes the number of different DAGs with respect to the number of nodes:
2.6. MONTE-CARLO-MARKOV-CHAIN

\[ G(0) = 1 \]
\[ G(n) = \sum_{i=1}^{n} (-1)^{i-1} \binom{n}{i} \cdot 2^{i(n-i)} \cdot G(n-i) \]

For some numbers of nodes the results are shown in Table 2.4. There we can see that for ten or more nodes it is intractable to compute the posterior for each \( \lambda \). According to [Chickering, 1996], the exponential increase of the number of DAGs makes the computation NP-complete. We can deal with the problem by using an approximation algorithm.

<table>
<thead>
<tr>
<th>( n )</th>
<th>( m = G(n) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>25</td>
</tr>
<tr>
<td>4</td>
<td>543</td>
</tr>
<tr>
<td>5</td>
<td>29281</td>
</tr>
<tr>
<td>10</td>
<td>( &gt; 10^{18} )</td>
</tr>
</tbody>
</table>

Table 2.4: Number of different DAGs with respect to the number of nodes.

2.6 Monte-Carlo-Markov-Chain

2.6.1 Motivation

The problem is to compute the posterior \( P(\lambda|x) \) for each \( \lambda \in \{\lambda_1, \ldots, \lambda_m\} \), which we call the search space. We use a MCMC approach which traverses the search space by some representative elements. Aside from this approach, there are also other methods which i.e. exclude uninteresting elements from the search space (see [Friedman et al., 1999] for an example).

We want to approximate the best structure. Each structure \( \lambda \) is a DAG which we can represent as an adjacency matrix \( A \in \{0, 1\}^{n \times n} \). An entry \( a_{ij} = 1 \) indicates an edge from node \( i \) to node \( j \). As the represented graph has to be acyclic, it is obvious that amongst others the diagonal has to be zero \( a_{ii} = 0 \) \( \forall i \). We are estimating the best structure by taking the maximum of the posterior over all structures \( \lambda \in \{\lambda_1, \ldots, \lambda_m\} \). If we cannot consider all \( \lambda \), it could happen that we miss the best structure. Thus, it is better to compute the probability of an edge \( a_{ij} \) leading to the adjacency matrix \( A \in [0, 1]^{n \times n} \). Instead of taking the argument maximum of the posterior, we compute the expectation according to the posterior:

\[ \lambda^* = E_{P(\lambda|x)} \lambda \]

2.6.2 The Algorithm

At this step, we can use MCMC. Monte-Carlo draws samples from a distribution to get an average of samples to approximate the expectation. MCMC is Monte-Carlo based on a Markov chain developed by [Metropolis et al., 1953, Hastings, 1970]. The constructed framework was reviewed by [Chib & Greenberg, 1995]. We estimate (2.104) by averaging samples \( \{\lambda^{(\tau)}, \tau = 1, \ldots, t\} \) from the distribution \( P(\lambda|x) \) where convergence is assured by the Strong Law of Large Numbers:

\[ E_{P(\lambda|x)} \lambda \approx \frac{1}{t} \sum_{\tau=1}^{t} \lambda^{(\tau)} \]
The samples are drawn according to a Markov chain which generally comprises a state space \( \{ \lambda_1, \ldots, \lambda_m \} \) where \( m = G(n) \) is the number of DAGs, an initial distribution over the states \( h_0(\lambda) \), and a transition matrix \( \mathcal{H} \). Each state represents a DAG, the initial distribution gives the probability for each state respectively DAG to be the initial state. The goal is to choose the transition matrix which gives us the probability \( P_\mathcal{H}(\Lambda_{\tau+1} = \lambda_i|\Lambda_{\tau} = \lambda_j) = h_{ji} \) to go from state \( \lambda_j \) to state \( \lambda_i \) in one time step. The chosen transition matrix must ensure that the visited states (the chain) denoted by \( \{ \lambda^{(\tau)} \} \tau = 1, \ldots, t \} \) are distributed according to the posterior \( P(\lambda|x) \). As we set the initial distribution to the uniform distribution, we ignore the first states in the chain. The number of ignored states is called burn-in. After the burn-in, the visited states have to be independent of the initial state. The distribution of states independent of the initial distribution after a long term run is called stationary distribution \( h_\infty(\cdot) \) of the Markov chain. It is defined by:

\[
P_\mathcal{H}(\Lambda_{\tau} = \lambda_i|\Lambda_0 = \lambda_j) \rightarrow h_\infty(\lambda_i), \quad (\tau \rightarrow \infty)
\]  

(2.106)

There are some properties which a Markov chain has to fulfil to have a unique stationary distribution, explained in more detail in [Gilks et al., 1996]:

- **Irreducibility:** The chain has nonzero probabilities to move from one state to any other state in a finite number of steps.
- **Aperiodicity:** The maximum common divider of the number of steps necessary to return to any starting point is equal to one. As we assume an irreducible Markov chain, it is proven that each state has the same period. Thus, if the probability to stay in one state is nonzero for at least one state, the Markov chain is aperiodic.

We can construct such a Markov chain with a stationary distribution equal to the posterior distribution \( h_\infty(\lambda) = P(\lambda|x) \) according to [Hastings, 1970], which is a generalization of the first proposed method by [Metropolis et al., 1953]. A candidate \( \lambda_i \) is drawn from another Markov chain with the same state space and the symmetric transition matrix \( \mathcal{Q} \) with elements \( q_{ij} = q_{ji} \). We accept the candidate \( \lambda_i \) as new state \( \lambda^{(\tau+1)} \) with probability \( \alpha(j, i) \) where

\[
\alpha(j, i) = \min \left( 1, \frac{P(\lambda_i|x) \cdot q_{ji}}{P(\lambda_j|x) \cdot q_{ij}} \right).
\]  

(2.107)

This means that the candidate is always accepted when its posterior probability is higher than the posterior probability of the actual state. In the other case, it is accepted proportionally to the ratio of the posterior probabilities. If we reject the candidate, we set \( \lambda^{(\tau+1)} = \lambda^{(\tau)} \). This intuitively explains how the method works: DAGs with a high posterior are drawn with a higher probability, and therefore they occur more often in the chain. The possibility of a rejection implies the aperiodicity of the Markov chain, which we use in the next subsection to show that the constructed Markov chain converges to the posterior distribution.

We finish with a summary of the developed Metropolis-Hastings-algorithm:

1. Take a DAG \( \lambda^{(\tau=0)} = \lambda_i \) and calculate its probability \( P(\lambda_i|x) \).
2. Modify DAG \( \lambda_i \) by removing, swapping, or adding an edge according to some probabilities: \( \lambda_i \rightarrow \lambda_j \).
3. Accept new DAG \( \lambda_j \) with probability \( \min \left( 1, \frac{P(\lambda_j|x)}{P(\lambda_i|x)} \right) \).
4. Increment \( \tau \) by one. Go to first step with \( \lambda^{(\tau)} := \lambda_j \) if new DAG was accepted. Otherwise set \( \lambda^{(\tau)} := \lambda_i \).

After an initial number of DAGs as burn-in, we obtain DAGs which are a representative set of all DAGs according to the posterior. To obtain the expectation value of these graphs, one simply takes the mean for each edge of all DAGs. The resulting graph which is not necessarily acyclic is the estimator \( \lambda^* \) for the best structure explaining the data \( x \). Thus, \( \lambda^* \) estimates the genetic regulatory network based on microarray data.
2.6.3 Proof of Convergence

Here, we prove the convergence of the Markov chain developed above. The acceptance rate of (2.107) can be generalized to the form

\[ \alpha(j, i) = \frac{\kappa_{ji}}{1 + \frac{P(\lambda_j|x) \cdot q_{ji}}{P(\lambda_i|x) \cdot q_{ij}}} \]  

(2.108)

with \( \kappa_{ji} = \kappa_{ij} \), and \( 0 \leq \alpha(j, i) \leq 1 \). We get (2.107) by setting

\[ \frac{P(\lambda_j|x) \cdot q_{ji}}{P(\lambda_j|x) \cdot q_{ij}} \geq 1 \Rightarrow \kappa_{ji} = 1 + \frac{P(\lambda_j|x) \cdot q_{ji}}{P(\lambda_i|x) \cdot q_{ij}}, \]  

(2.109)

\[ \frac{P(\lambda_j|x) \cdot q_{ji}}{P(\lambda_j|x) \cdot q_{ij}} < 1 \Rightarrow \kappa_{ji} = 1 + \frac{P(\lambda_j|x) \cdot q_{ij}}{P(\lambda_i|x) \cdot q_{ji}}. \]  

(2.110)

With the generalized form in (2.108) it can be easily proven that such a transition leads to the equality of the stationary and the posterior distribution. The elements \( h_{ij} \) of the transition matrix \( \mathcal{H} \) can be written with (2.108) as

\[ h_{ij} = P_{\mathcal{H}}(\Lambda_{r+1} = \lambda_i | \Lambda_r = \lambda_j) = q_{ji} \cdot \alpha(j, i) + 1_{i=j} \cdot \left[ 1 - \sum_{k=1}^{m} q(j, k) \cdot \alpha(j, k) \right] \]  

(2.111)

where \( 1_{i=j} \) denotes the identity function. It is one when \( i = j \), and otherwise zero. The first term is the successful probability of acceptance for a new candidate. The second term is the rejection probability for all possible candidates. Next, we show that the Detailed Balance Equation \( P(\lambda_i|x) \cdot h_{ij} = P(\lambda_j|x) \cdot h_{ji} \) holds, and we conclude from this that \( P(\lambda|x) \) is the stationary distribution. The Detailed Balance Equation obviously holds for \( i = j \). Thus, we assume \( i \neq j \):

\[ P(\lambda_i|x) \cdot h_{ij} = P(\lambda_i|x) \cdot \frac{q_{ij} \cdot \kappa_{ij}}{1 + \frac{P(\lambda_i|x) \cdot q_{ij}}{P(\lambda_j|x) \cdot q_{ji}}}, \]  

(2.112)

\[ = P(\lambda_j|x) \cdot q_{ji} \cdot \frac{P(\lambda_i|x) \cdot q_{ij} \cdot \kappa_{ij}}{P(\lambda_j|x) \cdot q_{ji} + P(\lambda_i|x) \cdot q_{ij}} \]  

\[ = P(\lambda_j|x) \cdot q_{ji} \cdot P(\lambda_i|x) \cdot q_{ij} + 1 + 1 \]  

\[ = P(\lambda_j|x) \cdot h_{ji} \]

The first step is done by substituting \( h_{ij} \) with (2.111). Next, we shift the denominator of the denominator \( P(\lambda_j|x) \cdot q_{ji} \) in front of the term. We do the opposite with \( P(\lambda_i|x) \cdot q_{ij} \), and use the symmetry of \( \kappa \). Then we can substitute with (2.111), again, and retrieve the Detailed Balance Equation. If we assume \( P_{\mathcal{H}}(\Lambda_r = \lambda_j) = P(\lambda_j|x) \) we get the invariance which means that after one sample is obtained from the posterior distribution, all succeeding samples will be from the posterior distribution as well:
\begin{equation}
    P_H(\Lambda_{\tau+1} = \lambda_i) = \sum_{j=1}^{m} P_H(\Lambda_{\tau+1} = \lambda_i, \Lambda_{\tau} = \lambda_j) \quad (2.113)
\end{equation}

\begin{align*}
    &\quad = \sum_{j=1}^{m} P_H(\Lambda_{\tau+1} = \lambda_i | \Lambda_{\tau} = \lambda_j) \cdot P_H(\Lambda_{\tau} = \lambda_j) \\
    &\quad = \sum_{j=1}^{m} h_{ji} \cdot P(\lambda_j | \mathbf{x}) = \sum_{j=1}^{m} h_{ij} \cdot P(\lambda_i | \mathbf{x}) \\
    &\quad = P(\lambda_i | \mathbf{x}) \cdot \sum_{j=1}^{m} h_{ij} = P(\lambda_i | \mathbf{x})
\end{align*}

After using the law of total probability, we substitute \( P_H(\Lambda_{\tau} = \lambda_j) \) with \( P(\lambda_j | \mathbf{x}) \) and apply the Detailed Balance Equation from (2.112). Then we can shift \( P(\lambda_j | \mathbf{x}) \) out of the sum, and get the sum of the transition probabilities from \( \lambda_i \) to any other state which is apparently one. As the Markov chain is assumed to be irreducible, and we have seen that it is aperiodic additionally, the Perron-Frobenius theorem states that under these assumptions the invariant distribution is the stationary where the chain converges to (for the proof see [Seneta, 1981]).

### 2.7 Method Validation - ROC curves

In the last section, we’ve learned something about Bayesian Networks and MCMC. Especially in MCMC, there are several parameters to be defined like size of burnin, number of steps, number of restarts, and so on. How does one assess the correct choice of these parameters? Or, in more general, how can we compare different models/predictors? Normally, one can have a look at the number of correct and incorrect predictions if we already know the true model (which is appropriate to assume if we do model assessment or comparison). Anyway, how can we deal with the normal situation that the number of correct predictions depend on a parameter (i.e. a threshold)? One possibility is the use of ROC curves.

In our case of network comparisons, we compare the retrieved DAGs. As we get a probability for each each of the resulting DAG, we have to decide the probability an edge has to have at least to be predicted. This is the threshold.

ROC curves supply a visualized overview of the performance of a method. The performance of an estimator can be tested according to its sensitivity and specificity. We consider only the skeleton of DAGs because otherwise equivalence classes make it difficult to decide whether an edge is correct or not. To get the undirected graph, we sum up the probabilities of both directions for an edge and divide them by two. Thus we retrieve a graph where each edge has a probability. We can compute sensitivity and specificity given a threshold \( \varepsilon \) to remove all edges which have a lower probability than \( \varepsilon \).

Next, we have to compute the difference between the predicted genetic regulatory network and the real genetic regulatory network. Thereby the latter ones are known because of published biological evidence in the case of our microarray data and because of the given structure of the simulated genetic regulatory network. Based on this, we can compute four values which describe the power of our prediction:

- True Positives (TP): The number of predicted edges which corresponds to an edge in the true model.

\footnote{ROC means Receiver Operator Characteristics and is a method from the research field of signal detection theory. The name comes from the development during the second world war where radar operators were used to detect enemy targets. The ability of a radar operator to distinguish between enemy targets, friendly ships, and noise was called the Receiver Operator Characteristic.}
2.7. METHOD VALIDATION - ROC CURVES

- False Positives (FP): The number of predicted edges which do not correspond to an edge in the true model.
- False Negatives (FN): The number of missing edges which corresponds to an edge in the true model.
- True Negatives (TN): The number of missing edges corresponding to a missing edge in the true model.

The sensitivity is the ratio of correctly identified edges to all identified edges. Specificity is the ratio of correctly predicted missing edges to all missing edges. Generally, there is a trade-off between sensitivity and specificity. A ROC curve visualizes this trade-off by plotting the sensitivity against the complementary specificity. The complementary specificity which is the proportion of erroneously predicted missing edges is taken to get a monetary increasing function. We can compute the needed values by:

\[
\text{Sensitivity} = \frac{TP}{TP + FN} \quad (2.114)
\]

\[
\text{Specificity} = \frac{TN}{TN + FP} \quad (2.115)
\]

Complementary specificity \( = 1 - \text{Sensitivity} = 1 - \frac{TN}{TN + FP} \)

\[
= \frac{TN + FP - TN}{TN + FP} = \frac{FP}{TN + FP} \quad (2.116)
\]

Figure 2.10 shows several ROC curves. The diagonal is the expected ROC curve from a random predictor. The bigger the area between the ROC curve of a model to the diagonal, the better the model is. A ROC curve which quickly raises to one has a very high sensitivity and a high specificity. The corresponding model is a good choice. A last remark: If the threshold cannot be shifted continuously from the lower left to the top right corner, it is highly recommended to plot points without connections instead of a line.

![ROC plot](image-url)

Figure 2.10: Example of some ROC curves. Model A is not as good as model B. Model C is theoretical and shows an nearly optimal model.
Chapter 3

Appendix

3.1 Quantile Normalization

The normalization in $T$ is done by quantile normalization (see [Bolstad et al., 2003]). Quantile normalization assumes that there is an underlying common distribution of intensities across chips. In figure 3.1A we have plotted three densities. The densities of $X$ and $Y$ are equal, and $X$ and $Z$ are unequal. The boxplot in the figure underlines the dissimilarity between $X$ and $Y$ to $Z$. In the qqplot (see figure 3.1C) which plots the sorted list of realizations of the distributions $X^{(s)}$ and $Y^{(s)}$. We denote $(X^{(s)}_i, Y^{(s)}_i)$ as the $i$th quantile. We see in the plot, that the quantiles of similar distributions lie upon the diagonal. Figure 3.1D shows that the quantiles for dissimilar distribution do not lie upon the diagonal.

![A: Densities](chart1.png)

![B: Boxplot](chart2.png)

![C: Same Distributions](chart3.png)

![D: Different Distribution](chart4.png)

Figure 3.1: Analysis of three distributions

Normally, we have more than two chips say $d$ which results in $d$ distributions we want to normalize. We organize the data in a matrix $X = (x_{i,j})_{i=1,...,n} \in \mathbb{R}^{n \times d}$ such that one row of the column-sorted matrix $X$ is a quantile $q_i = (q_{i1}, \ldots, q_{id})$. We have to imagine a $d$-dimensional qqplot for the comparison of the distributions. If we assume the realizations come from a common
underlying distribution, we then assume that the quantiles lie on the diagonal in the $d$-dimensional. Thus, for normalization we project the quantiles onto the unit diagonal $d = (1/\sqrt{d}, \ldots, 1/\sqrt{d}) \in \mathbb{R}^d$, which is equivalent to substituting each quantile by the average of the quantiles:

$$\text{proj}_d q_i = \frac{q_i \cdot d^T}{d \cdot d^T} \cdot d = \frac{1}{d} \left( \sum_{j=1}^{d} q_{ij}, \ldots, \sum_{j=1}^{d} q_{ij} \right) \quad (3.1)$$

Finally we have to put the values of the sorted matrix back to their original places. We can test the result by producing a qqplot of the normalized data, but it is apparent that we will see all quantiles on the diagonal as we have constructed it.

An artefact of this normalization is that a gene whose expression values respectively the corresponding probe values are always high but not equal will have a lower variance. This is due to the fact that quantile normalization shrinks differences for high values over proportionally.

### 3.2 Some Maths for VSN

#### Variances

Let $X$ and $Y$ denote random variables, then we get as the variance of the sum:

$$\text{Var}(X + Y) = E[X + Y - E(X + Y)]^2$$

$$= E[X + Y - EX - EY)^2 \quad (3.2)$$

$$= E[X - EX + Y - EY]^2$$

$$= E(X - EX)^2 + 2 \cdot E[(X - EX) \cdot (Y - EY)] + E(Y - EY)^2 \quad (3.3)$$

$$= \text{Var}(X) + 2 \cdot \text{Cov}(X, Y) + \text{Var}(Y) \quad (3.4)$$

#### Covariances

Let $X$ and $Y$ denote random variables, then we get as covariance:

$$\text{Cov}(X, Y) = E[(X - E(X)) \cdot (Y - E(Y))] \quad (3.5)$$

$$= E[X \cdot Y] - E[X \cdot E(Y)] - E[E(X) \cdot Y] + E[E(X) \cdot E(Y)] \quad (3.6)$$

$$= E[X \cdot Y] - E(X) \cdot E(Y) - E(X) \cdot E(Y) + E(X) \cdot E(Y) \quad (3.7)$$

$$= E[X \cdot Y] - E(X) \cdot E(Y) \quad (3.8)$$

It is obvious that $\text{Var}[X] = E[(X)^2] - [E(X)]^2$ follows from (3.7) by setting $Y = X$.

#### Variance Stabilizing Transformation

Here we show the derivation of (2.26). For a family of random variables $Y_u$ with $E(Y_u) = u$ and $\text{Var}(Y_u) = v(u) = c^2 \cdot (u - a)^2 + b^2$, we can write:

$$\text{Var}[h(Y_u)] \approx h'(u)^2 \cdot v(u) \quad (3.9)$$

To obtain a constant variance, we have to set $h'(u) = v^{-1/2}(u)$ which leads to $h(y) = \int^y 1/\sqrt{v(u)} \cdot du$ by integrating. Now, we have to substitute $v(u)$ and solve the integral:
\[ h(y) = \int_{y_1}^{y_2} \frac{1}{\sqrt{v(u) \cdot du}} \quad (3.12) \]

\[ = \int_{y_1}^{y_2} \frac{1}{\sqrt{c^2 \cdot (u - a)^2 + b^2}} \cdot du \quad (3.13) \]

\[ = \frac{1}{b} \cdot \int_{y_1}^{y_2} \frac{1}{\sqrt{\left(\frac{u}{b} - \frac{a}{b}\right)^2 + 1}} \cdot du \quad (3.14) \]

The term in the integral is similar to \((x^2 + 1)^{-1/2}\) whose integral is known to be \(\text{arsinh}(x)\). Therefore, we try to differentiate the inverse hyperbolic sine of the quadratic term in the square root of the integral to get correct constant factors:

\[ \frac{\text{darsinh} \left( \frac{u}{b} - \frac{a}{b} \right)}{du} = \frac{1}{\sqrt{\left(\frac{u}{b} - \frac{a}{b}\right)^2 + 1}} \cdot \frac{c}{b} \quad (3.15) \]

The last factor comes from the application of the chain rule. The difference to (3.12) is the \(c\) in the nominator. Now we can guess that the integral of (3.12) is given by

\[ \int_{y_1}^{y_2} \frac{1}{\sqrt{c^2 \cdot (u - a)^2 + b^2}} \cdot du = \frac{1}{c} \cdot \text{arsinh} \left( \frac{c \cdot u}{b} - \frac{c \cdot a}{b} \right) \quad (3.16) \]

and we can prove that by differentiating the right hand side term as we did in (3.15). The denominator of the first fraction \(c\) is cancelled by the \(c\) obtained from the chain rule. Therefore, (3.16) gives us the correct solution.

**Integration Rules**

Let \([a, b]\) an interval with \(a < b\), \(g\) in \([a, b]\) continuously differentiable, \(f\) continuous, then:

\[ \int_{g(a)}^{g(b)} f(x) \cdot dx = \int_{a}^{b} (f \circ g)(z) \cdot g'(z) \cdot dz \quad (3.17) \]

This is retrieved simply by substituting \(x = g(z)\). Then we get \(dx = g'(z) \cdot dz\) and with the chain rule \(f(x) \cdot dx = f(g(z)) \cdot g'(z)dz = (f \circ g)(z) \cdot g'(z)dz\).

### 3.3.1 Transpose of Matrices

Let \(A \in \mathbb{R}^{n\times m}\) and \(B \in \mathbb{R}^{m\times p}\) denote matrices, then it holds that

\[ A^{T\cdot T} = A^T \quad (3.18) \]

\[ (A \cdot B)^T = B^T \cdot A^T \quad (3.19) \]

If we have the transpose of three matrices, we can associate two matrices, apply the rules and then apply the rule again to the associated matrices. Let \(C \in \mathbb{R}^{p\times q}\), then it holds that

\[ (A \cdot B \cdot C)^T = C^T \cdot (A \cdot B)^T \]

\[ = C^T \cdot B^T \cdot A^T \quad (3.22) \]
3.3.2 Symmetric Matrices

Let \( A \in \mathbb{R}^{n \times m} \) denote a matrix. A matrix is symmetric if it holds that:

\[
A^T = A \tag{3.23}
\]

Thus, i.e. diagonal matrices are symmetric. Now, we do not assume \( A \) to be symmetric. Nevertheless, the product of the matrix with its transposed is symmetric, because we can show that in this case (3.23) always holds:

\[
(A^T \cdot A)^T = A^T \cdot A^T = A^T \cdot A \tag{3.24}
\]

Here, we apply (3.19) and then (3.18).

3.3.3 Orthogonality of Eigenvectors of a Symmetric Matrix

Let \( x, y \in \mathbb{R}^d \) be two vectors. We say that \( x \) and \( y \) are orthogonal, if and only if \( \langle x, y \rangle = 0 \). We define the scalar product as before with \( \langle x, y \rangle = x^T \cdot y \). To prove that the eigenvectors of a real valued symmetric matrix are orthogonal, first of all, we show that a symmetric matrix has only real eigenvalues. Then, we show that the eigenvectors of the matrix are orthogonal.

We denote the conjugate complex of \( z \in \mathbb{C} \) as \( \bar{z} \). As the conjugate complex of a complex number \( c = a + ib \) is \( \bar{c} = a - ib \), we see that \( c = \bar{c} \Leftrightarrow c \in \mathbb{R} \) because then the imaginary part of the complex number is zero. We begin with the eigen equation of matrix \( A \) and its eigenvector \( x \) to the eigenvalue \( \lambda \):

\[
A \cdot x = \lambda \cdot x \Leftrightarrow \tag{3.25}
\]

\[
\bar{A} \cdot \bar{x} = \bar{\lambda} \cdot \bar{x} \Leftrightarrow \tag{3.26}
\]

\[
\bar{x}^T \cdot A^T \cdot x = \bar{x}^T \cdot \bar{\lambda} \cdot x \tag{3.27}
\]

Firstly, we conjugate the equation and take \( A \) as a real valued matrix and therefore \( \bar{A} = A \). Next, we transpose the equation and use the symmetry of \( A \) with \( A^T = A \). Now, we multiply (3.25) with \( \bar{x}^T \) from the left and (3.27) with \( x \) from the right which gives us the two equations:

\[
\bar{x}^T \cdot A \cdot x = \bar{x}^T \cdot \lambda \cdot x \tag{3.28}
\]

\[
\bar{x}^T \cdot A \cdot x = \bar{x}^T \cdot \bar{\lambda} \cdot x \tag{3.29}
\]

As the left hand side is equal, we can equate (3.28) and (3.29):

\[
\bar{x}^T \cdot \lambda \cdot x = \bar{x}^T \cdot \bar{\lambda} \cdot x \Leftrightarrow \tag{3.30}
\]

\[
\bar{x}^T \cdot x \cdot \lambda = \bar{x}^T \cdot x \cdot \bar{\lambda} \Leftrightarrow \tag{3.31}
\]

\[
\langle x, x \rangle \cdot \lambda = \langle x, x \rangle \cdot \bar{\lambda} \Leftrightarrow \tag{3.32}
\]

\[
||x||^2 \cdot \lambda = ||x||^2 \cdot \bar{\lambda} \Leftrightarrow \tag{3.33}
\]

\[
\lambda = \bar{\lambda} \Leftrightarrow \lambda \in \mathbb{R} \tag{3.34}
\]

After equating we can shift the scalar eigenvalue and see that we have the scalar product of \( x \) which is the squared norm of \( x \). This is not zero, therefore we can conclude that \( \lambda = \bar{\lambda} \).

The second part is to show the orthogonality of eigenvectors corresponding two unequal eigenvalues of a symmetric matrix \( A \). We denote \( x \) as a eigenvector of the eigenvalue \( \lambda_1 \) and \( y \) as a eigenvector to the eigenvalue \( \lambda_2 \) with \( \lambda_1 \neq \lambda_2 \). Then we can write:

\[
\langle \lambda_1 \cdot x, y \rangle = \langle A \cdot x, y \rangle = \langle x, A^T \cdot y \rangle = \langle x, A \cdot y \rangle = \langle x, \lambda_2 \cdot y \rangle \tag{3.35}
\]

Now we can set the beginning and the end of the transformation sequence equal:
As we assumed $\lambda_1 \neq \lambda_2$ it follows that $x^T \cdot y = \langle x, y \rangle = 0$ which means that $x$ and $y$ are orthogonal.

### 3.4 Prior Selection for Bayesian Networks

The Bayesian approach to assess the probability of a model/parameters $\theta$ given data $x$ uses Bayes theorem in a way that we can compute the posterior by:

$$
P(\theta|x) = \frac{P(x|\theta) \cdot P(\theta)}{P(x)} \quad (3.39)
$$

The data prior $P(x)$ cannot be computed directly therefore we have to integrate over the parameters resulting in

$$
P(\theta|x) = \frac{\int P(x|\theta) \cdot P(\theta) \cdot d\theta}{P(x)} \quad (3.40)
$$

Now we select the parameter prior $P(\theta)$ so that we can solve the integral analytically. To do so we have to use the conjugate distribution of the likelihood distribution. Thus the prior depends on the likelihood distribution. In the following paragraphs we show the calculation for the Binomial distribution and the Multinomial distribution.

#### 3.4.1 Conjugate Prior for Binomial Distribution

The conjugate distribution of the Bernoulli-distribution is the $\beta$-distribution:

$$
\beta(\theta|\alpha_h, \alpha_t) = \frac{\Gamma(\alpha)}{\Gamma(\alpha_h) \cdot \Gamma(\alpha_t)} \cdot \theta^{\alpha_h-1} \cdot (1-\theta)^{\alpha_t-1} \quad (3.41)
$$

with $\alpha = \alpha_h + \alpha_t$. The $\Gamma(\cdot)$-function is defined as:

$$
\Gamma(t) = \int_{0}^{\infty} x^{t-1} \cdot e^{-x} \cdot dx \quad (3.42)
$$

With partial integration one can see that $\Gamma(t+1) = t \cdot \Gamma(t)$ for $t > 0$. For $t = 1$ we retrieve with (3.42) that $\Gamma(1) = 1$, thus we can define inductively

$$
\Gamma(n) = (n-1)! \quad (3.43)
$$

with $n \in \mathbb{N}$. Besides that, [Johnson et al., 1995] show that:

$$
\int_{0}^{1} (1-t)^{a-1} \cdot t^{b-1} dt = \frac{\Gamma(a) \cdot \Gamma(b)}{\Gamma(a + b)} \quad (3.44)
$$

Now we can solve (3.40):
\[ P(\theta|x) = \frac{P(x|\theta) \cdot P(\theta)}{\int_\Theta P(x|\theta) \cdot P(\theta) \cdot d\theta} \]  
(3.45)

\[ = \frac{\binom{n}{h} \cdot \theta^h \cdot (1-\theta)^{n-h} \cdot \beta(\alpha_h, \alpha_t)}{\int_0^1 \binom{n}{h} \cdot \theta^h \cdot (1-\theta)^{n-h} \cdot \frac{\Gamma(\alpha)}{\Gamma(\alpha_h) \Gamma(\alpha_t)} \cdot \theta^{\alpha_h-1} \cdot (1-\theta)^{\alpha_t-1} \cdot d\theta} \]  
(3.46)

\[ = \frac{\theta^{h+\alpha_h-1} \cdot (1-\theta)^{t+\alpha_t-1} \cdot \frac{\Gamma(\alpha)}{\Gamma(\alpha_h) \Gamma(\alpha_t)}}{\Gamma(h+\alpha_h) \cdot \Gamma(t+\alpha_t) \cdot \theta^{h+\alpha_h-1} \cdot (1-\theta)^{t+\alpha_t-1} \cdot d\theta} \]  
(3.47)

\[ = \frac{\Gamma(h+\alpha_h) \cdot \Gamma(t+\alpha_t)}{\Gamma(t+\alpha_t) \cdot \Gamma(h+\alpha_h) \cdot \theta^{h+\alpha_h-1} \cdot (1-\theta)^{t+\alpha_t-1}} \]  
(3.48)

\[ = \frac{\theta^{h+\alpha_h-1} \cdot (1-\theta)^{t+\alpha_t-1}}{\beta(\theta|\alpha_h+h, \alpha_t+t)} \]  
(3.49)

In the step from (3.45) to (3.46) we simply substitute the probabilities with the equation given above. The same is done in the next step and we can shift the binomial coefficient out of the integral and cancel it. In (3.48) we have combined the exponentials. In the denominator we can write the fraction of \( \Gamma \)'s in front of the integral because it is not dependent on \( \theta \). In the next step this fraction is cancelled by the same fraction in the nominator. And we can also solve the integral by substituting with (3.44). In (3.50) we reverse the \( \Gamma \)-fraction and recognize that this is a \( \beta \)-distribution with the parameters given in (3.51).

The prior does not solve only the integration but has an intuitive graphical interpretation. We make this clear with help of the coin toss example introduced on page 26. In figure 3.2 we draw \( \beta \)-distributions with different parameters \( \alpha_h \) and \( \alpha_t \). First we state that \( \beta(1,1) \) is equal to the uniform density function. Thus if we do not know anything about the models, we set \( \alpha_h = 1 \) and \( \alpha_t = 1 \). If we have already performed some experiments, we set the parameter \( \alpha_h \) to the number of heads we have observed and \( \alpha_t \) to the number of tails. If we have observed two heads and two tails, we see at the prior density of \( \beta(2,2) \) that the parameter \( \theta \) is 0.5 with highest probability. This maximum shifts if we have observed heads and tails unequal times. If we have about two thirds tails, it is not very probable that we have a fair coin. The maximum here is at \( \theta = 0.33 \).

Now we can go over to interpret the posterior. It is a \( \beta \)-function, as well. The only difference is that we add to \( \alpha_h \) respectively \( \alpha_t \) the newly observed number of heads and tails. The interpretation of the posterior \( \beta \)-function remains. If we do another sequence of experiments, we can take this posterior as the new prior. Now that we have eliminated some uncertainty in our knowledge about the model, which we can use as prior knowledge in following experiments.

If we want to compute the estimator \( \hat{\theta} \), we have to compute the expectation value with respect to the posterior:
3.4. PRIOR SELECTION FOR BAYESIAN NETWORKS

\[ \theta = E_{\theta|x} \theta \]
\[ = \int_0^1 \theta \cdot P(\theta|x) \cdot d\theta \] (3.52)
\[ = \int_0^1 \theta \cdot \beta(\theta|\alpha_h + h, \alpha_t + t) \cdot d\theta \] (3.53)
\[ = \frac{\Gamma(\alpha + h + t)}{\Gamma(h + \alpha_h) \cdot \Gamma(t + \alpha_t)} \cdot \int_0^1 \theta \cdot \theta^{h-1} \cdot 1^{t-1} \cdot d\theta \] (3.54)
\[ = \frac{\Gamma(h + \alpha_h + 1) \cdot \Gamma(t + \alpha_t)}{\Gamma(\alpha + h + t + 1)} \] (3.55)
\[ = \frac{\alpha_h + h}{\alpha + h + t} \] (3.56)

The calculation seems to be difficult, but due to our selection of the prior most terms cancel and we get an intuitive result. In (3.54) we have substituted the posterior by the calculated \( \beta \)-function from (3.51). As before we can plug in the \( \beta \)-function from (3.41) and shift the \( \Gamma \)-fraction out of the integral and then we combine the \( \theta \) with its exponential term. In (3.56) we have solved the integral with (3.44). Here we have to recognize that we added one to the exponential term of \( \theta \) resulting in a +1 in the first term of the nominator and in the denominator of the second fraction. With the inductive definition of \( \Gamma(n) \) in (3.43) we can decrement these two \( \Gamma \)-functions with one. Then we can cancel all \( \Gamma \)-functions and all that remains is shifted out of \( \Gamma \). It results in the number of heads we had observed a priori and the newly observed heads divided by the number of all experiments done (a priori and actually). This intuitive result is exactly what one is used to do by estimating the probability of an experiment with two outcomes. If we have observed both different outcomes (heads and tails) equally frequent, we will get \( \hat{\theta} = 0.5 \).

3.4.2 Conjugate Prior for Multinomial Distribution

The conjugate distribution of the multinomial distribution is the Dirichlet distribution. Here we show the integrals will be cancelled as it did with the \( \beta \)-distribution by assuming a Bernoulli (binomial) distribution. The Dirichlet-distribution is defined as:
\[ D(\theta|\alpha_1, \ldots, \alpha_r) = \frac{\Gamma(\alpha)}{\prod_{k=1}^{r} \Gamma(\alpha_k)} \cdot \prod_{k=1}^{r} \theta_k^{\alpha_k-1} \]  \hspace{1cm} (3.58)

It is the extension of the \( \beta \)-distribution with more than two states: \( |\Omega| = r \). Similarly, \( \alpha = \sum_{k=1}^{r} \alpha_k \). [Kotz et al., 2000] show that

\[ \int_{\Theta} \prod_{k=1}^{r} \theta_k^{\alpha_k+N_k-1} d\theta = \frac{1}{\Gamma(\alpha+N)} \cdot \prod_{k=1}^{r} \Gamma(\alpha_k + N_k) \]  \hspace{1cm} (3.59)

with \( N = \sum_{k=1}^{r} N_k \). Computing the posterior is similar to (3.45):

\[
P(\theta|x) = \frac{P(x|\theta) \cdot P(\theta)}{\int_{\Theta} P(x, \theta) \cdot d\theta} \hspace{1cm} (3.60)
\]

\[
= \frac{\prod_{k=1}^{N_1} \theta_k^{N_k} \cdot \prod_{k=1}^{N_r} \Gamma(\alpha_k)}{\prod_{k=1}^{r} \Gamma(\alpha_k + N_k)} \cdot \prod_{k=1}^{r} \theta_k^{\alpha_k-1} \cdot d\theta
\]

\[
= \frac{\prod_{k=1}^{r} \theta_k^{\alpha_k+N_k-1}}{\Gamma(\alpha+N) \cdot \prod_{k=1}^{r} \Gamma(\alpha_k + N_k)}
\]

\[
= \frac{\Gamma(\alpha+N)}{\prod_{k=1}^{r} \Gamma(\alpha_k + N_k)} \cdot \prod_{k=1}^{r} \theta_k^{\alpha_k+N_k-1}
\]

\[
= D(\theta|\alpha_1 + N_1, \ldots, \alpha_r + N_r)
\]

As before we retrieve as posterior the conjugate distribution. The interpretation remains and the expectation value with respect to the posterior is quite similar. Denoting the number of observed events \( k \) with \( N_k \) and let \( \alpha_k \) being the corresponding knowledge of earlier observed events \( k \), we get:

\[
\hat{\theta}_k = E_{P(\theta|x)} \theta_k
\]

\[
= \int_{\Theta} \theta_k \cdot D(\theta|\alpha_1 + N_1, \ldots, \alpha_r + N_r) \cdot d\theta
\]

\[
= \frac{\alpha_k + N_k}{\alpha + N}
\]

The interpretation is exactly as before.


