INAUGURAL-DISSERTATION

zur
Erlangung der Doktorwürde
der
Naturwissenschaftlich-Mathematischen Gesamtfakultät
der
Ruprecht-Karls-Universität
Heidelberg

vorgelegt von
Diplom-Mathematikerin Johanna Mazur
aus Oppeln (Polen)

Tag der mündlichen Prüfung: 24. Juli 2012

Bayesian Inference of Gene Regulatory Networks:
From Parameter Estimation to Experimental Design

Gutachter: Dr. Stefan Körkel
Prof. Dr. Lars Kaderali

Abstract

To learn the structure of gene regulatory networks is an interesting and important topic in systems biology. This structure could be used to specify key regulators and this knowledge may be used to develop new drugs which affect the expression of these regulators. However, the inference of gene regulatory networks, especially from time-series data is a challenging task. This is due to the limited amount of given data which additionally contain a lot of noise. These data cause from the technical point of view for the parameter estimation procedure problems like the non-identifiability and sloppiness of parameters.

To address these difficulties, in these thesis new methods for both, the parameter estimation task and the experimental design for gene regulatory networks, are developed for a non-linear ordinary differential equations model, which use a Bayesian procedure and generate samples of the underlying distribution of the parameters. These distributions are of high interest, since they do not provide only one network structure but give all network structures that are consistent with the given data. And all of these structures can then be examined in more detail.

The proposed method for Bayesian parameter estimation uses smoothing splines to circumvent the numerical integration of the underlying system of ordinary differential equations, which is usually used for parameter estimation procedures in systems of ordinary differential equations. An iterative Hybrid Monte Carlo and Metropolis-Hastings algorithm is used to sample the model parameters and the smoothing factor. This new method is applied to simulated data, which shows that it is able to reconstruct the topology of the underlying gene regulatory network with high accuracy. The approach was also applied to real experimental data, a synthetic designed 5-gene network (the DREAM 2 Challenge #3 data) and outperforms other methods.

For the Bayesian experimental design step, a full Bayesian approach was used which does not use any parametric assumption of the posterior distribution, nor linearizes around a point estimate. To make the full Bayesian approach computationally manageable, maximum entropy sampling is used together with a population-based Markov chain Monte Carlo algorithm. The approach was applied to simulated and real experimental data, the DREAM 2 Challenge #3 data, and outperforms the usage of random experiments and a classical experimental design method.

Zusammenfassung

Die Struktur von Genregulationsnetzwerken zu lernen ist eine interessante und wichtige Fragestellung in der Systemsbiologie. Diese gelernte Struktur könnte dazu dienen, essentielle Regulatoren zu spezifizieren und dieses Wissen könnte weiterhin für die Entwicklung neuer Arzneimittelwirkstoffe genutzt werden, die die Expression dieser Regulatoren beeinflussen. Die Inferenz von Genregulationsnetzwerken, speziell aus Zeitreihendaten, ist jedoch eine herausfordernde Aufgabe. Dies liegt daran, dass die Menge der gegebenen Daten klein ist und diese Daten zusätzlich auch sehr verrauscht sind. Aus technischer Sicht für die Parameterschätzung führt dies zu Problemen wie der Nicht-Identifizierbarkeit und der "Sloppiness" von Parametern.

Um diese Schwierigkeiten zu addressieren, werden in dieser Dissertation sowohl Methoden für die Parameterschätzung als auch für das Experimentdesign von Genregulationsnetzwerken entwickelt und zwar für ein nicht-lineares gewöhnliches Differentialgleichungs-Modell. Diese Methoden benutzen einen Bayes'schen Ansatz und erzeugen Stichproben der zugrunde liegenden Verteilungen der Modell-Parameter. Diese Verteilungen sind von hohem Interesse, da sie nicht nur eine Netzwerkstruktur als Lösung präsentieren, sondern alle Netzwerkstrukturen, die mit den gegebenen Daten konsistent sind. Und alle diese erhaltenen Strukturen können dann näher untersucht werden.

Die vorgestellte Methode für die Bayes'sche Parameterschätzung benutzt geglättete Splines, um die numerische Integration, die üblicherweise benutzt wird, des zugrunde liegenden Systems bestehend aus gewöhnlichen Differentialgleichungen, zu verhindern. Ein iterativer hybrider Monte-Carlo- und Metropolis-Hastings-Algorithmus wird benutzt, um Stichproben der Modellparamter und des Glättungsfaktors der geglättenen Splines zu generieren. Mit Anwendung dieser Methode auf simulierte Daten wird gezeigt, dass die Topologie des zugrunde liegenden Genregulationsnetzwerkes mit hoher Genauigkeit gelernt werden kann. Der gleiche Ansatz wird auch auf echte experimentelle Daten, einem synthetisch konstruierten 5-Gen-Netzwerk (den DREAM-2-Challenge-#3-Daten), angewendet und übertrifft die Resultate anderer Methoden.

Für das Bayes'sche Experimentdesign wird ein vollständiger Bayes'scher Ansatz benutzt, welcher weder eine parametrische Annahme über die a-posteriori-Verteilung annimmt noch um einen Punktschätzer herum linearisiert. Um diesen vollständigen Bayes'schen Ansatz rechnerisch handhabbar zu machen, wird "Maximum Entropy Sampling" zusammen mit einem populations-basierten Markov-Chain-Monte-Carlo-Algorithmus benutzt. Diese Methode wird

sowohl auf simulierten als auch echten experimentellen Daten, den DREAM-2-Challenge-#3-Daten, angewendet und übertrifft sowohl die Resultate der Anwendung zufälliger Experimente als auch einer klassischen Experimentdesignmethode.

Acknowledgments

> When eating bamboo sprouts, remember the man who planted them.
>
> *(Chinese Proverb)*

A work like this would have never been possible without the encouragement, help and just possibilities given by other people. According to the Chinese proverb the "bamboo sprouts" is this thesis and the "man who planted them" are the bulk of people that helped me in numerous ways such that this thesis came into existence.

I will start to express my gratitude to Prof. Dr. Lars Kaderali, who gave me the opportunity to start this PhD work in his research group. He found the perfect balance between guiding my work and giving me the freedom to perform my research in a way I wanted to perform it. He also encouraged me to present my work to other researchers and was interested in me being get noticed in the scientific community. Thank you, Lars!

Prof. Dr. Dr. h.c. Hans Georg Bock also deserves my biggest respect for seeing me and my work and for giving me the possibility to benefit from the expertise that is available in his research group at the Interdisciplinary Center for Scientific Computing. Thank you, Georg!

I thank the graduate school HGS MathComp and all their fellows for showing me that "Solus Numquam Vades!" on the long path to the PhD degree and how mathematical and computational methods help to answer scientific questions in diverse fields of research.

My colleague Bettina Knapp, who always stood my sorrows and worries but also pleasures on a daily basis in the last four and a half years, has become one of the best friends I've ever had. Thank you for being such a supportive friend! And thank you for explaining computer science to me and for being a living R manual and an excellent Swabian teacher.

Altogether, I want to thank the whole *Viroquant Research Group Modeling* for the nice time we spent. Being a member of this group from its beginning until its end, I appreciate all the different people that I was able to meet and work with. Coming from diverse disciplines and countries, first I learned a lot about different cultures and am able now to see that Germans are really complaining on a high level. Second, working with people with different backgrounds, me with a mathematical background, I gained a lot of expertise in topics from computer science, biology and bioinformatics. I thank all the people who answered me the

more or less stupid questions I had.

One person, that I have to mention explicitly is Daniel Ritter. It was a great honor for me to supervise your master's thesis and I thank you for your great and devoted work.

Since some other people in the group were also very important to me in my PhD time, I will mention them in alphabetical order: First, I want to thank Joanne Barry, being the only English native speaker in the group, for having the thankless task of proofreading all publications in the whole group and still always being kind and respectful to all people. Furthermore, I will always remember the time we spent together in Shanghai and the discussions we had which were always very deep and interesting although our Weltanschauung[1] is completely different in some cases. Thank you Dr. Diana Claußnitzer for sharing my passion for coffee, my sense of humor and being such a great and talented person and physicist. I want to thank Aditya Jitta for his trust in me to supervise the successful project he did in our group. Thank you Prabhav Kalaghatgi for sharing my passion for American TV series, for teaching me the pronunciation of Indian names and for being such a great company in room 541. Thank you Dr. Narsis Kiani for being an open-minded partner for discussions and a very hospitable person and, of course, thank you for your apartment in the Old Town! Many thanks are going to Douaa Mugahid: you helped the group to socialize with your excellent organizational skill and your lovable personality. Finally, I want to thank Nora Rieber for always giving me the feeling that everything is okay the way it is.

I am deeply indebted to Victoria Miller for being the best friend ever in tough but also in happy times. Especially her skill to make me laugh independent of the mood I am in, is a gift that deserves to be mentioned. Thank you for the opportunity to being your friend!

Last but not least, I want to thank my parents for being supportive in all phases of my career until now and for treating my work with such a great respect and honor. Without your backing and your patience my PhD time would have been a really hard time.

[1] according to Leo (`http://dict.leo.org`) one may use the German word also in English

Contents

List of Figures

List of Algorithms

List of Tables

Introduction

Every new beginning comes from some other beginning's end.

(Seneca)

Systems biology, as a relatively new scientific field, deals with the investigation of biological systems combining experimental data and mathematical and algorithmical methods. The main idea behind this new scientific area is the hope that looking not only at small parts of the system, but rather see the whole one will find out some *emergent properties* in the system that are not present in any of its subsystems. To see these properties can also be described by the saying: "The whole is more than the sum of its parts". Kitano [Kit02] summarized this excellently in his paper about computational systems biology in 2002 and also points out the need of computational methods for problems arising in systems biology:

> Molecular biology has uncovered a multitude of biological facts, such as genome sequences and protein properties, but this alone is not sufficient for interpreting biological systems. Cells, tissues, organs, organisms and ecological webs are systems of components whose specific interactions have been defined by evolution; thus a system-level understanding should be the prime goal of biology. Although advances in accurate, quantitative experimental approaches will doubtless continue, insights into the functioning of biological systems will not result from purely intuitive assaults. This is because of the intrinsic complexity of biological systems. A combination of experimental and computational approaches is expected to resolve this problem.

A general framework how biological modeling in systems biology looks like is depicted in Figure 0.1: with some biological knowledge a mathematical model is established which is then used to predict biological behavior. If the obtained results are not accurate for the desired purpose, methods for optimum experimental design are applied and the proposed experiments are performed in a wet laboratory. The obtained data are then used to improve the mathematical model. Now a new cycle of predicting biological behavior, applying methods for experimental design, performing wetlab experiments and improving the used mathematical model starts.

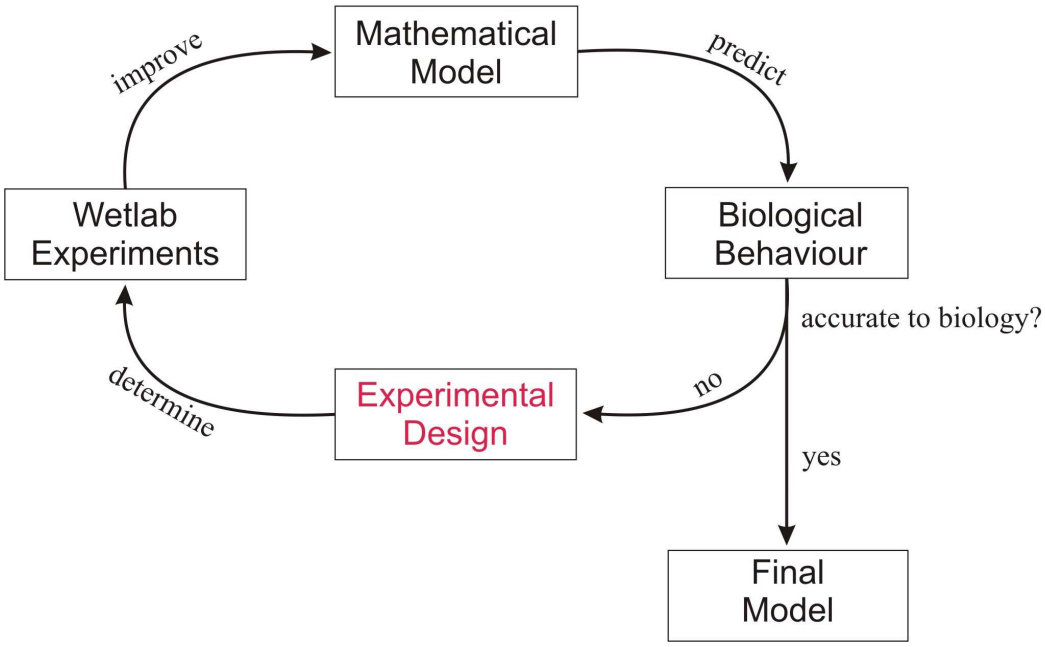


Figure 0.1.: *A sequential procedure for modeling biological processes. First, according to prior biological knowledge a mathematical model is established. This model is then used to predict biological behavior and if the results are not accurate to the desired purpose, experimental design is performed to obtain data from wetlab experiments, which will improve the mathematical model the most. After doing so, the next cycles begin until the model is accurate enough.*

Two essential features from the algorithmical side in this procedure are, first, parameter estimation methods that are used to infer parameters of the mathematical model for given data, and second, approaches for optimum experimental design are needed to specify the wetlab experiments that have to be performed.

In this thesis, learning the structure of gene regulatory networks from gene expression data is the task that will be addressed. Knowing the topology of gene regulatory networks is highly desired in systems biology. The structure of these networks may be used to predict key regulators and these regulators may be used as potential drug targets. A lot of models describing gene regulatory networks were already described and inference algorithms were also given. These models and algorithms will be described in Chapter 4. The model for gene regulatory networks used in this thesis will be a system of non-linear ordinary differential equations and the task is to estimate the model parameters given time-series gene expression data.

For parameter estimation problems in systems biology several technical problems are present. These problems arise, because only a limited amount of data which additionally contain a lot of noise is available for the inference task. Together with the fact, that usually a large number of parameters has to be estimated and thus the search space that has to be searched to find an optimal solution is large, the following three problems have been described in more detail for parameter estimation tasks in systems biology:

1. **Non-identifiability of parameters.** The non-identifiability of parameters is per se not only a problem arising for models in systems biology. It may appear in all kinds of models. Basically, one distinguishes between *practical non-identifiability* and *structural*

non-identifiability of parameters. The latter one specifies the fact, that even for the case with perfect data, the model parameters cannot be determined in a unique way and ambiguities in the parameters arise. It is a property that is inherent in the used model. On the contrary, practical non-identifiabilities of parameters arise because not enough data and not good enough data is available to estimate the parameters precisely. Once good data is available, practical non-identifiabilities can be resolved. Raue and colleagues developed sophisticated methods to do identifiability analysis for models in systems biology [RKM+09, RBKT10, RKM+11].

2. **Sloppiness of parameters.** Sethna and coworkers [GWC+07, DCS+08, TMS10] found out that a lot of models, especially in systems biology, show an effect which is called *sloppiness*. It is defined as the property that the eigenvalues of the covariance matrix around a point estimate span a large range and the corresponding eigenvectors are skewed. This behavior implies that the parameters are not independent and one can change the values of the sloppy parameters, i.e., the ones corresponding to a small eigenvalue of the covariance matrix, to a high extent without changing the model behavior. The authors present also examples that even for the direct measurement of the parameters the precision of the predictions may not be increased because of the inherent sloppiness property of the underlying model [GWC+07]. And because a wide range of parameters in sloppy models will describe the same dynamics of the system, they argue that one should focus more on the model predictions and less on precise parameter estimates.

3. Recently, Slezak *et al.* [SSC+10] performed a case study where they analyzed the landscape of the objective function used for parameter estimation. This landscape is rugged and contains a high number of local minima. Furthermore, they argue that a high number of these local minima, including the global minimum, are not reasonable from the biological point of view. The authors suggest to use experimental design simultaneously with optimization and furthermore, to generate independent data sets describing different regimes of the biological system under consideration that will hopefully constrain different parameters of the used model.

All these difficulties presented here can be nicely addressed using a Bayesian framework for parameter estimation tasks which also has been pointed out in our publication [MK11]. The Bayesian framework is the main algorithmical focus in this thesis and new methods to perform it will be introduced and examined here. A Bayesian framework incorporates the noise present in the data and one easily is able to incorporate prior knowledge into the learning procedure. However, the biggest benefit of Bayesian approaches is their ability to provide distributions over parameters. These distributions contain a lot of knowledge about different parameter sets that are able to describe the data in a similar way, and this knowledge, as a next step, can be used to perform Bayesian experimental design and to generate additional experimental data that will increase the information[2] of these distributions the most. Several authors already argued that Bayesian experimental design is needed and has to be explored in large, complex non-linear ordinary differential equations models [CG08, KR10, KHAR10][3].

In this thesis, a new algorithmical method for Bayesian parameter estimation for the inference of a non-linear model for gene regulatory networks is given. The work presented here

[2] the mathematical concept as defined in Definition 2.2.2

[3] these publications will be further discussed in Section 6.3.3

was published as [MRRK09] and is collaborative work together with Daniel Ritter who did his Master's thesis [Rit08] on the same topic. Additionally, a new algorithmical method for Bayesian experimental design is presented and examined for the same non-linear model describing gene regulatory networks. The method uses the maximum entropy sampling theory presented by Sebastiani and Wynn [SW00].

Structure of the thesis

The thesis is structured in five parts. In Part I the technical background is given, which is needed to understand the essential part of the thesis. It is subdivided into biological, mathematical and algorithmical background. Since the topic of the thesis is located in the interdisciplinary field of systems biology, this part gives the reader coming from different disciplines the possibility to be able to understand the new methods presented in this thesis. In Chapter 1 the essential terms and facts of the gene regulation process are introduced and described. Furthermore, experimental methods are described with their strengths and weaknesses that are used to measure gene products, i.e., mRNA and protein concentrations. Chapter 2 describes the mathematical background needed in probability, information and Markov chain theory. Furthermore, basic facts about ordinary differential equations and splines are given. In Chapter 3 algorithms for the generation of samples of a distribution, i.e., Markov chain Monte Carlo algorithms, and their efficiency are presented and discussed. Additionally, general aspects about parallel computing, which will be used in Chapter 7, and receiver operating characteristics that are needed for the evaluation of learned networks in Chapter 5, are described. Reading this thesis, the reader may first omit Part I and can directly start with Part II and then go back to Part I, if any background knowledge is needed.

Part II deals with the first topic of this thesis, the inference of gene regulatory networks from time-series data with Markov chain Monte Carlo algorithms and spline interpolation using a model of non-linear ordinary differential equations. This part starts with an overview of existing models for gene regulatory networks in Chapter 4 with the analysis of their pros and cons. In Chapter 5 the new method for Bayesian parameter estimation of non-linear ODE models for gene regulatory networks with smoothing splines combined with sampling algorithms is explained and results for simulated and real experimental data with the comparison to other inference methods are presented.

In Part III the second topic of this thesis, Bayesian experimental design for the parameter estimation of systems of non-linear ordinary differential equations by means of maximum entropy sampling is examined. This part starts with an introduction in classical and Bayesian experimental design with existing methods for parameter estimation for models based on ordinary differential equations in Chapter 6. In Chapter 7 the new method for Bayesian experimental design is given and applied to the model for gene regulatory networks described in Chapter 5 and the results are compared to random experiments and classical experimental design for simulated and real experimental data.

To close the circle of the thesis, which is also depicted in Figure 0.1, Part IV gives the discussion and the outlook of the presented Bayesian methods for the parameter estimation and the experimental design for non-linear ordinary differential equations examined in Part II and Part III.

In the last part, in Part V, the appendices are given.

Part I.

Technical background

CHAPTER 1

Biological background

> The essence of life is statistical improbability on a colossal scale.
>
> *(Richard Dawkins)*

Since the application of my work is settled in biology, I will give an overview of some biological facts concerning gene expression and gene regulation. These sections are geared to [ABL+94]. Furthermore, I will summarize experimental setups how to measure gene expression and the difficulties behind it.

1.1. The cell

What all cells have in common is, that they are surrounded by a *plasma membrane* which is about 5 nm thick and contains the genetic information in it. This genetic information is stored in the *DNA*. The information in the DNA encodes the "rules" which vital processes are required and which organic molecules have to be synthesized to accomplish the tasks needed for the cell to live. For this purpose, *evolution* has brought up three processes, how chemical energy in form of *ATP* is formed. ATP is the central source for energy present in cells.

1. *Glycolysis* is an inefficient process to degrade *glucose* in the absence of oxygen, i.e., it is an *anaerobic* process. As their was no oxygen in the atmosphere when the first cells evolved, this process was developed by the cells.

2. *Photosynthesis* is the process where sunlight is absorbed with the pigment molecule *chlorophyll* to use the energy of the sun to convert CO_2 from the atmosphere into organic compounds. As a by-product of photosynthesis, oxygen is produced.

3. *Respiration* is an efficient process for ATP formation. It is the *aerobic oxidation* of food molecules, i.e., it uses oxygen to degrade food molecules. This process developed, once photosynthesis caused oxygen to be present in the atmosphere.

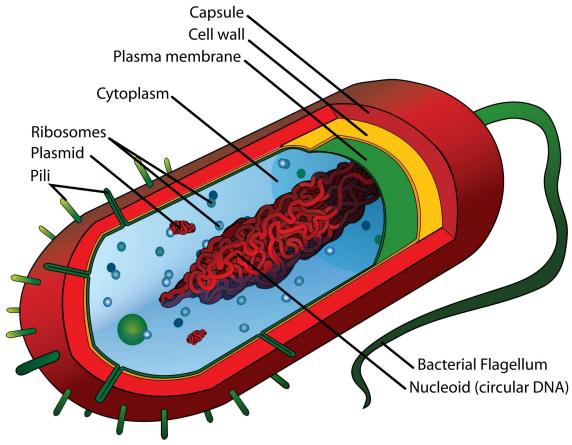


Figure 1.1.: *A typical example of a prokaryotic cell: a bacterium. Details of the bacterium components are described in Section 1.1.1. Source:* `http://en.wikipedia.org/wiki/Prokaryote`

Basically, one distinguishes between two different types of cells: cells without a *nucleus*, *prokaryotic cells*, and cells with a nucleus, *eukaryotic cells*. We will start with the former ones.

1.1.1. The prokaryotic cell

Prokaryotic cells are cells with relatively simple internal structures. Inside a prokaryotic cell one finds DNA, *RNA*, *proteins*, and other small molecules. Typical examples for prokaryotic cells are *bacteria*. Bacteria are small and can replicate quickly which enables them to rapidly adapt to changing environments. They are the most abundant type of cells on earth.

In Figure 1.1, a typical example for a prokaryotic cell is depicted: a bacterium. One see that the plasma membrane encloses the *circular DNA* and other small molecules. The "body" of circular DNA is also called the *nucleoid*. *Ribosomes* are used in all cells, prokaryotic and eukaryotic in the procedure of forming proteins which will be described in Section 1.2.3. Additionally, one may find *plasmids* in bacteria, which are DNA molecules separate and independent from the nucleoid. Around the cell membrane of the bacterium there is a *cell wall*. An additional surrounding of bacteria is the *capsule*, which protects the prokaryotic cell from engulfment by eukaryotic cells. The *pili* on top of bacteria are used to connect to another bacterium and transfer plasmids between them. To move, bacteria use *flagella*, which is a protein structure about 20 nm in diameter.

To produce chemical energy in form of ATP, different bacteria use different processes. Every process described in the previous section can be used by bacteria. Since bacteria adapted to a lot of environment conditions, every species of bacterium uses that energy producing process which is most advantageous in its environment.

1.1.2. The eukaryotic cell

In eukaryotic cells, the DNA is separated from the other contents of the cell, since most of it is contained in the nucleus. The nucleus itself is enclosed by a double layer of membrane. Thus,

a eukaryotic cell has two main regions, the nucleus and the *cytoplasm*, where most of the cell's metabolic reactions occur. Typical examples for a eukaryotic animal and a eukaryotic plant dell are shown in Figure 1.2. Different from prokaryotic cells, eukaryotic cells contain several *cell organelles*. Common in all eukaryotic cells, one finds *mitochondria*, which are responsible for respiration. In other words, one can say, that mitochondria are the energy producers of cells. Another cell organelle, which is found in plants, but not in animals and *fungi*, are the *chloroplasts*. This organelle is responsible for carrying out photosynthesis in their inherent *thylakoid membranes*. The obtained *glucose* is then stored in so-called *starch grains*. The evidence is high, that both, mitochondria and chloroplasts, were originally prokaryotic cells and were engulfed by eukaryotic cells and live now a symbiotic life.

To simplify the problem of entrance and leaving of materials for biosynthetic reactions through the plasma membrane, eukaryotic cells contain a lot of internal membranes. Thus, reactions occur inside the cell and the exchange with the environment is done by *exocytosis* (external material is brought into the cell) and *endocytosis* (internal material is brought outside of the cell). In both processes *vesicles* are formed containing the molecules which have to move out resp. move in. In the *endoplasmic reticulum* (ER) materials for the export from the cell are produced. The *Golgi apparatus* helps in the transport of the molecules made in the ER. Membranes also enclose *lysosomes* and *peroxisomes*. The former ones store *enzymes* needed for digestion, which are able to attack proteins. The latter ones break very long *fatty acids*. In this process *hydrogen peroxide* is generated, which is dangerous for the rest of the cell, since it is very reactive. Thus, in both cases the enclosing membrane protects the cell from severe damage, either by enzymes or hydrogen peroxide. In plant cells (see Figure 1.2 Bottom) there exist even a water-filled *vacuole* to, amongst other things, isolate harmful materials to the cell. The vacuole is surrounded by a membrane called the *tonoplast*.

To keep all the cell organelles in the proper place and control their movement, an eukaryotic cell contains a *cytosceleton*. Is is composed of a network of protein *filaments*. Flagella consist of these filaments and help the eukaryotic cell, as the prokaryotic cell, to move. *Centrioles*, as a part of the cytosceleton, which are only present in animal cells, play an essential role in the spatial arrangement of the cell.

1.2. Gene expression

We learned in the previous section, how a general cell looks like and want to focus now only on eukaryotic cells to describe the process of gene expression and gene regulation. The three main types of molecules involved in gene expression are DNA, RNA and proteins. We will discuss these molecules in detail and along the way the processes of gene expression and regulation will be described. Since it is out of the scope of this PhD thesis to explain basic knowledge in general and organic chemistry, I assume such facts as given. They can be found in any introductory book on general and organic chemistry.

1.2.1. DNA

The basic subunits that form *deoxyribonucleic acid* (DNA) are the four *nucleotides adenine* (A), *cytosine* (C), *guanine* (G) and *thymine* (T). Nucleotides consist of one of several nitrogen-containing *aromatic base* linked to a *five-carbon sugar* that carries a *phosphate group*. The nucleotides are named after the base they contain. These nucleotides are linked together by *covalent bonds* that join the $5'$ carbon of the sugar *deoxyribose* to the $3'$ carbon of the next.

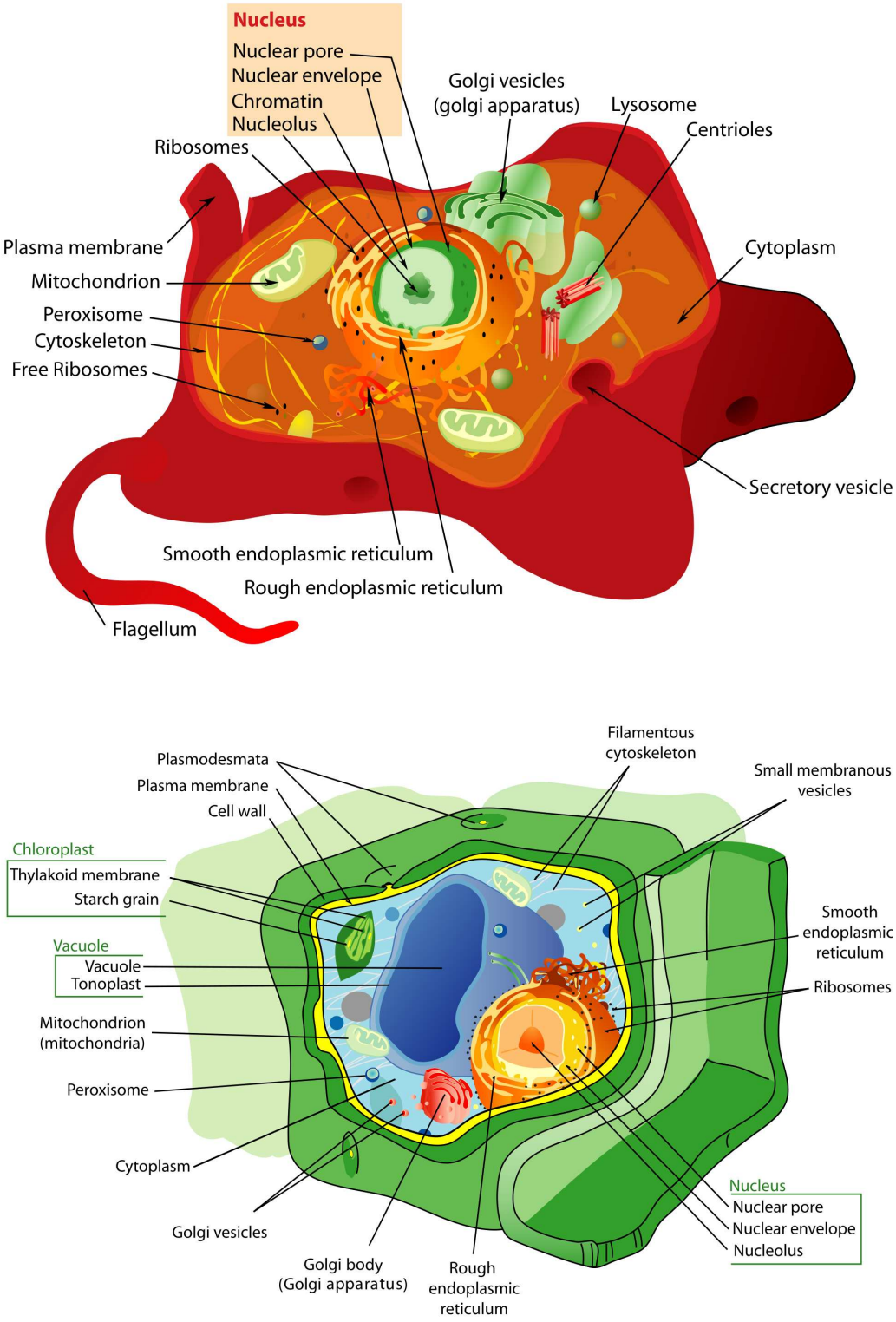


Figure 1.2.: *Eukaryotic cells: (Top) A typical animal cell. (Bottom) A typical plant cell. The cell organelles and components are described in more detail in Section 1.1.2. Source:* `http://en.wikipedia.org/wiki/Eukaryote`

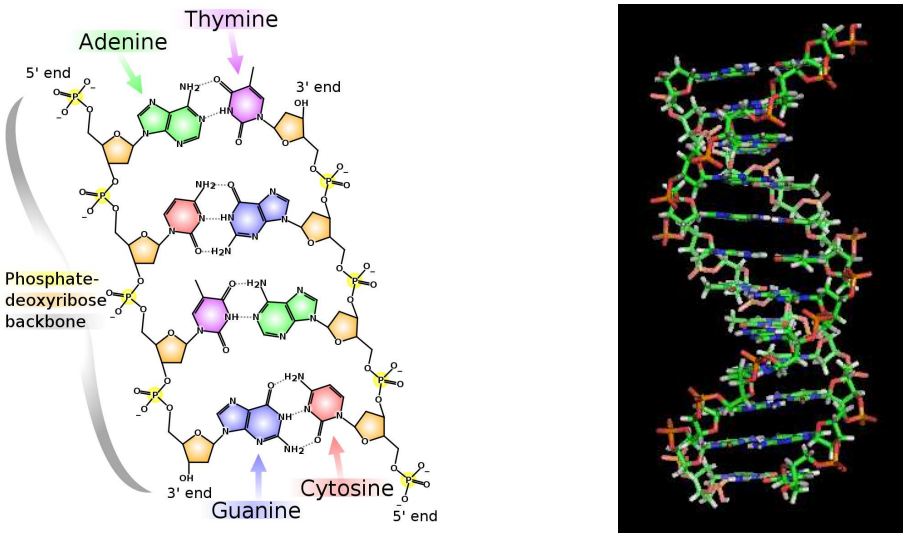


Figure 1.3.: *(Left) The chemical structure of a part of a DNA molecule, where the hydrogen bonds are denoted by dotted lines. Source:* `http://en.wikipedia.org/wiki/DNA`*, (Right) The double helix structure of a DNA molecule. Source:* `http://en.wikipedia.org/wiki/Nucleic_acid_double_helix`

All the genetic information contained in the DNA is structured as a linear sequence of these four nucleotides. A DNA molecule is built of two strands of such linear sequences that are *complementary* in their nucleotide sequence. Complementary means that the bases G and C are always opposite each other and connected via *hydrogen bonds*. And the same holds for the bases A and T. These complementary nucleotide sequences form a *double helix* with about 10 nucleotide pairs per helical turn. One such DNA molecule may contain several millions of such nucleotide pairs. Each DNA molecule is packaged in a separate *chromosome*. And all the information contained in all chromosomes in one organism is called the *genome* of this organism. In Figure 1.3 the chemical structure of DNA is depicted on the left and on the right one sees the double helix structure of DNA.

Now we know how the genetic information is stored in detail in every eukaryotic cell. But how is this information now used to organize the living of the cell, i.e., how are genes expressed? To answer this question, we first need to define what a gene is. Since this definition is not unique, we will define it here as a region of the DNA helix that produces a functional RNA molecule[1] (also called here *messenger RNA*, see also Section 1.2.2). Why do we say "functional RNA molecule" and not "RNA molecule"? This is, because for the gene expression process one needs some helper molecules like *transfer RNA* (tRNA), which carries one of the 20 amino acids needed for the synthesis of proteins. A second helper molecule produced from a gene is *ribosomal RNA* (rRNA), which together with *ribosomal proteins* forms ribosomes. And, of course, the information for these helper molecules is also encoded in the DNA. With the definition above, genes in higher eukaryotes can be up to 2 million nucleotide pairs long, whereas only about 1000 nucleotide pairs are needed to code for a protein of average size. The parts containing the necessary information for a specific protein are called *exons* and the non-informative parts are called *introns*.

[1] see [ABL+94] page 340

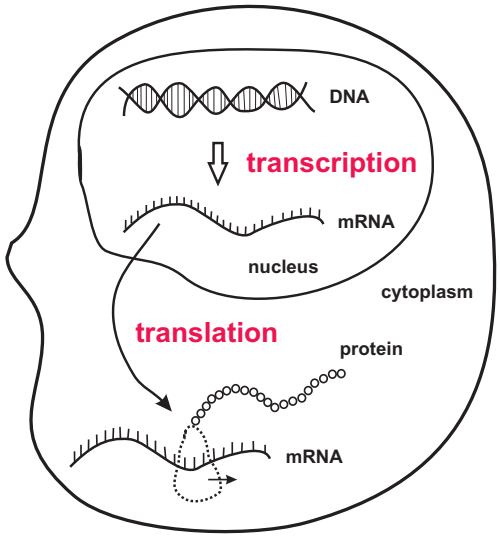


Figure 1.4.: *Schematic view over gene expression: In the nucleus the DNA is read and mRNA is created (transcription). The mRNA is then translated into the cytoplasm and a protein is created (translation).*

Now we can come back to the question, how genes are expressed. In Figure 1.4 the two main steps for gene expression are depicted:

1. *DNA transcription*
2. *translation* or *protein synthesis*

In the first step, a gene is read to produce *messenger RNA* (mRNA). This step will be explained in the next subsection. The second step deals, once we have an mRNA template, with the procedure of the synthesis of proteins. This will be explained in more detail in Subsection 1.2.3.

1.2.2. Messenger RNA

First, we want to answer the question, how *ribonucleic acid* (RNA) looks like in comparison to DNA. As DNA, the four basic subunits of RNA are nucleotides. But there are two basic differences between the subunits for DNA and RNA. The nucleotides which build RNA contain as sugar *ribose*, whereas the nucleotides which build DNA contain as sugar deoxyribose. Furthermore, instead of the base thymine, the corresponding nucleotide for the formation of RNA contains as base *uracil* (U). Another important difference between DNA and RNA is that RNA is always single-stranded and very short, compared to DNA molecules (between 70 and 10,000 nucleotides).

How is now the information contained on the DNA in a gene transcribed into an mRNA? To accomplish this task, an *enzyme* is used. An enzyme deals as a *catalyst* for chemical reactions. This enzyme is *RNA polymerase*. It collides randomly with the DNA molecule but binds tightly once it recognizes the *promoter* of a gene. This specific DNA sequence deals as a starting point for RNA synthesis. The RNA polymerase then opens the DNA double helix at that point and one of the DNA strands is used as a template for complementary base-pairing

with the incoming nucleotides A, C, G, and U. Which one of the strands is used as a template is determined by the promoter. The RNA polymerase molecule then moves along the DNA and opens the DNA helix to provide a new template region for the formation of RNA. At the same time, the enzyme closes the DNA helix to displace the newly formed part of the RNA molecule. Thus, there exists always only a short region of a DNA/RNA helix. When the enzyme passes a second special sequence in the DNA, the *stop signal*, the RNA polymerase releases the DNA and the produced RNA. This produced mRNA molecule contains exon and intron sequences. The intron parts have to be cut out to obtain an mRNA molecule which codes directly for a protein. This process is called *RNA splicing*. It is performed with help of a multicomponent protein complex, the *spliceosome*. This complex binds an intron sequence and catalyzes the breakage and formation of covalent bonds between the two exons formerly bound to the intron. The introns are then degraded inside the nucleus and the obtained mRNA molecule leaves then the nucleus through one of the *nuclear pores* (see Figure 1.2) and enters the cytoplasm.

We have to mention here, that the above described transcription procedure omits the very important detail of how the transcription of specific regions of the DNA is regulated. Until now, we only mentioned, that the RNA polymerase finds the promoter in a random fashion. However, this approach may transcribe genes which are not needed at the specific moment and, on the other hand, may not transcribe genes which are needed to be expressed, because specific proteins are needed by the cell. To regulate this, gene regulatory proteins, also called *transcription factors*, bind to the promoter regions of a gene to, either activate or inhibit the transcription of it. We will come to this point in more detail in the next subsection.

1.2.3. Proteins

The question to be answered in this subsection is: How do we obtain from the information contained in a gene the protein we need? First, we will describe, how proteins are synthesized from mRNA molecules. Second, we will give an overview about regulation of gene expression in general and how proteins play a crucial role in it.

Protein synthesis

Essential molecules for this process are the previously introduced tRNA and rRNA. The tRNA molecules carry *amino acids*. For each of the 20 used amino acids in biological organisms there exist at least one specific tRNA molecule. This is a good point to mention, that proteins are nothing different than long chains of amino acid molecules linked by *peptide bonds*. An amino acid always contains a *carboxyl group* and an *amino group* and the carboxyl group from one amino acid then binds via a peptide bond to the amino group of another amino acid.

The tRNA binds at one end to a *codon*, a sequence of three following nucleotides, of the mRNA and at the other ends it binds to the amino acid specified by that codon. Since there are 4 different nucleotides, there are $4^3 = 64$ different codons coding for 20 different amino acids. Three of the 64 codons are so-called *stop codons*. They serve as points on the mRNA where protein synthesis is terminated. The remaining codons all code for a specific amino acid. Thus, for almost all amino acids there is more than one codon associated with it.

The other important molecules for protein synthesis are ribosomes. They are complexes of rRNA and protein molecules. They serve as guide for the correct protein synthesis, i.e., that all codons are read in the right order and not single nucleotides are skipped. Ribosomes have

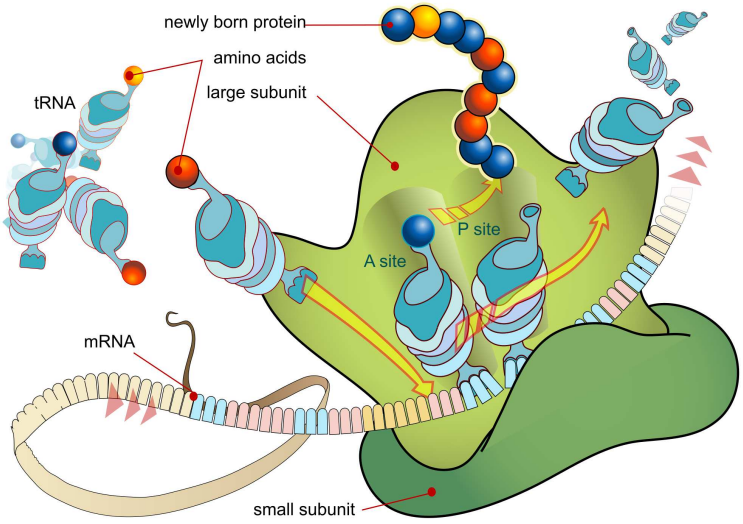


Figure 1.5.: *A schematic view of protein synthesis: tRNA molecules bind to the P- and the A-site of the ribosome and the corresponding amino acids form peptide bonds between them. The ribosome moves along the mRNA molecule and binds new tRNA molecules. While the ribosome is doing so, the protein grows and is released, once a stop codon on the mRNA is reached. Source:* `http://en.wikipedia.org/wiki/Translation_(biology)`

three binding sites for RNA molecules, one for the mRNA and two for two tRNA molecules, the *P-site* and the *A-site*. The two binding sites for the two tRNA molecules are that close, that the two tRNA molecules are forced to bind to adjacent codons and do not skip any nucleotides. Ribosomes consist of two subunits. The small subunit binds to the mRNA and tRNA molecules. The large subunit is used for the catalyzation of the formation of the peptide bonds in the protein.

The process of protein synthesis is depicted in detail in Figure 1.5. This process starts with the binding of an *initiator tRNA* to the P-site of the ribosome. The initiator tRNA always carries *methionine*, the amino acid that starts a protein chain. As a next step, the small subunit of the ribosome binds to the $5'$ end (compare with Figure 1.3 (Left)) of the mRNA molecule and scans this molecule, until the *start codon* is reached. The start codon is always a AUG codon, which is the codon that codes for methionine. Next, the initiator tRNA binds to this codon and a matching tRNA molecule[2] is bound to the next codon and to the A-site of the ribosome. Afterwards, the carboxyl group of methionine forms a peptide bond with the amino group of the amino acid linked to the tRNA molecule in the A-site. Subsequently, the ribosome moves exactly one codon along the mRNA molecule. Now the tRNA molecule in the P-site binds a chain of two amino acids. The tRNA occupying the A-site before is now occupying the P-site. The process described in the last few lines continues until one of the three stop codons is reached. When this happens, a *release factor* binds to the stop codon and adds a water molecule to the amino acid chain instead of another amino acid. This leads to the release of the protein chain into the cytoplasm. Furthermore, the ribosome releases the mRNA molecule and dissociates into two subunits.

[2] "Matching" is meant in the sense, that the tRNA molecule contains the complementary nucleotides given the codon on the mRNA molecule.

Regulation of gene expression

One distinguishes between two types of gene expression control:

1. *transcriptional control*: control when and how often a given gene is transcribed
2. *post-transcriptional control*: everything else in the process of gene expression, except the transcription of a gene, e.g., control which mRNA is transported into the nucleus, control which mRNA in the cytoplasm is translated into a protein, control of the degradation of mRNA molecules, alternative splicing, ...

Transcriptional control is the predominant form of gene regulation, which makes sense, since then no useless intermediate products are formed. Therefore, I will focus in more detail on transcriptional control.[3] However, one has to keep in mind, that the control of gene expression is a highly complex procedure and more complicated than shown in Figure 1.4.

Since this section is about proteins, you probably already guessed it, that for transcriptional control proteins play an essential role. As mentioned at the end of the previous section, three components are needed for transcriptional control of gene expression:

1. *gene control region*, which is a combination of short stretches of DNA of defined sequence, and consists of the promoter and *regulatory sequences*, which can be spread thousands of nucleotide pairs away from the promoter
2. general transcription factors
3. specific transcription factors

We will explain in the following, how transcription factors bind to these short stretches and how this binding enhances or represses the transcription of a specific gene.

In eukaryotic cells, the initiation of transcription cannot be made with the RNA polymerase enzyme alone. It has to form a complex together with general transcription factors, which are proteins that are abundant in the nucleus. This complex forming process can be slowed down or speeded up in response to regulatory signals[4] which are carried out by specific transcription factors. These specific proteins bind to the general transcription factors or the RNA polymerase enzyme and enhance resp. suppress the formation of the complex consisting of RNA polymerase and general transcription factors. This then enhances resp. suppresses the transcription of mRNA molecules.

An important fact is, that a specific transcription factor has to be bound to a regulatory sequence on the DNA to be able to influence transcription of its target gene. If the regulatory sequences are close to the starting point for the RNA polymerase enzyme, the binding of gene regulatory proteins to the RNA polymerase/transcription factors-complex is easy. However, what if the regulatory protein binds to a regulatory sequence far away from the promoter? In this situation, how does it regulate the complex formation needed for transcription? This is achieved by *DNA looping*. The name exactly says what it is about: the DNA is forming a loop to allow the protein to bind to a regulatory sequence far away from the promoter to interact directly with parts of the forming RNA polymerase/transcription factors-complex.

To make the control of gene regulation even more complicated, several gene regulatory proteins can assemble into small complexes and then activate or repress transcription. This

[3] For more details on post-transcriptional control see [ABL+94] pages 453 ff.
[4] For more information concerning cell signaling see [ABL+94] Chapter 15.

formation of a complex can either happen in solution, but also, if the proteins do not bind in solution, it can happen on the DNA. Thus, two proteins cooperate to bind to the DNA, and then they offer a new surface, where another regulatory protein is able to bind to and there is able to stimulate transcription.

As a final remark, it remains to mention, that for some genes in mammalian cells the gene control region may be up to $50,000$ nucleotide pairs long. This control region contains several regulatory sequences, each to regulatory proteins may bind to and all are able to participate in the gene regulation process.

Although, the focus was on transcriptional control, I will mention one aspect of post-transcriptional control, namely *alternative splicing*. We saw in Section 1.2.2 that an immature mRNA molecule contains introns and exons and with splicing the introns are removed such that mature mRNA molecules only contain exons with encode for a specific protein. But this splicing procedure is not unique. Some exons may be also cut out of the immature mRNA molecule, such that the corresponding mature mRNA is encoding for a different protein. The regulation of splicing in general, and also for alternative splicing, is also done by protein complexes. For more details on alternative splicing see [MCS05].

Altogether, we see that gene regulation is a highly complex procedure.

1.3. Experimental setups

In this section, the experimental setups will be described, which are used to measure the abundance of gene expression products, i.e., mRNA molecules and proteins, in the cell. The first three methods detect the amount of mRNA molecules, whereas with a *western blot* one detects the amount of proteins in the cell.

1.3.1. Northern Blot

The general principle of northern blotting is to separate RNA molecules by size and detect them by *hybridization*, i.e., complementary single strands of nucleotide chains will reform double helices of the sequence of the target mRNA. This trick for detection was proposed by Southern in 1975 [Sou75]. We will now describe northern blotting in more detail.

It consists of eight steps [Tra96], which are also outlined in Figure 1.6: First, the RNA from a specific tissue under consideration is extracted. Second, mRNA molecules are isolated. These are then separated on the basis of molecular size by *gel electrophoresis* as a third step. Fourth, the mRNA molecules of interest are blotted onto a membrane. Most often, a positively charged nylon membrane is used, since the negatively charged nucleic acids bind very well to it. To ensure covalent bonds between RNA molecules and membrane, the RNA molecules have to be immobilized on the membrane, either by exposure to heat or to *ultraviolet light* as a fifth step. Sixth, the hybridization probe with the *complementary DNA* (cDNA), i.e., a strand with the complementary nucleotides of the mRNA of interest, is prepared. These probes are labeled, most common, with radioactive isotopes or with molecules emitting light as a result of a chemical reaction. Labeling is needed for the detection in the last step. The next to last step is the hybridization with the probe. It is followed by washing away the non-binding hybridization probes. Here one sees the importance of the immobilization of the RNA molecules on the membrane in step five. Otherwise, everything would be washed away. The eighth and last step is the detection of the binded labeled probes and its quantification. The detection method depends, of course, on the labeling method in step six. The quantification

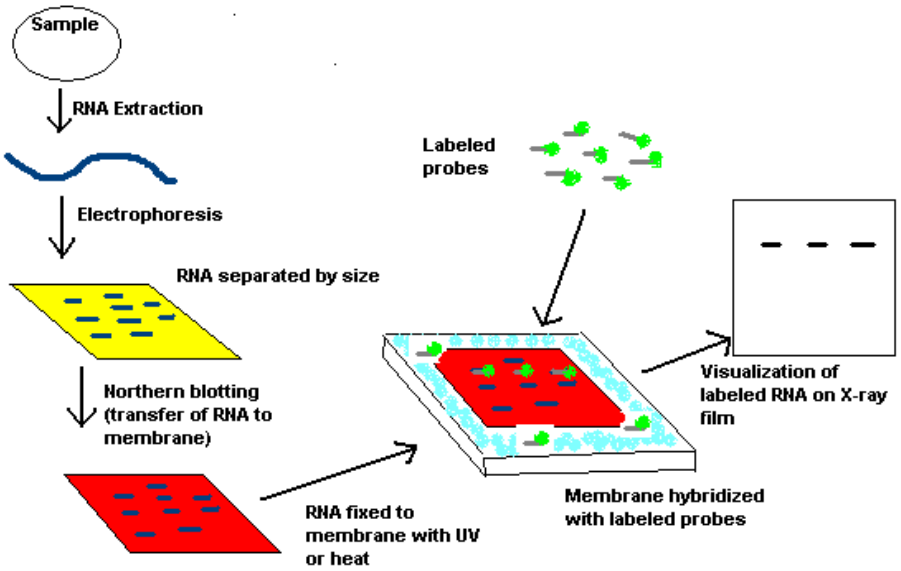


Figure 1.6.: *A schematic view of northern blotting. The eight steps of the northern blotting procedure are described in detail in Section 1.3.1. Source:* `http://en.wikipedia.org/wiki/Northern_blotting`

is done by *densitometry*, a procedure which measures how much light passes a material, in this case, the membrane with the bound labeled cDNA probes. The less light passes the membrane, the higher the amount of labeled cDNA probes and thus the higher was the amount of the mRNA molecule of interest in the tissue at the beginning.

A clear advantage of this method is its simplicity and its relatively high *specificity*, i.e., the method is conservative and specifies mostly mRNA molecules with high abundance[5]. The main disadvantage is the risk of mRNA degradation during gel electrophoresis which may lead to the second disadvantage, the low *sensitivity*, i.e., for a lot of mRNA molecules the given results are too low and thus not properly detected, compared to that of *qPCR* [SME+09]. Compared to *DNA microarrays*, northern blotting usually only considers a small number of different genes, whereas microarrays are able to visualize thousands of genes. However, northern blotting is capable to detect small changes in gene expression that microarrays cannot, at least for some examples [TMIY01].

1.3.2. Real-time quantitative polymerase chain reaction

Real-time quantitative polymerase chain reaction (qPCR) is a quantitative way to measure the amount of present mRNA molecules. This is done by, first, amplification of the corresponding cDNA molecules. During this amplification the amount of double-stranded DNA molecules is measured by the increase of the fluorescence of a dye, e.g., *ethidium bromide* or *SYBR green*, which only fluoresce when bound to a double-stranded DNA molecule. These steps and two

[5] Compare the terms "specificity" and "sensitivity" used here with the same terms used in Section 3.3.1

quantification methods will be described in the following in more detail. The description is based on [VVF08, Hun10]. For a more technical experiment protocol see [NHB06].

Polymerase chain reaction

The goal of a *polymerase chain reaction* (PCR) is the amplification of short DNA sequences. One starts with the double-stranded DNA molecule one wants to amplify, a pair of *primers* and a heat-stable DNA polymerase inside a solution. Primers are short sequences of nucleotides, of about 20 nucleotides in length, which serve as starting points for the amplification and are complementary to the region of the DNA that is of interest. PCR is performed in several cycles, where in each cycle the amount of the DNA molecule of interest is doubled in the optimal case of 100% reaction efficiency. Every cycle contains of three steps:

1. Denaturation
2. Annealing
3. Elongation

In the *denaturation* step the reaction is heated to a high temperature of more than 90 degrees such that the two strands of the DNA separate. In the *annealing* step, the reaction is cooled down to a temperature between 50 and 60 degrees where the primers bind to the specific sites of the single-stranded DNA molecules obtained from the denaturation step. As a third step, the temperature of the reaction is raised to usually 72 degrees, which is the temperature where the heat-stable DNA polymerase performs the elongation of the primer to a new DNA strand. This procedure is illustrated in Figure 1.7.

Of course, to quantify mRNA molecules one first has to reverse transcribe the mRNA into cDNA molecules. Otherwise, no double-strands could be created.

Quantification of mRNA

Now that several cycles of PCR are performed, how do we quantify the amount of mRNA molecules we had at the beginning? Mainly, there are two approaches to do this:

1. measure absolute levels (done with the *standard curve method*)
2. measure relative levels of a target gene versus a reference gene (done with the $2^{-\Delta\Delta C_T}$ *method* [LS01])

Before we will describe these quantification methods in more detail, we first need to know what the *crossing threshold* (C_T) means. We saw it already in the exponent of the $2^{-\Delta\Delta C_T}$ method. For this, we have to look at the three phases of the PCR process. In the first phase, the process is exponential, i.e., the amount of double-stranded DNA molecules doubles in every cycle of the PCR, since all reaction reagents are abundant and the reaction efficiency is close to 100%. In the second phase, the process is linear, because the reagents become limiting although the produced DNA molecules are accumulated. In the third phase, the process reaches a plateau where the reaction stops because of exhausted reagents.

It is obvious, that one wants to measure the amount of DNA molecules when the reaction is in the exponential phase, because of the perfect reaction efficiency. But at the beginning of a PCR the amount of products may be hard to detect. Thus, a crossing threshold C_T has

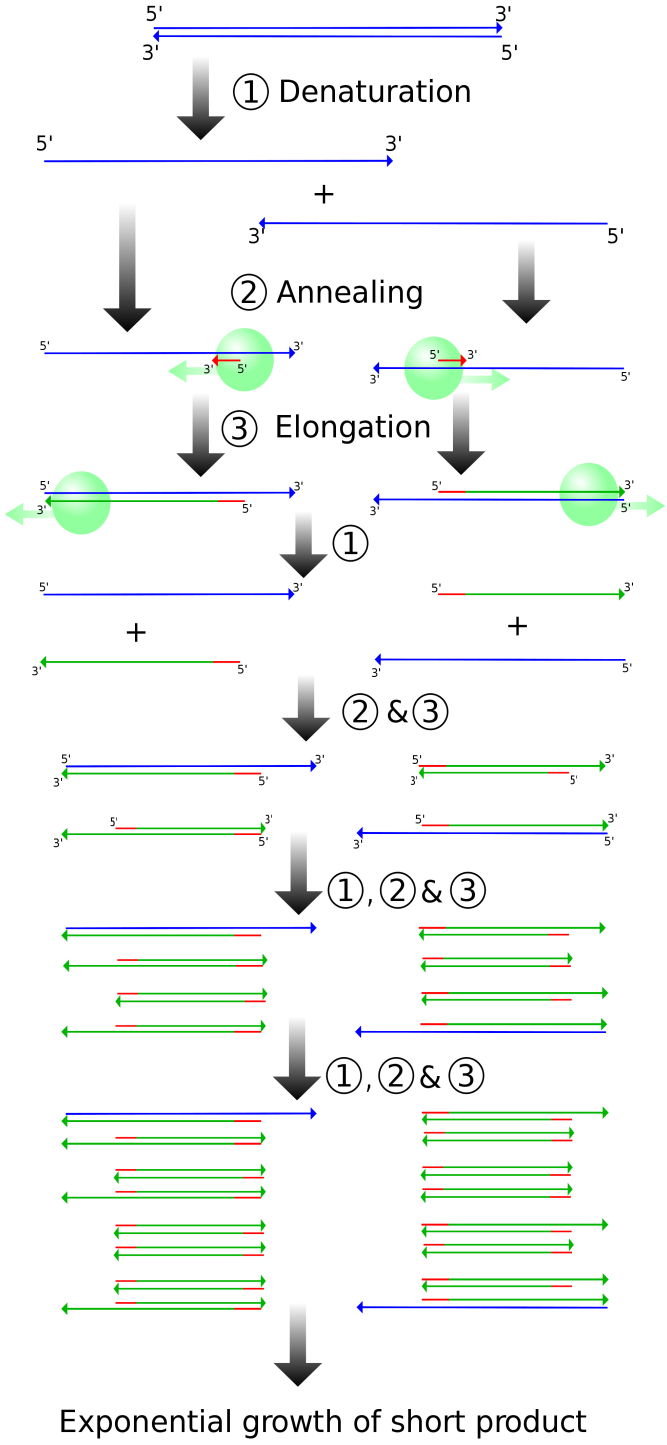


Figure 1.7.: *Illustration of the principle of a PCR. The three steps denaturation, annealing and elongation are described in more detail in Section 1.3.2. Source:* `http://en.wikipedia.org/wiki/Polymerase_chain_reaction`

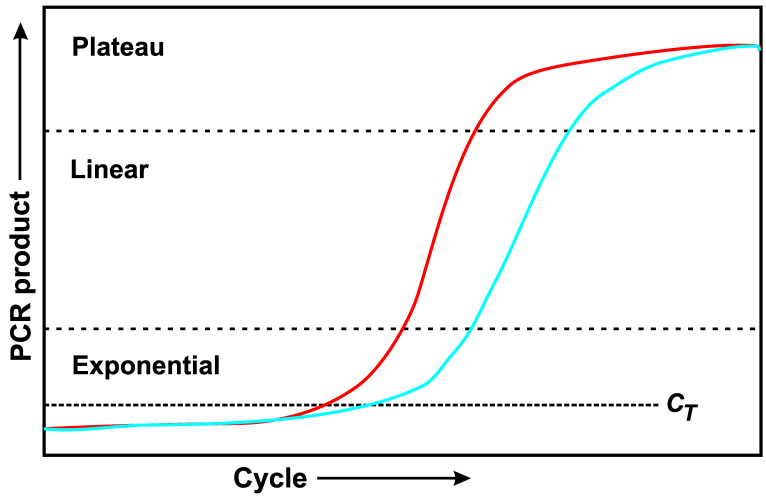


Figure 1.8.: *The three phases of a PCR reaction: It starts with an exponential phase. When the reagents start to become limiting, it is in the linear phase and when the reagents are exhausted, the reaction reaches a plateau. Also the crossing threshold C_T is depicted. This figure is adapted from [VVF08].*

to be specified, which gives the amount of products that are detectable properly. Of course, this threshold varies between different experimental settings and is usually a parameter to be set by the experimenter. It should be set as low as possible to ensure, that one is still in the exponential phase of the reaction. In Figure 1.8 the three phases of a PCR reaction are depicted. Additionally, one can see, that at the beginning of the reaction, the fluorescence signals are that low that the exact amount of DNA molecules is hard to detect.

Now we can begin to explain the two approaches for quantification introduced earlier. For both methods, we assume that the C_T is given by the experimenter. Let's start with the standard curve method. The basic idea of it is to perform PCRs with known amounts of DNA molecules. For this, one starts with a certain amount and then takes dilutions of this probe, e.g., in every dilution step, the amount is reduced by one half or by one tenth. For all this diluted probes a PCR is performed and from the fluorescence measurements resulting in curves like in Figure 1.8 one specifies the cycle number, where the curve crosses the C_T line. By plotting now the concentration of the probes against the obtained cycle numbers one will be able, in the optimal case, to draw a straight line through these points (see Figure 1.9). Since in reality, one does not have the ideal case, *linear regression* is performed to obtain this straight line. With this standard curve, one is able to determine the initial amount of DNA molecules of a probe of interest. One just has to observe the cycle number, where for this probe the PCR crosses the C_T line. With the standard curve, one immediately gets the initial amount that was present in the probe. This is illustrated in Figure 1.9 with the dotted lines: We observed that the PCR crossed the C_T line with the cycle number 13. We calculate with the standard curve, that the initial concentration had the value 3. It remains now only to calculate $2^3 = 8$ [a.u.] to obtain the concentration of interest. Of course, this quantification method depends crucially on the standard curve obtained.

The most common method for relative quantification given a reference sample, is the $2^{-\Delta\Delta C_T}$ method. This method is based on two crucial assumptions:

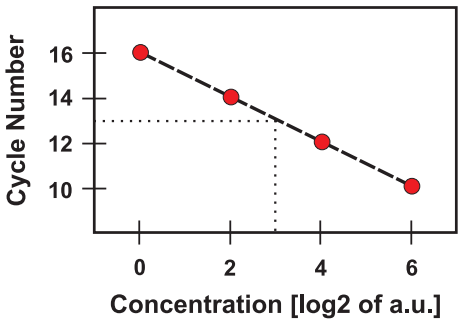


Figure 1.9.: *An idealized standard curve: On the x-axis the logarithmized concentration values and on the y-axis the corresponding observed cycle number at which the reaction curve crosses the C_T line are depicted. The dotted lines illustrate how the initial amount of a probe can be determined: First, one observes the C_T number of this probe and then one can extract from the straight line, what the concentration was.*

1. the reaction efficiency is 100%

2. it exists a gene which is expressed at a constant level in different samples

Concerning the first assumption one can say, that the crossing threshold C_T has to be set as low as possible to guarantee that the PCR is still in the exponential phase. To fulfill the second assumption, usually a *housekeeping gene* is used as a reference gene. Housekeeping genes are needed for the performance of basic functions of a cell. And some of them are expressed at constant levels in all cells such as *β-actin* which makes them ideal candidates to serve as reference genes for the relative quantification of mRNA molecules of interest. How does this method work now? Basically, one has two PCRs which deal as a *calibrator* for the reaction. For example, in the case of interest how a virus infection affects the expression of a certain gene of interest, one can use samples which are not infected with this virus as a calibrator. We need one PCR to measure the amount of the gene of interest and another one to measure the amount of the reference gene in the calibrator. Now we have to perform two PCRs for the gene of interest in the sample of interest: one for the gene itself and again one for the reference gene. For all these four reactions the corresponding values, where the C_T line is crossed, are measured. Let us name these values here $C_{T(Virus,I)}$, $C_{T(Virus,R)}$ for the virus infected samples for the gene of interest (I) and the reference gene (R). And furthermore, $C_{T(healthy,I)}$ and $C_{T(healthy,R)}$ for the healthy samples for the gene of interest and the reference gene. Then the relative amount of the gene of interest compared to the calibrator is

$$2^{-\left(C_{T(Virus,I)}-C_{T(Virus,R)}-(C_{T(healthy,I)}-C_{T(healthy,R)})\right)} =: 2^{-\left(\Delta C_{T(Virus)}-\Delta C_{T(healthy)}\right)} =: 2^{-\Delta\Delta C_T}.$$

Now one can clearly see why this method is called the $2^{-\Delta\Delta C_T}$ method. In Figure 1.8 we see two curves for a PCR reaction. Let's assume the red one represents the reference gene and the blue one represents the gene of interest in the case of healthy samples. Then, $\Delta C_{T(healthy)}$ is represented by length of the dashed line located between the crossing of the red and blue line with the C_T dashed line. A similar picture will appear, of course, for the case with the virus infected samples.

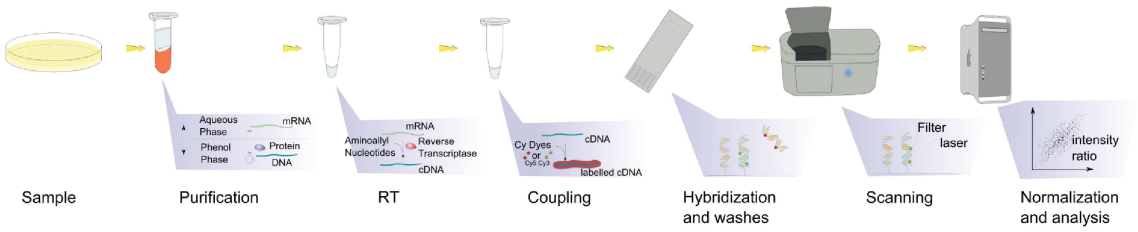


Figure 1.10.: *Scheme of the general flow of a DNA microarray experiment. From a sample, first the mRNA molecules have to be separated from the rest material in a cell. These are then reverse transcribed (RT) to get cDNA molecules. As a next step, these cDNA molecules are labeled and are ready to start the hybridization. After washing away the not binded molecules, the fluorescence signals are reported and, as a last step, it remains to normalize and analyze the obtained data. Source:* `http://en.wikipedia.org/wiki/DNA_microarray`

1.3.3. DNA Microarrays

The technology for DNA microarrays was introduced in 1995 by Schena *et al.* [SSDB95]. Since the literature for microarray experiments is vast, I stick to the three references [SD01, Sto05, WH08]. As northern blotting it is based on hybridization of cDNA molecules. Well, this is only half of the truth, since there exist two common platforms for DNA microarray experiments:

1. cDNA microarrays
2. oligonucleotide microarrays

In northern blotting the cDNA molecules of interest are immobilized and the hybridization probes are labeled with a radioactive or light-emitting marker. In contrast, and this is the main difference between these two methods, for DNA microarray analysis the probes are immobilized and the mRNA molecules are labeled with a fluorescent marker. This gives the possibility to locate different probes to different locations of a plate. And by labeling the whole cellular RNA, one is able to detect the expression of the whole genome.

Now let's come to the difference between the two platforms for DNA microarrays. In the first one, one creates cDNA molecules, usually by PCR. Then these probes are spotted onto a solid surface, most often a glass slide. In contrast, for oligonucleotide microarrays, oligonucleotides are synthesized directly on the plate, i.e., *in situ*.

In Figure 1.10 we see a typical procedure of a DNA microarray experiment after the probes are fixed on a plate. First, the mRNA molecules are isolated from the sample and reverse transcribed to obtain the corresponding cDNA molecules. These are then labeled with a fluorescent marker. The labeled molecules are used for hybridization and after washing away the unused experimental material, the fluorescent signals are detected with *high-throughput screening* methods[6]. The obtained data has now to be analyzed. This a very crucial step, because of the high-dimensionality of the data.

The first point in data analysis of DNA microarray experiments is, that usually not the absolute values are considered, but ratios between the sample of interest to a reference sample.

[6]Because of the vast amount of methods and protocols available, I will only give the reference [JB09] as a starting point to read.

A typical example in microarray studies is the comparison of normal samples and disease samples (see Section 2 in [WH08]). This comparison can be done in two separate microarray experiments or in one experiment, where RNA molecules from both samples are labeled, each with another fluorescent marker. Usually, for this *two-color microarrays* the markers Cy3 (emitting green light) and Cy5 (emitting red light) are used.

It has been shown in studies, that results between different platforms, different laboratories and also between different analysis methods differ a lot (see references in [WH08]). Of course, there are a lot of possibilities which may explain the different results from the technical point of view: different probe preparation, different labeling techniques, different hybridization protocols and as very important part the different quality of the tissue / sample, i.e., the amount of mRNA molecules expressed in the sample may be different. For two-color microarrays the additional bias of different chemical properties of the two dyes used is introduced. Also it may occur that targets hybridize with probes that are not exactly the complementary strand. This is also known as *cross hybridization*.

But how can we circumvent, at least partially, the mentioned problems? The problem with the bias introduced by the usage of two chemically different dyes can be easily solved by performing the same experiment again where the labeling of the samples is reversed and the obtained results from both experiments are averaged. In other words, a *dye-swap* is performed. To impede cross hybridization, good probe design is essential. There exist a lot of different methods to perform this task (see e.g. [RHMP05, GR08, SZK+08]). The Affymetrix[7] platform, in contrast, uses probes that are designed in pairs to avoid cross hybridization. One sequence is the complementary strand to the target and the other sequence differs from the exact complement by two mutations. The signal from the modified probe can be seen as background signal and accounts for non-specific binding [Sto05]. As a very promising approach to develop standards and quality controls for microarray experiments the *MicroArray Quality Control* (MAQC) project was established in 2006 [Con06, Con10]. In MAQC-I is has been shown that external RNA controls included in the microarray experiments can be used as predictors of technical performance. Another finding was that to produce reproducible and valid results, biologists have to increase their algorithmical skills [edi10]. The same was mentioned also by Walker and Hughes [WH08]. With the high complexity of the data obtained and the vast amount of methods and software packages available helping with the data analysis, this seems to be a reasonable concern.

Altogether, until no *golden standard* protocols for DNA microarray experiments are defined, the conclusions obtained for individual genes have to be validated with northern blotting, qPCR or western blotting.

1.3.4. Western Blot

Until now, we just described methods how to detect mRNA molecules. Western blotting, in contrast, is a method to detect the amount of present proteins.

The first steps of the underlying experimental protocol, first described in 1979 [TSG79], are similar to the first three steps of the northern blotting procedure. The difference, of course, is that proteins instead of mRNA molecules are separated. After separation the proteins are blotted on a membrane. Usually, *electroblotting* is used to pass proteins from the gel to a membrane made of *nitrocellulose*. The passing is done by putting the gel together with the

[7] `http://www.affymetrix.com`

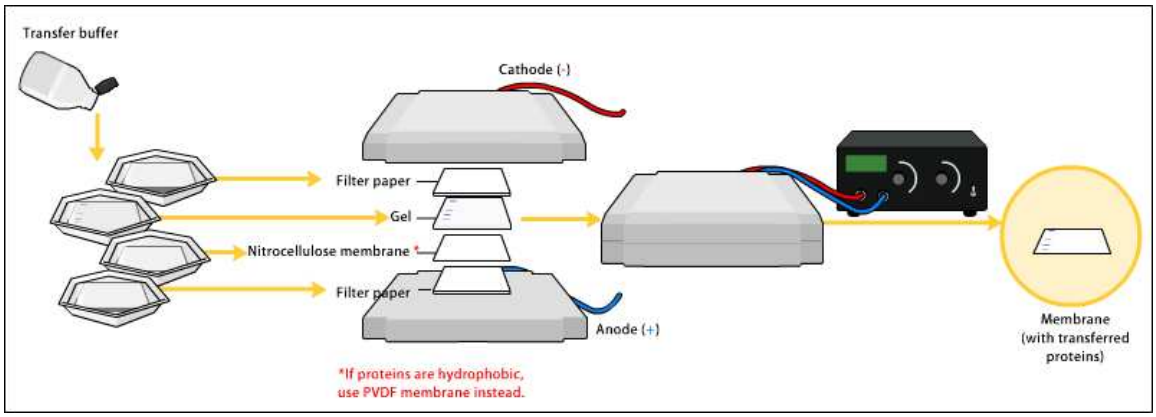


Figure 1.11.: *The electroblotting procedure to fix proteins to a membrane: The gel, which contains the proteins together with a membrane and filter paper are put into a magnetic field. The proteins then move from the cathode to the anode and stick to the membrane. Source:* `http://en.wikipedia.org/wiki/Western_blot#cite_ref-Corley2005_5-0`

proteins, the membrane and some filter papers into an *electrical field*, where the proteins then move from the *cathode* (+ pole) to the *anode* (− pole) and stick to the membrane (see Figure 1.11).

The detection of the amount of proteins bound to the membrane is usually done in two steps. First, a *primary antibody* is used to bind specifically to the protein of interest. After washing away the unbound primary antibody, as the second step a labeled *secondary antibody* is used to bind to the primary antibody. The labeling is most often done with a fluorescent dye. Of course, the unbound secondary antibody has to be washed away to detect the amount of protein correctly.

CHAPTER 2

Mathematical background

> Mathematics, rightly viewed, possesses not only truth, but supreme beauty.
>
> *(Bertrand Russell)*

In this chapter we introduce the mathematical fundamentals to understand the methods described in Part II and Part III. We start with the basic definitions and concepts in probability theory. With this foundation, we will provide some basic knowledge in information theory and Markov chain theory. Additionally, we will introduce the concept and notations of ordinary differential equations and splines and provide general ideas about the numerical methods to solve differential equations and to obtain splines and smoothing splines.

2.1. Probability theory

In this section, I will give the main definitions and results needed to understand the following sections on information and Markov chain theory. This section is based on [LW00, Dur05]. Of course, there are a lot of other books dealing with probability theory, which can also be used to get an overview and deeper understanding of the topic.

2.1.1. Probability spaces

The essential concept in probability theory is a *probability space*:

Definition 2.1.1. A *probability space* is a triple $(\Omega, \mathcal{F}, p)$, where Ω is a set of outcomes, $\mathcal{F}$ is a set of events (a σ-algebra), and $p : \mathcal{F} \to [0, 1]$ is a function that assigns probabilities to events (a *probability measure*).

Definition 2.1.2. A *σ-algebra* $\mathcal{F}$ is a collection of subsets of Ω that satisfy

1. if $A \in \mathcal{F}$, then $A^c \in \mathcal{F}$, and

2. if $A_i \in \mathcal{F}$ is a countable sequence of sets, then $\cup_i A_i \in \mathcal{F}$.

Definition 2.1.3. A *probability measure* p is a function $p : \mathcal{F} \to \mathbb{R}$ with

1. $p(\Omega) = 1$, and
2. $p(A) \geq p(\emptyset) = 0$ for all $A \in \mathcal{F}$, and
3. if $A_i \in \mathcal{F}$ is a countable sequence of disjoint sets, then

$$p\big(\cup_i A_i\big) = \sum_i p(A_i).$$

A probability space of special interest is the probability space on the real line $(0, 1) \subset \mathbb{R}$. As a σ-algebra on the real numbers $\mathbb{R}$ the *Borel sets* $\mathcal{B}$ are used. The Borel sets are defined as the smallest σ-algebra containing the open sets of $\mathbb{R}$. As a measure on $\mathbb{R}$ the *Lebesgue measure* $\lambda_\mathbb{R}$ is used. It is defined as the only measure on $\mathbb{R}$ with $\lambda_\mathbb{R}\big((a, b]\big) = b - a$ for all $a < b$. Note, that we have $\lambda_\mathbb{R}(\mathbb{R}) = \infty$. Thus, the space $(\mathbb{R}, \mathcal{B}, \lambda_\mathbb{R})$ is NOT a probability space. But with the restriction to the unit interval, we will obtain the probability space $(\Omega, \mathcal{F}, p) = \big((0, 1), \{A \cap (0, 1) : A \in \mathcal{B}\}, \lambda_\mathbb{R}(B) \text{ for } B \in \mathcal{F}\big)$.

Remark 2.1.4. The Borel sets $\mathcal{B}$ and the Lebesgue measure $\lambda_\mathbb{R}$ are also defined on the n-dimensional real space $\mathbb{R}^n$. The Borel sets $\mathcal{B}$ are defined as for the real numbers as the smallest σ-algebra containing the open sets of $\mathbb{R}^n$. Furthermore, as for the real numbers the Lebesgue measure on $\mathbb{R}^n$ is defined as the only measure with

$$\lambda_\mathbb{R}\big((a_1, b_1] \times \cdots \times (a_n, b_n]\big) = (b_1 - a_1) \cdot \cdots \cdot (b_n - a_n)$$

for all a_i, $b_i \in \mathbb{R}$ and $a_i < b_i$ for all $i \in \{1, \ldots, n\}$.

Sometimes we will equip the same set of outcomes Ω and a σ-algebra $\mathcal{F}$ with different probability measures. For this purpose, we will call the pair $(\Omega, \mathcal{F})$ a *measurable space*.

2.1.2. Random variables and random vectors

Now that we defined a σ-algebra and a measure on $\mathbb{R}$, we are able to give the definition of a *random variable*:

Definition 2.1.5. Let $(\Omega, \mathcal{F}, p)$ be a probability space. A function $X : \Omega \to \mathbb{R}$ is called a *random variable*, if for every Borel set $B \subset \mathbb{R}$ we have

$$X^{-1}(B) = \{\omega : X(\omega) \in B\} \in \mathcal{F}.$$

How can we now describe random variables in a more descriptive way? As a first point, we look at the *distribution function* of a random variable:

Definition 2.1.6. Let X be a random variable on the probability space $(\Omega, \mathcal{F}, p)$. The *distribution function* $F : \mathbb{R} \to [0, 1]$ of X is defined as $F(x) = p(X \leq x)$.

Remark 2.1.7. For continuous random variables X we can characterize the distribution function with help of a *density function* $f : \mathbb{R} \to \mathbb{R}$. If and only if $f(x) \geq 0$ for all $x \in \mathbb{R}$ and $\int f(x)\,dx = 1$, then

$$F(x) := \int_{-\infty}^{x} f(y)\,dy$$

defines a distribution function of X.

Example 2.1.8. *1. Of special interest is the* normal distribution*, also called* Gaussian distribution*, $\mathcal{N}(\mu, \sigma^2)$ with* mean μ *and* variance σ^2*. It has the density function*

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}}\,\mathrm{e}^{-\frac{(x-\mu)^2}{2\sigma^2}}\,. \tag{2.1}$$

If we consider random variables with density function (2.1)*, we will write $X \sim \mathcal{N}(\mu, \sigma^2)$.*

2. Another important density function belongs to the gamma distribution $\mathrm{Gamma}(r, a)$ *with* shape parameter r *and* rate parameter a*. It has the density function*

$$f(x) = \frac{1}{\Gamma(r)} a^r x^{r-1}\,\mathrm{e}^{-ax} \tag{2.2}$$

for $x \geq 0$ and $f(x) = 0$ otherwise, where $\Gamma(r)$ denotes the gamma function

$$\Gamma(r) = \int_0^\infty t^{r-1}\,\mathrm{e}^{-t}\,dt. \tag{2.3}$$

We will write $X \sim \mathrm{Gamma}(r, a)$, if we consider random variables with density function (2.2)*.*

3. To introduce one more example for density functions, I take the beta distribution $\mathrm{Beta}(\alpha, \beta)$ *with* shape parameters α *and* β*. It has the density function*

$$f(x) = \frac{1}{\mathrm{B}(\alpha, \beta)} x^{\alpha-1}(1-x)^{\beta-1} \tag{2.4}$$

for $x \in [0, 1]$ and $f(x) = 0$ otherwise, where $\mathrm{B}(\alpha, \beta)$ denotes the beta function

$$\mathrm{B}(\alpha, \beta) = \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)}. \tag{2.5}$$

We will write $X \sim \mathrm{Beta}(\alpha, \beta)$, if we consider random variables with density function (2.4)*.*

Another possibility to describe random variables is with their *mean* and their *variance*, which we already saw for the normal distribution $\mathcal{N}(\mu, \sigma^2)$ in Example 2.1.8 (1).

Definition 2.1.9. Let X be a random variable on the probability space $(\Omega, \mathcal{F}, p)$.

1. The *expected value*, also called the *mean*, of X is defined as

$$E(X) := \int X\,dp,^{1} \tag{2.6}$$

if the integral is finite. It is often also denoted by μ.

[1] For more information about this type of integral see [Bau92] Chapter 2.

2. The *variance* of X is defined as
$$\mathrm{Var}(X) := E\Big(\big[X - E(X)\big]^2\Big) = E(X^2) - \mu^2, \tag{2.7}$$
if $E(X^2) < \infty$. It is often also denoted by σ^2.

Remark 2.1.10. 1. For discrete random variables X the mean is calculated as
$$E(X) = \frac{1}{n}\sum_{i=1}^{n} x_i,$$
where $\Omega = \{x_1, \ldots, x_n\}$.

2. For continuous random variables X the mean is calculated as
$$E(X) = \int x \cdot f(x)\, dx,$$
where $f(x)$ denotes the density function of X.

Example 2.1.11. *1. The mean of a normally distributed random variable $X \sim \mathcal{N}(\mu, \sigma^2)$ is $E(X) = \mu$ and the variance is $\mathrm{Var}(X) = \sigma^2$.*

2. The mean of a gamma distributed random variable $X \sim \mathrm{Gamma}(r, a)$ is $E(X) = r/a$ and the variance is $\mathrm{Var}(X) = r/a^2$.

3. The mean of a beta distributed random variable $X \sim \mathrm{Beta}(\alpha, \beta)$ is
$$E(X) = \frac{\alpha}{\alpha + \beta}$$
and the variance is
$$\mathrm{Var}(X) = \frac{\alpha\beta}{(\alpha+\beta)^2(\alpha+\beta+1)}.$$

Until now, we only looked at one random variable X. But we are also interested in *random vectors* $\mathbf{X} = (X_1, \ldots, X_n)$, $n \in \mathbb{N}$. The expected value of $\mathbf{X}$, also called the *mean vector*, is defined as
$$E(\mathbf{X}) := \big(E(X_1), \ldots, E(X_n)\big).$$
Analogously, the variance of $\mathbf{X}$ is defined as
$$\mathrm{Var}(\mathbf{X}) := E\big[(\mathbf{X} - \mu)^T(\mathbf{X} - \mu)\big],$$
if $\mu = E(\mathbf{X}) < \infty$ and $E(X_iX_j) < \infty$ for all $i, j \in \{1, \ldots, n\}$. This symmetric $(n \times n)$-matrix is also denoted by Σ and called *covariance matrix*. The diagonal entries in Σ are the variances $\mathrm{Var}(X_i)$, $i \in \{1, \ldots, n\}$ and the other entries are the *covariances* $\mathrm{Cov}(X_i, X_j)$, $i, j \in \{1, \ldots, n\}$, which are defined as
$$\mathrm{Cov}(X_i, X_j) := E\big[(X_i - \mu_i)(X_j - \mu_j)\big], \tag{2.8}$$
where $\mu_i = E(X_i)$ and $\mu_j = E(X_j)$. If $\mathrm{Cov}(X_i, X_j) = 0$, we say X_i and X_j are *uncorrelated*. To have a normalized measure for the covariance between two random variables, we introduce the *correlation coefficient* $\mathrm{Corr}(X_i, X_j)$ as
$$\mathrm{Corr}(X_i, X_j) := \frac{\mathrm{Cov}(X_i, X_j)}{\sqrt{\mathrm{Var}(X_i)\,\mathrm{Var}(X_j)}}, \tag{2.9}$$
which is also denoted by ρ_{ij}.

Remark 2.1.12. 1. The correlation coefficient ρ_{ij} is always a number between -1 and 1.

2. The correlation coefficient $\mathrm{Corr}(X_1, X_2)$ between two random variables describes the linear dependency between them, i.e., if $X_1 = aX_2 + b$, we have $\mathrm{Cov}(X_1, X_2) = a\,\mathrm{Var}(X_1)$ and $\mathrm{Var}(X_2) = a^2\,\mathrm{Var}(X_1)$ and thus $\mathrm{Corr}(X_1, X_2) = 1$ for $a > 0$ and $\mathrm{Corr}(X_1, X_2) = -1$ for $a < 0$. We see that, if X_1 and X_2 are linearly dependent, the correlation coefficient takes one of the extreme values.

Now that we described elementary properties of random vectors, we will give the definition of the important property of *independence* of random variables:

Definition 2.1.13. 1. Let $X_1, \ldots, X_n$ be random variables defined on the same probability space $(\Omega, \mathcal{F}, p)$. They are called *independent*, if for all Borel sets $B_1, \ldots, B_n \in \mathcal{B}$ we have

$$p(B_1 \cap B_2 \cap \ldots \cap B_n) = p(B_1) \cdot p(B_2) \cdots p(B_n).$$

2. If $X_1, \ldots, X_n$ are independent and follow the same distribution, they are called *i.i.d.* (*independent and identically distributed*).

Remark 2.1.14. The definition of independence in 2.1.13 is equivalent to saying that

$$F_{X_1,\ldots,X_n} = F_{X_1} \cdots F_{X_n},$$

where $F_{X_1,\ldots,X_n}$ is the *joint distribution function* of $\mathbf{X}$ and $F_{X_1}, \ldots, F_{X_n}$ are the distribution functions of $X_1, \ldots, X_n$.

If the density functions $f_{X_1,\ldots,X_n}$ and $f_{X_1}, \ldots, f_{X_n}$ exist, another equivalent condition for independence is

$$f_{X_1,\ldots,X_n} = f_{X_1} \cdots f_{X_n}.$$

Remark 2.1.15. If X_1 and X_2 are independent, then they are also uncorrelated. But the inverse is not always true!

2.1.3. Conditional probability and Bayes' formula

An important concept in probability theory, which we will use in Chapter 5 and 7, is the concept of a *conditional probability*. We ask the question: What is the probability of an event A, if we already observed the event B? A formal definition is:

Definition 2.1.16. Let $(\Omega, \mathcal{F}, p)$ be a probability space, A, $B \in \mathcal{F}$ and $p(B) > 0$. The *probability of A under the condition B*, $p(A|B)$, is given by

$$p(A|B) := \frac{p(A \cap B)}{p(B)}. \tag{2.10}$$

Remark 2.1.17. 1. Let $(B_n)_{n\in\mathbb{N}}$ be a series of events with *pairwise disjoint* events, i.e., $B_i \cap B_j = \emptyset$ for all i, $j \in \mathbb{N}$, with $\Omega = \bigcup B_n$ and $p(B_n) > 0$ for all n. With (2.10) we now obtain the *formula of the total probability*:

$$p(A) = \sum_{n\in\mathbb{N}} p(B_n)p(A|B_n) \qquad \text{for all} \qquad A \in \mathcal{F}.$$

For continuous random vectors $\mathbf{X} = (X_1, \ldots, X_n)$ we will write, e.g., for $n = 2$ also

$$p(x_1) = \int p(x_1|x_2)p(x_2)\, dx_2.$$

In this case, we will say, that we are *integrating out* the variable x_2, where p denotes here the density of $\mathbf{X}$. Since later we will deal with parameterized densities, we will also say, that we are *integrating out the parameter* x_2.

2. Another important formula we get immediately with (2.10) is *Bayes' formula* for two events A and B:

$$p(B|A) = \frac{p(A|B)p(B)}{p(A)}. \tag{2.11}$$

2.1.4. Estimators

In the previous sections, we fixed a special random variable and defined properties of it. Now we want to look the other way around: we have *observations* $x_1, \ldots, x_n$ of a phenomenon under the same experimental conditions, and we want to infer properties about the underlying random variable X. Such observations are also called *samples* of X. We now define, that every x_i is a *realization* of a random variable X_i and all $X_1, \ldots, X_n$ are i.i.d. like X. Furthermore, we say that the distribution function of X is *parameterized* with a parameter vector $\theta \in \Theta \subset \mathbb{R}^k$, $k \in \mathbb{N}$. We now want to learn approximately with the observations $x_1, \ldots, x_n$ the parameter vector θ or a function of it $\tau(\theta)$. Now we are able to define an *estimator*:

Definition 2.1.18. Let $x_1, \ldots, x_n$ be realizations of the random variables $X_1, \ldots, X_n$ and let $\tau : \Theta \to \mathbb{R}$. We call the random variable

$$T_n^e : \mathbb{R}^n \to \mathbb{R} \tag{2.12}$$

the *estimator* for τ and the value $T_n^e(x_1, \ldots, x_n)$ is called the *estimate* for $\tau(\theta)$.

We are now interested in the quality of the estimators and their estimates. A reasonable measure for the quality of an estimator is the expected value of the squared error between it and $\tau(\theta)$. We have (see [LW00]):

$$E_\theta\Big(\big[T_n^e - \tau(\theta)\big]^2\Big) = \mathrm{Var}_\theta(T_n^e) + \big[E_\theta(T_n^e) - \tau(\theta)\big]^2. \tag{2.13}$$

The last term in (2.13) is called the *bias* of the estimator T_n^e. If the bias is zero for all possible $\theta \in \Theta$, we call an estimator *unbiased*. With (2.13) we thus see, that a good estimator is an unbiased estimator with low variance.

The next question we may ask concerning the quality of estimators is, how it improves for more realizations, i.e., for $n \to \infty$. For this purpose, we first have to introduce two different types of convergence for random variables: *convergence in probability* and *almost sure convergence*:

Definition 2.1.19. Let $(X_n)_{n=1}^\infty$ be a sequence of random variables defined on a probability space $(\Omega, \mathcal{F}, p)$ and X be a random variable defined on the same probability space.

1. We say, $(X_n)_{n=1}^{\infty}$ *converges towards* X *in probability*, if for all $\varepsilon > 0$ we have

$$\lim_{n\to\infty} p\big(|X_n - X| > \varepsilon\big) = 0.$$

   We will also denote this type of convergence by $X_n \xrightarrow{p} X$.

2. We say, $(X_n)_{n=1}^{\infty}$ *converges almost surely towards* X, if for all $\varepsilon > 0$ we have

$$p\Big(\limsup_{n\to\infty} \big(|X_n - X| > \varepsilon\big)\Big) = 0.$$

   We will also denote this type of convergence by $X_n \xrightarrow{a.s.} X$.

Remark 2.1.20. Almost sure convergence implies convergence in probability, but the inverse may not always be true! Intuitively, we can see, that almost sure convergence is a stronger property than convergence in probability, since for the latter only the probability that X_n and X are not equal decreases to zero for growing n, whereas for the former the probability for the events, for which, for growing n, X_n does not equal X, is zero (for more details see [Bau01]).

With Definition 2.1.19 we are now able to define quality measures for estimators for growing number of observations: *weak consistency* and *strong consistency*.

Definition 2.1.21. Let $(T_n^e)_{n=1}^{\infty}$ be a sequence of estimators and let $\tau : \Theta \to \mathbb{R}$.

1. We say, $(T_n^e)_{n=1}^{\infty}$ is *weakly consistent for* τ, if for all $\theta \in \Theta$ we have

$$T_n^e \xrightarrow{p} \tau(\theta).$$

2. We say, $(T_n^e)_{n=1}^{\infty}$ is *strongly consistent for* τ, if for all $\theta \in \Theta$ we have

$$T_n^e \xrightarrow{a.s.} \tau(\theta).$$

Example 2.1.22 (Maximum likelihood estimator)**.** *An often used estimator is the* Maximum Likelihood (ML) Estimator. *To define the* ML *estimator, we first have to define the* likelihood. *For this purpose, let* $(x_1, \ldots, x_n)$ *be a realization and let* $f(x_i) = p(X_i = x_i)$ *for all* $i \in \{1, \ldots, n\}$. *For continuous random variables the function* $f(\cdot)$ *thus denotes the density function of* X. *The* likelihood function *is defined as*

$$L(\theta; x_1, \ldots, x_n) = f_\theta(x_1) \cdots f_\theta(x_n), \qquad \theta \in \Theta. \tag{2.14}$$

A parameter value $\hat{\theta}$ *with*

$$L(\hat{\theta}; x_1, \ldots, x_n) \geq L(\theta; x_1, \ldots, x_n) \qquad \textit{for all} \qquad \theta \in \Theta$$

is called Maximum Likelihood estimate *for the parameter* θ *with the realization* $(x_1, \ldots, x_n)$. *An estimator, which gives us for all realizations the maximum likelihood estimate* $\hat{\theta}$*, is called the* Maximum Likelihood estimator *and we will sometimes write* $\mathrm{ML}(\theta) = L(\hat{\theta})$.

One may now ask: Are all ML *estimators (weakly or strongly) consistent? Unfortunately, this is not the case (see e.g. [Fer82]). But Wald [Wal49] gives conditions under which* ML *estimates are strongly consistent.*

2.2. Information theory

2.2.1. Definition of Shannon's information

In his seminal paper in 1948, Shannon introduced information theory to answer questions in communication theory [Sha48]. I give an overview over basic definitions and results from information theory that are needed to understand the experimental design method in Part III. Here the notation is used as in [CT06], which is also a good reference for further reading concerning the topic of information theory.

The main goal of this section is to give a measure for the uncertainty and information of a random variable. First we will do this for a discrete random variable X on a probability space $(\Omega, \mathcal{F}, p)$ with $\mid \Omega \mid = n$. Shannon introduced the measure of the uncertainty of X in an axiomatic way. He gave three properties which are reasonable for such a measure H:

1. H should be continuous in $p(x_i)$ for all $i \in \{1, \dots, n\}$
2. If all the $p(x_i)$ are equal, i.e., $p(x_i) = \frac{1}{n}$, then H should be a monotonic increasing function of n.
3. If a choice can be broken down into two successive choices, the original H should be the weighted sum of the individual values of H.

Then he proved:

Theorem 2.2.1. *The only H satisfying the three above assumptions is of the form*

$$H(X) := -K \sum_{i=1}^{n} p(x_i) \log p(x_i), \tag{2.15}$$

where K is a positive constant.

Proof. See [Sha48], Appendix 2. □

Definition 2.2.2. We call $H(X)$, as defined in (2.15), the *entropy* of a discrete random variable X. Furthermore, we define the *information* of X to be

$$\mathrm{I}(X) := -H(X). \tag{2.16}$$

Remark 2.2.3. We will now illustrate property 3 from above for a special case with three possibilities with corresponding probabilities $p_1 = \frac{1}{2}$, $p_2 = \frac{1}{3}$ and $p_3 = \frac{1}{6}$.

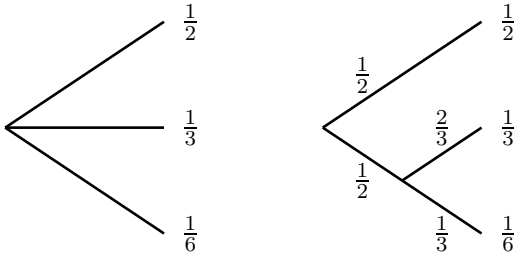


Figure 2.1.: *Decomposition of a choice with three possibilities*

On the left in Figure 2.1 we will write for the entropy $H(\frac{1}{2}, \frac{1}{3}, \frac{1}{6})$. Whereas on the right we can write for the entropy $H(\frac{1}{2}, \frac{1}{2}) + \frac{1}{2}H(\frac{2}{3}, \frac{1}{3})$. Note, that the second term is weighted with $\frac{1}{2}$, because we will make this choice only with the probability of $\frac{1}{2}$. The property 3 from above now states that we want to have

$$H\Big(\frac{1}{2}, \frac{1}{3}, \frac{1}{6}\Big) = H\Big(\frac{1}{2}, \frac{1}{2}\Big) + \frac{1}{2}H\Big(\frac{2}{3}, \frac{1}{3}\Big),$$

because although we take different routes, we end up with the same probabilities.

We have to note here, that the logarithm in the definition of entropy in (2.15) can be to the base 2, sometimes denoted by H_2, or to the base e, sometimes denoted by H_e. For the former, the entropy is calculated in *bits* and for the latter the entropy is calculated in *nats*. Because of the equality

$$\log_b x = \log_b a \log_a x,$$

we are always able to change the base of the logarithm in the definition of entropy and obtain

$$H_b(X) = (\log_b a) H_a(X)$$

for every discrete random variable X and all bases a, $b \in \mathbb{R}^+$.

The definition of entropy is still not satisfying, since we have a constant K in it. In the following we will use $K = 1$. This can be seen as a normalization for the case $n = 2$, where the logarithm is to the base 2 and we want to have

$$H\Big(\frac{1}{2}, \frac{1}{2}\Big) = 1.$$

In other words, this normalization says, that the entropy of a fair coin is 1 bit.

One very important property of H is, that we always have $H(X) \geq 0$. This can be easily seen with help of the property $H(X) = E\big(-\log p(X)\big)$.

2.2.2. Kullback-Leibler divergence and mutual information

We will now look at the definitions of two related concepts of entropy: the *Kullback-Leibler divergence*, which measures a distance between two distributions, and *mutual information*, which measures the information a random variable contains about another random variable.

Definition 2.2.4. 1. For two discrete random variables X and Y, defined on the same measurable space $(\Omega, \mathcal{F})$ with $|\Omega| = n$, with probability functions $p(x)$ and $q(y)$ the *relative entropy* or *Kullback-Leibler divergence* between X and Y is defined as

$$D_{\mathrm{KL}}(p \parallel q) := \sum_{i=1}^{n} p(x_i) \log \frac{p(x_i)}{q(y_i)}. \tag{2.17}$$

2. Let X and Y be two discrete random variables with joint probability function $p(x, y)$ and marginal probability functions $p(x)$ and $p(y)$ defined on the measurable spaces $(\Omega_1, \mathcal{F}_1)$ with $|\Omega| = n_1$ and $(\Omega_2, \mathcal{F}_2)$ with $|\Omega| = n_2$. The *mutual information* $\mathrm{MI}(X; Y)$ is defined

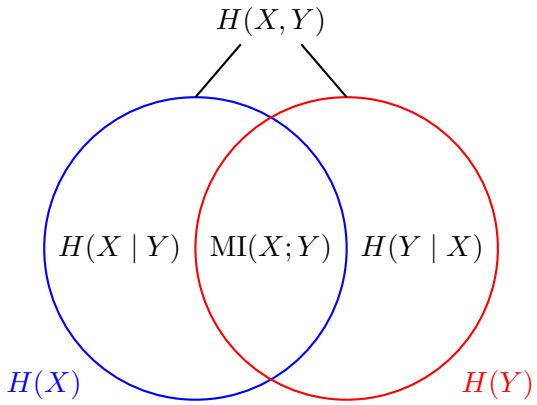


Figure 2.2.: *Relationship between entropy and mutual information*

as the Kullback-Leibler divergence between the joint probability distribution and the product distribution $p(x)p(y)$. As a formula:

$$\mathrm{MI}(X;Y) := \sum_{i=1}^{n_1}\sum_{j=1}^{n_2} p(x_i, y_j) \log \frac{p(x_i, y_j)}{p(x_i)p(y_j)}. \tag{2.18}$$

We have to note, that the Kullback-Leibler divergence D_{KL} between two probability distributions is not a true distance function, since it is not symmetric and does not satisfy the triangle inequality. But still it is useful to have it in mind as a "distance" between distributions. This is also justified with the next proposition, which states that D_{KL} is always greater than or equal to zero, and zero only if the distributions are equal.

Proposition 2.2.5. *1. Let $p(x)$ and $q(x)$ be two probability functions for discrete random variables. Then we have*

$$D_{\mathrm{KL}}(p \parallel q) \geq 0,$$

with equality if and only if $p(x) = q(x)$ for all x.

2. For any two discrete random variables X and Y we have

$$\mathrm{MI}(X;Y) \geq 0,$$

with equality if and only if X and Y are independent.

Proof. See [CT06], Section 2.6. □

In Figure 2.2 we depicted the relation between entropy and mutual information. For understanding this figure, one needs the additional definition of *conditional entropy*.

Definition 2.2.6. Let X and Y be as in Definition 2.2.4 for mutual information. Then the *conditional entropy* $H(X \mid Y)$ is defined as

$$H(X \mid Y) := -\sum_{i=1}^{n_1}\sum_{j=1}^{n_2} p(x_i, y_j) \log p(x_i \mid y_j). \tag{2.19}$$

Figure 2.2 illustrates the following properties, which hold for the general case.

Proposition 2.2.7. *1. Relations between entropy and mutual information:*

$$\begin{aligned} \mathrm{MI}(X;Y) &= H(X) - H(X \mid Y) \\ &= H(Y) - H(Y \mid X) \\ &= H(X) + H(Y) - H(X,Y) \\ \mathrm{MI}(X;X) &= H(X) \end{aligned}$$

2. (Information can't hurt)

$$H(X \mid Y) \leq H(X),$$

with equality if and only if X and Y are independent.

3. (Independence bound on entropy) Let $X_1, \ldots, X_n$ be discrete random variables. Then we have

$$H(X_1, \ldots, X_n) \leq \sum_{i=1}^{n} H(X_i),$$

with equality if and only if the X_i are independent.

Proof. See [CT06], Section 2.4 and 2.6. □

2.2.3. Differential entropy

Until now we have only defined the entropy as a measure for uncertainty for discrete random variables. Of course, it would be very useful to have such a measure also for continuous random variables. For this purpose, we first need to define the *support set* of continuous random variables.

Definition 2.2.8. Let X be a continuous random variable with density function $f(x)$. The set where $f(x) > 0$ is called the *support set* of X.

With this definition we are now able to define the measures of interest: *differential entropy* and *conditional differential entropy*. Instead of sums, as in (2.15) and (2.19), we use integrals over the support sets:

Definition 2.2.9. 1. The *differential entropy* $h(X)$, also denoted by $\mathrm{Ent}(X)$, of a continuous random variable X with density function $f(x)$ is defined as

$$h(X) := -\int_S f(x) \log f(x)\, dx, \tag{2.20}$$

where S is the support set of X.

2. If X and Y are two continuous random variables, which have a joint density function $f(x,y)$, we define the *conditional differential entropy* $h(X \mid Y)$, which is also denoted by $\mathrm{Ent}(X \mid Y)$, as

$$h(X \mid Y) := -\int_S f(x,y) \log f(x \mid y)\, d(x,y), \tag{2.21}$$

where S is the support set of (X,Y).

We are now interested, how entropy and differential entropy are related. Do they have the same properties, or are there differences? For this purpose, we give the following theorem.

Theorem 2.2.10. *If the density $f(x)$ of a continuous random variable X is Riemann integrable, then*

$$\lim_{\Delta \to 0} \big(H(X^\Delta) + \log \Delta \big) = h(X),$$

where X^Δ denotes the quantized random variable *of X, i.e., we divide the range of X into bins of length Δ and take in every bin a value x_i such that*

$$f(x_i)\Delta = \int_{(i-1)\Delta}^{i\Delta} f(x)\, dx,$$

whose existence is guaranteed with the mean value theorem and set X^Δ to x_i in the i-th bin.

Proof. See [CT06], Section 8.3. □

With Theorem 2.2.10 we now see, that the differential entropy is *not* the limit of entropy for increasing number of bins, but differs with the term $\log \Delta$, which of course is $-\infty$ for $\Delta \to 0$. One important property of entropy, namely that it is always greater zero, does not hold for differential entropy anymore. This is easily seen by considering X to be uniformly distributed on $[0, \frac{1}{2}]$. Then for the density we have $f \equiv 2$ and thus

$$h(X) = -\int_0^{\frac{1}{2}} 2 \log 2 \, dx = -\log 2 < 0.$$

In general, we can say that the differential entropy of a discrete random variable is $-\infty$.

One nice property of differential entropy is, that it is *translation invariant*.

Theorem 2.2.11. *For a continuous random variable X and $c \in \mathbb{R}$ we have*

$$h(X + c) = h(X).$$

Proof. Follows directly from Definition 2.2.9. □

The next step now is to define Kullback-Leibler divergence and mutual information for continuous random variables. Since the properties are exactly the same as for the discrete case, we will denote them with the same symbol. The formal definitions are:

Definition 2.2.12. 1. The *relative entropy* or *Kullback-Leibler divergence* $D_{\mathrm{KL}}(f \parallel g)$ between two densities $f(x)$ of a random variable X and $g(x)$ of a random variable Y is defined as

$$D_{\mathrm{KL}}(f \parallel g) := \int_S f(x) \log \frac{f(x)}{g(x)}\, dx,$$

where S denotes the support set of Y.

2. The *mutual information* $\mathrm{MI}(X;Y)$ between two continuous random variables X and Y with joint density $f(x,y)$ is defined as

$$\mathrm{MI}(X;Y) := \int_S f(x,y) \log \frac{f(x,y)}{f(x)f(y)}\, d(x,y),$$

where S denotes the intersection of the support sets of X and Y.

Remark 2.2.13. The properties of differential entropy, conditional differential entropy, Kullback-Leibler divergence and mutual information as described in Proposition 2.2.5 and 2.2.7 also hold for the continuous case. The only exception is, that $D_{\mathrm{KL}}(f \parallel g) = 0$ if and only if $f = g$ *almost everywhere*, i.e., the densities don't need to be equal for all x, but may be unequal on sets $D \subset \Omega$, where $p(D) = 0$.

2.3. Markov chain theory

At the beginning of this section, we notice that we will talk about probability distributions and distributions of random variables interchangeably. We will give some fundamental properties of Markov chains and under which conditions they converge to a unique *invariant distribution*. These special class of Markov chains are of special importance for the construction of the *sampling algorithms* in Section 3.1 to ensure that the sampling algorithms sample from the right probability distribution. This section is based on [Nea93]. We will only consider Markov chains with finite *state space* and *discrete time*. But the theory can be extended (easily) to a countably infinite state space. The theory for continuous state spaces is more difficult, but since a computer considers every state space (maximally) as a countably infinite state space, we skip this theory. The interested reader is referred to [MT09].

2.3.1. Definition of a Markov chain

A Markov process, also called *Markov chain* here, is a special *stochastic process*[2]. Stochastic processes are a collection of random variables indexed by a parameter representing the time. Thus, stochastic processes over time might show several possible dynamics according to some given probability. For Markov processes the current state X_{i+1}, $i \in \mathbb{N}$, only depends on the previous state X_i and not on all other previous states $X_0, \ldots, X_{i-1}$. More formally:

Definition 2.3.1. A *Markov process* or *Markov chain* is a series of random variables X_0, X_1, $X_2, \ldots$ satisfying the following rule:

$$p(X_{n+1}|X_n, X_{n-1}, \ldots, X_0) = p(X_{n+1}|X_n). \tag{2.22}$$

for all $n \in \mathbb{N}$. A stochastic process which satisfies (2.22) is also said to have the *Markov property*. The range of all X_n is called the *state space* of the Markov chain.

Usually, a Markov chain is given by the initial distribution of the various states X_0 and the *transition probability*, the conditional distributions for X_{n+1} given X_n. For $x \in X_0$ the initial probability is written as $p_0(x)$ and for $x \in X_n$ and $x' \in X_{n+1}$ the transition probability is written as $T_n(x, x')$ for all $n \in \mathbb{N}$. If the transition probabilities are not dependent on the time, the corresponding Markov chain is called *homogeneous* and the index n is skipped in the notation.

By applying the transition probabilities to p_0 we obtain p_1, i.e., the probabilities for all the states $x \in X_1$. The distribution p_1 may be different from p_0. However, since our goal is to sample from one specific probability distribution, there has to be at least one $m' \in \mathbb{N}$ such that for all $m > m'$ we have $p_m = p_m T_n$, i.e., the probability distribution of the state space

[2] For general theory on stochastic processes see [Doo53].

is not changed by the transition probabilities. More precise, this gives us the definition of an *invariant distribution*.

Definition 2.3.2. The distribution p is *invariant* (or *stationary*) with respect to the Markov chain with transition probabilities $T_n(x, x')$, if for all $n \in \mathbb{N}$, we have

$$p(x) = \sum_{x'} p(x')T_n(x', x).$$

Remark 2.3.3.
1. For homogeneous Markov chains with finite state space, the transition probabilities, denoted by T, are also called a *transition matrix*. The entries t_{ij} of the matrix T correspond to the probability that for a Markov chain currently being in the state x_i the next state of the Markov chain will be x_j. Therefore, the entries of T are all non-negative and the rows sum up to one. Such matrices are also called *stochastic matrices*.
2. In the case of a continuous state space T is also called a *transition kernel*. A transition kernel describes the density for a transition to state x' from a state x.

To make the testing simpler, if a distribution is invariant, we introduce the property of *detailed balance*.

Definition 2.3.4. For a homogeneous Markov chain with transition matrix T, the *detailed balance* property is fulfilled, if

$$p(x)T(x, x') = p(x')T(x', x)$$

for all x, $x' \in X$.

Theorem 2.3.5. *A homogeneous Markov chain with transition matrix T where the underlying distribution p fulfills the detailed balance property is also the invariant distribution of this Markov chain.*

Proof. We have

$$\sum_{x'} p(x')T(x', x) = \sum_{x'} p(x)T(x, x') = p(x) \sum_{x'} T(x, x') = p(x).$$

Thus, the proof is complete. □

Remark 2.3.6. The reverse of Theorem 2.3.5 is not true. To see this, let p be the uniform distribution on the state space $\{0, 1, 2\}$. It is invariant with respect to the homogeneous Markov chain with the transition probabilities $T(0, 1) = T(1, 2) = T(2, 0) = 1$ and all others zero. But the detailed balance property is not fulfilled.

2.3.2. Ergodicity

The aim now is to construct a transition matrix such that we finally sample from our desired distribution. Since we do not know at the beginning how our desired distribution looks like, we want to have a transition matrix which guarantees us, that the resulting Markov chain will sample from our desired distribution independent of which starting distribution we use, i.e.,

the Markov chain needs to have a unique invariant distribution. It obviously has to be the distribution we wish to sample from and it has to be achieved independent of which starting distribution we use. In the literature such a Markov chain is often called to be *ergodic*. We now give a theorem, which guarantees us for a large number of Markov chains their ergodicity.

Theorem 2.3.7. *Let be given a Markov chain on a finite state space X with transition probabilities $T(x, x')$ with π as invariant distribution and the following equation has to hold:*

$$\nu = \min_{x} \min_{x' | \pi(x') > 0} \frac{T(x, x')}{\pi(x')} > 0. \tag{2.23}$$

Then the Markov chain is ergodic, i.e., regardless of the initial probabilities, $p_0(x)$, we have

$$\lim_{n \to \infty} p_n(x) = \pi(x)$$

for all $x \in X$. A bound on the rate of convergence is given by

$$| \pi(x) - p_n(x) | \leq (1 - \nu)^n. \tag{2.24}$$

Furthermore, if $a(x)$ is any real-valued function of the state, then the expectation of a with respect to the distribution p_n, written $E_n[a]$, converges to its expectation with respect to π, written $\langle a \rangle$, with

$$| \langle a \rangle - E_n[a] | \leq (1 - \nu)^n \max_{x, x'} | a(x) - a(x') | . \tag{2.25}$$

Proof. See [Nea93], Chapter 3.3. □

Remark 2.3.8. The constant ν in Theorem 2.3.7 may be very small. Therefore, the convergence rate to the invariant distribution may be not very fast.

Now one can ask, which transition matrices satisfy equation (2.23)? A matrix with only non-zero entries would definitely do.

One drawback is, that Theorem 2.3.7 is only valid for homogeneous Markov chains. However, since in many applications one uses transition matrices which are cyclic in the following way: there exists a $d \in \mathbb{N}$ such that $T_{n+d} = T_n$ for all $n \in \mathbb{N}$, considering now the Markov chain only at time points that are multiples of d, we obtain a homogeneous Markov chain with transition matrix $T_0 \cdots T_{d-1}$. Sometimes we have transition matrices T such that equation (2.23) does not hold for T but for T^k for one $k \in \mathbb{N}$. Then we still get convergence to the invariant distribution, but the exponent n in (2.24) has to be replaced by $\lfloor n/k \rfloor$[3]

With this remark we can now give another characterization which transition matrices satisfy equation (2.23), namely, transition matrices T such that there exists a $k \in \mathbb{N}$ such that T^k has only non-zero entries. These Markov chains are called *regular*.

Another often mentioned type of Markov chains are *irreducible* Markov chains. They satisfy, compared to regular Markov chains, a weaker condition, namely, for all $x, x' \in X$ there exists a $k \in \mathbb{N}$ such that $T^k(x, x') > 0$. This condition is not sufficient for ergodicity of the Markov chain according to Theorem 2.3.7, because the k-th powers of the transition matrix may all contain one or more entries which are zero, because to ensure irreducibility we only have to find one special power of the transition matrix where one specific entry is non-zero. This

[3] the bracket $\lfloor x \rfloor$ denotes the *floor function* and is the largest integer not greater than x

problem will be solved by considering *aperiodicity* of a Markov chain. We will give now a precise formulation how an aperiodic Markov chain is defined to understand why we need it to ensure ergodicity. This part is geared to [Beh00].

Definition 2.3.9. 1. Let $x \in X$ be a state of a finite Markov chain with transition matrix $T = (t_{ij})_{i,j=1}^{n}$ such that $t_{ii}^{(k)} > 0$ for some $k > 0$, i.e., there is a positive probability that a walk which starts at i returns to i. Then the *period of* i is the greatest common divisor of the set $N_i := \{k \mid k \geq 0, t_{ii}^{(k)} > 0\}$.

2. If i has period 1, it will be called *aperiodic*.

3. A Markov chain where all states are aperiodic is called an *aperiodic* Markov chain.

Remark 2.3.10. It has to be noted, that for a state with period d it does not mean, that a walk starting in i will be back in i after d, $2d$, $3d, \ldots$ steps with a probability that is non-zero. It rather implies that it is for sure that the walk does not occupy position i again after k steps whenever k is not in $\{d, 2d, 3d, \ldots\}$.

With Definition 2.3.9 and Remark 2.3.10 it is now clear, why we need aperiodicity of a Markov chain to ensure that (2.23) holds when we have irreducibility of the Markov chain. We thus showed:

Theorem 2.3.11. *A Markov chain on a finite state space is ergodic, if it is irreducible and aperiodic.*

2.4. Ordinary differential equations

We will give in this section a definition of ordinary differential equations (ODEs) in general and give some properties. This is done to get a feeling concerning ODEs, since the model that is used for the modeling of gene regulatory networks in Part II and Part III is based on ODEs. In general, processes appearing in nature can often be and are extensively described with ODEs. This section is based on [Kna06] and [Heu06] where a detailed and exhaustive introduction into ODEs can be found.

2.4.1. Definition of an ODE

Definition 2.4.1. 1. A *differential equation* is defined as an equation, where the derivatives of one or more functions dependent on one or more variables are present.

2. If the functions are dependent on only one variable, the corresponding differential equation is called an *ordinary differential equation*, otherwise it is called a *partial differential equation*.

3. If only one function is used, we call the differential equation a *single differential equation*, otherwise it is called a *system of differential equations*.

4. The highest order of the derivatives that appears in the differential equation is called the *order of the differential equation*.

The definition above uses only words to explain intuitively, what is meant with a differential equation. We will now give a more formal view of it, to be familiar with it, since it will be used in the following parts of this thesis. We will give the notation for systems of ordinary differential equations of order 1, since these are the systems we need in this thesis[4]. Let x_i, $t \mapsto x(t)$, $i \in \{1, \dots, n\}$, be real-valued functions. Then we will write a system of ordinary differential equations as an equation of the form

$$\frac{dx}{dt} = f(t, x), \tag{2.26}$$

where $x = (x_1, \dots, x_n)$ and $f : \mathbb{R}^{n+1} \to \mathbb{R}^n$, $(t, x_1, \dots, x_n) \mapsto f(t, x_1, \dots, x_n)$. Of course, we can write for every function x_i in the system a single ODE of the form

$$\frac{dx_i}{dt} = f(t, x_1, \dots, x_n).$$

2.4.2. Existence and uniqueness of solutions of ODEs

The next interesting question that arises, is if and under which conditions ODEs have a solution. Firstly, we note that there exist ODEs with no solution at all, e.g.,

$$\left(\frac{dx}{dt}\right)^2 = -1.$$

Also there exist ODEs with exactly one solution, e.g.,

$$\left(\frac{dx}{dt}\right)^2 = -x^2.$$

However, the most common case is the one where there exist infinitely many solutions. As an example we will use the process of *exponential growth* for a population. The corresponding ODE is of the form

$$\frac{dx}{dt} = \alpha x, \qquad \text{with} \qquad \alpha > 0. \tag{2.27}$$

The ODE (2.27) is also an example for a *linear differential equation*, which is in general defined as an ODE where the function f in (2.26) is linear in x.

One sees immediately, that all functions of the form $x(t) = c\,\mathrm{e}^{\alpha t}$ with $c \in \mathbb{R}$ are solutions of the ODE for exponential growth, since we have

$$\frac{dx}{dt} = \alpha c\,\mathrm{e}^{\alpha t} = \alpha x(t). \tag{2.28}$$

By fixation of a value for the population at the beginning of the exponential growth process $x_0 := x(0)$ we obtain $x(0) = c\,\mathrm{e}^{\alpha 0} = c$ and thus the single function from the set of solutions (2.28) that satisfies the value x_0 at the beginning of the process is $x(t) = x_0\,\mathrm{e}^{\alpha t}$. In general, a problem of the form

$$\frac{dx}{dt} = f(t, x), \quad x(t_0) = x_0, \tag{2.29}$$

[4]The interested reader in partial differential equations is referred to, e.g., [Jos07].

where t_0 denotes the starting point of the process under consideration, is called an *initial value problem*. In the example above for a linear differential equation we saw that from the set of solutions (2.28) we obtained a unique solution after specification of the initial value.

However, can we be sure that the specified set of solutions is the only possibility for solutions for the exponential growth process? And furthermore, in case of general initial value problems is there always a unique solution? In the following we give two theorems which guarantee the existence of solutions of initial value problems. We will see that it depends on the properties of the function $f(t, x)$. Let us start with a theorem that guarantees existence.

Theorem 2.4.2 (Peano existence theorem)**.**

1. *(Local version) Let the function $f(t, x)$ be continuous on the compact cuboid*

$$R := \{(t, x) \mid |t - t_0| \leq a, \|x - x_0\| \leq b\} \quad (a, b > 0),$$

and let

$$M := \max_{(t,x)\in R} \|f(t, x)\|, \quad \alpha := \min\Big(a, \frac{b}{M}\Big).$$

Then there exists at least one solution of the initial value problem (2.29) *on the cuboid* $[x_0 - \alpha, x_0 + \alpha]$.

2. *(Global version) Let the function $f(t, x)$ be continuous and bounded on the cuboid $[c, d] \times \mathbb{R}^n$. Then there exists at least on solution of the initial value problem* (2.29) *on the cuboid $[c, d]$ with $t_0 \in [c, d]$ and $x_0 \in \mathbb{R}^n$.*

Proof. See [Heu06] Chapter 60. □

With the Peano existence theorem we see that the continuity of the function $f(t, x)$ guarantees that there exists a solution for the initial value problem (2.29). Existence is nice, but what about uniqueness of the solution? As one would expect, the requirements on the function $f(t, x)$ will be stricter to guarantee uniqueness. This is in fact the case. The stricter requirement is that not only continuity but also *Lipschitz continuity* has to hold for $f(t, x)$.

Definition 2.4.3. Let $U \subset \mathbb{R} \times \mathbb{R}^n$ and $f : U \to \mathbb{R}^n$, $(t, x) \mapsto x$. Then the function f fulfills a (global) *Lipschitz condition* in the second variable on U, if and only if there exists a constant $L_C \geq 0$, such that for all $t \in \mathbb{R}$ and for all $x_1, x_2 \in \mathbb{R}^n$ with $(t, x_1), (t, x_2) \in U$ the equation

$$\|f(t, x_1) - f(t, x_2)\| \leq L_C \|x_1 - x_2\| \tag{2.30}$$

is valid.

With this condition for the function $f(t, x)$ we are able to give a theorem which also guarantees the uniqueness of the solution for the initial value problem (2.29).

Theorem 2.4.4 (Picard-Lindelöf theorem)**.**

1. *(Local version) Let the function $f(t, x)$ be continuous and let it fulfill a Lipschitz condition in the second variable for all (t, x_1), $(t, x_2) \in R$, where R is defined as in Theorem 2.4.2. Then there exists a unique solution for the initial value problem* (2.29) *on the cuboid $[x_0 - \alpha, x_0 + \alpha]$, where α is defined as in Theorem 2.4.2.*

2. *(Global version) Let the function $f(t,x)$ be continuous on the cuboid $[c,d]\times\mathbb{R}^n$ and let it fulfill there a Lipschitz condition. Then there exists a unique solution of the initial value problem (2.29) on the cuboid $[c,d]$ with $t_0\in[c,d]$ and $x_0\in\mathbb{R}^n$.*

Proof. See [Heu06] Chapter 60. □

As a final theorem for the solution of initial value problems we show that the solutions depend continuously on the initial values and on the function $f(t,x)$. This is, of course, a desirable result and guarantees that the solution does not change much in cases where the initial values and the function $f(t,x)$ are only slightly varied.

Theorem 2.4.5. *Let the function $f(t,x)$ fulfill the requirements of the local version of the Picard-Lindelöf theorem 2.4.4. Let $\tilde{f}(t,x)$ be continuous on the cuboid R as defined in Theorem 2.4.4. Furthermore, let $x(t)$ be the unique solution of the initial value problem (2.29) on the interval $J:=[t_0-\alpha,t_0+\alpha]$ and let $\tilde{x}(t)$ be any solution on the interval $\tilde{J}\subseteq J$, which lies completely in R, of the initial value problem*

$$\frac{dx}{dt}=\tilde{f}(t,x),\qquad x(t_0)=\tilde{x}_0.$$

Finally, the following estimates with the constants σ and ω should hold on R

$$\begin{aligned}\|x_0-\tilde{x}_0\|&\leq\sigma\\ \|f(t,x)-\tilde{f}(t,x)\|&\leq\omega.\end{aligned}$$

Then we have with the Lipschitz constant L_C of $f(t,x)$ the equation

$$\|x(t)-\tilde{x}(t)\|\leq\sigma\,\mathrm{e}^{L_C|t-t_0|}+\frac{\omega}{L_C}(\mathrm{e}^{L_C|t-t_0|}-1) \tag{2.31}$$

for all $t\in\tilde{J}$. Thus, the difference between the solutions $x(t)$ and $\tilde{x}(t)$ is arbitrarily small, if σ and ω are small.

Proof. See [Heu06] Chapter 13. □

2.4.3. Solving initial value problems in practice

Having now the theoretical foundation, that for many systems of differential equations there exist a unique solution, we have to say, that analytical solutions are only available for few ordinary differential equations, e.g., for linear ODE systems[5]. However, for the most cases methods for the *numerical integration* of ODE systems have to used. This is a wide topic and we will not give details but rather some general ideas and which difficulties may arise while applying methods for the numerical integration of ordinary differential equations. The interested reader in the details is referred to [SB05].

Looking at the initial value problem (2.29), we see that the function $f(t,x)$ just describes the slope of the solution x. Thus, we can write (we consider the case where $n=1$)

$$\frac{x(t+h)-x(t)}{h}\approx f\big(t,x(t)\big)$$

[5] see [Heu06] Parts II, III, VII and VIII for the derivation of solutions for linear ODE systems

with $h > 0$. Reformulating this, we obtain the approximating value for the solution of x at the point $t + h$

$$x(t + h) \approx x(t) + hf\big(t, x(t)\big).$$

Having now introduced the general idea of methods for numerical integration we see that we introduce in every step an error with the usage of $h > 0$. The goal of efficient methods for numerical integration is now to use as step size h values that are large, such that not many calculations have to be performed, but at the same time the step size has to be small, such that the error introduced is small. Looking at Theorem 2.4.5, the problem of introducing error (because of discretization errors according to the step size and because of rounding errors arising with the calculation with *floating points*) in the numerical integration approach may get really bad for some problems. Because of the term $\mathrm{e}^{L_C|t-t_0|}$ in equation (2.31), even small differences in the initial values and the function $f(t, x)$ may lead to substantial differences in the solutions, in case one considers huge time intervals $t - t_0$.

Special difficulties arise for the case of *stiff differential equations*. Stiff differential equations are characterized as differential equations, where the *Jacobian* of the function $f(t, x)$ has at least one *eigenvalue* $m \in \mathbb{C}$ with $|m| \gg 0$. This will then lead to numerical instabilities, since these eigenvalues determine the step size h of the approach for the numerical integration to be very small. Thus, a lot of calculations have to be performed. For a more detailed overview see [SB05] Section 7.2.16.

2.5. Splines

This section deals with the introduction and definition of *splines*. They will be used in Part II. The scope of this section is the definition and illustration of splines to get a feeling for the usage of them.

2.5.1. Definition of splines

Splines are often used to provide an easy to handle function which connects given data points and thus gives a functional relationship between these points. We will focus here only on *cubic splines*, since these are the most widely used ones.

Definition 2.5.1.

1. Let $\Delta := \{a = x_0 < x_1 < \ldots < x_n = b\}$ be a finite sequence of points in the interval $[a, b]$. A *cubic spline* S_Δ corresponding to the sequence Δ is defined as a function $S_\Delta : [a, b] \to \mathbb{R}$ with the properties
   a) S_Δ is continuously differentiable twice on the interval $[a, b]$
   b) on every subinterval $[x_i, x_{i+1}]$, $i \in \{0, \ldots, n-1\}$, the function S_Δ is a cubic polynomial
2. Let be given data $Y := \{y_0, y_1, \ldots, y_n\}$. Then we define $S_\Delta(Y; \cdot)$ to be an *interpolating cubic spline* with $S_\Delta(Y; x_i) = y_i$ for all $i \in \{0, \ldots, n\}$.

In Figure 2.3 an interpolating spline is shown which connects 11 data points. We see there that the interpolating spline perfectly fits the data points as desired. However, in case of noisy data one does not always want to fit the data perfectly, but rather smooth out the noise and

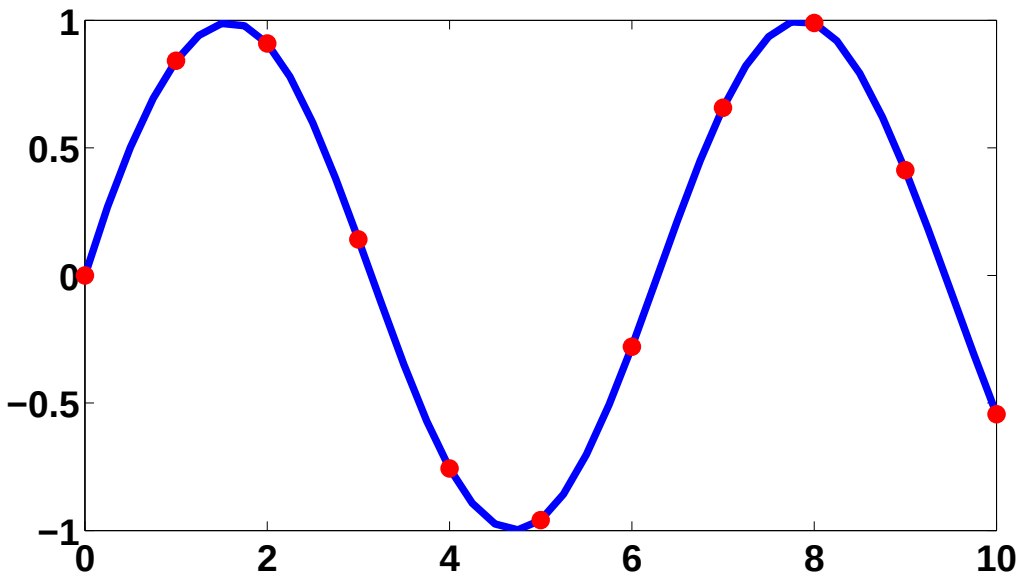


Figure 2.3.: *An example for an interpolating spline fitting* 11 *data points generated with the function* `spline` *in Matlab.*

fit the underlying dynamic behavior of the process the data is describing. Thus, we introduce here also *smoothing splines* which provide exactly the desired results.

Definition 2.5.2. Let Δ and Y be given as in Definition 2.5.1. The function $g : \mathbb{R} \to \mathbb{R}$ which minimizes

$$\sum_{i=0}^{n} \big(y_i - g(x_i)\big)^2 + \lambda \int \big(g''(z)\big)^2 \, dz, \tag{2.32}$$

where λ is called the *smoothing factor*, is called a *cubic smoothing spline*.

Well, one question arises immediately after reading Definition 2.5.2. The question is: Is a cubic smoothing spline a cubic interpolating spline? The answer to this question is: yes. A proof that the function g that minimizes (2.32) is a cubic interpolating spline can be found in [Rei67].

The smoothing factor λ plays a role of how much we want to smooth the data. To be more precise, for $\lambda = 0$ the solution will be an interpolating function and no noise will be smoothed out. For $\lambda \to \infty$ the solution will be a straight line and no dynamics of the system under consideration will be captured.

In Figure 2.4 a smoothing spline is depicted smoothing 11 data points. We see there that the general dynamics are covered and single noisy peaks are smoothed out.

2.5.2. Calculating splines in practice

The points that remain to be discussed are, if the cubic interpolating spline $S_\Delta(Y;\cdot)$ exists and is unique. To discuss these points in a bit more detail, we will point out here that the

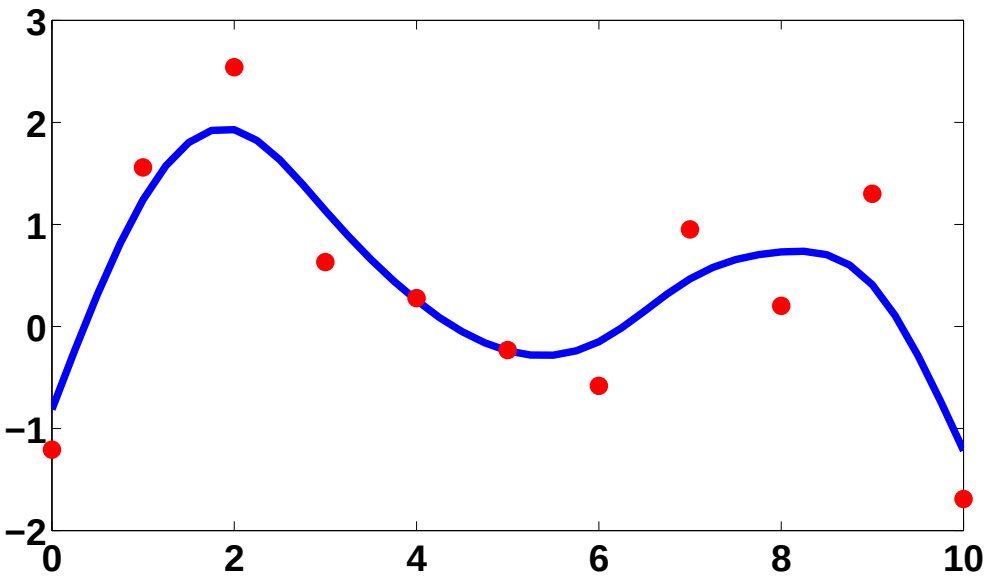


Figure 2.4.: *An example for a smoothing spline fitting* 11 *data points generated with the function* `csaps` *in Matlab.*

generation of cubic interpolating splines is done simply by solving *systems of linear equations*[6]. To guarantee the unique existing solution of a system of linear equations, the corresponding *matrix* needs to have *full rank*. The data points Y alone does not give a matrix with full rank. However, two additional equations are needed to guarantee a unique solution. There are three typical constraints that are used for this purpose, where as a example one can demand the second derivatives of the interpolating spline at the border points a and b to be zero[7]. The proof that by doing so one obtains a unique cubic interpolating spline is found in [SB05] Chapter 2.5.

What about smoothing splines? Since a smoothing spline is the solution of an optimization problem, we may think, that the computational effort is high. Fortunately, Buja *et al.* [BHT89] gave a closed formula for the unique solution of the optimization problem, which basically results in matrix calculation with a matrix where the structure can be exploited and thus the computational effort is low.

[6]The interested reader in numerical issues concerning the solution of systems of linear equations is referred to [SB05] Chapter 4.

[7]for the other two possible constraints see [SB05] Section 2.5.1

CHAPTER 3

Algorithmical background

> Anyone who considers arithmetic methods of producing random digits is, of course, in a state of sin.
>
> *(John von Neumann)*

In this chapter we give the background for the algorithms that are used in Part II and Part III. Firstly, an overview of *sampling algorithms* is given, which are used to obtain samples from a distribution of interest. Secondly, some basics about *parallel computing* are described, that are used in Part III. Furthermore, *receiver operating characteristics* are described, which are used in Part II as evaluation method for the performance of the proposed method.

3.1. Sampling algorithms

In many (statistical) applications one is interested how special distributions look like and one wants to generate samples from them. The ideal case would be, to have independent samples from the desired distribution. Since most distributions of applications are high-dimensional and multi-modal, this is not an easy task. With the theory given in Section 2.3 we will now give sampling algorithms which are based on Markov chains. The samples produced are not independent, since the current sample depends on the previous sample (see Definition 2.3.1), but we can say that samples far away form each other can be assumed to be independent. In more mathematical terms: let x_n and x_m be members of a sampled Markov chain. We can say that if $m \gg n$, then x_n and x_m are independent samples of the desired distribution.

But first, we want to give some general properties how one can construct ergodic Markov chains by giving some ideas how to use/find suitable transition matrices such that they converge to their invariant distribution at as fast a rate as possible. Often one uses a set of base transition matrices $B_1, \ldots, B_k$, where each holds the desired distribution invariant but may not be ergodic. For example, each B_j may only change a subset of the state space. There are two main reasons to combine base transition matrices, first, to obtain an ergodic

Algorithm 3.1 Gibbs sampling

Require: desired distribution $p(\cdot)$, starting value $x^{(0)} = (x_1^{(0)}, \ldots, x_n^{(0)})$, number of Markov chain samples T

1: $t \leftarrow 0$
2: **while** $t < T$ **do**
3: **for** $k = 1$ **to** n **do**
4: Sample $x_k^{(t+1)}$ from $p\big(\cdot \mid x_1^{(t+1)}, \ldots, x_{k-1}^{(t+1)}, x_{k+1}^{(t)}, \ldots, x_n^{(t)}\big)$
5: **end for**
6: Append $x^{(t+1)}$ to Markov chain $\big(x^{(i)}\big)_{i=0}^{t}$
7: $t \leftarrow t + 1$
8: **end while**
9: **return** Markov chain $\big(x^{(i)}\big)_{i=0}^{T}$

Markov chain and second, to speed up the convergence rate. We saw in Theorem 2.3.11, that ergodicity of a Markov chain follows from irreducibility and aperiodicity. We can guarantee aperiodicity easily by combining base transition matrices with the identity to ensure that in each step there is a non-zero probability that a state will remain in its current state.

A general introduction to Markov chain Monte Carlo (MCMC) algorithms for *machine learning* approaches is given in [AFDJ03].

For the sampling algorithms presented in the next sections, we will show that they produce ergodic Markov chains only for the case of a finite state space. This is done for simplicity reasons, since the case where one deals with transition kernels instead of transition matrices is technically more demanding. However, the results also hold for the case of an infinite state space. This section is based on [Nea93], except for the subsection on population-based MCMC algorithms which uses [JSH07] as a reference.

3.1.1. Gibbs sampling

The Gibbs sampler is a simple sampler suitable for high-dimensional distributions, where the full conditionals $p(x_j \mid x_1, \ldots, x_{j-1}, x_{j+1}, \ldots, x_n)$ for all $j \in \{1, \ldots, n\}$ are available and it is easy to get independent samples from them.[1] Later we will see that the Gibbs sampler is a special case of the Metropolis-Hastings algorithms, but because of its simplicity, we will explain it first to give an idea how sampling algorithms work.

One generates a homogeneous Markov chain $X^{(0)}$, $X^{(1)}$, $X^{(2)}, \ldots$ by sampling from the conditional distributions for each variable $x_j^{(t)}$ as mentioned above where the recent sampled value is directly used to sample the next values $x_k^{(t)}$, $k > j$. A pseudocode version is given in Algorithm 3.1.

One may now ask why this procedure leaves the desired distribution invariant. For the answer of this question consider base transition probabilities B_k for $k \in \{1, \ldots, n\}$ given by

$$B_k(x, x') = p(x'_k \mid x_1, \ldots, x_{k-1}, x_{k+1}, \ldots x_n) \cdot \prod_{i \neq k} \delta(x_i, x'_i)$$

[1] As examples for distributions where independent samples can be got easily are the Gaussian and the gamma distribution.

where δ denotes *Euler's delta function* such that $\delta(x, y) = 1$ if $x = y$ and otherwise it is zero. By sampling with the above procedure one gets a Markov chain with the transition matrix $T = B_1 B_2 \cdots B_n$. It is enough to show that each B_k holds the desired distribution invariant to show that this procedure holds the desired distribution invariant. Since B_k leaves all components except for x_k invariant, it leaves the marginal distribution for this components invariant. For the remaining component x_k it samples from the marginal distribution of x_k, which also leaves the desired distribution invariant. Thus, we are done.

The next question to answer is to show that such a constructed Markov chain is ergodic. This will be the case, if all B_k's are non-zero. To be more precise, the conditional probabilities $p(\cdot \mid x_1, \ldots, x_{k-1}, x_{k+1}, \ldots x_n)$ have to be non-zero everywhere. Because in that case it is possible to reach each state of the state space after the usage of each B_k once. And furthermore, there will be a non-zero probability for each variable to stay in the current state, which guarantees aperiodicity. Now the ergodicity is proven.

A typical application of the Gibbs sampler is the inference of Bayesian networks[2], since the variables have usually a simple distribution and it is easy to sample from the full conditionals.

3.1.2. Metropolis-Hastings algorithm

We have seen in the previous section that the Gibbs sampler is only applicable when it is easy to sample from the full conditionals. However, in many applications one has distributions which are multi-modal and not of a standard form. In such cases one uses sampling algorithms for which one only has to be able to calculate the desired distribution at each point up to the normalization constant. Since the calculation of the normalization constant is computational expensive, this is really an improvement for sampling.

You may now think: Nice, but how is this done? One way to do this was proposed by Metropolis in 1953 [MRRT53] and refined by Hastings in 1970 [Has70], also called the *Metropolis-Hastings algorithm*. For this, one needs first a proposal distribution $q(\cdot \mid \cdot)$. The second step is to sample from this distribution to get a new proposed point x' from the desired distribution. This point is now accepted with the probability

$$A(x, x') = \min\left\{1, \frac{p(x')q(x \mid x')}{p(x)q(x' \mid x)}\right\} \tag{3.1}$$

otherwise the Markov chain stays in the old point x. One often uses a symmetric proposal distribution, i.e., $q(x \mid x') = q(x' \mid x)$ for all $x, x' \in X$ such that equation (3.1) is reduced to

$$A(x, x') = \min\left\{1, \frac{p(x')}{p(x)}\right\}. \tag{3.2}$$

To prove that we really sample from the desired distribution, we look at the underlying base transition probabilities

$$B_k(x, x') = q(x'_k \mid x)A(x, x') \prod_{i \neq k} \delta(x_i, x'_i) + \delta(x, x')\Big(1 - \sum_{\tilde{x}} q(\tilde{x}_k \mid x)A(x, \tilde{x}) \prod_{i \neq k} \delta(x_i, \tilde{x}_i)\Big),$$

where x'_k is different from x only in the k-th component. The first term is the probability to change component k from x_k to x'_k. The second term gives the probability to reject x'_k

[2] See Section 4.1.2 for more details on Bayesian networks.

and stay in the current state x_k. As for the Gibbs sampling we will obtain with these base transition matrices a homogeneous Markov chain with transition matrix $T = B_1 \cdots B_n$. For a symmetric proposal distribution we can show that for every B_k, $k \in \{1, \ldots, n\}$, the detailed balance property holds. Thus with Theorem 2.3.5 the algorithm has the desired distribution as invariant distribution. To show the ergodicity of the obtained Markov chain we note that the probability of staying in the same state is non-zero, which gives as aperiodicity. And irreducibility is guaranteed, if the proposal distribution and p are non-zero everywhere. With Theorem 2.3.11 we now obtain ergodicity.

Have proven the ergodicity for the case of a finite state space, we now say something about some practical issues of the usage of the Metropolis-Hastings algorithm. In practice, most often a normal distribution $\mathcal{N}(x, \sigma^2)$, where the mean is the current state x and σ^2 gives a user-specified variance for the proposal distribution, is used as proposal distribution. Using a large value for σ^2 may lead to a high rejection rate of the algorithm. Using, on the other hand, small values for σ^2 will decrease the rejection rate, but one needs to produce a big number of samples to explore the whole distribution. Another issue to mention for this algorithm is its random walk behavior, i.e., after performing N steps, the mean distance from the starting point is only $\sqrt{N}$. This can be interpreted in the way, that because of choosing the direction for every proposed step independently from each other, some steps may cancel out each other.

To avoid the random walk behavior, several possibilities exist for proposing better states in MCMC algorithms. In the next two sections, we will describe the *hybrid Monte Carlo algorithm* which was designed to simulate physical systems. It proposes states in a more global way. Another possibility is to use *population-based MCMC algorithms* which run several Markov chains in parallel and exchanges the information between them to provide more global steps.

3.1.3. Hybrid Monte Carlo algorithm

The hybrid Monte Carlo algorithm (HMC) was introduced by Duane in 1987 [DKPR87] and was motivated with *molecular dynamics*. It is based on the physical property that for a *closed system* the *total energy* of it is the sum of its *kinetic energy* and its *potential energy*. And to change the total energy of the system, it interacts with its surroundings in a stochastic manner. For the physical details and background concerning this property we refer to the literature dealing with *classical* and *statistical mechanics*.

Let's come back to the case, that we want to generate samples of a distribution $p(x)$, where $x = (x_1, \ldots, x_n)$. How can we now use these physical properties for generating the samples of interest? First, we consider the vector x to describe the "position coordinates" of the molecules to be modeled and say that their potential energy $E(x)$ is $E(x) = -\ln p(x)$. The *canonical distribution* of x is then[3]

$$p(x) \sim \exp\big(-E(x)\big) = p(x).$$

Second, new variables $\rho = (\rho_1, \ldots, \rho_n)$ are introduced describing the *momentum* of the system which possesses the kinetic energy

$$K(\rho) = \frac{1}{2} \sum_i \rho_i^2. \tag{3.3}$$

[3] we take here the temperature to be 1 for convenience and omit the normalization constant

The canonical distribution of the momentum variables ρ is then

$$p(\rho) = \frac{1}{\sqrt{(2\pi)^n}} \exp\Big(-\frac{1}{2}\sum_i \rho_i^2\Big),$$

i.e., the normal distribution. The combination of the position variables x and the momentum variables ρ is called the *phase space*. The total energy of the phase space, also known as the *Hamiltonian*, is

$$H(x, \rho) = E(x) + K(\rho) \tag{3.4}$$

and the canonical distribution according to this energy function is given by

$$p(x, \rho) \sim \exp\big(H(x, \rho)\big) \sim \exp\big(-E(x)\big) \cdot \exp\big(-K(\rho)\big) \sim p(x) \cdot p(\rho). \tag{3.5}$$

Thus, the desired distribution is just the marginal distribution for x of the distribution $p(x, \rho)$. This means, that getting samples from the phase space, we obtain the desired values for x by just ignoring the obtained values for ρ.

To generate the samples for the phase space the HMC algorithms proceeds in two steps

1. following the *Hamiltonian dynamics* to generate samples with the same total energy
2. perform *stochastic transitions* to get samples with changed total energy

The next obvious point is to say more about Hamiltonian dynamics. They are characterized by the following two equations

$$\begin{aligned} \frac{dx_i}{d\tau} &= \frac{\partial H}{\partial \rho_i} = \rho_i \\ \frac{d\rho_i}{d\tau} &= -\frac{\partial H}{\partial x_i} = \frac{\partial E}{\partial x_i}, \end{aligned} \tag{3.6}$$

where τ is a time parameter. In physical problems, it describes the real physical time. For statistical problems it describes just an artificial parameter. To see that these dynamics really conserve the value of the Hamiltonian, we just look at

$$\frac{dH}{d\tau} = \sum_i \left(\frac{\partial H}{\partial x_i}\frac{dx_i}{d\tau} + \frac{\partial H}{\partial \rho_i}\frac{d\rho_i}{d\tau} \right) = \sum_i \left(\frac{\partial H}{\partial x_i}\frac{\partial H}{\partial \rho_i} - \frac{\partial H}{\partial \rho_i}\frac{\partial H}{\partial x_i} \right) = 0.$$

In practice, to follow the Hamiltonian dynamics (3.6) exactly is not possible. Instead, the equations have to be discretized with a non-zero time step ε, which introduces of course some error into the dynamics. This discretization is usually done with the *leapfrog discretization* which is described in Algorithm 3.2.

We will now come to the second step in the HMC algorithm: the stochastic transitions. The purpose of this is to get samples with different total energy. It is convenient just to change the values for the momentum variables ρ, because of (3.5) and the property that ρ follows a normal distribution, which is easy to sample from. This can also be seen as a Gibbs sampling step, where we sample from ρ given the current values of x.

A last point we have to manage is how we want to deal with the error that is introduced in the leapfrog discretization. This is easily solved by considering the point obtained after the

Algorithm 3.2 Leapfrog discretization

Require: desired distribution $p(\cdot)$, starting values (x, ρ), step size ε, number of leapfrog steps L

$\ell \leftarrow 1$
$\rho_i \leftarrow \rho_i - \frac{\varepsilon}{2}\frac{\partial p(x)}{\partial x_i}$ for all $i \in \{1, \dots, n\}$
while $\ell < L$ **do**
  $x_i \leftarrow x_i + \varepsilon\rho_i$ for all $i \in \{1, \dots, n\}$
  $\rho_i \leftarrow \rho_i - \varepsilon\frac{\partial p(x)}{\partial x_i}$ for all $i \in \{1, \dots, n\}$
  $\ell \leftarrow \ell + 1$
end while
$\rho_i \leftarrow \rho_i + \frac{\varepsilon}{2}\frac{\partial p(x)}{\partial x_i}$ for all $i \in \{1, \dots, n\}$
return (x, ρ)

discretization as a candidate step $(\hat{x}, \hat{\rho})$, which is then accepted or rejected according to the acceptance function

$$A\big((x,\rho),(\hat{x},\hat{\rho})\big) = \min\Big(1, \exp\big(H(x,\rho) - H(\hat{x},\hat{\rho})\big)\Big),$$

which is the same as (3.2) as can be seen with the help of (3.5). Pseudocode for the HMC algorithm is given in Algorithm 3.3.

The HMC algorithm can also be seen as a Metropolis-Hastings algorithm where the proposal distribution is defined by the leapfrog steps. To show now that we really generate samples from the desired distribution, Neal [Nea93] showed that the detailed balance property is fulfilled. Thus, our desired distribution is the invariant distribution of the produced Markov chain. For the ergodicity of the Markov chain the same conditions have to be fulfilled as for the Metropolis-Hastings algorithm.

The efficiency of this algorithm depends crucially on the number of leapfrog steps L performed and on the step size ε. Generally, a large number of leapfrog steps should be performed to explore the state space in a global way and thus to avoid the random walk behavior. Furthermore, the step size should be small such that the introduced error is not too large. However, in every single leapfrog step we need to calculate the derivative of $p(x)$ with respect to all of its components $x_1, \dots, x_n$. Thus, we need to find a tradeoff between a step size not being too large, which enables the exploration of the state space globally and still does not need many evaluations of the derivative. And this is not a trivial task.

3.1.4. Population-based Markov chain Monte Carlo algorithms

Instead of running one Markov chain, in population-based Markov chain Monte Carlo algorithms one runs several chains in parallel and, thus, in every step one generates a population of samples. These samples are generated using so-called *genetic operators*. The main idea of this approach is to explore the state space in a more efficient way by exchanging information between chains. A nice review is given in [JSH07].

We will put this in a more formal context: Instead of considering only the desired target density $p(\cdot)$, we consider a new target measure

$$p^*(\cdot) := \prod_{n=1}^{N} p_n(\cdot) \tag{3.7}$$

Algorithm 3.3 Hybrid Monte Carlo algorithm

Require: desired distribution $p(\cdot)$, starting value x^0, number of leapfrog steps L, step size ε for leapfrog steps, standard deviation σ for the sampling of the momentum variables ρ, number of Markov chain samples T

$t \leftarrow 0$
while $t < T$ **do**
  Sample $\rho_i^{(t)}$ from $\mathcal{N}(0, \sigma^2)$ for all $i \in \{1, \dots, n\}$
  Perform L leapfrog steps with step size ε at state $(x^{(t)}, \rho^{(t)})$ (see Algorithm 3.2)
  Store resulting candidate state in $(\hat{x}, \hat{\rho})$
  Sample u from $\mathcal{U}(0, 1)$
  $\alpha \leftarrow \min\{1, \exp(H(x^{(t)}, \rho^{(t)}) - H(\hat{x}, \hat{\rho}))\}$
  if $u < \alpha$ **then**
    $x^{(t+1)} \leftarrow \hat{x}$
  else
    $x^{(t+1)} \leftarrow x^{(t)}$
  end if
  Append $x^{(t+1)}$ to Markov chain $\left(x^{(i)}\right)_{i=0}^{t}$
  $t \leftarrow t + 1$
end while
return Markov chain $\left(x^{(i)}\right)_{i=0}^{T}$

where $N \in \mathbb{N}$ denotes the number of chains that are run in parallel. We will assume that at least for one $n \in \{1, \dots, N\}$ we have $p(\cdot) \equiv p_n(\cdot)$. At the end, only the samples of the desired distribution $p(\cdot)$ are taken for further investigation. Of course, if all distributions $p_n(\cdot)$ are identical, all samples should be used.

The genetic operators or *population moves* used to generate the samples in the next step are *mutation*, which explores the state space in a local way. And furthermore, *exchange* and *crossover* explore the state space in a more global way.

Mutation

This step explores the state space in a local way. It is applied to a single member of the population and is essentially the same as a step of the Metropolis-Hastings algorithm as described in Section 3.1.2.

Exchange

This step is used to exchange information between chains. It is also a Metropolis-Hastings step, where for two chains n, $m \in \{1, \dots, N\}$ the current values x_n and x_m are exchanged with the probability $\min\{1, A\}$, where

$$A = \frac{p_n(x_m)p_m(x_n)}{p_n(x_n)p_m(x_m)}.$$

Especially in the case where the sequence of distributions $\left(p_n(\cdot)\right)_{n=1}^{N}$ is selected in such a way that the distributions are related but are easier to sample from than $p(\cdot)$, this step is very useful.

Crossover

There are two types of crossover operators, *real crossover* and *snooker crossover*, where we use the terminology of Goswami and Liu [GL07].

The real crossover is based on the crossover operator used in genetic algorithms[4]. Assume we have $x_n = (x_{n1}, x_{n2}, \ldots, x_{nd})$ and $x_m = (x_{m1}, x_{m2}, \ldots, x_{md})$ and we want to use for crossover the l-th position. The proposed new values are then

$$x'_n = (x_{n1}, \ldots, x_{n(l-1)}, x_{ml}, \ldots, x_{md}),$$
$$x'_m = (x_{m1}, \ldots, x_{m(l-1)}, x_{nl}, \ldots, x_{nd}).$$

These proposed values are accepted via a Metropolis-Hastings step with the probability $\min\{1, A\}$ where

$$A = \frac{p_n(x'_n)p_m(x'_m)}{p_n(x_n)p_m(x_m)}$$

under the assumption that the choices for the chains and the crossover position are made with uniform probability. Of course the crossover step is not limited to do a crossover at one point, one can also use several break points to perform *k-point crossover*.

Since for real-valued problems the performance of the real crossover operator is poor [LW01], so-called snooker crossover operators were suggested. The idea of this steps is to move population members towards each other. From a computational perspective the use of the crossover operator is that we expect that some individuals should have high target density, and thus, it is reasonable to propose samples near other individuals of the population.

The snooker crossover proposed by Liang and Wong [LW01] chooses two "parental" individuals x_n and x_m randomly and proposes as new sample x'_n the point

$$x'_n = x_n + r \cdot \frac{x_m - x_n}{||x_m - x_n||} \tag{3.8}$$

where r follows the density

$$f(r) \propto |1 - r|^{d-1} \, p_n\left(x_n + r \cdot \frac{x_m - x_n}{||x_m - x_n||}\right). \tag{3.9}$$

This step is illustrated in Figure 3.1 on the left.

Another direction for the snooker crossover proposed by ter Braak [tB06] chooses instead of the parent x_n used in (3.8) another randomly chosen individual x_l different from x_n and x_m. The proposed sample x'_n is then

$$x'_n = x_n + r \cdot \frac{x_m - x_l}{||x_m - x_l||} = x_n + r \cdot \mathbf{e}.$$

This step is illustrated in Figure 3.1 on the right. One has to note, that the red arrow is parallel to the dashed line between x_l and x_m. And furthermore, we have to consider that if x_l and x_m had been chosen the other way around, then the red arrow would go into the opposite direction. Hu and Tsui [HT10] proved that, if you draw r from the distribution

$$g(r) = \frac{p_n(x_n + r \cdot \mathbf{e})}{\int_{-\infty}^{\infty} p_n(x_n + r' \cdot \mathbf{e}) \, dr'}, \tag{3.10}$$

[4]For more information on genetic algorithms see [Gol89].

Figure 3.1.: *Geometrical illustration for the snooker crossover operator proposed by [LW01] (left) and [tB06] (right)*

Algorithm 3.4 A basic population-based MCMC algorithm

Require: desired distribution $p(\cdot)$, size of population N, starting values $(x_1, \ldots, x_N)$ for the N chains, new target measure $p^* = \prod_{i=1}^{N} p_i$ (as in (3.7)), number of Markov chain samples T, rules R how to use the mutation, exchange and crossover operators

1: $t \leftarrow 0$
2: **while** $t < T$ **do**
3: According to rules R do:
4: Perform mutation operator
5: Perform exchange operator
6: Perform crossover operator
7: $t \leftarrow t + 1$
8: **end while**
9: **return** Markov chain $(x_i^{(t)})_{t=0}^{T}$, where i corresponds to the index of the chain that sampled from the desired distribution $p(\cdot)$

then x'_n follows the desired distribution $p_n(\cdot)$. It has to be noted, that it might not be easy to sample from the densities (3.9) and (3.10). In this situation, one has to take a constant r and perform an additional Metropolis-Hastings acceptance step to ensure sampling from the desired distribution. This of course lowers the acceptance probability and may lead to poor mixing performance [HT10].

In Algorithm 3.4 we give a pseudocode for a basic population-based Markov chain Monte Carlo algorithm. This algorithm does not give any details but just mentions rules R, which describe, how one should apply the mutation, exchange and crossover operators. This is done, because the algorithm can and should be used very flexibly to obtain reliable samples of the desired distribution $p(\cdot)$. The rules R may contain:

- which crossover operator is used, i.e., real crossover or snooker crossover
- if the snooker crossover operator is used, which direction is used, e.g., the direction proposed by Liang and Wong [LW01] or the one proposed by ter Braak [tB06]
- which of the genetic operators are performed in one sampling step, e.g., only one genetic operator can be performed in one sampling step or several consecutively
- probabilities for genetic operators to occur in one sampling step, e.g., one maybe wants to do more crossover steps than mutation steps

3.1.5. General aspects concerning efficiency of sampling algorithms

As we have seen in the previous four sections, it is not easy to create efficient sampling algorithms. For the Metropolis-Hastings algorithm one has to define a proposal distribution such that the rejection rate is small and the whole state space is explored with a running time as fast as possible. For the hybrid Monte Carlo algorithm one additionally has to fix a number of leapfrog steps and a step size of the leapfrog steps. And for population-based MCMC algorithms the number of chains N and several parameters for the genetic operators used have to be set. There are two questions arising concerning this topic:

1. How many steps do we need to reach the invariant distribution?
2. How many steps do we have to sample to obtain a reliable description of the distribution we are interested in?

Both questions are, in general, not easy to answer. As a summary of the literature one can say:

It depends on the underlying distribution.

For finite Markov chains, [Beh00] gives a good overview of the theoretical background of an answer to the first question, called *rapid mixing* of sampling algorithms. Since for finite Markov chains the transition matrix describing the invariant distribution (note that for finite Markov chains there is always an invariant distribution) has the invariant distribution as its left eigenvector. And because of the form of the distribution matrix (sum of rows is equal to one), all eigenvalues are smaller than or equal to one where one is really an eigenvalue. He proves that the convergence rate of the Metropolis-Hastings algorithms to the invariant distribution depends on the size of the second largest eigenvalue, the smaller it is, the faster the convergence rate.

For general state space Markov chains, the theory is similar to the theory of finite Markov chains (see [GRS98] Chapter 4): Instead of a transition matrix one has a transition kernel and considers the *spectrum* of it instead of the eigenvalues.

One important term to mention and introduce for the usage of MCMC algorithms in practice is the obvious need of neglecting samples at the beginning of the sampling procedure, called the *burn-in phase*, since these samples do not follow the distribution of interest. The samples in the burn-in phase will not be used for further analysis of the samples generated from the invariant distribution, i.e., the distribution of interest. How to set the length of the burn-in phase is not a trivial task and problem-specific.

Beyond these theoretical considerations, there are several modifications of the classical MCMC algorithms, as described in the previous four subsections, called *adaptive* MCMC algorithms. The different authors define the word "adaptive" in a different way suitable for their purposes. As an idea one can say that "adaptive" mean to select a proposal distribution which depends on the previous sampled points such that the ergodicity property is still satisfied and the running time is more efficient.

General theoretical considerations of the efficiency of adaptive MCMC algorithms can be found in [AR05, AA06]. The first paper also gives an adaptive MCMC algorithm where the proposal distribution is $\mathcal{N}(x, \sigma I)$ with x as the current point and $\sigma > 0$ and the parameter σ is updated in such a way that the asymptotic acceptance rate τ is $\tau = 0.234$, which is noted in [RGG97, RR01] to be optimal under some technical conditions. Another way to update

the proposal distribution is also to consider the covariance matrix of the previous sampled points, which has been done, for example, by [HST01]. This algorithm satisfies the necessary ergodicity properties, but does not satisfy the Markov property anymore.

3.2. Parallel Computing

The general idea of *parallel computing* is to run a computational program on several *processors* at the same time, i.e., in parallel, to save *computational time* and/or divide the *memory load* onto several processors. Although advantageous, parallel programming is more difficult compared to *serial programming*. This section is based on [Kep09, KMK]. For a detailed information on parallel computing, especially on the hardware needed, see [GKKG03].

Several reasons for parallel computing being more difficult than serial computing are that one has to keep track of on which processor specific data is located and if other processors need also the same data. Thus, it has to be considered, if data have to be distributed from one processor to another where it is needed. This distribution of data and the distribution of a program to several processors produce an *overhead*, i.e., additional computational time and memory is needed, which is not present in serial programs. The overhead may be very large, if a lot of data has to be distributed, i.e., the processors have to communicate a lot. This may even lead to the strange situation, that the computational time is even longer for a parallel program distributed on more processors as compared to a serial program run on one processor. This is the case if a lot of communication has to be performed between the processors, which produces a huge overhead although the work load per processor is smaller. For this reason, it is essential for efficient parallel computing to find a trade-off between the number of processors the work is distributed to and the amount of communication needed between the processors. Finding this trade-off is not always intuitive and often requires some effort of iterative thinking and running with different numbers of processors.

Now, that we know that communication between processors is a crucial step, we may ask: How is this communication organized in a parallel program? Basically, there are three models:

1. *manager/worker* model
2. *message passing* model
3. *distributed array* model

The manager/worker model is the simplest model for parallel computing. There the work is distributed into several independent tasks and the workers do not communicate with each other. The only communication taking place is the communication between the workers and the manager.

However, for a lot of problems, the communication between all processors is needed. The message passing model accounts for this. This is typically done with the *Message Passing Interface* (MPI) standard [GLS99]. In the MPI every processor needs to have a unique identifier P_{ID} and know how many other processors also work on the same parallel program. The drawback of this model is that the effort for programming is increased, since every message that has to be sent between different P_{ID}s has to be programmed individually by the programmer. And this causes a lot of additional *code* and complicates *debugging*, i.e., testing the program.

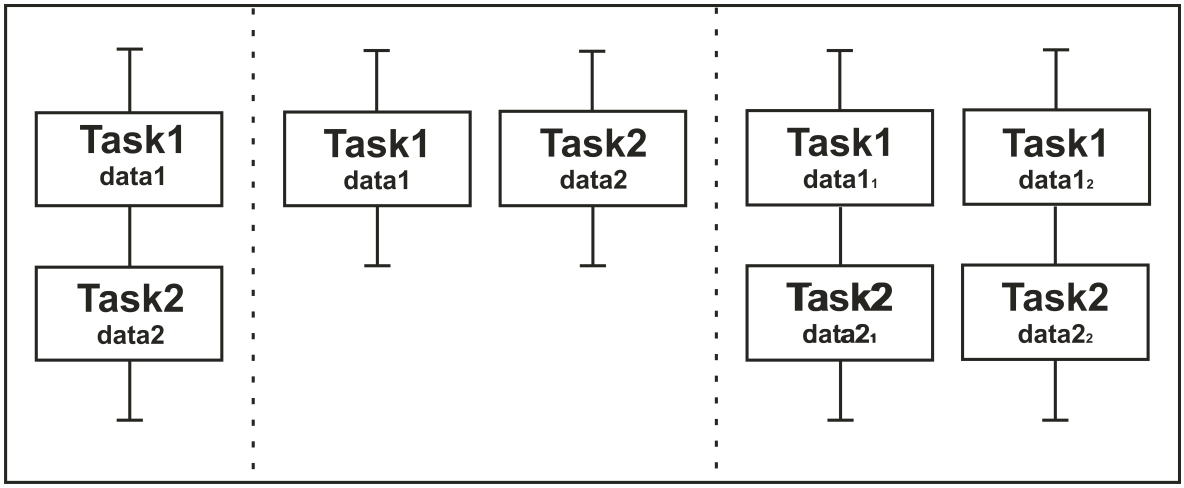


Figure 3.2.: *Scheme for the three parallel computing models: a single-program single-data model on the left, a MPMD model in the middle and a SPMD model on the right. In the serial program on the left two tasks are performed one after the other on data*1 *and data*2*, respectively. In the parallel program based on a MPMD model two processors are used, where on the first one, Task*1 *is performed on data*1 *and the second calculates Task*2 *on data*2*. On the right a parallel program based on the SPMD model is depicted. There, the same program as in the serial program is run on two processors, but the data needed for the two tasks is divided across the two processors.*

The distributed array model is a compromise between the two other models. Well, in fact, it needs the MPI standard as basis. However, the usage of MPI is implicit and hided from the user. So, how does the distributed array model work? Basically, the user sees one big global array, where different parts of the array are located on different processors. How the global array is distributed is defined by the user, but the needed data distribution and communication is done by the MPI layer beneath automatically. Thus, the user does not have to take care of it and the code is shorter and better readable. Every P_{ID} is then responsible for the calculation on the data stored locally on it.

Now we know how communication is organized. The next step is to understand how the parallel program can be organized itself. There are to main models to organize parallel programs:

1. *multiple-program multiple-data* (MPMD) model
2. *single-program multiple-data* (SPMD) model

To understand this two models better, we first mention, that a serial program will be organized according to this nomenclature as a *single-program single-data* model. This name becomes clear, if we consider that we run one program with one dataset and perform one task of the program after the other. In a MPMD the different tasks of the serial program are divided across different processors and the different datasets needed for this tasks are located where the task is performed. In a SPMD model, the same program is run on all processors, but the data needed for the different tasks is distributed to the processors. Figure 3.2 gives a scheme for this three models where the parallel programs are run on two processors each.

It has to be mentioned, that in Figure 3.2 no information about the *runtime* of the programs is depicted. It does not mean, that in the MPMD model the runtime is much shorter than

in the other models. In general, one cannot say much about the runtime for the different models. Of course, one goal of parallel computing is to reduce the computational time of the algorithm. Thus, both, parallel programs based on the MPMD model as well parallel programs based on the SPMD model should have a shorter runtime compared to the serial program. To measure the performance of parallel programs in general, the most often used measure is *speedup* S_p, which is defined as

$$S_p = \frac{\text{time}_{serial}}{\text{time}_{parallel}(p)}, \tag{3.11}$$

i.e., the runtime of the serial program is divided by the runtime of the parallel program performed on p processors. The ideal speedup is when the parallel program runs p times faster on p processors than the serial program runs on one processor. This speedup is called *linear*. A common observed speedup behavior is *sublinear* speedup. This speedup is not as good as linear speedup, but for increasing number of processors the runtime of the parallel program decreases. There are two main reasons for a sublinear speedup. First, as we mentioned before, communication creates overheads which increases the runtime. Second, not all parts of the serial program are parallelizable. Thus, a linear speedup can never be achieved for programs with non-parallelizable parts in it. An upper bound for the speedup for parallel programs is given by *Amdahl's law* [Amd67]

$$S_p \leq \frac{1}{\text{frac}_{\text{serial}} + (1 - \text{frac}_{\text{serial}})/p}, \tag{3.12}$$

where $\text{frac}_{\text{serial}}$ denotes the fraction of time spent on serial operations, i.e., the non-parallelized part of the code, and p denotes the number of processors. From this formula the maximum possible speedup can be derived by

$$\lim_{p \to \infty} S_p \leq \frac{1}{\text{frac}_{\text{serial}}}.$$

As an example, we can assume that only half of the serial code can be parallelized. Thus, the maximum possible speedup is 2 and cannot be increased with the usage of more processors. In Figure 3.3, Amdahl's law and the maximum possible speedup is depicted for different proportions of parallelizable code.

Another common observed speedup behavior is *saturation*. We mentioned this earlier as the strange case, where the runtime of the process is higher for a parallel program distributed on more processors. In more detail, we first see for increased number of processors an increase in speedup, then for increasing number of processors the speedup reaches a maximum and decreases for splitting up the program to more processors. This speedup behavior is observed in cases the communication load, and thus the overheads, exceeds the saving of computation time due to parallelization.

3.3. Receiver Operating Characteristics

Receiver operating characteristics (ROC) graphs are a method to visualize the performance of *classifiers*. This section is based on [Faw06]. We will first describe the two-class problem where we classify between two classes. Since this thesis deals with gene regulatory networks, we will focus on the example of present/absent edges to describe ROC graphs for the two-class problem. For the three-class problem we will focus on the example of absent/positive/negative edges in a network, which have to be classified.

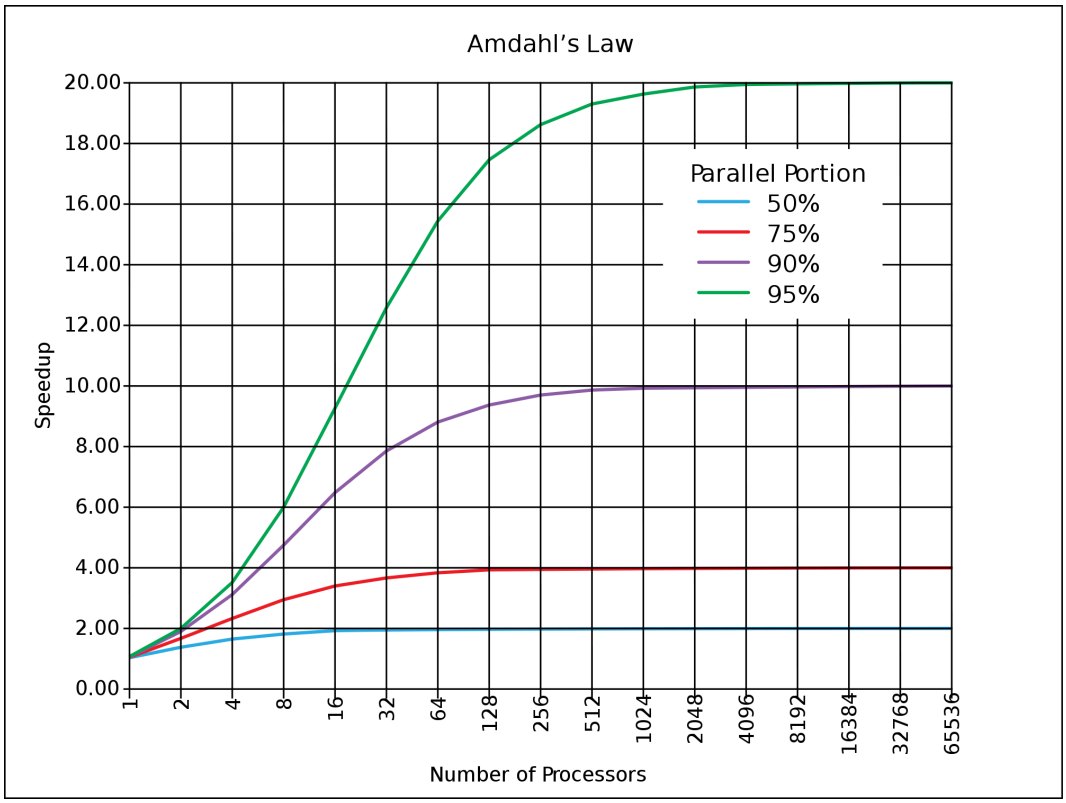


Figure 3.3.: *Amdahl's law for different amounts of parallelizable code. Source:* `http://en.wikipedia.org/wiki/Amdahl's_law`

3.3.1. Two-class problem

As mentioned above, we consider classifiers which give as output, if an edge is present or absent. We will call these classifiers also *discrete classifiers*. There exist also classification models, which produce continuous outputs. We will call these classifiers *continuous classifiers*. For them, thresholds can be applied to predict membership to classes of the output. The classifier of a two-class problem can have four possible outcomes, which can be denoted in a *confusion matrix* as given in Table 3.1. With the confusion matrix we define several performance metrics of a classifier we will use. The first metric is the *true positive rate*, which is also called *recall* and *sensitivity* and defined as

$$\text{true positive rate} = \frac{\text{TP}}{\text{TP} + \text{FN}}.$$

The *false positive rate* (fpr) is defined as

$$\text{false positive rate} = \frac{\text{FP}}{\text{FP} + \text{TN}}.$$

Another used metrics are *specificity*, which is just $1 - \text{fpr}$ and *precision*, which is defined as

$$\text{precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}.$$

		predicted	
		present edge	no edge
actual	present edge	True Positives (TP)	False Negatives (FN)
	no edge	False Positives (FP)	True Negatives (TN)

Table 3.1.: *Confusion matrix of the two-class problem (no edge, present edge).*

How can this metrics be now used to measure the performance of a classifier? Of course, we want to have a high sensitivity, since we want to find all the present edges in the true network, also called the *golden standard*, and do not classify the edges present in the golden standard as being no edge. But we also want to have a low false positive rate, since we do not want to classify not present edges in the golden standard network as being a present edge. In one type of a ROC graph we plot the sensitivity on the y-axis and $1-$ specificity on the x-axis. This can be seen as depicting the benefits of a classifier (true positives) versus its costs (false positives).

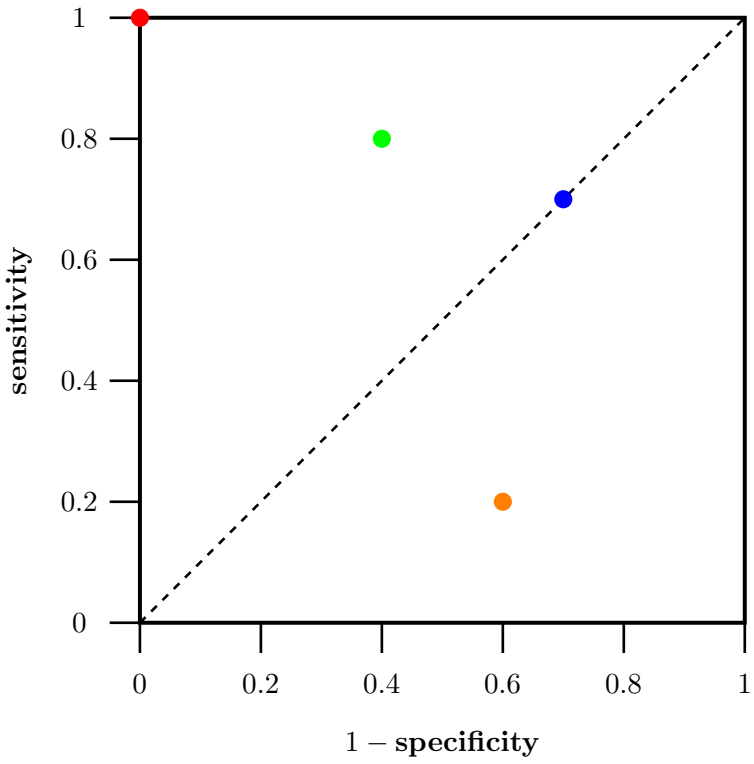


Figure 3.4.: *A basic ROC graph and the performance of different discrete classifiers. The red dot represents a perfect classifier. The blue dot represents a classifier which is as good as random guessing. The orange dot represents a classifier worse than random guessing. But if the labels of this classifier are reversed, the new classifier is represented as the green dot. This figure is adapted from [Faw06].*

In Figure 3.4 the performance of several discrete classifiers is depicted. The red dot represents a perfect classifier, since it learns all present edges and no absent edge is learned as being

a present one, i.e., sensitivity = 1 and 1 − specificity = 0. The blue dot represents a classifier which is as good as random guessing. In general, the diagonal line in Figure 3.4 indicates the performance of a classifier being as good as random guessing. Thus, a classifier with a good performance is found in the upper left corner of the ROC graph. Why represents the diagonal line the performance of random guessing? Let's assume that a random classifier learns half of the present edges right. Then we can expect that is also learns half of the non-present edges right. In general, one can say, that a random classifier will learn $x\%$, $x \in [0, 100]$, of the possible edges in the golden standard network as present edges, which leads then to a sensitivity of $x/100$ and to a $1 -$ specificity of also $x/100$. And this is exactly represented with a diagonal line.

It remains to explain the remaining two dots in Figure 3.4. The orange dot represents a classifier worse than random guessing. This is, of course, a bad classifier. But it basically means that the classification outputs have just to be reversed to make true negatives from false positives and true positives from false negatives. Doing so, we get a classifier with a performance better than random guessing which is depicted as the green dot.

How can we now measure the performance of a continuous classifier? We can specify a threshold and fix that all values greater than the threshold represent a present edge and all values lower than the threshold represent an absent edge. With this procedure we get the same situation as for a discrete classifier. We can now obtain a point in the ROC graph giving the performance of the continuous classifier for the specified threshold. However, how can we decide that we choose the right threshold? Since for different threshold we get different performance points and we do not know which of them will be the one giving the best performance[5] beforehand, we will calculate points in the ROC graph for varying thresholds. An exemplary ROC graph for a continuous classifier is depicted in Figure 3.5. The obtained points in the ROC graph for varying thresholds are then connected and the *area under curve* (AUC) can be calculated as the integral of the ROC curve. In Figure 3.5 we have AUC = 0.73. The theoretical maximum for a perfect classifier is, of course, 1. With the AUC we are now able to compare different continuous classifiers: the greater the AUC value, the better the performance of the classifier.

The ROC curve can also be used to specify the "best" threshold, i.e., the threshold, which gives a point in the ROC graph with the best performance. The blue dashed line in Figure 3.5 is the line furthest away from the diagonal in the ROC graph which touches points in the ROC curve. Thus, the two points $(0.1, 0.5)$ and $(0.4, 0.8)$ represent the points with the best performance of the classifier and the associated thresholds are then the "best" thresholds.

An important property of the sensitivity vs. $1 -$ specificity ROC graphs is that they are insensitive to the distribution of present and absent edges in the golden standard network. It doesn't matter what the percentage of the edges in the golden standard network are present or absent. The corresponding ROC curve and the AUC value will be always the same for the same classifier. However, since, especially for biological networks, the percentage of present edges in the golden standard network is much smaller than the percentage of absent edges, i.e., biological networks are *sparse*, we also want to take this knowledge into account to be sure that the classifiers we propose are suitable for biological networks.

This can be done with precision vs. recall ROC graphs. There on the y-axis we plot precision and on the x-axis we plot recall, i.e., the true positive rate. The difference between the previous ROC graphs is now that with the consideration of precision we take into account the

[5] "best performance" in the sense, that the point is furthest away from the diagonal in the ROC graph

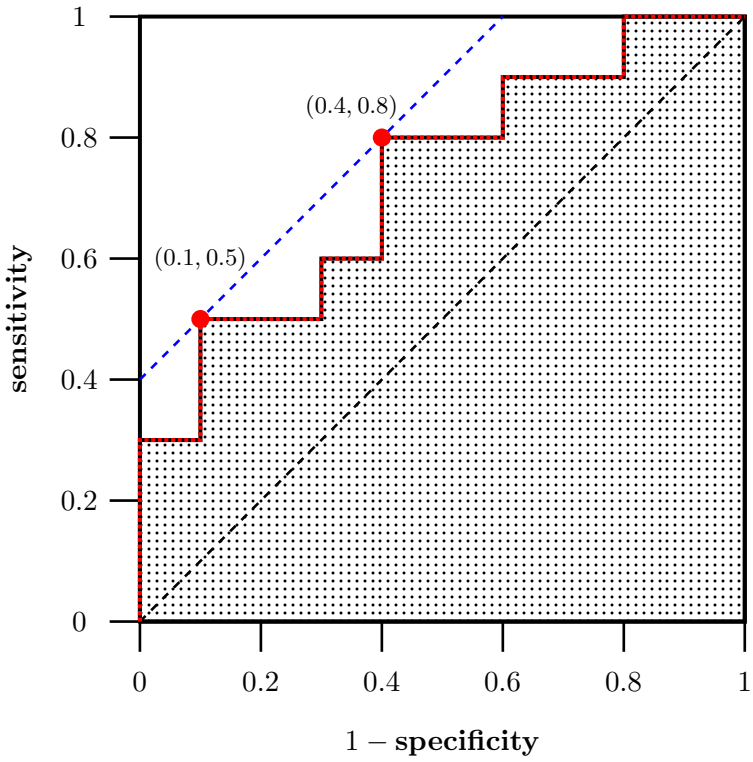


Figure 3.5.: *A basic ROC graph for a continuous classifier with the corresponding ROC curve depicted in red. The dotted area depicts the corresponding AUC of* 0.73 *and the dashed black line represents the (theoretical) ROC curve for random guessing. The blue dashed line touches the points of the ROC curve which represent the points with the best performance.*

distribution of the present and absent edges. This can be seen by looking at the confusion matrix in Table 3.1. Whereas in the sensitivity vs. $1 -$ specificity ROC graphs we depict the relationships of the rows of the confusion matrix against each other, we plot in precision vs. recall ROC graph the relationship of the first column of the confusion matrix. And thus, it considers the sparsity of biological networks. The value for random guessing is not 0.5 anymore, as in the sensitivity vs. $1-$specificity ROC graphs, but is dependent on the amount of present edges in the golden standard. Why is this the case? Let's assume we have a golden standard network with 100 possible edges where 90 are absent and 10 are present. Now we take a random classifier which characterizes possible edges with the probability of x, $x \in [0, 1]$, as present. We can then except, that $x \cdot 90$ of the absent edges are learned as being present and $x \cdot 10$ of the present edges are learned as being present. Thus, for precision we have

$$\text{precision} = \frac{x \cdot 10}{x \cdot 10 + x \cdot 90} = \frac{1}{10},$$

which is independent of x and is the proportion of present edges of the possible edges.

In Figure 3.6 an exemplary precision vs. recall ROC graph for a continuous classifier is depicted. As in the sensitivity vs. $1 -$ specificity ROC graph, we can determine the "best" threshold which gives a point in the ROC graph with the best performance. However, since the best classifiers are found in the upper right corner of the precision vs. recall ROC graph,

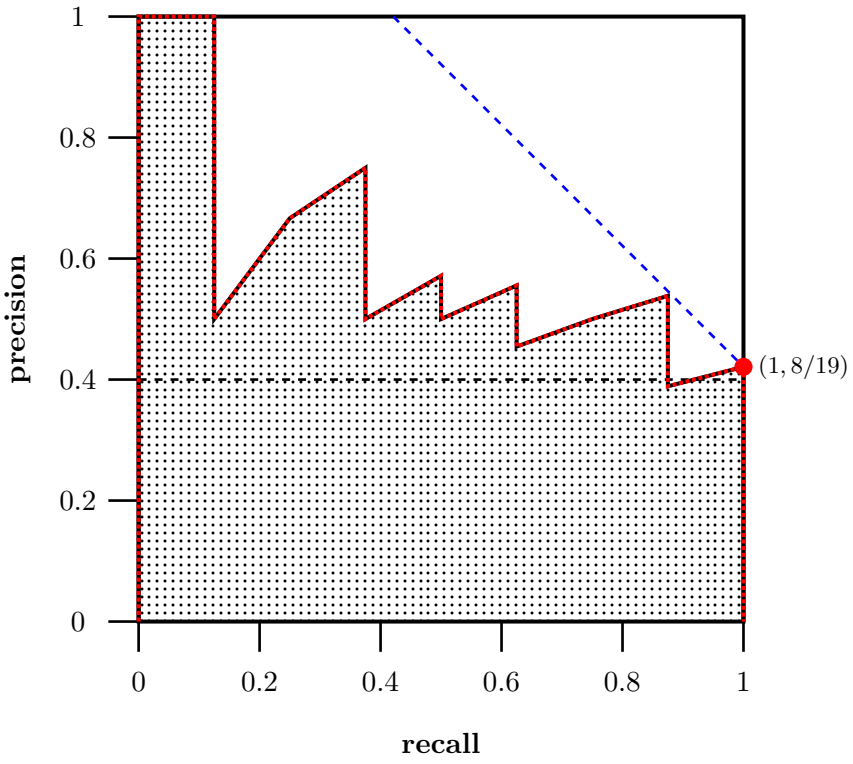


Figure 3.6.: *An exemplary precision vs. recall ROC graph for a continuous classifier with the corresponding ROC curve depicted in red. The dotted area depicts the corresponding AUC and the dashed black line represents the (theoretical) ROC curve for random guessing. The blue dashed line touches the point of the ROC curve which represent the point with the best performance.*

the best classifiers are the ones which are touched by the line which is furthest away from the function $f(x) = -x + 1$. In Figure 3.6 this line is depicted as the blue dashed line and the optimal threshold can be obtained at the point $(1, 8/19)$ for this example.

3.3.2. Three-class problem

Now we will come to the case, where our classifier gives as output, if an edge in the network is absent, positive or negative. Thus, we do not have a two-class problem anymore, but a three-class problem. The usual procedure for classification problems with more than two classes is to reduce it to the case of two classes by calculating several ROC graphs and AUC values where in every one, one of the classes is considered as the positive class and all the other classes are considered as the negative class. The overall AUC value is then calculated as the weighted sum of all the single AUC values. See Section 9 in [Faw06] for more details.

We will use a different approach. The similarity to other methods is that we also reduce it to the case of a two-class problem. But instead of averaging the different AUC values, we change the confusion matrix as given in Table 3.2. This confusion matrix was first used by Böck [Böc08] and Ritter [Rit08] for the analysis of three-class problems. With this confusion matrix the values for random guessing change and are now dependent on the proportion of the absent edges in the golden standard network. The exact formula and the proof can be

Algorithm 3.5 ROC point generation and AUC calculation

Require: set of learned edges L, golden standard network, number of ROC points p_{ROC}

$l_{\max} \leftarrow \max\{l \mid l \in L\}$
$l_{\min} \leftarrow \min\{l \mid l \in L\}$
$t_s \leftarrow \frac{l_{\max} - l_{\min}}{p_{\text{ROC}}}$ is the threshold varying size
$t_h \leftarrow l_{\max}$ is the starting threshold
sensitivity $\leftarrow 0$
specificity $\leftarrow 1$
precision $\leftarrow 1$
$\text{AUC}_{\text{ROC}} \leftarrow 0$
$\text{AUC}_{\text{P2R}} \leftarrow 0$
for $i = 1$ **to** $p_{\text{ROC}} + 1$ **do**
  $\text{sensitivity}_{\text{old}} \leftarrow$ sensitivity
  $\text{specificity}_{\text{old}} \leftarrow$ specificity
  $\text{precision}_{\text{old}} \leftarrow$ precision
  consider all learned edges with learned value $\geq t_h$ to be present edges and the others to be absent edges
  determine according to the confusion matrix in Table 3.1 and the golden standard network the values for sensitivity, specificity and precision
  $\text{AUC}_{\text{ROC}} \leftarrow \text{AUC}_{\text{ROC}} + \frac{\text{sensitivity}_{\text{old}} + \text{sensitivity}}{2} \cdot \mid \text{specificity}_{\text{old}} - \text{specificity} \mid$
  $\text{AUC}_{\text{P2R}} \leftarrow \text{AUC}_{\text{P2R}} + \frac{\text{precision}_{\text{old}} + \text{precision}}{2} \cdot \mid \text{sensitivity}_{\text{old}} - \text{sensitivity} \mid$
  $t_h \leftarrow t_h - t_s$
end for
return area under curve values for the sensitivity vs. $1-$specificity ROC curve AUC_{ROC} and for the precision vs. recall ROC curve AUC_{P2R}

found in Appendix A.

		predicted		
		positive edge	negative edge	no edge
actual	positive edge	TP	FP	FN
	negative edge	FP	TP	FN
	no edge	FP	FP	TN

Table 3.2.: *Mapping of three-class classification problem (no edge, positive edge, negative edge) onto two-class* ROC / PR *evaluation.*

Part II.

Inference of gene regulatory networks using stochasting sampling

CHAPTER 4

Models for gene regulatory networks

> Essentially, all models are wrong, but some are useful.
>
> *(George E. P. Box)*

In this chapter we will describe different possibilities to model *gene regulatory networks* (GRNs). We will look at the advantages and disadvantages of different kind of deterministic models (*statistical models*, *discrete models* and *continuous models*) and of stochastic models (*single-molecule models*). We do not make a claim of completeness of all possible methods to model GRNs. Instead, we show the most often used types to illustrate the diversity of the models used. There is a vast amount of reviews available describing models for gene regulatory networks. An excellent one was written by Karlebach and Shamir in 2008 [KS08]. However, there are also another good ones, i.e., [MS07, SB07, KR08, HLT+09].

The aim of all models of GRNs is to find the structure between different genes. To state it more formally, we say that we have a set $V = \{v_1, \ldots, v_n\}$ of n genes given. We now want to find interactions $\mathcal{E}$ which describe the structure between the genes V. This can be nicely represented in a graph $G = (V, \mathcal{E})$, where the genes V denote the nodes and the interactions $\mathcal{E}$ denote the edges. Depending on the model, the graph G can be *directed* or *undirected*.

Definition 4.0.1. The graph $G = (V, \mathcal{E})$ of a gene regulatory network is called the *topology* of a gene regulatory network.

4.1. Statistical models

Statistical models all have in common, that the nodes V are treated as random variables and the edges $\mathcal{E}$ represent probabilistic relationships between these random variables. The main advantage of statistical models is the ability to capture the stochasticity of gene expression data. I refer the explanation of this section to the second chapter in the doctoral thesis by Florian Markowetz [Mar05].

4.1.1. Coexpression networks

The underlying biological view of coexpression networks is the *guilt-by-association heuristic*. What it means, is that genes which show similar expression profiles are supposed to have similar functions. This follows the hypothesis, that several genes are responsible for the same cellular function and are activated at the same time to perform the desired task.

There are several possibilities to calculate the similarity between gene expression profiles. The most simple measure for similarity is *correlation*. However, since correlation measures only a linear relationship between random variables, it neglects nonlinear properties. Another possibility is *mutual information* which was used by Butte and Kohane to introduce *relevance networks* [BK00]. Margolin *et al.* [MNB+06] implemented the ARACNE-algorithm, which improves relevance networks in that way, that it removes the vast majority of indirect interactions. Recently, a version of the ARACNE-algorithm was presented, which is able to reconstruct directed graphs with time-series data [ZMC10]. Kaleta *et al.* [KGS+10] and Chaitankar *et al.* [CGP+10] used a similar information theoretic approach to infer a directed network with time-series data for *Escherichia coli* and *Saccharomyces cerevisiae*, respectively.

Except for the last example, the obtained topologies with coexpression networks are undirected, since they just provide a similarity between gene profiles and do not model regulation between genes. Because coexpression networks are easy to calculate, there are a lot of applications present in the literature, e.g., [SSDK03, MHDD09, XCP+10].

A clear drawback of coexpression networks is, that genes may have similar expression profiles by chance where the genes are not biologically related. Wolfe *et al.* [WKB05] tested the general applicability of the guilt-by-association heuristic to the transcriptome. The authors calculated probabilistic scores between each gene and each Gene Ontology (GO) category and tested their results with ROC curves. Their results show that the AUC values are much better than random guessing, but are also away from being perfect.

Recently, a new statistic called *local correlation* was developed to identify nonlinear relationships [CAR+10]. The authors show that local correlation outperforms correlation and has higher statistical power than mutual information. Thus, it seems to be a promising measure that can be exploited in the future.

4.1.2. Graphical models

In comparison to coexpression networks, graphical models used to model GRNs calculate *conditional dependencies* between genes, given information of other genes. Thus, one only connects two genes, if their correlation cannot be explained by other genes. This is also implicitly done in the ARACNE-algorithm described in Section 4.1.1, where no edge is drawn, if the correlation between two genes can be explained by a single third gene.

The question, one is facing now is: What about the correlation between two genes that can be explained by an *arbitrary* subset of other genes? There are two often used methodologies to model GRNs which deal with this question, which will be described now. Graphical models are able to give undirected but even directed graphs. Gaussian graphical models give undirected graphs, whereas Bayesian networks are able to provide directed graphs.

Gaussian graphical models

The question answered with Gaussian graphical models (GGMs) is, if the correlation between two genes can be explained by *all other* genes in the model. Thus, we only draw an edge

between two genes, if their correlation cannot be explained by all other genes. The name Gaussian already implies that the set of the nodes $\mathbf{X} = (X_1, \ldots, X_n)$ follows a multivariate Gaussian distribution, i.e., $\mathbf{X} \sim \mathcal{N}(\mu, \Sigma)$. It can be shown, that the conditional distribution of two random variables X_i, X_j, $i, j \in \{1, \ldots, n\}$, of a GGM given all other variables $\mathbf{X} \backslash \{X_i, X_j\}$ follows a bivariate normal distribution with *correlation coefficient* $\rho_{ij} = -k_{ij}/\sqrt{k_{ii}k_{jj}}$, where

$$K = \left(k_{ij}\right)_{i,j=1}^{n} := \Sigma^{-1} \tag{4.1}$$

denotes the *precision matrix* of the distribution $\mathbf{X}$. The value ρ_{ij} defined above is also called the *partial correlation coefficient* between X_i and X_j. Thus, we have shown, that two random variables X_i and X_j are independent given the remaining random variables $\mathbf{X} \setminus \{X_i, X_j\}$ if and only if the corresponding element of K is zero [Edw00].

Compared to coexpression networks, where the correlation is used, partial correlation coefficients usually vanish. Thus, they give a strong measure for dependence and a weak measure for independence, whereas for correlation alone it is the other way around [SS05a].

The procedure to obtain GRNs with GGMs is performed in four steps:

1. Calculate an estimate $\hat{\Sigma}$ of the covariance matrix Σ, usually done with the *sample covariance matrix*
$$\hat{\Sigma} = \frac{1}{M-1}\left(\mathbf{X} - \bar{\mathbf{X}}\right)^T\left(\mathbf{X} - \bar{\mathbf{X}}\right) \tag{4.2}$$
where $\bar{\mathbf{X}} = (\bar{X}_1, \ldots, \bar{X}_n)$ denotes the sample mean and M denotes the number of samples
2. Invert $\hat{\Sigma}$ to obtain an estimate $\hat{K}$ for the precision matrix K
3. Perform statistical tests to deduce which elements of $\hat{K}$ are significantly different from zero
4. Put an edge between two genes, where the corresponding entry of $\hat{K}$ was found to be different from zero in Step 3

One problem with this procedure is, that to obtain accurate results, the number of samples M has to be larger than the number of genes n considered. Otherwise, the sample covariance matrix $\hat{\Sigma}$ is not positive definite and cannot be inverted [Fri89, SS05a]. Since it is a usual problem in modeling GRNs, that there are more genes than different samples ($n \gg M$), several methods have been proposed to deal with this problem, e.g.,:

- Wille *et al.* [WZV+04] applied modified GGMs to analyze the isoprenoid gene network in *Arabidopsis thaliana*. For this purpose, they applied GGMs to subnetworks of three genes and combined the obtained results to restore the whole network.
- Schäfer and Strimmer [SS05a] used first *bootstrap aggregation* [Bre96] to stabilize the estimator of the sample covariance matrix. Second, they calculated the *Moore-Penrose pseudoinverse* [Pen55] of this obtained estimate of the sample covariance matrix. The Moore-Penrose pseudoinverse is also applicable to singular matrices. Their method is implemented in the R-package *GeneNet* [SORS06] and was extended by Opgen-Rhein and Strimmer [ORS07] to obtain also directed graphs.
- Schäfer and Strimmer [SS05b] review the three general used methods to circumvent the "few-samples problem": dimension reduction prior to analysis (e.g., clustering of genes [TH02]), computation of low order partial correlation coefficients and regularized variants of graphical Gaussian modeling.

- Kontos [Kon09] proposes a non-biased estimator for the covariance matrix after proving that the estimator used by Schäfer and Strimmer [SS05a] is biased. Furthermore, he gives more efficient algorithms for the inference of GGMs than previous known algorithms.

GGMs were applied by de la Fuente *et al.* [dlFBHM04] on *Saccharomyces cerevisiae* microarray data. And Ma *et al.* [MGB07] investigated the *Arabidopsis thaliana* transcriptome with GGMs. Wang *et al.* [WCD05] combined GGMs with *graph orientation rules* to obtain directed networks and applied their method on microarray data for the mitogen-activated protein kinase (MAPK) pathways in yeast.

Bayesian networks

We said in the previous section, that GGMs are useful graphical models with a lot of applications. But they only provide undirected networks. Bayesian networks (BNs), in contrast, are able to give directed networks.

For GGMs we draw an edge, if the correlation between two genes cannot be explained by *all other* genes. In Bayesian networks we will draw an edge, where the correlation between two genes cannot be explained by *any subset* of the other genes. This collection of independencies already implies directions of some edges in the network [Pea00, SGS00]. We give a formal definition:

Definition 4.1.1. A *Bayesian network* for a set of random variables $\mathbf{X} = (X_1, \ldots, X_n)$ consists of

1. a directed acyclic graph (DAG) S that encodes a set of conditional independence assertions about variables in $\mathbf{X}$
2. a set P of local probability distributions associated with each variable.

Together, these two components define the joint probability distribution for $\mathbf{X}$ by

$$p(\mathbf{X}) = \prod_{i=1}^{n} P(X_i \mid \mathrm{pa}_i, \theta), \tag{4.3}$$

where pa_i denotes the *parent nodes* of node X_i in S and θ denotes the parameters which describe the set of local probability distributions P.

To learn a Bayesian network, given data D, we thus have to learn the underlying DAG S given local probability distributions P. Chickering *et al.* [Chi96, CHM04] showed, that this inference problem is a difficult task, i.e., they proved that it is, at least, NP-complete. A tutorial for the inference of Bayesian networks was given by Heckerman [Hec96].

To score different network structures S, several methods can be used:

1. The *Maximum Likelihood* (ML) estimate:

$$\mathrm{ML}(S) = \max_{\theta} p(D \mid S, \theta) \tag{4.4}$$

   The ML estimate has the drawback, that is tends to *overfitting*, i.e., the score $\mathrm{ML}(S)$ is higher for larger networks with more edges, which capture all details (especially noise) in the data.

2. To control overfitting, a regularized version of the ML estimate was proposed: the *Bayesian information criterion* (BIC):

$$\mathrm{BIC}(S) = \max_{\theta} p(D \mid S, \theta) - \frac{d}{2} \log M, \tag{4.5}$$

where d denotes the number of parameters θ used to parameterize the local probability distributions P and M the number of samples as denoted in Section 4.1.2.

3. Another score, circumventing the overfitting problem [Mar05], is to average over parameters θ instead of maximize over them. Thus, we obtain a *marginal likelihood* $p(D \mid S)$ by treating the parameters θ as *nuisance parameters* and integrating them out, i.e.,

$$p(D \mid S) = \int p(D \mid S, \theta) p(\theta \mid S) \, d\theta. \tag{4.6}$$

With the usage of Bayes' theorem, we obtain the desired probability of a network structure S given the data D:

$$p(S \mid D) = \frac{p(D \mid S) p(S)}{p(D)}. \tag{4.7}$$

The marginal likelihood (4.6) can either be calculated analytically using *conjugate priors* [GCSR04], i.e., the posterior is of the same distributional form as the prior, or has to be approximated.

For the case, when we have to deal with missing or hidden data, which is very often the case for biological problems, we can maximize $p(D \mid S, \theta)$ in (4.4) and (4.5) and $p(S \mid D)$ in (4.7) using the *Expectation-Maximization algorithm* (EM-Algorithm). The EM-algorithm was introduced by Dempster *et al.* [DLR77] in 1977 and proposes an iterative approach of the expectation of missing data, given measured data and current parameters θ (E-step) and the maximization of the likelihood given the missing data obtained in the E-step and the measured data (M-step).

Now that we can calculate scores for given network structures S, we have to find S with the highest score. But since the number of DAGs on n edges explodes with growing n, we have to use either an informative prior $p(S)$ in the Bayes' score (4.7) or/and reduce the search space or/and use a heuristic to find the best network structure S. Several approaches have been proposed to learn network structures efficiently, e.g., [HGC95, NRF04, PBT05, KZRZ09, NMB+09]. As a special example, Friedman [Fri97, Fri98] introduced the *Structural Expectation-Maximization-Algorithm*, where the network structure S is learned inside the EM-Algorithm.

For the inference task it has to be considered, that several Bayesian network structures with the same underlying undirected graph, but different direction of edges may represent the same statements of conditional independence [VP90]. These network structures are said to be in the same *equivalence class*. Thus, even with infinitely many samples, we are not able to distinguish between network structures in one equivalence class. For the interpretation of gene regulatory networks, this drawback of Bayesian networks is crucial: we cannot be sure, that the obtained directed edges really represent regulation between genes.

In the literature, there is a lot of applications of BNs for the inference of GRNs. Friedman *et al.* [FLNP00] give one of the first papers applying BNs to infer the GRN of *Saccharomyces cerevisiae*. Liu *et al.* [LLT+09] use BNs and heterogeneous data to infer miRNA-mRNA interactions.

Another drawback of Bayesian networks is, that they have to be acyclic to fulfill the factorization of the joint probability distribution in (4.3). Thus, feedback loops and cycles, as often present in biological systems [LBK+07, AJL+08], cannot be modeled. This can be circumvented by using time-series data and assuming, that a gene x at time point $t+1$ is only allowed to have parents at time point t. With this assumption we are able to model feedback loops with BNs, too. These kind of BNs are called *Dynamic Bayesian Networks* (DBNs) [FMR98,MM99]. DBNs were extensively used for the inference of GRNs, e.g., [NRF04, ZDJJ06, DEO+09]. Dojer *et al.* [DGM+06] extended the usage of DBNs also to data obtained with perturbation experiments.

4.2. Discrete models

In discrete models the nodes V can take only discrete values and for discrete time steps the values for nodes V are updated according to some *regulation function*.

4.2.1. Boolean networks

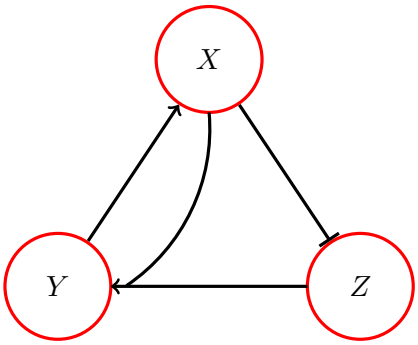
Let's start with the simplest form of discrete values that the nodes of the gene regulatory network can have: 1 for active or "on" and 0 for inactive or "off". These kind of discrete models are called *Boolean networks* and were described first by Kauffmann in 1969 [Kau69]. Let us first give a formal definition of Boolean networks and then discuss the used terms in detail. The following definition and considerations are based on [AMML11].

Definition 4.2.1. A *Boolean network of order* n is completely defined by an operator $A : \{0,1\}^n \to \{0,1\}^n$. The operator A can be specified by *transition functions*, also called *Boolean functions*, $f_i : \{0,1\}^n \to \{0,1\}$ for all $i \in \{1,\dots,n\}$ such that for $x = (x_1,\dots,x_n)$ and $x' = (x'_1,\dots,x'_n)$ the equation $x' = A(x)$ and the set of equations $x'_i = f_i(x)$, $i \in \{1,\dots,n\}$, are equivalent.

The question that comes up seeing this definition is: How do we obtain a graphical interpretation in form of a topology of the underlying gene regulatory network just with the operator A? To do this, we assume that for every node $i \in \{1,\dots,n\}$ there is a subset $S_i \subseteq V$ such that the transition function f_i depends on $x \in \{0,1\}^n$ only through the components x_j where $j \in S_i$. The graph that represents the topology is the one where the parents of node i are the nodes in S_i and this holds for all $i \in \{1,\dots,n\}$. The minimal subset S_i with this property is then called the *minimal graph of* A.

A nice property of Boolean networks is their ability to represent the dynamic behavior of the underlying gene regulatory network. For this purpose, one starts with an initial network state x_0 and applies the operator A to x_0 several times. Thus one obtains a sequence $\text{bool}_{x_0} := \left(A^t(x_0)\right)_{t=0}^{\infty}$. Since there are only 2^n states possible of the system, there have to be states which appear infinitely often in bool_{x_0}. This set of states is called an *attractor* of the system. Two types of attractors are possible. The first one is called a *fixed point* and is defined such that $x = A(x)$, where $x \in \{0,1\}^n$. The second type for an attractor is a *cycle*. It is defined as a finite sequence of states $x^1,\dots,x^m \in \{0,1\}^n$ such that each state is unique, it holds $x^{t+1} = A(x^t)$ for all $t \in \{1,\dots,m-1\}$ and $x^1 = A(x^m)$.

We will give a small example for a three node Boolean network to illustrate all the introduced terms. In Figure 4.1 (Left) the representation as a graph is depicted. The transition



t			$t+1$		
X	Y	Z	X	Y	Z
0	0	0	0	0	1
0	0	1	0	0	1
0	1	0	1	0	1
0	1	1	1	0	1
1	0	0	0	0	0
1	0	1	0	1	0
1	1	0	1	0	0
1	1	1	1	1	0

Figure 4.1.: *(Left) An example for a Boolean network of order* 3 *represented as a graph. The nodes* X*,* Y *and* Z *are representing genes and the topology shows that gene* X *inhibits gene* Z*, gene* Y *activates gene* X *and genes* X *and* Z *together activate gene* Y*. (Right) Representation for the same Boolean network as on the left with the operator* A *where for all* 8 *possible states the states for the three genes after applying the operator* A *are given.*

functions for this example can be written as the *logical operators*

$$X = Y \qquad Y = X\&Z \qquad \text{and} \qquad Z = \neg X. \tag{4.8}$$

The operator A that defines completely this Boolean network can be given in the form of a table, where for the $2^3 = 8$ elements of the set $\{0,1\}^3$ the states for the three genes are given as the output of the operator A. This table is shown in Figure 4.1 (Right).

It remains now to look at possible sequences bool_{x_0}. For this example only two different sequences are possible. One leads to a fixed point as an attractor and the other one leads to a cycle consisting of two states. These sequences are illustrated in Figure 4.2.

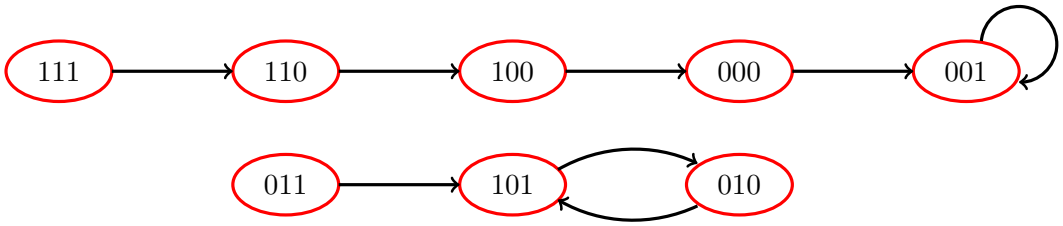


Figure 4.2.: *Sequences* bool_{x_0} *for the Boolean network represented in Figure 4.1 and* (4.8)*. The one on top shows a sequence which ends in the fixed point attractor* (001)*. The one on bottom shows a sequence which ends in a cycle attractor consisting of the two states* (101) *and* (010)*.*

For a more thorough description Boolean network models and its analysis, we refer the reader to [Kau93].

Boolean networks were used to analyze the robustness and the dynamics for a given topology of a gene regulatory network and such for given Boolean functions. This was done, for example, for the yeast transcriptional network by Kauffman *et al.* [KPST03] and by Li *et al.* [LLL+04] for the cell-cycle regulation in yeast. However, as pointed out by Karlebach and Shamir [KS08], these procedures are only practically possible for small networks. Giacomantonio *et al.* [GG10] enumerated all possible Boolean networks with five genes defining a small gene regulatory network underlying mammalian cortical area development and compared the interactions obtained with existing knowledge from literature.

The other common usage for Boolean networks is to infer gene regulatory networks with experimental data. To do so, first the data, of course, has to be discretized. This inference task needs to find the Boolean functions which define the Boolean network that is *consistent* with the data. Not always there are such Boolean functions, because of incomplete and/or noisy data.

Liang *et al.* [LFS98] developed the algorithm REVEAL which uses mutual information to estimate the underlying transition functions of Boolean networks from time-series data. Their algorithm performs well for networks with a small *in-degree*, i.e., the number of parents in the graph of a node, number per gene.

Lähdesmäki *et al.* [LSYH03] introduced an algorithm which finds in polynomial time Boolean functions which make as few "errors" as possible according to the given data. They used their algorithm to find all possible Boolean functions with a pre-defined error for a five-gene yeast network using time-series gene expression data from Spellman *et al.* [SSZ+98].

The two former algorithms are implemented in the R-package *BoolNet* [MHK10]. The BoolNet package provides also functions for the generation and the analysis in terms of attractors of Boolean networks.

Wittmann *et al.* [WKSR+09] transformed Boolean models into a system of ordinary differential equations using polynomial interpolation. They examined their approach on a model describing the activation of T-cells and proved that their method preserves the steady-state behavior of the Boolean model. The same approach was also used to predict interactions at the mid-hindbrain boundary [WBT+09].

Recently, Almudevar *et al.* [AMML11] used a Bayesian inference approach to infer Boolean networks with steady-state data obtained with perturbation experiments. Maucher *et al.* [MKKK11] proposed a fast algorithm for the inference of Boolean networks based on the calculation of correlation coefficients between two consecutive states. The authors applied their method to synthetic data representing a published *E. coli* network and to real data for a yeast cell-cycle network.

probabilistic Boolean networks

Boolean networks as defined in Definition 4.2.1 and described in the previous section are thoroughly deterministic. To include the uncertainty about the regulation of certain genes, Shmulevich *et al.* [SDKZ02] introduced *probabilistic Boolean networks* (PBN) as a new class of models for gene regulatory networks where they may exist more than one Boolean function for every gene each with a certain probability. The authors also propose an inference algorithm for PBNs based on *coefficients of determination*. The analysis of the dynamics of PBNs can be done with the rich theory on Markov chains[1]. This was done for gene expression data for a sub-network describing human glioma, where the joint steady-state probabilities for several groups of genes were obtained with MCMC algorithms [SGH+03]. A thorough overview of PBNs and their analysis can be found in [SD10].

Li *et al.* [LZP+07] compared PBN and dynamic Bayesian network approaches for the inference of GRNs from time-series data describing a 12 gene network of *Drosophila melanogaster*. Dynamic Bayesian networks were able to learn more interactions and gave a higher recall value. Kaderali *et al.* [KDZ+09] used a combination of Bayesian networks and PBNs to infer signaling pathways from RNAi data. Qian *et al.* [QGID10] introduced a method for the

[1] for a glimpse on this theory see Section 2.3

reduction of states present in the Markov chains arising in the analysis of the dynamics of PBNs. The authors prune states having only a small probability to appear in the stationary distribution and thus reduce the computational complexity of searching for optimal Boolean functions.

Lähdesmäki and Shmulevich provided a Matlab Toolbox called BN/PBN, which offers functions for simulation, inference and analysis of Boolean networks and PBNs [LS11].

Algebraic models

Laubenbacher and Stigler [LS04] were the first that developed first an algorithm for reverse engineering of gene regulatory networks using the rich theory from *algebra*, a subject in pure mathematics. They used the fact that functions $f_i : \mathbb{F}^n \to \mathbb{F}$, where $\mathbb{F}$ denotes a *finite field*, can be expressed as polynomial functions in n variables [LN96]. Since we are in the Boolean network section, a reasonable question is the one how this now corresponds to Boolean networks? If we set $\mathbb{F} := \{0, 1\}$, then the functions f_i will be Boolean functions. However, the results from algebra allows it to discretize the expression of genes to more than the two "on" and "off" states as used in Boolean networks. However, not all possible numbers of discretization steps are possible, because a finite set can only be a finite field, if the number of elements in it is a power of a *prime number*. This is not such an limiting constraint, since we still can discretize the data in $2, 3, 4, 5, 7, 8, 9, 11, 13, \ldots$ levels. Since the handling of polynomial functions is quite easy, the inference task can be performed quickly. Laubenbacher and Stigler applied their algorithm with time-series data to a model for *Drosophila melanogaster* and found out that this algorithm is quite sensitive to noise.

Veliz-Cuba *et al.* [VCJL10] introduced the approach by Laubenbacher and Stigler as a new model type describing gene regulatory networks. They originated the name "algebraic models". Furthermore, they showed that Boolean networks as well as Petri nets are special cases of algebraic models.

4.2.2. Petri nets

Petri nets were introduced by Petri in 1962 in his PhD thesis [Pet62]. Let us give first a (mathematically not exact) definition of Petri nets and proceed with a small example. We will orient ourselves for the introduction on Petri nets on the review paper by Murata [Mur89].

Definition 4.2.2. A *Petri net* is a directed graph together with an initial state called the *initial marking* M_0 and two kinds of nodes, *places* and *transitions*. The edges are called *arcs* and can only be either from a place to a transition or from a transition to a place. Arcs are labeled with positive integer weights. To a place a nonnegative integer of *tokens* is assigned, which is called a *marking*.

To obtain a better feeling for all the terms introduced in the previous definition, we will use the example of a chemical reaction.

Example 4.2.3. *The chemical reaction we look at is the* oxyhydrogen reaction $O_2 + 2H_2 \to 2H_2O$. *In Figure 4.3 we illustrate the terms used in the definition above. The red circles represent the three places* O_2, H_2 *and* H_2O *in our case. The rectangle in the middle represents the transition and there are arcs from the two reactants* O_2 *and* H_2 *to the*

transition and also an arc from the transition to the product H_2O of the oxyhydrogen reaction. Two arcs are labeled with the integer 2 representing the stoichiometry *of the reaction: the one from H_2 to the transition and the one from the transition to H_2O. Now let us look more closely on (a): the places H_2 and O_2 are marked each with two tokens symbolizing in each case two molecules. With this marking the transition is said to be* enabled*, since the number of the tokens is greater than or equal to the weights on the arcs to the transition. In (b) the marking of the Petri net is shown after the transition has* fired*. The firing of a transition can only occur if this transition is enabled. According to the stoichiometry of the reaction the marking after firing of the transition in (b) changes to one remaining token in O_2 and two tokens in H_2O. The transition is now* disabled*, because the required number of tokens to perform the reaction, i.e., the number of tokens is smaller than the weight on the arc connecting H_2 and the transition, in not available in H_2 anymore.*

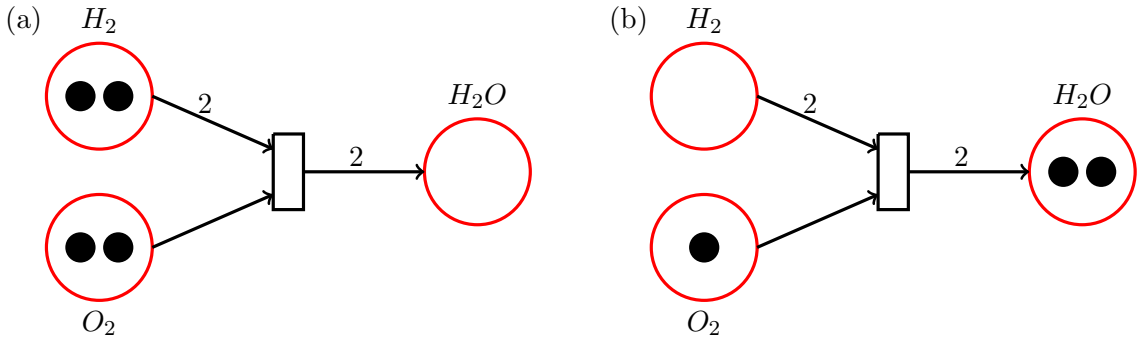


Figure 4.3.: *Petri net for the oxyhydrogen reaction. Details of the symbols are described in Example 4.2.3. (a) The marking of the Petri net before firing of the enabled transition. (b) The marking of the Petri net after firing the transition. The transition is now disabled.*

The usage and applicability for Petri nets is vast to study properties and dynamics of biological systems in general. Peleg *et al.* [PRA05] listed properties of Petri nets like *reachability*, *liveness* and *boundedness* and how they can be used to deduce biological implications from them. The concept of liveness examines if there are deadlocks in the system under considerations. Basically, in an informal way, a Petri net is *live*, if all transitions in the system can be fired independent of the initial marking. For biological systems a live Petri net implies that all processes modeled can be executed. The concept of boundedness examines the maximal number of tokens that can be in each place. A Petri net is said to be k-bounded, if the number of tokens in each place is never higher than k, $k \in \mathbb{N}$, independent of the initial marking. For biological systems the examination of boundedness can be useful in cases where substances can become toxic if present at a high concentration. The maybe most useful concept for Petri nets is the concept of reachability. It explores if a specified marking can be reached from another marking. Thus, reachability deals with the analysis of the dynamic properties of the underlying system. The so-called *reachability problem* can be solved in exponential time in the general case [Lip76]. In biological systems it can be used to determine if certain outcomes are possible for both, the modeled system in general or the perturbed system. The amount of software tools present to analyze Petri nets is also vast. For an overview of tools available look at the webpage [Wor11].

Chaouiya *et al.* [CRRT04] introduced the Petri net also for modeling of gene regulatory networks using Boolean functions. This approach was automized by Steggles *et al.* [SBSW07]

and applied to a regulatory network controlling sporulation in the bacterium *Bacillus subtilis*. It has to be pointed out that the network used was not inferred form data but modeled with literature knowledge.

Two extensions of Petri nets were also used to describe gene regulatory networks. Matuno *et al.* [MDNM00] used a *hybrid Petri net* representation of the gene regulatory network of the genetic switch mechanism of λ phage. In a hybrid Petri net places and transitions can be either discrete or continuous. The authors also introduced *delay times* for discrete transitions to reflect the time needed to transcribe a gene and the time needed for the RNA polymerase molecule to move to the right position inside the nucleus. As a second extension, Hamed *et al.* [HAP10] introduced *fuzzy Petri nets* for the modeling of gene regulatory networks. In fuzzy Petri nets tokens are associated with a real number between 0 and 1 and a transition is associated with a *certain factor* between 0 and 1. They applied their modeling approach to predict the change of expression in a target gene that is regulated by two other genes. The authors used microarray data for known gene triplets for this purpose.

As a last example for the usage of Petri nets the work by Durzinsky *et al.* [DWWM08] is presented, which provides an algorithm that is able to reconstruct gene regulatory networks from time-series data. The algorithm gives as the output all minimal networks which are consistent with the data and are encoded as Petri nets. An improvement of this algorithm which uses extended Petri nets in such a way that also inhibitory effects can be learned was proposed by the same authors in 2011 [DWM11].

4.3. Continuous models

Up to now, gene regulatory networks are looking at the nodes and treat them either as random variables as for the statistical models described in Section 4.1 or as entity that can only have discrete values as for the discrete models described in Section 4.2. In this section the nodes of the network will have continuous values and the dynamics of the nodes will be described in detail. From the biological point of view, these models are more detailed, since real experimental data appears in continuous values. We will focus here on models based on ordinary differential equations and use [KR08, PHdB09] as references to describe them.

4.3.1. Linear ordinary differential equations

Let's start to describe linear ODEs. Thus, the dynamics for a gene x_i, $i \in \{1, \ldots, n\}$, are described by

$$\frac{dx_i}{dt}(t) = A \cdot x(t) + b,$$

where A denotes a $(n \times n)$-matrix, b is an $(n \times 1)$-vector, both with constant entries, and $x = (x_1, \ldots, x_n)$. Chen *et al.* [CHC99] were the first that used linear ODEs to model GRNs. The main advantage for the usage of linear ODE models is that they are analytically solvable and no numerical integration has to be performed. This is the main reason, why linear ODE models for GRNs were used to reverse engineer GRNs from experimental data with a huge number of genes, e.g., [vSWR00, YTC02, GHL05, BGdB06, SJ06].

However, as was also discussed by Kaderali and Radde [KR08], linear ODE models do not show rich dynamic behavior, as they only have one steady state, if the matrix A is invertible. Furthermore, if the steady state is not stable the solution of the linear ODE is not bounded

which is not suitable to biological systems where the concentrations of the considered species are known to be bounded.

To use the simplicity of linear ODEs, several authors linearized their system under consideration at a specific point of interest [KKB+02, SKK04, SCJ05, SST07] and thus investigated the local behavior of the system.

4.3.2. Non-linear ordinary differential equations

To be able to capture more complex dynamics like *oscillations*, *multi-stationarity* and *biochemical switches*, nonlinear ODE models are needed. There are two types of models often used for gene regulation: *additive models* and *multiplicative models*. The multiplicative models used for the description of gene regulatory networks are called *S-systems*. Furthermore, we will give examples of publications using other non-linear ODE models. The parameter inference of models based on non-linear ODEs will be described in more detail in Chapter 5 in Section 5.2.

Additive Models

Let us start with additive models. In these models, the change of the concentration of each gene is described in the form

$$\frac{dx_i}{dt}(t) = \sum_{j=1}^{n} f_{ij}\big(x_j(t)\big) - \gamma_i x_i(t),$$

where $f_{ij} : \mathbb{R} \to \mathbb{R}$ are nonlinear *regulation functions* and $\gamma_i x_i$ denotes a first order *degradation term*, see e.g [MPO95, dJP00]. In additive models, one assumes that the regulations that influence gene i act independently. From the biological point of view this assumption is not necessarily fulfilled. For example, some proteins have to made complexes to be able to regulate a gene[2]. How are these nonlinear regulations functions now looking like? An often used regulation function for activation is the *Hill function*

$$f_{ij}\big(x_j(t)\big) = \frac{x_j^{m_{ij}}}{x_j^{m_{ij}} + \theta_{ij}^{m_{ij}}}, \tag{4.9}$$

where θ_{ij} denotes the *threshold parameter* describing the concentration of x_j needed to significantly regulate the expression of gene x_i and m_{ij} denotes the *Hill coefficient* which gives the steepness of the Hill function. This function was introduced by Hill in 1910 as a function describing the *dissociation* curve of *hemoglobin* [Hil10]. This regulation function is derived from *chemical reaction kinetics* and a detailed derivation can be found in [Rad07, KR08] and more details will be discussed in the next chapter in Section 5.1. The Hill function is used commonly as regulation function for GRNs in the literature, e.g., [QBd07, BC09, NK09, MDBA11].

To overcome the problem that the system of non-linear ODEs cannot be, in general, analytically solved, as an approximation of Hill functions, *piecewise-linear functions* were also widely used. Either step functions (here for activation)

$$f_{ij}\big(x_j(t)\big) = \begin{cases} 0, & \text{if } x_j < \theta_{ij} \\ 1, & \text{otherwise} \end{cases} \tag{4.10}$$

[2] compare to Section 1.2.3

are used, e.g., [GK73, EG00, dJGH+04, CdJG06, AJCG11], which are obtained for the case where the Hill coefficient tends to infinity. Alternatively, a continuous piecewise-linear function (here for activation) are used

$$f_{ij}\big(x_j(t)\big) = \begin{cases} 0, & \text{if } x_j(t) \leq \theta_{ij}^1 \\ \mu x_j(t) + \nu, & \text{if } \theta_{ij}^1 < x_j(t) < \theta_{ij}^2 \\ 1, & \text{if } x_j(t) \geq \theta_{ij}^2, \end{cases} \tag{4.11}$$

where two threshold parameters θ_{ij}^1 and θ_{ij}^2 and two real parameters $\mu > 0$ and $\nu < 0$ defining the straight line between the two threshold parameters have to be defined. These regulation functions were used, for example, in [PK05, GRW07].

The regulation functions were only given for the case where gene j is activating gene i. What about gene j inhibiting the expression of gene i? The answer is straightforward: the regulation functions for the inhibitory effect are defined as

$$f_{ij}^{inh}\big(x_j(t)\big) := 1 - f_{ij}\big(x_j(t)\big),$$

which holds for all functions f_{ij} defined in (4.9),(4.10) and (4.11).

S-systems

Another class of non-linear ODE models for GRNs are multiplicative models, where the regulatory terms are described by *power law functions*, i.e., the change of the expression of gene i is described by

$$\frac{dx_i}{dt}(t) = \alpha_i \prod_{j=1}^{n} x_j(t)^{g_{ij}} - \beta_i \prod_{j=1}^{n} x_j(t)^{h_{ij}}, \tag{4.12}$$

where g_{ij} and $h_{ij} \in \mathbb{R}$ are called the *kinetic orders* and α_i and $\beta_i \geq 0$ are *rate constants* [Sav70, Sav91]. The first term in (4.12) describes the activating effects and the second term the inhibiting effects. For these type of models it was demonstrated that they are able to capture dynamics present in biological systems [KTA+03]. The inference of S-systems from time-series data was predominantly done with genetic algorithms, e.g., [KTA+03, SSSZ04, SSSZ05, KIK+05, SHS06]. More advanced methods, like the inclusion of regularization terms [LWZ11] or *alternating regression* [VCV+08] have recently been introduced. It has to be mentioned that exponents in general are hard to estimate numerically [WEG08].

Other models

Perkins *et al.* [PJRG06] used partial differential equations to model the *Gap* gene network in *Drosophila melanogaster*.

Busch *et al.* [BCTR+08] used a *continuous time recurrent neural network*, which can be seen as a system of delay differential equations, to describe the GRN of *keratinocyte migration*. They inferred the model parameters with a genetic algorithm.

Äijö and Lähdesmäki [ÄL09] modeled gene regulatory networks with a general ODE model with a synthesis rate, a degradation rate and an unknown non-parametric regulation function. They used *Gaussian processes* and Bayesian learning to infer the underlying regulatory network for the IRMA network proposed by Cantone *et al.* [CMI+09] and used as the DREAM 2 Challenge #3 data.

Also a non-parametric ODE model was learned recently by Aswani *et al.* [AKB+10] from expression data with methods based on correlation.

4.4. Single-molecule models

Models looking on single-molecules are the most detailed models and thus are also the computationally most demanding ones. Having a larger number of molecules involved in the biological process under consideration, little or no stochastic effects can be observed, or rather, are averaged out, and thus, deterministic ordinary differential equations can be used. However, if the number of molecules is small, significant stochastic effects can be observed. Especially, it was observed with experiments that examined single-cells that the gene regulation processes for translation and transcription are stochastic and only a small number of needed molecules are present, e.g., [MA99, OTK+02, RPT+06, CFX06, YXR+06]. A recent review on modeling stochasticity in biological systems was published by Wilkinson [Wil09]. Furthermore, an excellent introductory book into stochastic modeling is also given by Wilkinson [Wil06].

4.4.1. Gillespie algorithm

The algorithm that is used to simulate stochasticity on the molecular level is known as *Gillespie's stochastic simulation algorithm* (SSA). The algorithm was developed by Gillespie in 1976 to simulate chemical reactions [Gil76, Gil77]. A very nice overview of the different simulation algorithms for chemical reactions is given in [Gil07], which we will also use as source to give an introduction into the SSA and its extensions.

We will start with N chemical species $\{S_1, \ldots, S_N\}$, which are involved in M chemical reactions $\{R_1, \ldots, R_M\}$ in a constant volume Ω_V. We denote by $X_i(t)$ the number of molecules of species S_i, $i \in \{1, \ldots, N\}$, at time t and want to estimate $X_i(t)$ for all species i at time t given an initial state $\mathbf{X}(t_0) = \mathbf{x}_0$ and an initial time t_0. We denote here $\mathbf{X}(t) := \big(X_1(t), \ldots, X_N(t)\big)$. Before we introduce the SSA, we have to say something about the assumptions that are made to use the SSA. The fundamental assumption is that the system under consideration is *well-stirred*. A well-stirred system can be characterized as a system where the majority of molecular collisions is non-reactive and the molecules are uniformly distributed over Ω_V.

Let us start now with the deduction of the SSA. For this purpose we introduce two quantities for every reaction R_j, $j \in \{1, \ldots, M\}$, which is first its *state-change vector* $\nu_j := (\nu_{1j}, \ldots, \nu_{Nj})$ describing the change in the number of molecules of all species caused by reaction R_j. As a small example we will have a look at the oxyhydrogen reaction again which was also used to explain Petri nets in Example 4.2.3. Setting $S_1 := O_2$, $S_2 := H_2$ and $S_3 := H_2O$ and according to the stoichiometry of the reaction we have $\nu = (-1, -2, 2)$. Thus, if the system is in state $\mathbf{X} := (2, 2, 0)$ and then the oxyhydrogen reaction occurs, the new state of the system will be $\mathbf{X} + \nu = (1, 0, 2)$.

The second quantity describing the reaction R_j is its *propensity function* a_j. The propensity function describes the probability that the reaction R_j will occur in the next infinitesimal time interval $[t, t + dt)$. Gillespie gives the propensity function for *unimolecular* and *bimolecular reactions* [Gil07]. For both cases one first needs to specify a constant c_j such that $c_j\, dt$ gives the probability that any molecules that are needed for the reaction R_j will react in the next time dt. If there are now x_1 molecules present in Ω_V, the probability that an unimolecular reaction will occur is $a_j = x_1 c_j\, dt$. Looking at the bimolecular reaction, we say that we have $x_1 \cdot x_2$ pairs of molecules in Ω. The probability that the reaction R_j occurs is then $a_j = x_1 x_2 c_j\, dt$. The constant c_j can be interpreted as a *stochastic reaction constant*.

Having not all the ingredients, we want to simulate *trajectories* of $\mathbf{X}(t)$. The "key secret" to do so is to consider these trajectories as Markov chains. Thus, we only need to consider the

Algorithm 4.1 Gillespie's stochastic simulation algorithm

Require: species $S_1, \ldots, S_N$, chemical reactions $R_1, \ldots, R_M$, initial time t_0, initial state x_0, time T to stop the simulation, propensity functions $a_j(\mathbf{x})$, $j \in \{0, \ldots, M\}$, state-change vectors ν_j for all reactions R_j

1: Start trajectory $\mathbf{X} = x_0(t_0)$
2: $t \leftarrow t_0$
3: $\mathbf{x} \leftarrow x_0$
4: **while** $t < T$ **do**
5: Sample r_1 from $\mathcal{U}(0,1)$
6: $\tau \leftarrow \frac{1}{a_0(\mathbf{x})} \ln\left(\frac{1}{r_1}\right)$
7: Sample r_2 from $\mathcal{U}(0,1)$
8: $j \leftarrow$ the smallest integer satisfying $\sum_{j'=1}^{j} a_{j'}(\mathbf{x}) > r_2 \cdot a_0(\mathbf{x})$
9: $t \leftarrow t + \tau$
10: $\mathbf{x} \leftarrow \mathbf{x} + \nu_j$
11: Append $\mathbf{x}(t)$ to trajectory $\mathbf{X}$
12: **end while**
13: **return** trajectory $\mathbf{X}$

system in the current state and need to know the probability that the next reaction will be reaction R_j and occurs in the time interval $[t+\tau, t+\tau+dt)$. We will denote this probability by $p(\tau, j \mid \mathbf{x}, t)\, d\tau$. Gillespie [Gil77] showed that it holds

$$p(\tau, j \mid \mathbf{x}, t) = a_j(\mathbf{x}) \exp\bigl(-a_0(\mathbf{x})\tau\bigr),$$

where

$$a_0(\mathbf{x}) := \sum_{j'=1}^{M} a_{j'}(\mathbf{x}).$$

We thus just need to generate samples for τ and j. Pseudocode for the generation of samples for τ and j and thus for the SSA is given in Algorithm 4.1. Software packages implementing the SSA are available for R [PK08] as well as for Matlab [Vej10].

In a single step of the SSA always only one reaction event is simulated. Since this procedure is very slow for large systems, Gillespie introduced the *tau-leaping algorithm* [Gil01]. In this algorithm one estimates the largest value for τ such that the following *leap condition* is satisfied:

> During $[t, t+\tau)$ no propensity function is likely to change its value by a significant amount.

With constant propensity functions on the interval $[t, t+\tau)$, Gillespie showed that the number of times the chemical reaction R_j fires in the time interval $[t, t+\tau)$ follows a *Poisson distribution*. Thus, in one step of the tau-leaping algorithm several reactions may be simulated and the state of the system at time point $t+\tau$ is

$$\mathbf{X}(t+\tau) := \mathbf{X}(t) + \sum_{j=1}^{M} \mathcal{P}_j\bigl(a_j(\mathbf{x})\tau\bigr)\nu_j), \tag{4.13}$$

where $\mathcal{P}_j\big(a_j(\mathbf{x})\tau\big)$ is a Poisson distributed random variable with mean $a_j(\mathbf{x})\tau$. The most efficient version for the selection of the largest τ satisfying the leap condition can be found in Cao *et al.* [CGP06].

Two further stochastic simulation algorithms were developed for stiff systems, i.e., systems with fast and slow reactions[3], the *implicit tau-leaping algorithm* [RPCG03] and the *slow-scale SSA* [CGP05].

4.4.2. Applications of the SSA

The SSA was used vastly for the description of small regulatory networks and the analysis of their dynamics, e.g., [ARM98, NNL+05, WBT+05, GG06, SJOW07]. Additionally, we will give some recent approaches to model general gene regulatory networks.

Roussel and Zhu [RZ06] developed a generalization of the SSA that is able to deal with delays arising in gene expression systems. The authors give a detailed model for transcription and apply it to a gene under the control of a *lac* promoter in *E. coli* cells. This algorithm was implemented in the *CellLine* program [RCLP07]. Ribeiro and co-workers also developed a general stochastic modeling strategy for gene regulatory networks where elementary chemical reactions are combined into one delayed chemical reaction [RZK06].

Fournier *et al.* [FGP+09] modeled self-regulated genes which are building blocks of gene regulatory networks and gave efficient algorithms to compute the invariant distribution of the trajectories obtained with the SSA.

Recently, Ribeiro [Rib10] gave a review of existing stochastic models describing gene expression and gene regulation.

4.4.3. Inference of stochastic models

There are not many inference methods for stochastic models available (see also [Wil09]). The main problem is that for stochastic models there is no obvious objective function that can be used for optimization of the model parameters, since no distance between model predictions and experimental data can be measured for varying simulated trajectories of the underlying biological system. Several non-Bayesian, e.g., [RAT06, TXGB07], and Bayesian approaches, e.g., [RRK06, BWK08], exist for this purpose. However, all these approaches are computationally expensive and only applicable to small networks.

Wilkinson and co-workers approximated the stochastic model with a *chemical Langevin equation* which ends up in estimating parameters in a nonlinear diffusion process, i.e., in a *stochastic differential equation* [GW05, HFR07]. This approach was applied to a small auto-regulatory gene network estimated from simulated data [GW08a]. However, according to Wilkinson [Wil09] these parameter estimation procedures have not been applied extensively to experimental data.

[3] compare to stiff differential equations in Section 2.4.3

CHAPTER 5

Inference of gene regulatory networks using ordinary differential equations embedded into a stochastic framework

> With four parameters I can fit an elephant, and with five I can make him wiggle his trunk.
>
> *(John von Neumann)*

Gene regulatory networks are represented as *directed graphs* $G = (V, \mathcal{E})$, with *vertices* $V = \{x_1, \ldots, x_n\}$ corresponding to genes and *directed edges* $\mathcal{E}$ corresponding to regulations between genes. An edge from gene x_i to gene x_j indicates that the product of gene x_i, i.e., mRNA or protein, influences the expression of gene x_j either by activating or by inhibiting it. The work presented here was published as [MRRK09] and the details will be described in the following sections.

5.1. Differential Equations Model for Gene Regulatory Networks

Our model for gene regulatory networks is a nonlinear ordinary differential equations model assuming that different regulators act independently. It was developed and first introduced by Radde in 2007 in her PhD thesis [Rad07]. There, the total effect on the expression of gene x_i can be written as the sum of the individual effects. For all x_i, $i \in \{1, \ldots, n\}$, the ODE model is written as

$$\frac{dx_i}{dt}(t) = s_i - \gamma_i x_i(t) + \sum_{j=1}^{n} \beta_{ij} f_{ij}\big(x_j(t)\big), \tag{5.1}$$

where $x_i(t)$ denotes the concentration of gene product x_i at time t. Furthermore, s_i and γ_i are basal *synthesis* and *degradation rates* for each gene x_i, which in the absence of regulations

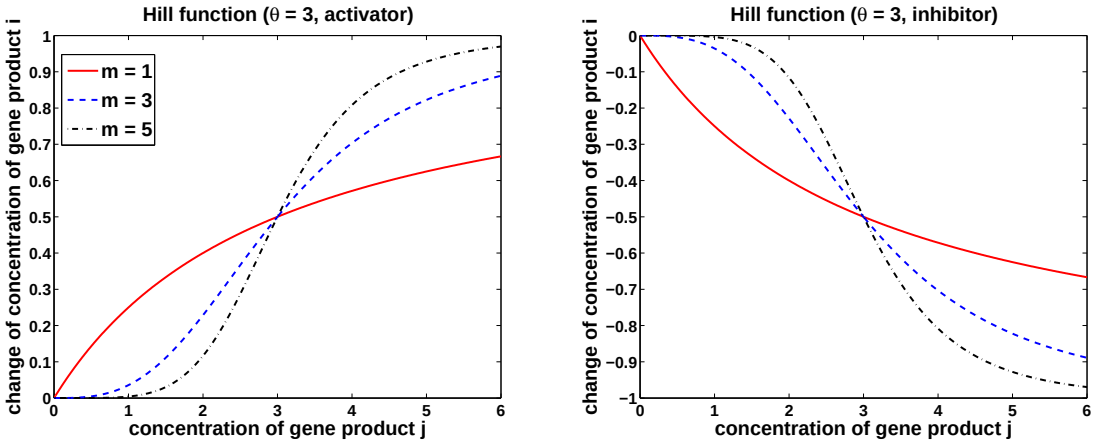


Figure 5.1.: *Hill functions f_{ij} for different Hill coefficients $m = 1, 3, 5$ as used in [Rad07]. The left plot shows an activation, the right plot an inhibitory effect. The threshold parameter θ_j was chosen equal to 3 for both plots. At this concentration of the regulating gene j, half of the maximum effect on gene i is achieved.*

from other genes determine the dynamic behavior of component x_i. The variables β_{ij}, $i, j \in \{1, \dots, n\}$, denote the regulation strengths from gene x_j on gene x_i such that $\beta_{ij} > 0$ denotes an *activation*, $\beta_{ij} < 0$ an *inhibition* and $\beta_{ij} = 0$ means that there is no regulation from gene x_j to gene x_i.

The *regulation functions* f_{ij} are *Hill functions* obtained from *chemical reaction kinetics* by considering the binding process of a transcription factor to a promoter as a *reversible chemical reaction*, see for details [JM61, YY71, Alo06, Rad07]. They are written as

$$f_{ij}\big(x_j(t)\big) = \frac{x_j(t)^{m_{ij}}}{x_j(t)^{m_{ij}} + \theta_{ij}^{m_{ij}}} \tag{5.2}$$

and depicted on the left in Figure 5.1. The parameter m_{ij} denotes the *Hill coefficients* and θ_{ij}, $i, j \in \{1, \dots, n\}$, is related to the reaction rate by describing the concentration of x_j needed to significantly activate or inhibit expression of x_i. We call it also the *threshold parameters*. The Hill coefficients describe the *cooperativity* of transcription factors, e.g., a transcription factor may only bind to the promoter, if there is already another transcription factor bound to the promoter. This leads to a Hill coefficient being greater than 1. On the other hand, the binding to the promoter may be blocked by another transcription factor leading to $m < 1$.

Radde investigated this model in detail and showed that it is able to provide interesting dynamic behavior like *multi-stationarity* and *oscillations*. One drawback of this model is, that it may provide negative values for the concentration values x_i, $i \in \{1, \dots, \}$, if the synthesis rate s_i is smaller than the sum of the negative regulations $\sum_{\beta_{ij}<0} \beta_{ij}$[1]. Since in practice, we will only have positive concentrations we will use a different regulation function for the inhibiting regulations. It was also proposed in [Alo06] as a function for the inhibiting effect. Furthermore, we want to simplify the model more by reducing the number of parameters:

1. We only consider one Hill coefficient for all the regulations. This is justified by the fact, that the dynamics of the Hill function do not significantly change for varying

[1] compare to equation (2.68) in [Rad07]

Hill coefficient, as can be seen in Figure 5.1 for the case where $m = 3$ and $m = 5$. Additionally, we noticed in the parameter estimation results, that the parameter m does not have a huge effect on the results. However, this change of the model reduced the parameter space to be inferred by $n^2 - 1$ dimensions.

2. We only consider one threshold parameter per gene. We observed in inference runs that this threshold parameters are very crucial for the learning of the regulation strengths parameters β_{ij}. However, one threshold parameter per interaction makes the inference procedure very unstable. And since the range of the data we use is quite small, we see it to be a reasonable justification to simplify the model where we only consider one threshold parameter per gene. Thus, we reduce the parameter space to be inferred by $n^2 - n$ dimensions.

With these changes, we use in the following the regulation functions

$$f_{ij}\big(x_j(t)\big) = \begin{cases} \frac{x_j(t)^m}{x_j(t)^m + \theta_j^m} & \text{for } \beta_{ij} > 0 \\ 1 - \frac{x_j(t)^m}{x_j(t)^m + \theta_j^m} & \text{for } \beta_{ij} < 0, \end{cases} \tag{5.3}$$

which are depicted in Figure 5.2. With this notation, we have to mention, that we have to change the ODE model in (5.1) slightly by taking the absolute values of the interaction strengths parameters β_{ij}, i.e., our ODE model is

$$\frac{dx_i}{dt}(t) = s_i - \gamma_i x_i(t) + \sum_{j=1}^{n} |\beta_{ij}| f_{ij}\big(x_j(t)\big) \tag{5.4}$$

with the regulation functions f_{ij} as in (5.3). Otherwise, we would also have to deal with the problem of obtaining negative concentration values. This proposed regulation function for inhibition can be interpreted in the way, that we can only inhibit something, if there is already a certain amount of a gene product present.

5.2. Parameter Estimation of ODE systems

We will describe here some general procedures how parameter are estimated in ODE systems with their benefits and drawbacks. Additionally, motivated by these facts, we provide the objective function according to which we will do our estimation procedure.

5.2.1. Based on numerical integration

The usual procedure for the estimation of parameters for differential equation models from experimental time-series data consists of the following two steps carries out iteratively:

1. solve the differential equations numerically for the time interval of interest
2. compare the predicted values from the first step with the experimental data

The comparison of the predicted values of the model with the experimental data is done by calculating an error, usually the squared difference between model prediction and data points for all data points $\tau \in \{1, \dots, T\}$ and all genes $i \in \{1, \dots, n\}$

$$\sum_{i=1}^{n} \sum_{\tau=0}^{T} \big(x_i(\tau, \omega) - d_{i\tau}\big)^2, \tag{5.5}$$

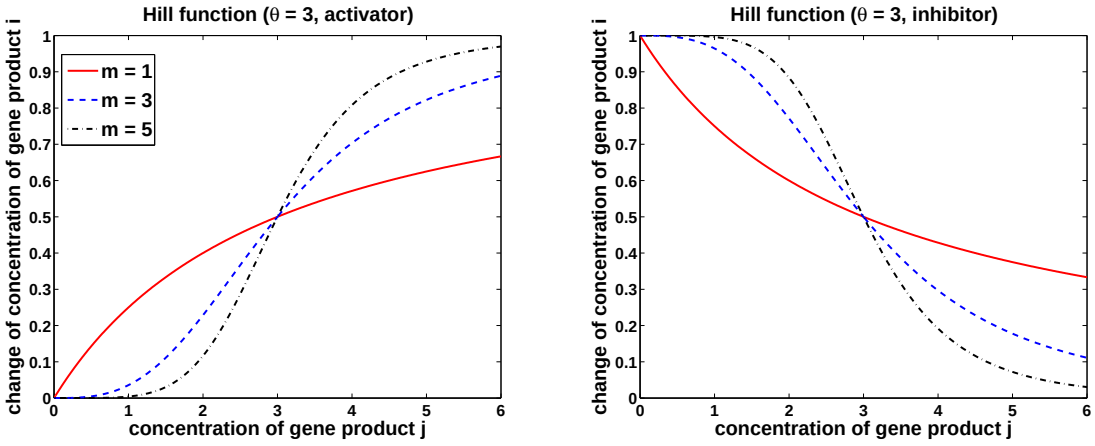


Figure 5.2.: *Hill functions f_{ij} for different Hill coefficients $m = 1, 3, 5$ as we use it. The left plot shows the behavior of gene i, if it is activated by gene j. The right plot shows the behavior of gene i being inhibited by gene j.*

where $x_i(\tau, \omega)$ denotes the value of time-series i after numerical integration of the ODE model of interest which is parameterized with the parameters ω and $d_{i\tau}$ denotes the experimental data points being used.

In every iteration of the two-step procedure above one changes the parameters ω of the model and calculates the error according to (5.5). One chooses at the end the parameter values where this error is minimal, i.e., one performs a *least squares* approach. Thus, one has to deal with an *optimization problem* where (5.5) is minimized according to the model parameters ω. There exist a huge number of different optimization algorithms which mainly differ by the way the *search space*, i.e., the parameter space of interest, is searched to get the optimal value of the parameters in an efficient way. Since we will not use optimization algorithms, we will not describe them here. For more information, see e.g. [Nel01]. The one thing, one has to keep in mind concerning optimization algorithms in relation to the work we did, is that they only provide ONE optimal parameter set at the end[2].

The drawback of this procedure for parameter estimation in ODE systems is that in every iteration one has to numerically solve the ODEs of interest. And this is a time consuming procedure, especially in the case of *stiff differential equations*[3]. And Radde [Rad07] argued that the model we use is stiff around the threshold values θ_j, $j \in \{1, \ldots, n\}$ for the change of the Hill coefficient m.

5.2.2. Based on spline interpolation

To circumvent the need to solve of the ordinary differential equations numerically, Varah proposed in 1982 the following method for parameter estimation with ODEs [Var82]:

1. fit interpolating cubic splines with varying knots to the data and calculate the derivatives of the spline at the time points of the data

[2] For *genetic algorithms* we get several parameter sets at the end, but not a distribution over parameter sets.
[3] compare to Section 2.4.3

2. compare the derivatives of the spline obtained in the first step with the model outcome derivatives (in our case it are the values (5.4) for all genes)

The *spline interpolation* will be done numerically with the methods implemented in Matlab in the *Curve Fitting Toolbox*. For details on these algorithms see [dB01].

For the comparison in the second step, Varah used the squared difference between the derivatives of the spline $z(t, D)$ obtained from the spline interpolation to the data D and the model outcome derivatives, i.e., the ODE system of interest parameterized with the parameters ω, for all data points $\tau \in \{1, \ldots, T\}$ and all genes $i \in \{1, \ldots, n\}$

$$\sum_{i=1}^{n} \sum_{\tau=0}^{T} \Big(\frac{dx_i}{dt}(t_\tau, \omega) - \dot{z}_i(t_\tau, D) \Big)^2. \tag{5.6}$$

One important point to note here, is that for the calculation of $\frac{dx_i}{dt}(t, \omega)$ the spline estimates $z_i(t, D)$ are included as values for $x_i(t)$. Having only to calculate the values for the derivatives once at the beginning we have to deal with an optimization problem, where we want to find the parameter values where the error function (5.6) is minimal.

Varah suggests to adjust the number and position of the knots of the spline adaptively and argues

> ..., that the user is a good judge of whether a given spline fit represents the data properly.

Well, this is a very crucial assumption. And because the spline interpolation is only done once at the beginning, the parameter estimates obtained for the model can only be as good as the spline fit is. In particular, for the case of noisy data, which is the normal scenario for biological data, this is not a reliable method. Daniel Ritter [Rit08] used in his master's thesis Varah's approach.

Poyton *et al.* [PVM$^+$06] developed in 2006 an improvement of Varah's method by iterating the two steps of Varah. This method is called *iterative principal differential analysis* (iPDA). The term *principal differential analysis* was first used by Ramsay in 1996 for the inference of linear differential equations [Ram96]. This name was chosen because of the analogy to the *principal component analysis*[4] techniques, where also linear algebraic-equation models are fitted using data. The work presented in [Ram96] has the same underlying idea as the work of Varah [Var82], but gives more technical details for the usage in estimating the parameters in linear differential equations.

As the word *iterative* in the iPDA method already suggests, the two steps of Varah's method are iterated, with the difference that *smoothing splines* with a *smoothing factor* λ are fitted instead of just vary the knots in cubic splines. A further important point to mention is that not the second derivative will be used as penalty term in the smoothing spline as was described in Section 2.5, but a model-based penalty is used, i.e., the model predictions are fed back into the spline estimation. As an example, the objective function for spline fitting for our ODE model is

$$\sum_{\tau=1}^{T} \big(d_{i\tau} - z_i(t_{\tau,D}) \big)^2 + \lambda \int \left(\frac{dx_i(t)}{dt} - s_i + \gamma_i x_i(t) - \sum_{j=1}^{n} |\beta_{ij}| f_{ij}\big(x_j(t)\big) \right)^2 dt \tag{5.7}$$

[4] for more details see e.g. [Jol02]

and one minimizes over the coefficients of the spline basis functions. And the objective function for parameter estimation for one gene in our ODE model would be

$$\int \Big(\frac{dx_i}{dt}(t,\omega) - \dot{z}_i(t,D) \Big)^2 \, dt,$$

which is similar to the objective function Varah used (see (5.6)). However, Poyton *et al.* only used a linear differential equation as a proof-of-principle of their methodology.

A further improvement of Varah's method has been proposed by Ramsay *et al.* in 2007 [RHCC07], where they use two nested levels of optimization. What is now meant by this? Whereas Poyton *et al.* did a joint estimation of the model parameters (in our model ω) and of the coefficients of the spline basis functions, Ramsay *et al.* just had an *outer* optimization problem of estimating the model parameters. Furthermore, for every change in the model parameters, the spline was estimated according to an objective function similar to (5.7) as an *inner* optimization problem. To clarify this a bit more, one can say that Poyton *et al.* estimate the model parameters in one step for a FIXED spline, and with the obtained optimal values for the model parameters a new spline is estimated. Ramsay *et al.*, on the other hand, just optimize the model parameters, but in every optimization step, i.e., every time the model parameter are changed, the spline is refitted again according to the objective function (5.7).

A more refined extension even exists, where the smoothing factor of normal smoothing splines[5] is also time-dependent and estimated in the estimation procedure [CR09].

The objective function we use

In our procedure we orient ourselves on the methods described above and consider an objective function for the simultaneous estimation of model parameters and the smoothing factor of the smoothing spline. We do this according to the objective function

$$\sum_{i=1}^{n} \sum_{\tau=0}^{T_1} \big(d_{i\tau} - z_i(t_\tau, D, \lambda) \big)^2 + \sum_{i=1}^{n} \sum_{\tau=0}^{T_2} \Big(\frac{dx_i}{dt}(t_\tau, \omega) - \dot{z}_i(t_\tau, D, \lambda) \Big)^2, \tag{5.8}$$

where T_1 denotes the number of time points in the experimental data and T_2 denotes the number of points to be used in the parameter fitting of the slopes. Because it is necessary to have a large number of slope estimates to describe the dynamics of a system adequately, we will have $T_2 \gg T_1$. Furthermore, we need to note that, as in (5.6), the spline values $z_i(t_\tau, D, \lambda)$ are needed to calculate the values $x_i(t_\tau)$ present in (5.4) and thus in (5.8).

5.3. Bayesian Learning Framework

Having now introduced the objective function we want to use in our estimation procedure, we additionally want to embed our estimation framework into a Bayesian setting to explore its benefits as described in the introduction of this thesis: the inclusion of prior knowledge and its ability to provide the whole distribution over model parameters. The first question to answer is now: what is meant by a "Bayesian setting"? Simply speaking, we want to have a distribution of the parameter space we are interested in. The second question that follows immediately is: how can this be achieved? Well, since the data used for the parameter

[5] "normal" in the sense, that as penalty term the second derivative is used

estimation procedure contains a lot of noise, we will just assume that this noise follows a certain distribution. Most often, it is assumed that the data is corrupted by independent mean zero Gaussian noise with variance σ^2. We will also follow this assumption to keep the model simple. Of course, other noise models could be used, but this is out of the scope of this thesis. Altogether, if we do our parameter estimation procedure of ODE systems based on numerical integration and use the objective function (5.5) where

$$x_i(\tau, \omega) = d_{i\tau} + \varepsilon_{i\tau} \quad \text{with} \quad \varepsilon_{i\tau} \sim \mathcal{N}(0, \sigma^2)$$

for all $i \in \{1, \dots, n\}$, $\tau \in \{1, \dots, T\}$, then we will obtain as likelihood for certain data D given the model parameters ω with $\dim(\omega) = d$

$$p(D \mid \omega, \sigma^2) = \prod_{i=1}^{n} \prod_{\tau=1}^{T} \frac{\mathrm{e}^{-\frac{1}{2\sigma^2}\left(d_{i\tau} - x_i(\tau,\omega)\right)^2}}{\sqrt{2\pi\sigma^2}}. \tag{5.9}$$

And we want to find the parameters, where the least squares term $\left(d_{i\tau} - x_i(\tau, \omega)\right)^2$ is minimal for all species i and all time points τ. Thus, the probability $p(D \mid \omega, \sigma^2)$ would be maximal. A common used approach to find the parameters of interest is the maximum likelihood approach as presented in Example 2.1.22 which gives one point estimate of the parameter vector ω of interest which fits the data best.

However, the likelihood (5.9) only says something about the probability of some data given the model parameters ω. We are more interested in the probability of the model parameters ω given the experimental data D. In (2.11) we saw how we can obtain the probability of interest with Bayes' formula. Let us note first, that we fix the variance σ^2 to some predefined value and do not want to estimate it. This makes the notation also simpler. Applying Bayes' formula, we thus obtain

$$p(\omega \mid D) = \frac{p(D \mid \omega) p(\omega)}{p(D)},$$

where $p(\omega)$ denotes a *prior distribution* of the model parameters and $p(D)$ denotes a *normalizing constant*, which guarantees, that integrating $p(\omega \mid D)$ over $\mathbb{R}^d$ will give the result 1, i.e., to guarantee that $p(\omega \mid D)$ is a density. We are now directly confronted with the two problems:

1. How do we know the prior distribution of the model parameters ω?
2. How do we calculate the normalizing constant?

Let us first deal with the second question. The normalizing constant $p(D)$ can also be written as

$$p(D) = \int_{\mathbb{R}^d} p(D \mid \omega) p(\omega) \, d\omega.$$

Since for the most cases, there exist no *analytical solution* for such an integral, we would need to generate independent samples, e.g., with MCMC algorithms, from the distributions $p(D \mid \omega)$ and $p(\omega)$ and then calculate the integral. This may be very time consuming. Additionally, in some optimization or sampling procedure, this would have to be done in every step. Which makes it not tractable anymore. The good point is now, that $p(D)$ is independent of the model parameters ω. In other words, one can say that the parameters ω

are integrated out (compare to Remark 2.1.17). And thus, we can simply neglect this term in the optimization/sampling procedure.

Let us come back to the first question now. At the first glance it seems to be a drawback of the Bayesian setting, that one has to specify some prior distributions for the model parameters. However, in practice one usually knows already something about the model parameters, i.e., one has some *prior knowledge* about the model. As in our model, we can say that the synthesis rates and the degradation rates are always positive. Even in the case, where no prior knowledge is available, one can always assume the uniform distribution for the parameter space. Thus, we are able to apply the Bayesian setting even in the case where no knowledge about the model is available.

The inclusion of prior knowledge instead has a lot of advantages. Firstly, doing so, we force the parameter estimation procedure to produce plausible results. In our case for the model of a GRN, we want to obtain model parameters, which are biologically plausible and are consistent with the literature. Secondly, from the algorithmic point of view the search space for the optimization/sampling procedure is reduced. And thus the algorithm is able to search the parameter space more efficiently.

The Bayesian setting how we use it

In the previous section we introduced the method that we will use to estimate the parameters in the ODE model (5.4) describing the behavior of gene regulatory networks. We argued that we will use the method based on spline interpolation, which led us to the objective function (5.8). To embed this now into a Bayesian setting we first assume that the experimental data is corrupted by independent mean zero Gaussian noise with variance σ_1^2. Furthermore, we assume that the differences between slope estimates and model predictions to be normally distributed with mean zero and variance σ_2^2. We noted in the previous section, that we need much more slope estimates than experimental data time points to describe the dynamic behavior of the system properly. To weight the two terms in (5.8) properly, the ratio between σ_1^2 and σ_2^2 can be used. With these assumptions we obtain as likelihood

$$p(D \mid \omega, \lambda, \sigma_1^2, \sigma_2^2) = \prod_{i=1}^{n} \prod_{\tau=1}^{T_1} \frac{\mathrm{e}^{-\frac{1}{2\sigma_1^2}\left(d_{i\tau} - z_i(t_\tau, D, \lambda)\right)^2}}{\sqrt{2\pi\sigma_1^2}} \times \prod_{i=1}^{n} \prod_{\tau=1}^{T_2} \frac{\mathrm{e}^{-\frac{1}{2\sigma_2^2}\left(\frac{dx_i}{dt}(t_\tau, \omega) - \dot{z}_i(t_\tau, D, \lambda)\right)^2}}{\sqrt{2\pi\sigma_2^2}},$$

which is equivalent to the objective function (5.8) up to log-transformation and scaling. By using Bayes' formula we now obtain for the probabilities of interest

$$p(\omega, \lambda, \sigma_1^2, \sigma_2^2 \mid D) = \frac{p(D \mid \omega, \lambda, \sigma_1^2, \sigma_2^2) p(\omega, \lambda, \sigma_1^2, \sigma_2^2)}{p(D)}. \tag{5.10}$$

5.4. Inclusion of Prior Knowledge

We argued in the previous section, that the prior distribution can be used to incorporate available knowledge about the underlying system and the model parameters. We also saw which advantages the inclusion of prior knowledge has.

Which prior knowledge do we want to incorporate in the estimation procedure? How does the distribution of $p(\omega, \lambda, \sigma_1^2, \sigma_2^2)$ look like? First of all, we do not want to estimate the noise variances σ_1^2 and σ_2^2, but set them in advance to weight the two terms in (5.8) equally. For

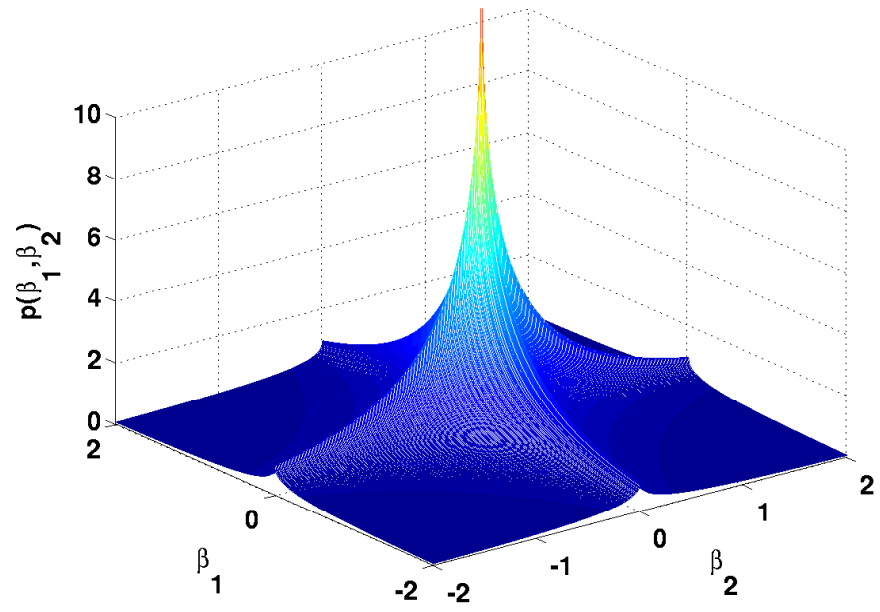


Figure 5.3.: *Plot for the two-dimensional* L_q*-prior* $p(\beta_1, \beta_2) := L_q(\beta_1; q, s) \cdot L_q(\beta_2; q, s)$ *for* $q = 0.5$ *and* $s = 1$*. One can clearly see that the prior favors points* (β_1, β_2) *where one of the components is close to zero over points where both components are far away from zero.*

this, we set $\sigma_2^2 = 1$ and $\sigma_1^2 = T_1/T_2$. As a next point we notice that we will use independent priors for all parameters. Thus, for our ODE model, the prior distribution will be

$$p(\omega, \lambda) = p(s_1) \cdots p(s_n) p(\gamma_1) \cdots p(\gamma_n) p(\beta_{11}) \cdots p(\beta_{nn}) p(k_1) \cdots p(k_n) p(m) p(\lambda). \tag{5.11}$$

Thus, we have to give formulas for all the factors on the right side of equation (5.11). Since the synthesis rates s, the degradation rates γ, the threshold parameters k and the Hill coefficient m have to be larger than zero, we use for them as prior distribution the gamma distribution $\text{Gamma}(r, a)$ as introduced in Example 2.1.8. The gamma distribution is a very flexible distribution for varying shape parameter r and rate parameter a.

Since we use for the calculation of the smoothing spline the Matlab function `csaps`[6], the smoothing factor λ varies only between zero and one. Thus, a suitable prior distribution is the beta distribution $\text{Beta}(\alpha, \beta)$ as introduced in Example 2.1.8.

Furthermore, we want to incorporate as prior knowledge the sparsity of gene regulatory networks [AD97] into the learning framework. For this purpose, we use a prior based on the L_q-norm [LZPA07] for the interaction strengths parameters β_{ij}, $i, j \in \{1, \ldots, n\}$:

$$L_q(\beta; q, s) = \mathcal{N}(q, s) \exp\left(-\frac{1}{q s^q} |\beta|^q\right), \tag{5.12}$$

[6] the smoothing parameter is denoted by p in the documentation of this function in Matlab; for the value of the function $\lambda(t)$ in the documentation the default constant function 1 is used

for $\beta \in \mathbb{R}$ and $q, s > 0$, where $\mathcal{N}(q, s)$ denotes the normalizing factor

$$\mathcal{N}(q, s) = \frac{q^{(q-1)/q}}{qs\Gamma(1/q)}.$$

For $q = 2$, equation (5.12) is a normal distribution, for $q = 1$ it corresponds to the Laplace distribution. Values of $q < 1$ enforce stronger sparsity constraints. In Figure 5.3 a two-dimensional L_q-distribution is depicted with $q = 0.5$ and $s = 1$. One clearly sees how points where at least one component is close to zero have a higher probability compared to points where both components are different from zero. Another prior used for integration of sparsity was introduced by Kaderali *et al.* in 2006 [KZF$^+$06] to predict survival times for cancer from gene expression data. Using this prior would mean to assume a multi-dimensional mean-zero normal distribution with variances Var_{ij} on the regulation strengths parameters $(\beta_{11}, \ldots, \beta_{nn})$ and use as prior distribution for each variance Var_{ij} a gamma distribution. Then the prior distribution for each regulation strengths parameter β_{ij} will be obtained by integrating out the variances Var_{ij}

$$p(\beta_{ij}) = \int p(\beta_{ij} \mid \mathrm{Var}_{ij}) p(\mathrm{Var}_{ij})\, d\,\mathrm{Var}_{ij}\,. \tag{5.13}$$

This sparsity prior was used by Radde and Kaderali [RK07] to infer the gene network structure from the ODE model (5.1) with the regulation functions (5.2) with a maximum likelihood approach. The L_q-prior we use here avoids the numerical integration (see (5.13)) and leads to similar sparsity constraints.

5.5. MCMC sampling from the posterior

Having now all ingredients for the Bayesian framework together, we will generate samples from the desired distribution $p(\omega, \lambda \mid D)$ using Markov chain Monte Carlo methods. For this purpose, we use an iterative approach. Firstly, the model parameters ω are sampled with the Hybrid Monte Carlo algorithm (see Section 3.1.3) with fixed smoothing factor λ. Secondly, the smoothing factor λ is sampled with the Metropolis-Hastings algorithm (see Section 3.1.2) with fixed model parameters ω. Pseudocode for this iterative procedure is given in Algorithm 5.1.

5.6. Evaluation of Reconstructed Networks

For evaluation of the results we only consider the mean value of the obtained Markov chain for each model parameter after neglecting points from the burn-in phase. This of course neglects completely the whole distribution over model parameters that we sampled from. We do this simplification to be able to evaluate the obtained results in an automated and quantitative way.

For quantitative evaluation we use receiver operating characteristics. We consider here the three class problem where we have the three classes of no edge, activating edge and inhibiting edge. We will calculate the AUC of the sensitivity vs. $1-$specificity ROC graph $\mathrm{AUC}_{\mathrm{ROC}}$ and the precision vs. recall ROC graph $\mathrm{AUC}_{\mathrm{P2R}}$. For more details on general receiver operating characteristics see Section 3.3 and for our analysis with the three-class problem see Subsection 3.3.2.

Algorithm 5.1 Iterative Hybrid Monte Carlo and Metropolis Hastings algorithm

Require: desired distribution $p(\cdot)$, starting value (ω^0, λ^0), proposal distribution $q_\lambda(\cdot \mid \lambda^{(t)})$, number of leapfrog steps for HMC L, proposal distribution for step size ϵ of leapfrog steps $q_\epsilon(\cdot)$, standard deviation σ_ρ for the sampling of the momentum variables ρ, number of Markov chain samples T

$t \leftarrow 0$
while $t < T$ **do**
  Sample $\bar{\epsilon}$ from $q_\epsilon(\cdot)$
  Sample $\rho_i^{(t)}$ from $\mathcal{N}(0, \sigma_\rho^2)$ for all $i \in \{1, \ldots, n\}$
  Perform L leapfrog steps with step size $\bar{\epsilon}$ starting at state $(\omega^{(t)}, \rho^{(t)})$
  Store resulting candidate state in $(\hat{\omega}, \hat{\rho})$
  Sample u_1 from $\mathcal{U}(0, 1)$
  $\alpha_1 \leftarrow \min\left\{1, \exp\left(H(\omega^{(t)}, \rho^{(t)}) - H(\hat{\omega}, \hat{\rho})\right)\right\}$
  if $u_1 < \alpha_1$ **then**
    $\omega^{(t+1)} \leftarrow \hat{\omega}$
  else
    $\omega^{(t+1)} \leftarrow \omega^{(t)}$
  end if
  Sample $\bar{\lambda}$ from $q_\lambda(\cdot \mid \lambda^{(t)})$
  Sample u_2 from $\mathcal{U}(0, 1)$
  $\alpha_2 \leftarrow \min\left\{1, \frac{p(\bar{\lambda}|\omega^{(t+1)}) q_\lambda(\lambda^{(t)}|\bar{\lambda})}{p(\lambda^{(t)}|\omega^{(t+1)}) q_\lambda(\bar{\lambda}|\lambda^{(t)})}\right\}$
  if $u_2 < \alpha_2$ **then**
    $\lambda^{(t+1)} \leftarrow \bar{\lambda}$
  else
    $\lambda^{(t+1)} \leftarrow \lambda^{(t)}$
  end if
  Append $\left(\omega^{(t+1)}, \lambda^{(t+1)}\right)$ to Markov chain $\left(\omega^{(k)}, \lambda^{(k)}\right)_{k=0}^{t}$
  $t \leftarrow t + 1$
end while
return Markov chain $\left(\omega^{(k)}, \lambda^{(k)}\right)_{k=0}^{T}$

5.7. Implementation

We implemented our algorithm in Matlab, Release 2008b (The Mathworks), using the *Statistics Toolbox* and the *Spline Toolbox*. The Spline Toolbox was recently included into the Curve Fitting Toolbox. As mentioned before, we use for the generation of the smoothing spline the function `csaps` and the derivative of the obtained spline was calculated with the function `fnder`. To obtain the values of the spline and its derivative at specific time points we use the function `fnval`. We need the Statistics Toolbox to generate random numbers from the gamma distribution with the function `gamrnd`. We need these to propose the step size for the leapfrog steps of the Hybrid Monte Carlo algorithm and the standard deviation of the normal distribution which is used to propose new samples in the Metropolis-Hastings algorithm.

The calculations were done on a Linux *cluster* with dual-processor 3.1 GHz XEON quad-core machines with 32 GB RAM, running each Markov chain in a single thread without parallelization of the code.

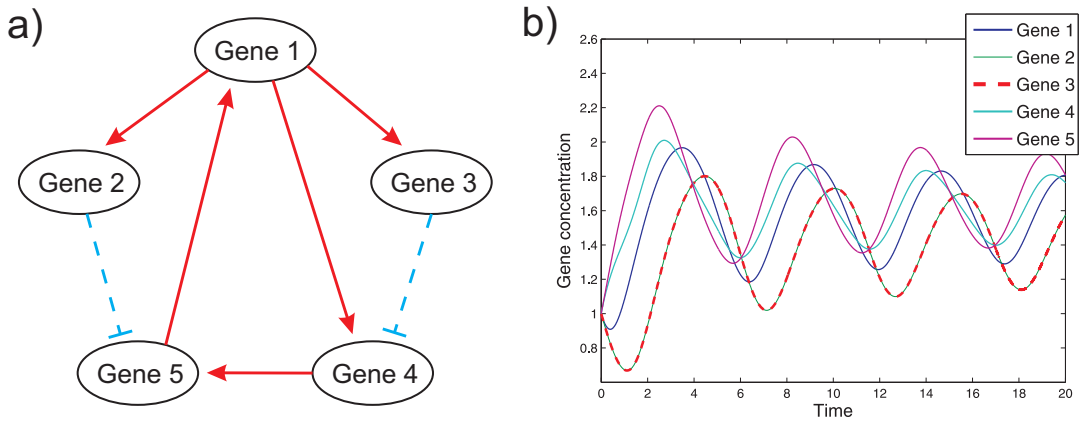


Figure 5.4.: *(a) True network of the DREAM 2 Challenge #3 five gene time-series data, showing the bio-engineered interactions between the five genes artificially inserted into yeast. (b) Time course of simulation with model in arbitrary time and concentration units, for the simulated five gene model. Different numbers of equidistant time points from this data were used for network reconstruction in the simulation study. The time courses of gene 2 and gene 3 are the same.*

5.8. Results

5.8.1. Simulated Data: Five Gene Network

To evaluate our approach we first generated simulated data, where we know which values the parameters of the ODE system have and how the golden standard network looks like. As golden standard network we used the true network structure of the DREAM 2 Challenge #3. It is shown in Figure 5.4 (a). The AUC values for random guessing for this network structure are calculated with the formulas provided in Section A.4 and they are $\mathrm{AUC}_{\mathrm{P2R}} = 0.14$ and $\mathrm{AUC}_{\mathrm{ROC}} = 0.358$. According to this network structure, the regulation strengths parameters were set to $\beta_{ij} = 2$ for activations, i.e., $\beta_{21} = \beta_{31} = \beta_{41} = \beta_{54} = \beta_{15} = 2$. For inhibitions, the regulations strengths parameters were set to $\beta_{ij} = -2$, i.e., $\beta_{52} = \beta_{43} = -2$. For all other indices of the matrix $\beta = (\beta_{ij})_{i,j=1}^{n}$ the values β_{ij} are zero. We adjusted the other parameters of the ODE system (5.4) to obtain oscillations as shown in Figure 5.4 (b). We point out here that learning network structure from oscillating data is a hard problem, because models tend to learn a steady state since the range of parameters providing oscillations is usually very small [RK09]. To obtain this behavior, we used for the synthesis rates the values $s = (0.2, 0.2, 0.2, 0.2, 0.2)$ and for the degradation rates the values $\gamma = (0.9, 0.9, 0.9, 1.5, 1.5)$ for the five genes. The threshold parameters of the Hill functions were set to $\theta = (1.5, 1.5, 1.5, 1.5, 1.5)$ with Hill coefficient $m = 5$.

The data were simulated by numerical integration with the function `ode45` in Matlab with the initial value $(1, 1, 1, 1, 1)$. To show the robustness of our method, we simulated data with different number of time points and different levels of noise. We used equidistant time points in the range between 0 and 20 [a.u.] (compare to Figure 5.4 (b)) and added mean-zero normally distributed noise with different standard deviations to the concentration values. Then we used our method as described in the previous sections and used Algorithm 5.1 so sample 110,000 samples from the distribution $p(\omega, \lambda \mid D)$. We used a burn-in phase of 10,000 steps to guarantee, that our Markov chain reached the desired distribution as stationary

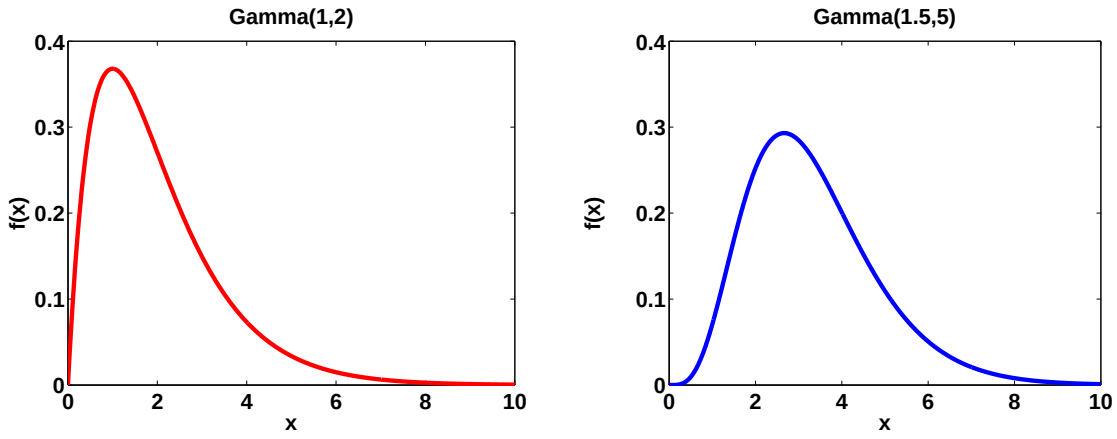


Figure 5.5.: *The prior distributions used for our Bayesian parameter estimation framework. On the left the prior for the synthesis and degradation rates is depicted. On the right the prior for the Hill coefficient m is shown.*

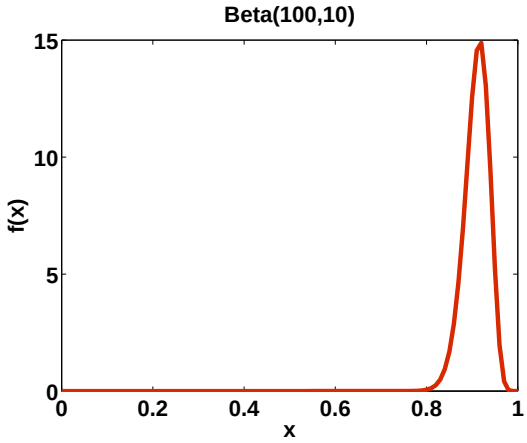


Figure 5.6.: *The density function of the beta distribution as was used as prior distribution for our Bayesian parameter estimation framework for the smoothing factor λ for the simulated data.*

distribution.

The parameters for the prior distribution were set as follows. For the gamma prior on the synthesis and degradation rates we used $a = 1$ and $r = 2$. The density function of this prior is depicted in Figure 5.5 on the left side. The gamma prior on the Hill coefficient m used the parameters $a = 1.5$ and $r = 5$. The density function for this gamma distribution is shown in Figure 5.5 on the right side. For the beta prior Beta(α, β) on the smoothing factor λ we used $\alpha = 100$ and $\beta = 10$. The corresponding density function is shown in Figure 5.6. To enforce sparsity in the learned network structure we used for the L_q prior the parameters $q = 0.5$ and $s = 1$ (compare to Figure 5.3). The gamma priors on the threshold parameters θ_j for each gene j are set in such a way that the mean and the variance of the priors corresponds to the mean and the variance of the dataset used.

The number of slope estimates T_2 is set to 1000 and the corresponding variance σ_2^2 is set to 1. The variance σ_1^2 is set to T_1/T_2, where T_1 denotes the number of time points in the used

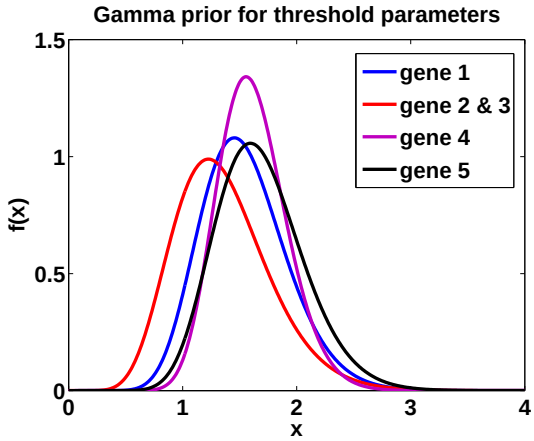


Figure 5.7.: *Shown are the density functions of the gamma priors of the threshold parameters* θ_j *for* $j \in \{1, 2, 3, 4, 5\}$ *for the simulated optimal data of* 40 *time points without noise. Since the time course data are the same for gene* 2 *and gene* 3*, the priors are also the same.*

dataset.

Results on 40 time points

As first results we will use optimal data in the sense that we have 40 time points and no noise. The prior distributions for the threshold parameters obtained with this data are shown in Figure 5.7. The mean values we obtained for the synthesis rates were $s = (0.23, 0.20, 0.29, 0.26, 0.15)$ and for the degradation rates were $\gamma = (1.17, 1.14, 1.33, 1.00, 0.99)$. For the Hill coefficient we obtained the mean value of 4.76 and the means for the threshold parameters θ_j ranged from 1.38 to 1.78. As mean smoothing factor λ we obtained 0.92. In Table 5.1 the mean values together with the standard deviations of the regulations strengths parameters β are shown. The large standard deviations of some regulations indicate that either several network structures exist that describe the data in a similar way or that the model dynamics are not sensitive to changes of this parameters. For example, the self-regulation of gene 3 may be activating, inhibiting or not present according to the large standard deviation of the learned parameter value. At this point, one can see perfectly the strength of our proposed method. By analyzing the learned distributions of the parameters one obtains a lot of additional information about the underlying ODE model and the quality of the data[7].

With the mean values from Table 5.1 we calculated AUC values for precision vs. recall curves and sensitivity vs. 1−specificity curves. We obtained the values $\text{AUC}_{\text{P2R}} = 0.516$ and $\text{AUC}_{\text{ROC}} = 0.706$. Compared to the values for random guessing we calculated above, these results show that we are much better than random guessing.

Effect of Noise and Dataset Size

After showing results for an optimal dataset, we applied our algorithm to datasets with different numbers of time points and different levels of noise. To be more precise, we used datasets

[7] by quality of the data we mean its ability to estimate the parameters of the underlying model in a satisfactory manner

To ↓ / From →	Gene 1	Gene 2	Gene 3	Gene 4	Gene 5
Gene 1	0.53 ± 0.72	-0.22 ± 0.41	-0.10 ± 0.43	1.37 ± 1.16	$\mathbf{0.88 \pm 0.74}$
Gene 2	$\mathbf{1.54 \pm 0.78}$	0.28 ± 0.42	0.49 ± 0.62	0.33 ± 0.55	0.28 ± 0.53
Gene 3	$\mathbf{1.35 \pm 0.80}$	0.34 ± 0.57	1.07 ± 1.59	0.55 ± 0.63	0.26 ± 0.46
Gene 4	$\mathbf{0.23 \pm 0.57}$	-0.35 ± 0.61	$\mathbf{-0.51 \pm 0.64}$	0.40 ± 0.69	0.84 ± 0.74
Gene 5	-0.01 ± 0.31	$\mathbf{-0.62 \pm 0.67}$	-0.91 ± 0.69	$\mathbf{0.55 \pm 0.81}$	0.65 ± 0.63

Table 5.1.: *Learned regulation strength parameters β for the simulated dataset with 40 time points. Given are mean and the standard deviation of the sampled interaction parameters. True edges which are present in the reference network are indicated in bold.*

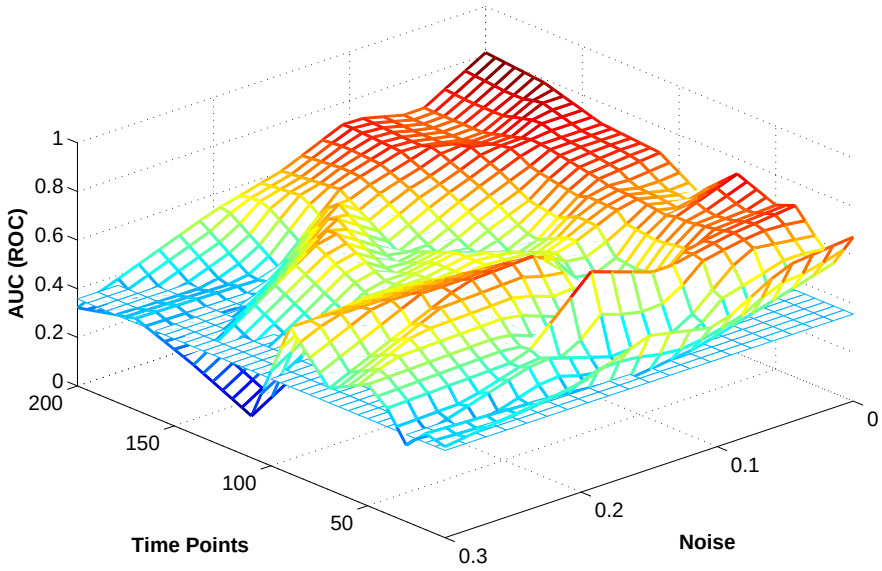


Figure 5.8.: AUC *values for different noise levels and different numbers of time points used for network reconstruction. The standard deviation of the noise was varied from $\sigma = 0$ to $\sigma = 0.3$, the number of time points from $T_1 = 10$ to $T_1 = 200$. The plots show* AUC *values under the* ROC *curve. The blue surface indicates the* $\mathrm{AUC}_{\mathrm{ROC}}$ *values that would follow for random guessing.*

with $T_1 = 10, 20, 30, 40, 50, 70, 90, 110, 140, 170$ and 200 time points. To all of the eleven different time points datasets we added mean zero Gaussian noise with standard deviations $\sigma = 0.05, 0.1, 0.15, 0.2$ and 0.3. The purpose of this procedure is to show the reliability of our method and also the limitations of our approach.

In Figure 5.8 the AUC values for all the 55 datasets obtained with our inference method are depicted for the sensitivity vs. 1−specificity ROC analysis. In Figure 5.9 the AUC values for the same datasets are shown for the precision vs. recall ROC analysis. The blue surface in both graphs shows the values that will be obtained for random guessing. We see the expected behavior of decreasing AUC values for decreasing number of time points and increasing levels of noise. Important to note here is the fact that even for low numbers of time points, the obtained AUC values are reasonably high. For the highest level of noise, our method does perform as good as random guessing. However, the *amplitude* of the oscillations present in the simulated data is between 0.4 and 1. Thus, noise with standard deviation of 0.3 will, at

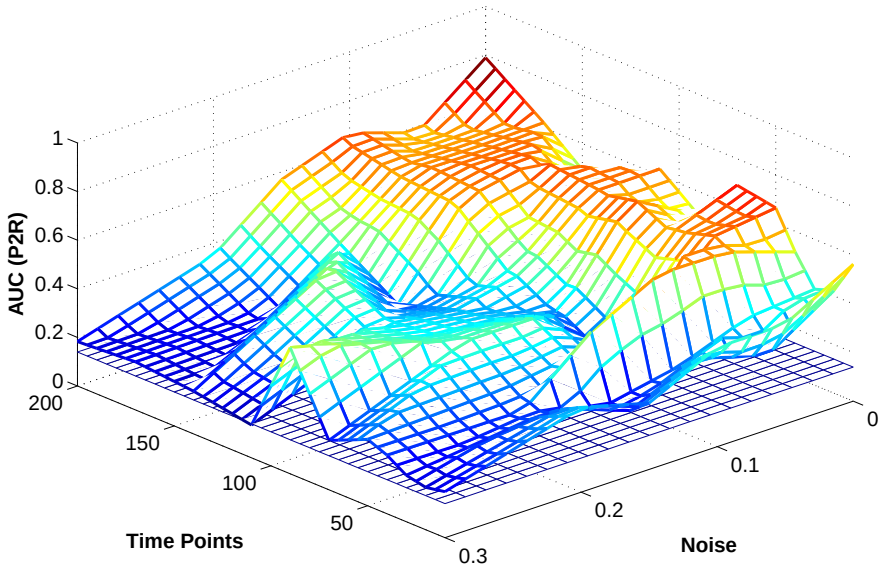


Figure 5.9.: AUC *values for different noise levels and different numbers of time points used for network reconstruction. The standard deviation of the noise was varied from* $\sigma = 0$ *to* $\sigma = 0.3$*, the number of time points from* $T_1 = 10$ *to* $T_1 = 200$*. The plots show* AUC *values for* PR *curves for varying* T *and* σ*. The blue surface indicates the* $\mathrm{AUC}_{\mathrm{P2R}}$ *values that would follow for random guessing.*

least partially, destroy the oscillating behavior of the data. And since estimating parameters from oscillating data is a challenging task as was mentioned before, our method performs well in general.

5.8.2. Experimental Data: The DREAM 2 Challenge #3 Dataset

To evaluate our approach on real biological data, we used a dataset of the DREAM initiative. The DREAM initiative was fathered by Stolovitzky *et al.* [SMC07] to understand the limitations of reverse engineering methods for the inference of cellular networks from high-throughput data. Furthermore, the aim is to see which methods are able to perform this inference task and to which extent concerning reliability and biological predictability. The acronym DREAM stands for *Dialogue on Reverse Engineering Assessment and Methods*. Every year, the DREAM initiative provides datasets for different tasks for network reconstruction and a *double-blind* procedure is performed, where researchers/scientists/engineers/... apply their methods on these datasets without knowing the underlying golden standard network. Thus, the applicant is "blind". Furthermore, the persons of the DREAM initiative do only know the alias of the applicants and not their real identities. Thus, the evaluators are also "blind". In the last year, already the sixth round of posted DREAM challenges was accomplished [Pro11a].

The biggest problem to evaluate reverse engineering methods for network reconstruction on experimental data is the fact that usually no golden standard network is available. In the Challenge #3 of DREAM 2, Cantone *et al.* [CMI+09] provided a dataset which was generated in-vivo, i.e., provides real biological data, and also possesses a golden standard

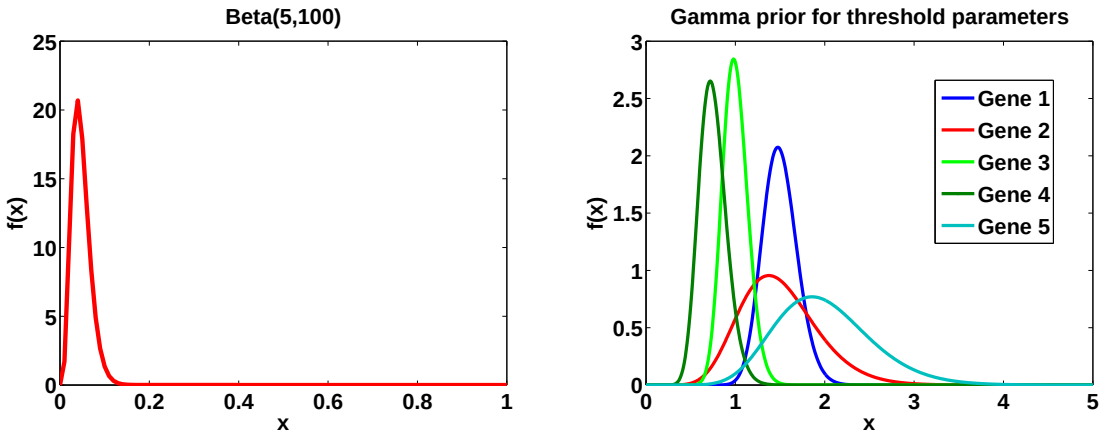


Figure 5.10.: *(Left) The density function of the beta distribution* Beta$(5, 100)$ *which was used as a prior distribution for the smoothing factor* λ *for the DREAM* 2 *Challenge* #3 *data. (Right) The density function of the gamma distributions which were used as prior distributions for the threshold parameters manually set for each gene for the DREAM* 2 *Challenge* #3 *data.*

network structure. Nice, but how was this achieved? The authors bio-engineered a five-gene network with the underlying topology as in Figure 5.4 (a). For this purpose they inserted new promoter/gene combinations into the chromosomal DNA of *budding yeast*. This dataset consists of two time-series. After stimulation, the gene expression was measured for the five involved genes using qPCR. The first time-series consists of 15 time points measured in 3 minutes intervals and the second time-series consists of 11 time points measured in 5 minutes intervals.

The datasets consists of negative log-ratios to the base 2 of the genes of interest to housekeeping genes. We transformed the data to get the original ratios. We applied our inference approach to the first time series (3 minute interval data), i.e., we have $T_1 = 15$.

To obtain samples from the desired distribution, we ran a Markov chain with the Algorithm 5.1 and performed $60,000$ samples with a burn-in phase of $10,000$ samples. The hyperparameters for the gamma prior for the synthesis and degradation rates were the same as were used for the simulated data. The same holds true for the hyperparameters for the gamma prior for the Hill coefficient and for the parameters T_2, σ_1^2 and σ_2^2. The hyperparameters of the beta prior for the smoothing factor were set to $\alpha = 5$ and $\beta = 100$. The density function of the beta distribution is shown on the left in Figure 5.10. The hyperparameters for the gamma priors on the threshold parameters were set manually for each gene individually. This was done in the way that the probability mass of the density functions is the mean value of the concentration values of every time-series for each gene. The density functions for the gamma priors for all five genes are depicted on the right in Figure 5.10. The hyperparameters for the L_q-prior were set to $q = 1$ and $s = 2$. The density function of this prior is shown in Figure 5.11.

We performed an evaluation on the mean of the obtained Markov chain for each parameter, as was done for the simulated data. The resulting AUC values are $\text{AUC}_{\text{ROC}} = 0.199$ and $\text{AUC}_{\text{P2R}} = 0.15$, which are both not better than random guessing. This may have several reasons. One may be that the level of noise in the present data is too high and thus the reverse engineering procedure is not able to reconstruct the correct network. This point will be later discussed in more detail. Another reason can be that the posterior distribution contains

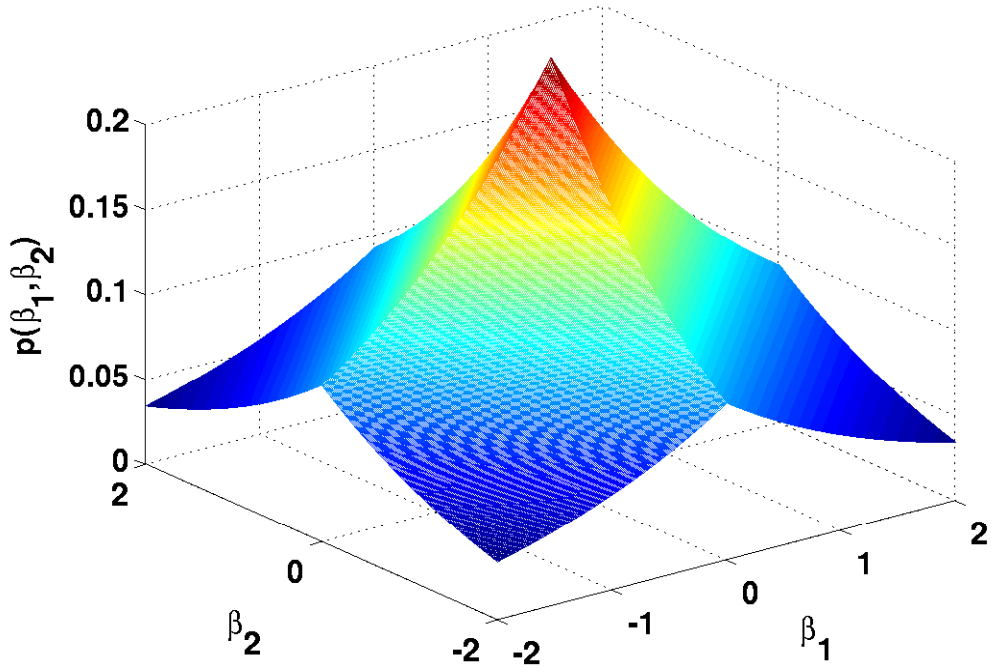


Figure 5.11.: *Plot for the two-dimensional* $\mathrm{L_q}$*-prior* $p(\beta_1, \beta_2) := \mathrm{L_q}(\beta_1; q, s) \cdot \mathrm{L_q}(\beta_2; q, s)$ *for* $q = 1$ *and* $s = 2$.

multiple modes and the calculated mean being not an appropriate summary statistic. We therefore searched the sampled points with a low value for the objective function and high AUC values. To be more precise, we search for the sample where the values for the objective function are smaller than $(178, 135)$, where we have to note that our iterative sampling algorithm 5.1 has two objective functions, one for the Hybrid Monte Carlo algorithm part and one for the Metropolis-Hastings algorithm part.

The regulation strengths parameters for this sample are shown in Table 5.2 and the AUC values for this sample are $\mathrm{AUC_{ROC}} = 0.532$ and $\mathrm{AUC_{P2R}} = 0.2554$. The dynamics for all five genes with this sample are shown in Figure 5.12. The blue dots represent the experimental data, the blue lines show the smoothing spline fitted to them and the red lines show the dynamics obtained with the used sample. The general dynamics of the data are well represented

To ↓ / From →	**Gene 1**	**Gene 2**	**Gene 3**	**Gene 4**	**Gene 5**
Gene 1	1.03	0.03	−0.05	−0.62	**0.12**
Gene 2	**1.09**	1.55	−0.09	0.00	−0.10
Gene 3	**0.10**	−0.04	0.56	0.16	−0.11
Gene 4	**−0.15**	−0.03	**−0.43**	0.26	0.09
Gene 5	0.77	**−0.16**	0.01	**0.01**	0.44

Table 5.2.: *Reconstructed maximum-a-posteriori regulation strengths parameters* β *for the DREAM* 2 *Challenge #3 data. True edges present in the reference topology are marked in bold.*

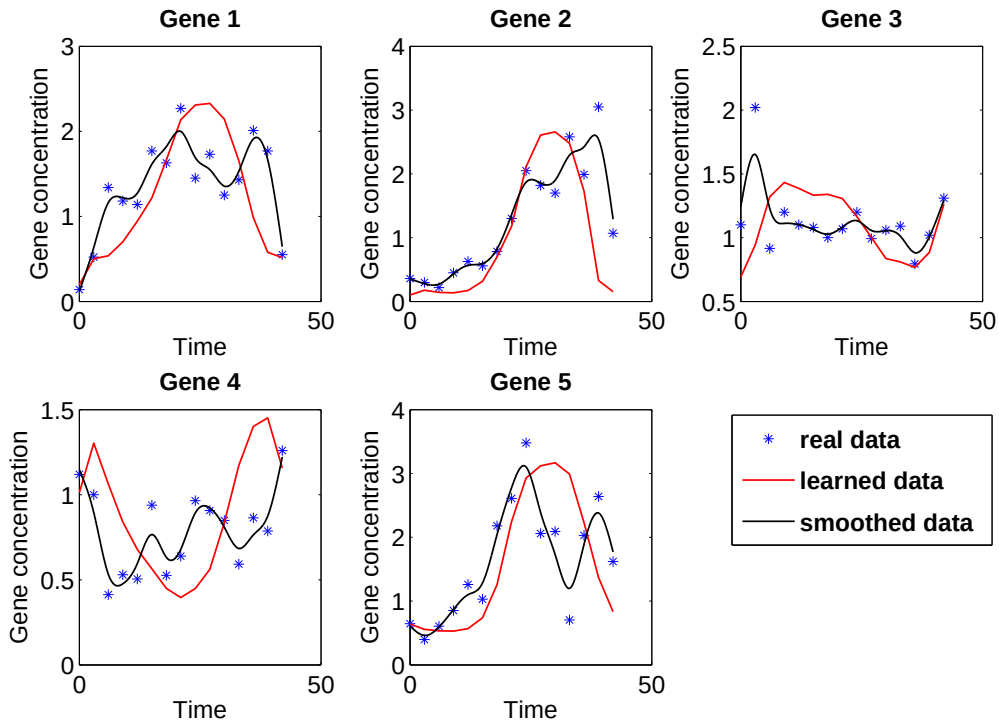


Figure 5.12.: *Plot of the experimental data from the DREAM 2 challenge, in comparison to time courses simulated with reconstructed model parameters. Shown in black is the smoothed data. Additionally, the function* `fminsearch` *in Matlab was used with a least squares objective function between the interpolated data and the results from numerical integration with* `ode45` *to find the optimum starting point.*

with a moderate amount of smoothing.

To compare our results with the results from other methods, we computed performance measures as were used in the DREAM 2 challenge for our inferred network parameters β as shown in Table 5.2. With these values we are able to compare our performance to the performance of other methods. In Table 5.3 we show the AUC values of our method (second column) and the best submitted results (first column). It remains now to explain how the AUC values were calculated. We used for this purpose the scoring methods as described in [SMC07]. Basically, they use a ROC analysis for a two-class problem (compare to Section 3.3.1) with present and absent edges. However, our method gives us activating and inhibiting edges as results for the network structure. Thus, we need to modify our results to be suitable for the evaluation methods of the DREAM 2 challenge. This was done as follows:

- we skipped the sign of the learned regulation strengths parameters β and divided by the largest regulation strength for the DIRECTED-UNSIGNED challenge
- for the two DIRECTED-SIGNED challenges we only took the regulation strengths parameters with the appropriate sign, skipped the sign and divided them by the highest absolute regulation strength

Our method outperforms all submitted results in the challenge DIRECTED-SIGNED-INHIBITORY. We observed as one difficulty in our estimation procedure, that our approach learned

Challenge (DIRECTED-)	Best submitted	Our method	No self-regulations
SIGNED-EXCITATORY	AUC = 0.79 AUC_{PR} = 0.72	AUC = 0.61 AUC_{PR} = 0.25	AUC = 0.79 AUC_{PR} = 0.54
SIGNED-INHIBITORY	AUC = 0.63 AUC_{PR} = 0.14	AUC = 0.96 AUC_{PR} = 0.45	AUC = 0.96 AUC_{PR} = 0.45
UNSIGNED	AUC = 0.73 AUC_{PR} = 0.55	AUC = 0.56 AUC_{PR} = 0.30	AUC = 0.79 AUC_{PR} = 0.57

Table 5.3.: *Results of the DREAM* 2 *Challenge #*3 *data of our method (second column) compared to submitted best results from [Pro11b] (first column). The third column gives the* AUC *values for our method when self-regulations are omitted. Our method clearly outperforms all submitted methods in the DIRECTED-SIGNED-INHIBITORY challenge; furthermore, when self-regulations are neglected, we also beat the best submitted method in the DIRECTED-UNSIGNED challenge.*

a lot of strong self-regulations of genes, possibly because of an improper balancing of the priors on synthesis/degradation rates and regulation strengths. Since there are no self-regulations in the DREAM 2 Challenge #3 data, we provide an additional evaluation when disregarding self-regulations. These results are shown in Table 5.3 in the third column. Doing so, we not only outperform the best submitted results for the DIRECTED-SIGNED-INHIBITORY challenge, but also beat the best methods in the DIRECTED-UNSIGNED challenge. Furthermore, the results for the DIRECTED-SIGNED-EXCITATORY challenge are comparable to the best submitted results.

5.9. Discussion

Relevance

We presented a new methodological procedure for the inference of gene regulatory networks from gene expression time-series data and applied it to simulated data and to experimental data generated for a synthetically designed 5-gene regulatory network in yeast. The proposed approach combines ordinary differential equations with a Bayesian inference technique which is optimally suited for the quantitative analysis of complex biological processes. The usage of non-linear differential equations makes it possible to describe complex dynamics of biological systems and analyze it with a rich mathematically well established theory. We combine the advantages of the deterministic world of ordinary differential equations with the advantages of a Bayesian framework, which is capable to capture the noise in the data, gives a straightforward framework to include prior biological knowledge into the inference procedure, and gives as output the probability distributions over model parameters and GRN topologies.

Generating distributions over model parameters and GRN topologies is the main advantage of the Bayesian approach. The information contained in these distributions can be used to predict future states of the system for different parameter sets according to their probability. These predictions may be used to design optimal experiments such that the topology of the GRN and the model parameters can be inferred more precisely and, at least some, alternative topologies and model parameter sets can be eliminated. Thus, according to our opinion the proposed method will be highly useful to perform an iterative approach as shown in Figure 0.1 in the introduction with wetlab experiments, network reconstruction and experimental design.

In our sampling procedure samples are more likely generated where the differences between the model slopes and the experimental slopes are small. Doing so, we avoid numerical integra-

tion, which is needed for the usually used case where samples are more likely generated where the differences between the experimental data and the model predictions are small. Additionally, we include the estimation of a smoothing spline into our sampling procedure, which enables an optimal tradeoff between describing the experimental data perfectly and smoothing out the noise present in the given data. The estimation of smoothing splines and the estimation of its corresponding smoothing factor can be performed much faster than numerical integration and thus makes a sampling procedure of the posterior distribution possible.

We evaluated our method on simulated data as well as on the DREAM 2 Challenge #3 data, a synthetically designed 5-gene network in yeast. The ability of our approach to reconstruct the underlying network topology with high accuracy was shown on the simulated data. As expected, this accuracy decreases with decreasing number of data points and increasing amount of noise present in the data. However, the simulated data used describes an oscillating system, which is known to be an extremely difficult task for parameter estimation and network reconstruction since a high number of data points are needed to describe the oscillating dynamics precisely, and with a high amount of noise present in the data the oscillations collapse quickly.

Applying our method on the DREAM 2 Challenge #3 data we outperformed the network reconstruction results of other methods that were submitted to the DREAM 2 challenge in the DIRECTED-SIGNED-INHIBITORY and DIRECTED-UNSIGNED categories. Taking a look at the posterior distribution we obtained with our sampling procedure, we see that there are several modes describing the data well. This observation fits perfectly to the comment of Stolovitzky *et al.* [SMC07] that none of the submitted methods were able to infer the network precisely. With our results we may explain this comment with the presence of multiple modes in the posterior distribution such that optimization procedures searching for ONE optimal topology may get trapped in suboptimal solutions while exploring the huge search space.

Limitations and Future Work

Having now pointed out the importance of our method, we will state some crucial points of our method, which will have to be considered in more detail in future work.

Firstly, we note that we did not perform an exhaustive study of the hyperparameters for the priors we used. However, we gained some knowledge, which prior distributions are of importance for the inference procedure and for which we just can even use some broad distribution. These considerations will come from an intuitive point of view without mathematical proof.

- Let us first look at the prior distributions for the threshold parameters θ_j, $j \in \{1, \ldots, n\}$, and why these are of importance to be able to learn oscillating behavior in the system. We will consider for this purpose an example: Let's assume that gene j activates gene i and the concentration values for gene j vary between 1 and 2. Look now at the regulation functions shown in Figure 5.2 we use in our ODE model and assume a strict prior for the threshold parameter θ_j to be around 3. Then the change of concentration for gene i will only be very small unless the regulation strengths parameter β_{ij} is not very large. However, this is prevented using the sparsity L_q-prior on the regulation strengths parameters. On the other hand, the change of concentration for gene i will be quite large even for small absolute values of the regulation strengths parameter, if the inhibiting regulation function is used. Altogether, the learning of an inhibiting regulation for this

example may be favorable over the learning of an activating regulation, although the true regulation is an activating one. A similar situation will occur, if gene j inhibits gene i and the concentration values for gene i vary between 4 and 5 with the same prior used for the threshold parameter. Then an activating regulation may be favorable over an inhibiting regulation. This two examples of course do not explore the behavior of the whole ODE system and the parameter estimation issues to the full extent. They rather should illustrate which problems may occur during the parameter estimation procedure in the case the priors for the threshold parameters are not set appropriately.

- We further notice that the prior distribution for the Hill coefficient m is not of big importance, since this parameter does not change the dynamics of the system much as it is varied. This was already mentioned in Section 5.1.

- Another important observation we made is that for the DREAM 2 Challenge #3 data the estimated smoothing factor λ is much higher (between 0.25 and 0.4) than the beta distribution as depicted on the left in Figure 5.10 would imply. We also made the observation, where we used a weaker prior for the smoothing factor, that the estimated values were close to 1. Thus, the data is overfitted and the noise is not smoothed out of the data at all. The problem, that our algorithm tends to sample higher values for the smoothing factor can be explained as follows. In the Metropolis-Hastings step to sample the smoothing factor, higher values are preferable, since the first term of the used objective function (5.8) will be minimal for the case where the data is perfectly fitted, i.e., the smoothing factor is 1[8]. Of course, in this sampling step we fix the other model parameters ω to the current values and also have to minimize the second term in the objective function (5.8). Thus, there is a balance between fitting the data perfectly and fitting the model dynamics perfectly. However, the parameter ω which describe the model dynamics are only fitted to the derivatives of the obtained smoothing spline. Thus, at some point the parameter estimates for ω are only as good as the smoothing spline represents the dynamics of the underlying biological process. We observed for our inference procedure on the DREAM data, that for a weak prior on the smoothing factor the sampled values were close to 1 (results not shown). Hence, a strong prior on λ is necessary to guarantee that the data is not overfitted and the underlying dynamics and not the noise is used for parameter estimation of the ODE system. Concerning this point it remains to be clarified in future work, if this problem of overfitting can be avoided by increasing the number of slope estimates T_2 used. However, this does not change the problem, that the parameter estimates are only as good as the spline fit is. Maybe, a more promising future step will be the usage of the smoothing splines as were used by Poyton *et al.* [PVM$^+$06] and by Ramsay *et al.* [RHCC07], where the model predictions are fed back into the spline estimation step instead of using the second derivative as penalty term (compare to (5.7)). Of course, all these considerations should be investigated more from the theoretical point of view.

As a second big point to mention is the fact, that we do not take into account the noise in the data during the learning procedure. Well, we introduced the variables σ_1 and σ_2 as variables representing noise in the data and noise in the slope estimates. However, we only used these

[8] for the Matlab function `csaps` the obtained smoothing spline will be the natural cubic spline for the smoothing factor being 1

parameters in the inference procedure to weight the two terms of the objective function (5.8) such that they are balanced. From this we obtain a further topic for future work, namely the investigation how the noise present in the data and in the slope estimates obtained with the smoothing splines can be incorporated into the estimation process and which consequences this has for the results of the inference process.

The findings we obtain on the DREAM 2 Challenge #3 data point out the need for optimum experimental design approaches. With the obtained samples of the posterior distribution a Bayesian experimental design procedure may and should be approached in the future, since we provided with our method presented here a lot of information[9] contained in the posterior distribution. In Chapter 7 we will use this information to propose and perform a sequential Bayesian experimental design framework.

[9] "information" is meant here in the usual meaning of the word in the English language as well as the mathematical term defined in Definition 2.2.2

Part III.

Bayesian experimental design for the inference of gene regulatory networks

CHAPTER 6

Experimental design

Experiment and you'll see.

(Cole Porter)

Cole Porter, an American songwriter, seems to be a wise man stating that you should "experiment and you'll see". One cannot obtain any information about the world and especially about topics in systems biology without performing experiments. However, the central question in a general context is

> How do we perform useful experiments that will help us to understand a specific problem better?

The next question that arises immediately following this one is of course

> What do we mean with the word "useful" and how can we concretize it for practical issues?

In this chapter we will give answers to the second question for the typical problems arising in systems biology that are described by ordinary differential equations models. Firstly, we will discuss why experimental design is a crucial issue with ODE models in systems biology, both from the biological but also from the theoretical side. Secondly, we will give the basic mathematical theory for experimental design, both for *classical experimental design* and *Bayesian experimental design*. We will focus in this chapter on experimental design for parameter estimation purposes and do not consider experimental design techniques for the *model discrimination* purpose.[1]

6.1. Why we need experimental design

The difficulties that arise in parameter estimation methods for models used in systems biology are vast. In the introduction we gave a motivation why Bayesian parameter estimation and

[1] for an introduction into methods for experimental design to discriminate between models see [ADT07] Chapter 20

Bayesian experimental design are highly needed for problems in systems biology. We will review some points again and give more detailed arguments.

6.1.1. Systems biology approach

Biological experiments need in general a vast amount of resources. The chemicals needed and the machines used are very expensive. Furthermore, a lot of time and effort of scientists is needed to simply do the experiments. Not to mention, the literature search and the design of the experiments that has to be done before the experiment can be started. Thus, optimal experimental design is useful to save resources, since only optimal experiments have to be performed.

A further point why optimal experimental design is needed from the systems biology point of view is that biological processes are complex and the knowledge about them is constantly increasing. Thus, all this information cannot be overlooked by one person and experimental design may help with suggesting the parts of a biological process to be explored which will give the most information. However, what a suitable definition of information is, is problem-specific and may be different for different aims of investigation of biological processes.

From a more technical point of view, optimum experimental design together with modeling is also useful to make predictions about some model parameters which cannot be measured directly.

6.1.2. Mathematical and algorithmical reasons

The most important issue why optimum experimental design is essential for the purpose of parameter estimation from the mathematical and algorithmical point of view is that good parameter estimates can only be found with good data! With excellent data less numerical instabilities will arise, which increases the reliability of the estimated parameters and decreases the runtime of the optimization algorithm used to find the optimal parameter values.

What does it now mean to have good data in the context of models in systems biology? Since we pointed out in the previous section that biological experiments are expensive, usually only limited data is available. Experimental design will help to perform the experiments needed with a minimum amount of resources, which will produce the best data according to the purpose we want to address with our model we have.

Furthermore, experimental design will help to address the problems of non-identifiability [RKM+09, RKM+11] and sloppiness [GWC+07] of parameters providing data which will be complementary in the sense that from two different measurements one obtains almost double the information that is present in the biological system under consideration compared to where only one measurement is present. Such complementary data were generated with the classical experimental design approach by Apgar *et al.* [AWWT10].

6.2. Classical experimental design

This section is based on [ADT07]. For the theory and the proofs behind classical experimental design see [Puk06]. As mentioned earlier we remind the reader here that we will focus only on the case for experimental design for the purpose of parameter estimation.

We start with a parameterized model $f(x, p)$, where $x = (x_1, \ldots, x_m)$ are called *factors* or *explanatory variables* and $p = (p_1, \ldots, p_k)$ is the parameter vector that describes the model.

The factors are described by the experimental design. As an example one may consider the model to be a system of ordinary differential equations modeling the dynamic behavior of gene products and let the factors be time points at which mRNA concentrations are measured. With a parameterized model and factors the *observations* from reality can be described by

$$y_i = f(x,p) + \varepsilon_i \qquad i = 1,\dots,N$$

where ε_i denotes the additive experimental error. In the absence of *systematic error* it holds $E(\varepsilon_i) = 0$ for all i. Another common used assumption is that of independent errors with constant variance, i.e.,

$$\begin{aligned} \mathrm{Cov}(\varepsilon_i,\varepsilon_j) &= 0 \qquad (i \neq j) \qquad \text{and} \\ \mathrm{Var}(\varepsilon_i) &= \sigma^2 \qquad \forall i \end{aligned}$$

With this notation we want to estimate the parameters p using a least squares approach, i.e., the searched parameter vector $\hat{p}$ is such that the value

$$S(p) = \sum_{i=1}^{N} \big(y_i - f(x,p)\big)^2 \tag{6.1}$$

is minimal for $p = \hat{p}$.

6.2.1. Linear models

We will first explain the mathematical theory of classical experimental design for the case of linear models. For linear models, we are able to find explicit expressions for the parameter estimate $\hat{p}$. For this purpose let $E(y) = Fp$ with F being a $(N \times k)$ matrix, where the i-th row of F is $f^T(x)$, a function of the m factors. With this model for the least squares approach one has to minimize the equation

$$S(p) = (y - Fp)^T (y - Fp).$$

One can obtain the search parameter vector $\hat{p}$ by differentiation of $S(p)$ and setting it to zero. Then, by rearranging, the *normal equations*

$$F^T F\hat{p} = F^T y$$

hold. In the last equation the very important $(k \times k)$ matrix $M_{\hat{p}} := F^T F$ arises which will be called the *information matrix* for $\hat{p}$. Why is this matrix of that much importance and why is it called the information matrix? To answer this important question, we will reformulate the problem a bit. First, if $F^T F$ has full rank, we obtain for the estimator of the parameters

$$\hat{p} = (F^T F)^{-1} F^T y.$$

As a further assumption we now say that $\varepsilon_i \sim \mathcal{N}(0,\sigma^2)$ for all $i \in \{1,\dots,N\}$. Since $\hat{p}$ is a linear combination of normally distributed observations, it is also normally distributed and the covariance matrix of $\hat{p}$ is

$$\mathrm{Var}(\hat{p}) = \sigma^2 (F^T F)^{-1}. \tag{6.2}$$

Now we are beginning to get a feeling why the matrix $F^T F$ is of importance. This is the case because we want to design an experiment such that we obtain a parameter estimate $\hat{p}$ which

is as precise as possible. Referring to (6.2) we will obtain this, if we can somehow choose an experimental design which "minimizes" the matrix $(F^TF)^{-1}$. Then the variance of $\hat{p}$ would be minimal. Of course, there is no obvious way how to "minimize" a matrix. However, what can be done is the minimization of a scalar-valued function of this matrix. Common used examples for scalar-valued functions of matrices are the *determinant* and the averaged *trace*. To get a better feeling for these functions, we will give a geometrical interpretation of them. For this purpose, we mention that the $100(1-\alpha)\%$ *confidence ellipsoid* for the parameter vector p is of the form

$$(p-\hat{p})^T F^T F(p-\hat{p}) \leq ps^2 F_{k,\nu,\alpha},$$

where s^2 is an estimate of σ^2 on ν *degrees of freedom* and $F_{k,\nu,\alpha}$ is the $\alpha\%$ point of the *F-distribution* on k and ν degrees of freedom. The volume of this ellipsoid is given now by the determinant of the covariance matrix $(F^TF)^{-1}$. Thus, minimizing the determinant of $(F^TF)^{-1}$ will give us a confidence region of the parameter estimates with small volume. Designs which minimize the determinant of the covariance matrix $(F^TF)^{-1}$ are called *D-optimal*. The "*D*" stands for "determinant".

Let us now come to the geometrical interpretation for the case that we use the average of the trace of the covariance matrix as scalar-valued function to be minimized. Since the trace of a matrix is the sum of its *eigenvalues* and the *eigenvectors* describe the *principal components* of the underlying ellipsoid, minimizing the averaged trace of the covariance matrix $(F^TF)^{-1}$ implies that the averaged lengths of the principal components of the confidence ellipsoid is minimized. Designs which minimize the averaged trace of the covariance matrix $(F^TF)^{-1}$ are called *A-optimal*. The "*A*" stands for "average".

Another related experimental design that may be used is called *E-optimal*. In this design the optimal design is used where the maximal eigenvalue of the covariance matrix $(F^TF)^{-1}$ is minimal. The "*E*" here stands for "eigenvalue".

In Figure 6.1 the geometrical interpretation of the three *alphabetical* design criteria *D*-, *A*- and *E*-optimality is depicted.

As a very important point to mention considering linear models is that the covariance matrix $(F^TF)^{-1}$ is not dependent on the model parameter estimate $\hat{p}$ (see (6.2)).

Multivariate response

Until now we only considered the case where we have one *response*, i.e., only one output is measured, of the model. Now we will say something about optimum experimental design where in one experiment several outputs, i.e., responses, are measured. For this purpose let $E(y_u) = F_u p$, $u \in \{1,\ldots,h\}$, be the linear models that describe the h different responses. We further assume that the h responses for observation i are correlated and that observations i and l are independent, i.e., the noise ε_{iu} follows are multivariate normal distribution $\mathcal{N}(0,\Sigma)$, where $\Sigma = \{\sigma_{uv}\}_{u,v=1}^{h}$ denotes the covariance matrix. As information matrix $M_{\hat{p}}$ for the parameter estimate $\hat{p}$ one now uses

$$M_{\hat{p}} = \sum_{u=1}^{h}\sum_{v=1}^{h} \frac{1}{\sigma_{uv}} M_{uv}, \tag{6.3}$$

where $M_{uv} = (F_u)^T F_v$.

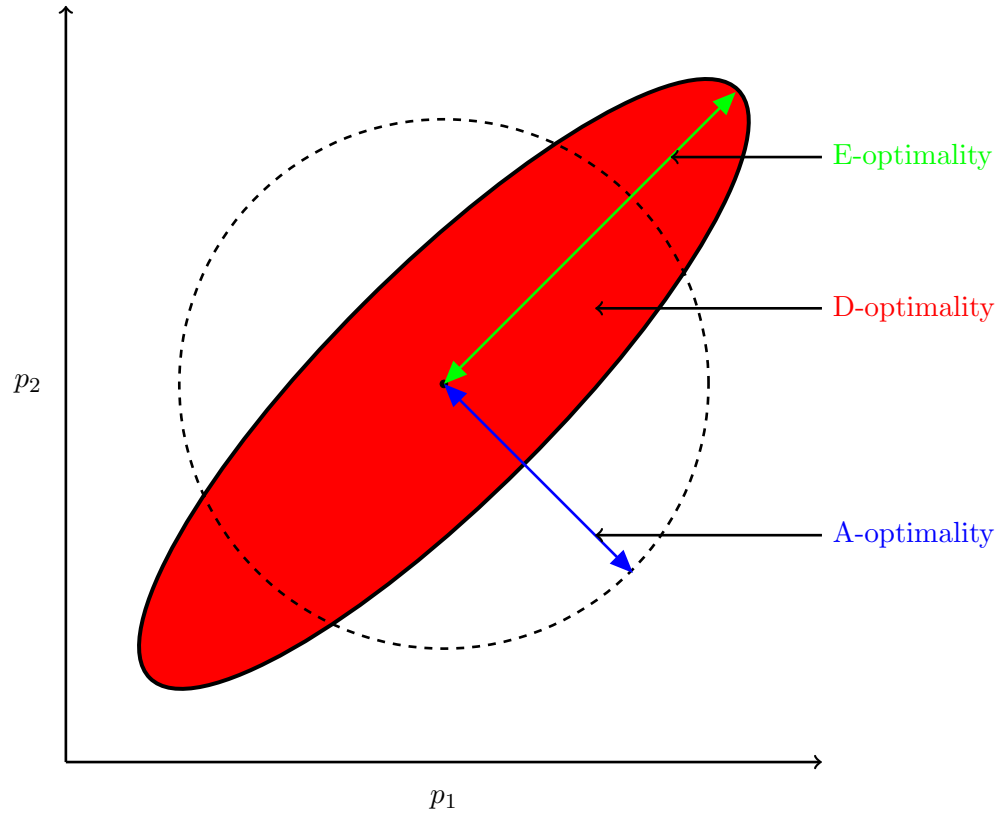


Figure 6.1.: *Geometrical interpretation of three optimality criteria of the classical alphabetical experimental design shown for a two-dimensional case where $p = (p_1, p_2)$. The optimality criteria used are D-optimality, A-optimality and E-optimality. The red area is the confidence ellipsoid and the dashed circle has as radius the averaged lengths of the eigenvectors of the confidence ellipsoid.*

6.2.2. Non-linear models

For non-linear models there is no general analytical solution for the estimator of the parameters $\hat{p}$ that minimizes (6.1) as was the case for linear models. In general, one has to use an optimization algorithm to obtain a point estimate $\hat{p}$. However, how will we do optimum experimental design for non-linear models having an estimator for the parameters? Or, to phrase the question in a different way: can we, as for the linear case, use the information matrix for $\hat{p}$ to obtain optimal experiments? The answer is: Yes, we can! Box and Lucas first introduced in 1959 the following procedure to perform design of experiments in non-linear situations [BL59]. The main trick is to trace back the non-linear case to the linear case using *Taylor's theorem*. We will explain this in more detail.

For this purpose, let

$$E(y) = \begin{pmatrix} f(x_1, p) \\ \vdots \\ f(x_m, p) \end{pmatrix}$$

be an univariate response model. The following procedure works analogously for the case with multivariate responses. Using Taylor's theorem at the point $\hat{p}$ and ignoring the derivatives of

order higher than 1 leads to

$$E(y) \approx \begin{pmatrix} f(x_1,\hat{p}) + \left.\frac{\partial f(x_1,p)}{\partial p_1}\right|_{p=\hat{p}} \cdot (p_1 - \hat{p}_1) + \ldots + \left.\frac{\partial f(x_1,p)}{\partial p_k}\right|_{p=\hat{p}} \cdot (p_k - \hat{p}_k) \\ \vdots \\ f(x_m,\hat{p}) + \left.\frac{\partial f(x_m,p)}{\partial p_1}\right|_{p=\hat{p}} \cdot (p_1 - \hat{p}_1) + \cdots + \left.\frac{\partial f(x_m,p)}{\partial p_k}\right|_{p=\hat{p}} \cdot (p_k - \hat{p}_k) \end{pmatrix}.$$

Bringing the last equation into matrix form, one immediately sees the relation to linear models:

$$E(y) - \begin{pmatrix} f(x_1,\hat{p}) \\ \vdots \\ f(x_m,\hat{p}) \end{pmatrix} = \left. \begin{pmatrix} \frac{\partial f(x_1,p)}{\partial p_1} & \ldots & \frac{\partial f(x_1,p)}{\partial p_k} \\ \vdots & & \vdots \\ \frac{\partial f(x_m,p)}{\partial p_1} & \ldots & \frac{\partial f(x_m,p)}{\partial p_k} \end{pmatrix} \right|_{p=\hat{p}} \cdot (p - \hat{p}) =: F(p - \hat{p})$$

The entries of the matrix F for the non-linear case here are also called the *parameter sensitivities*. As for the linear case one now minimizes scalar valued functions of the covariance matrix $(F^T F)^{-1}$ to perform optimum experimental design. Since by definition the parameter sensitivities depend on the model parameter estimate $\hat{p}$, the experimental design step crucially depends on $\hat{p}$. Atkinson *et al.* [ADT07] gives three possibilities how the dependence on the unknown parameter estimate $\hat{p}$ can be overcome. These methods are also known as methods to perform *robust experimental design*.

1. Sequential designs: Performing several rounds of experimental design and parameter estimation where the data obtained of the experiment is used to estimate the new parameter estimate and this estimate is then used to linearize the underlying model around this point

2. Maximin designs: First the value of the parameter estimate is found such that the optimality criterion used is maximal; second for this parameter estimate the design is found which minimizes the used optimality criterion

3. Bayesian designs: The distribution of the parameter estimates is taken into account in the experiment design process; these designs will be described in more detail in Section 6.3

6.2.3. Existing methods for parameter estimation with ODE systems

For models based on ordinary differential equations the observations y from reality are obtained as the solutions of the differential equations given as

$$\frac{dz_i}{dt}(t) = g_i(z_1,\ldots,z_n,t,p), \qquad i \in \{1,\ldots,n\}, \tag{6.4}$$

where n denotes the number of entities present in the system that is described. Thus, the expected multivariate response at some time point t is

$$E(y) = \big(z_1(t,p),\ldots,z_n(t,p)\big)^T.$$

To perform optimum experimental design we need the parameter sensitivities $\frac{\partial z_i(t,p)}{\partial p_j}$ for all entities z_i with $i \in \{1,\ldots,n\}$ and for all parameters p_j with $j \in \{1,\ldots,k\}$. These parameter

sensitivities are obtained by differentiating (6.4) with respect to every parameter p_j and using the rule of *total differentiation*. Doing so, we derive for all $i \in \{1, \ldots, n\}$ and $j \in \{1, \ldots, k\}$

$$\frac{\partial}{\partial p_j}\left(\frac{dz_i(t,p)}{dt}\right) = \frac{d}{dt}\left(\frac{\partial z_i(t,p)}{\partial p_j}\right) = \frac{\partial g_i(z,t,p)}{\partial z_i(t,p)} \cdot \frac{\partial z_i(t,p)}{\partial p_j} + \frac{\partial g_i(z,t,p)}{\partial p_j} \cdot \frac{\partial p_j}{\partial p_j} \tag{6.5}$$

Using the matrix notation

$$G_{p_j} := \left(\frac{\partial z_1(t,p)}{\partial p_j}, \ldots, \frac{\partial z_n(t,p)}{\partial p_j}\right)^T,$$

$$\frac{\partial g(z,t,p)}{\partial p_j} := \left(\frac{\partial g_1(z,t,p)}{\partial p_j}, \ldots, \frac{\partial g_n(z,t,p)}{\partial p_j}\right)^T \quad \text{with} \quad z := (z_1, \ldots, z_n)$$

and

$$\frac{\partial g(z,t,p)}{\partial z} = \begin{pmatrix} \frac{\partial g_1(z,t,p)}{\partial z_1} & \cdots & \frac{\partial g_1(z,t,p)}{\partial z_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial g_n(z,t,p)}{\partial z_1} & \cdots & \frac{\partial g_n(z,t,p)}{\partial z_n} \end{pmatrix}$$

we obtain from (6.5) the differential equation system

$$\frac{dG_{p_j}}{dt} = \frac{\partial g(z,t,p)}{\partial z} G_{p_j} + \frac{\partial g(z,t,p)}{\partial p_j},$$

which has to be solved by numerical integration to obtain the desired parameter sensitivities.

There is a vast literature on applications for classical experimental design methods with ODE systems. I will give here only a few exemplary publications.

Bernaerts *et al.* [BVI00] examined the growth temperatures on the growth rates of microorganisms. Their ODE model consists of two parameters and their experiment design consists of finding an optimally designed temperature input. As optimality criterion they used a modified E-optimality criterion, which is the minimization of the maximal eigenvalue of the information matrix divided by the smallest eigenvalue of the information matrix[2]. Banga *et al.* [BVI02] performed experimental design for the same model. However, their contribution is the usage of a stochastic global optimization algorithm instead of a local Newton-type optimization algorithm as was used by Bernaerts *et al.*

Körkel [Kör02] developed in his PhD thesis the sophisticated software package VPLAN, which uses a *multiple shooting* framework to parameterize the dynamics of the underlying system and uses *sequential quadratic programming* tools for the optimization task. This software was applied in his thesis to chemical reaction systems present in *chemical engineering* processes using the A-optimality criterion. Recently, Bandara *et al.* [BSE+09] applied VPLAN with a D-optimality criterion to a cell signaling model with six parameters to be estimated and showed for the first time that optimal experimental design for parameter estimation problems works excellently in model development in systems biology.

Asprey and Macchietto [AM02] designed two optimality criteria for experimental design purposes for dynamical systems where only poor knowledge is known about a parameter estimate $\hat{p}$. The one criteria corresponds to the maximin design described in Section 6.2.2, and

[2] This fraction is also called the *condition number* of a matrix.

the other one corresponds to a Bayesian design, which are both used for the robustification of the experimental design to circumvent the dependence of the unknown parameter estimate $\hat{p}$.

Faller *et al.* [FKT03] performed a simulation study with a model describing the MAP-kinase signaling pathway to explore to what extent experimental design improves the accuracy of the parameter estimates of models in systems biology. They found that the error in the parameter estimates can be reduced by 60% for optimal experimental design and additional experimental design can help in the detection of practical non-identifiabilities.

Kutalik *et al.* [KCW04] applied a multiple shooting framework for the estimation of parameters and performing experimental design to obtain optimal time points at which the four species of a single step in a signal transduction pathway cascade shall be measured. In their model three parameters had to be estimated, which was done with 16 measurements at different time points per species. The experimental design consists of minimizing the determinant of the inverse of the information matrix, i.e., the D-optimality criterion is used.

Gadkar *et al.* [GGI05] applied an iterative procedure to an ODE model describing the caspase function in apoptosis[3] to perform optimum experimental design and find the optimal experiment such that the number of the identifiable parameters in the model is maximal. Their most important finding was that optimal experiments performed with suboptimal measurements yields much better results than performing suboptimal experiments with optimal measurements.

Casey *et al.* [CBF$^+$07] examined a model with 56 parameters for the *epidermal growth factor receptor signaling*. Amongst other things they performed experimental design to reduce the dynamics of one entity in the model that cannot be measured and do not focus on reducing the uncertainty in the parameter estimates. They obtained that even only with one additionally measured data point of one entity in the system, that can be easily measured, reduced the uncertainty of the dynamics of the entity of interest to a high degree, whereas at the same time the uncertainty in the parameter estimates remained almost the same.

Apgar *et al.* [AWWT10] performed classical experimental design for a model for the epidermal and *neuronal growth factor* signaling which contains 48 parameters. The authors show that with the performance of five complementary experiments including *perturbation experiments* like the *overexpression* or *knockdown* of single or multiple genes present in the model, they were able to estimate all parameters within 10% of their nominal value. However, the authors needed a high number of data points to do so, which was also pointed out by Chachra *et al.* [CTS11].

6.3. Bayesian experimental design

The main difference between Bayesian and classical experimental design is the incorporation of prior knowledge into the learning process. We will describe in more detail the theory behind it according to the review papers by Chaloner and Verdinelli [CV95] and by Clyde [Cly01]. We will use the acronym BED from now on for *Bayesian experimental design*.

The main ingredient to perform BED does not differ that much from the ingredient to perform classical experimental design: one needs a model dependent on model parameters θ which describes the observations Y from reality for experiments e we want to perform. The main difference is that we need to embed this model into a probabilistic framework to have a distribution $Y \sim p_e(y \mid \theta)$. This can be easily obtained assuming the measurement noise

[3] description for non-biologists: the function of a specific enzyme on cell death

to be independent and follow a Gaussian distribution such that one gets a likelihood as was also done in Part II Chapter 5 for the purpose of parameter estimation. The second essential ingredient to perform BED missing in classical experimental design is a prior distribution over the model parameters θ, i.e., the distribution $p(\theta)$, which is, for example, obtained from expert knowledge or previous performed experiments.

Lindley [Lin72] presented in 1972 the theory now most often used to do BED. It is based on *decision theory* and consists of two parts. To do so, first one needs a *utility function* $U(d, \theta, e, Y)$, which is dependent on a *terminal decision* d, the model parameters θ and the experiment performed e with the data Y. The utility function should reflect the purpose and the costs of the experiment. We will see later different choices for utility functions used for different purposes.

This first part of this decision theoretic framework works like follows and assumes that we knew the data Y that will be obtained with the experiment e:

1. choose an experiment e
2. observe data Y for experiment e
3. select terminal decision d such that

$$\int U(d, \theta, e, Y) p(\theta \mid Y, e)\, d\theta$$

   is maximal, where the unknown parameter values θ are integrated out, and this is called *posterior expected utility* and is denoted by $U(e, Y)$

However, we want to decide which experiment to perform, such that we do not know the data Y before the experiment is performed in reality. Now the second part of the decision theoretic framework by Lindley comes into play: the experiment e is chosen such that the *pre-posterior expected utility*

$$U(e) = \int U(e, Y) p(Y \mid e)\, dY \tag{6.6}$$

is maximal, where the possible model outcomes Y for the experiment e are integrated out and $p(Y \mid e)$ is called the *predictive distribution* for model outcomes Y given the experiment e. The problem that occurs now is that we do not have the predictive distribution given. How do we now maximize (6.6) without this knowledge?

To overcome this problem we remind ourselves that the ingredients to perform BED are the distribution $p_e(Y \mid \theta)$ for model outcomes with parameter values θ and a prior distribution $p(\theta)$ for these model parameters. Thus, we obtain the predictive distribution integrating out the model parameters θ, i.e., for the predictive distribution we have

$$p(Y \mid e) = \int p_e(Y \mid \theta) p(\theta)\, d\theta.$$

Altogether, for the optimal experiment e_{best} it holds

$$U(e_{\text{best}}) = \max_e \int \left(\max_d \int U(d, \theta, e, Y) p(\theta \mid Y, e)\, d\theta \right) p(Y \mid e)\, dY.$$

6.3.1. BED for the normal linear model

For the *normal linear model* we will show the relation between classical and Bayesian alphabetical criteria. First, we need to specify what a normal linear model is.

Definition 6.3.1. A *normal linear model* is defined such that the observations Y follow a normal distribution, i.e.,

$$Y \sim p_e(y \mid \theta) = \mathcal{N}(F\theta, \sigma^2 \,\mathrm{Id}),$$

where θ is a vector of k unknown parameter, σ^2 is known, Id is the $(n \times n)$-identity matrix for the n observations and F denotes the $(n \times k)$-matrix describing the linear model[4]. Furthermore, the prior distribution for the parameter vector follows also a normal distribution, i.e.,

$$p(\theta) \sim \mathcal{N}(\theta_0, \sigma^2 R^{-1}),$$

where θ_0 and the $(k \times k)$-matrix R are known.

The normal linear model is used vastly in the literature for BED, since for it the preposterior expected utility has a closed formula. In the following two subsections we give two different utility functions which correspond to the D- and A-optimality criteria known from classical experimental design.

Shannon information

Shannon information was introduced by Lindley in 1956 [Lin56] as an appropriate utility function, if one is interested in the model parameters θ or functions of it. In more detail, we have

$$U(e, Y) = \int p(\theta \mid Y, e) \log p(\theta \mid Y, e)\, d\theta,$$

i.e., the information of the posterior distribution of the model parameters is used as utility function for BED.

For the normal linear model one obtains

$$U(e) \propto \det(nF^T F + R),$$

where the matrix F depends on the experiment e as in Section 6.2.1 and n denotes the sample size used. With this the similarity to the D-optimality criterion in classical experimental design can be seen immediately. One main difference is that the Bayesian D-optimality criterion depends on the sample size n.

Quadratic loss

If the purpose of the experiment is to obtain a point estimate of the model parameters, one uses as utility function the quadratic loss, i.e.,

$$U(e, Y) = (\theta - \hat{\theta})^T A (\theta - \hat{\theta}),$$

where A is a symmetric non-negative definite matrix.

[4] compare to the matrix F used in Section 6.2.1

For the normal linear model one obtains

$$U(e) \propto -\operatorname{tr}\big(A(nF^TF + R)^{-1}\big).$$

Here also the similarity to the A-optimality criterion in classical experimental design can be seen immediately. As above, the Bayesian A-optimality criterion depends on the sample size n.

Bayesian E-optimality

The Bayesian E-optimality criterion is referred to be that one where for a normal linear model one minimizes the maximal eigenvalue of $(nF^TF + R)^{-1}$. However, Chaloner and Verdinelli [CV95] pointed out that this criterion does not appear to correspond to any utility function and thus its justification from a decision theoretic point of view is unclear.

Main relations between Bayesian and non-Bayesian optimality criteria

We saw already that one main difference between Bayesian optimality criteria and optimality criteria from classical experimental design is that Bayesian optimality criteria depend on the sample size n. Since the identity

$$nF^TF + R = n(F^TF + \frac{1}{n}R)$$

holds, we see that the larger the sample size n the less difference is between the Bayesian and the non-Bayesian optimality criteria since the term $(1/n)R$ vanishes for $n \to \infty$. This is reasonable, since for more data available the posterior distribution of the model parameters will be described more by the data and less by the prior distribution used.

Another important issue is that we may consider singular matrices F^TF, since if we take an informative prior, i.e., the matrix R is a regular matrix, the matrix $nF^TF + R$ will always be regular, independent of the regularity or singularity of the matrix F^TF. Altogether, we only need to consider an informative prior distribution to ensure proper results for the experimental design procedure in a Bayesian framework.

6.3.2. BED for non-linear models

As seen for the case of non-linear models in classical experimental design, BED for non-linear models also traces back the non-linear case to the linear case. There are several ways how this was done in the literature.

One common approximation that is used in BED for non-linear models is to use a normal approximation for the posterior distribution of the model parameters. For this purpose, first a maximum likelihood estimate $\hat{\theta}$ is needed. As a common approximation it is used

$$p(\theta \mid y, e) \sim \mathcal{N}\big(\hat{\theta}, [n\,\mathrm{FIM}(\hat{\theta}, e)]^{-1}\big),$$

where $\mathrm{FIM}(\hat{\theta}, e)$ denotes the *Fisher information matrix*, which is defined for a likelihood $f(Y;\theta)$ as[5]

$$\big(\mathrm{FIM}(\theta, e)\big)_{i,j} = E_\theta\left[\Big(\frac{\partial}{\partial\theta_i}\log f(Y;\theta)\Big)\Big(\frac{\partial}{\partial\theta_j}\log f(Y;\theta)\Big)\right]. \tag{6.7}$$

[5] compare the Fisher information matrix to the matrix F^TF as used for the non-linear case in classical experimental design

With this approximation, the expected utility

$$U(e) \propto \int \log\det\big[n\,\mathrm{FIM}(\theta, e)\big]p(\theta)\,d\theta$$

is known as *Bayesian D-optimality* for the non-linear case.

For further optimality criteria for BED for non-linear models and other approximations used see [CV95]. Here I only wanted to introduce the Fisher information matrix, its relevance for BED for non-linear models and its similarity to the matrix F^TF as obtained for non-linear models with classical experimental design with the approximation using Taylor's theorem.

6.3.3. Existing methods for parameter estimation with ODE systems

Although Müller and Parmigiani [MP95] did not apply their approach to an ODE model, their idea of generating samples from the expected utility function $EU(d)$ for designs d is worth to mention, since it was a first attempt to solve the complete problem of BED, i.e., consider general prior distributions, consider general posterior distributions and perform an optimization to obtain the optimal design. In their approach the authors first generate ONE sample from the prior distribution for the parameters for a moderate number of designs and with this sample together with the likelihood the data d is generated. For each of these samples the value for the utility function is calculated and a curve is fitted through all these values. This curve is then used to perform a deterministic optimization to obtain the optimal design. The authors show that if the expected utility function is unimodal their obtained design is a consistent estimate of the optimal design.

Müller [Mül99] proposed three schemes to perform BED for the general case, i.e., where no analytical solution of the posterior distribution and the expected utility function is available:

1. Prior Simulation: Given a prior distribution of the model parameters θ and a likelihood $p(D \mid \theta)$, the expected utility function can be approximated by the average of the utility function evaluated with the samples for θ and D.

2. Augmented Probability Simulation: Given a proposal distribution for the designs e, a Markov chain Monte Carlo algorithm is applied to obtain samples from the distribution $h(e, d) \propto u(e, d)p(D \mid \theta)p(\theta)$.

3. Tightening the Expected Utility Surface: The distribution $h(e, d)$ introduced above is replaced with a power transformation $h^J(e, d)$. These kind of sampling puts more weight on samples with a higher probability and thus samples around the modes of $h(e, d)$.

However, all three schemes are only useful for problems with a small number of parameters and were not applied by Müller to ODE systems.

Cho *et al.* [CSKW03] applied a *multiparametric sensitivity analysis* to the NFκB signaling pathway to find out important parameter values of the underlying nonlinear ODE model. The most important parameter was used to decide manually which protein concentrations are measured. Although the authors do not use the term Bayesian experimental design, they use a uniform distribution over some range as prior knowledge for the parameter values.

Loredo [Lor04] introduced *Bayesian adaptive exploration*, which is a sequential BED based on maximum entropy sampling and applied it to determine 3 parameters of a model describing

the orbit of an extrasolar planet. Maximum entropy sampling will be described in more detail in the next chapter.

Steinke *et al.* [SST07] used Bayesian experimental design for the inference of gene regulatory networks. The authors linearized their nonlinear ODE model around the steady state, performed perturbation experiments and choose the experiment as optimal where the Kullback-Leibler divergence between the old posterior distribution and the posterior distribution containing the data from the additional experiment is maximal. The authors use as sparsity prior on the interaction strengths parameters the Laplace distribution. With this choice, the log density of the posterior is a *concave* function and thus has a single global maximum. This justifies the usage of an approximation of the posterior distribution as a product of Gaussian distributions which yields to a fast algorithm for experimental design for the purpose of the inference of gene regulatory networks.

Calderhead and Girolami [CG08] examined on the example of the *Repressilator* model the importance of optimally chosen measurements[6] to obtain a high value for the Kullback-Leibler divergence between the prior and the posterior distribution. To do so, the authors generated samples from a 10-dimensional space with a population-based Markov chain Monte Carlo algorithm, where for each species 49 data points were available. The posterior distribution was approximated with a *kernel density estimator*.

Busetto and Buhmann [BB09] divided the posterior distribution into a fixed number of clusters with a weighted K-means algorithm and wanted to find the optimal intervention such that the data obtained from this intervention will minimize the entropy of the global weights of the clusters, i.e., the intervention is searched such that the discrimination between alternative modes over the posterior distribution for the parameters is maximal. They applied their approach to the *Goodwin model* which consists of 4 parameters.

Busetto *et al.* [BOB09] consider the problem of finding the entities in nonlinear ODE models for the *TOR pathway* to be measured sequentially such that the mutual information between the models and the data is maximized, which is equivalent to maximize the expected Kullback-Leibler divergence between the posterior and the prior distribution. The authors tested their algorithms against three classical experimental designs (A-, D- and E-optimality) and were able to obtain 33% more information with their method for the same amount of data points. Their work is a good starting point for BED for model selection problems with nonlinear ODE models.

He *et al.* [HYB10] proved mathematically that a modified Morris *global sensitivity analysis* is equivalent to a Bayesian A-optimality criterion and performed a case study with a signaling pathway model, where 5 parameters were estimated.

Kramer and Radde [KR10] used a nonlinear ODE model for the regulation of the *secretory transport at the trans-Golgi network* and considered perturbation experiments to perform BED and looked at 3 parameters maximizing the information of the posterior distribution using steady-state measurements of the system. Their main contribution is the development of a new estimator for the calculation of the information of the posterior distribution which is described in more detail in [KHAR10] and is suitable even for small sample sizes whereas usually used kernel density estimators may fail in this context, at least for the exemplary results on the model for the secretory transport at the trans-Golgi network.

Terejanu *et al.* [TUM11] performed Bayesian experimental design maximizing the mutual information between the model parameters and the model predictions. The authors applied

[6] in their example the authors concern which species is measured and how many species are measured

their approach to a non-linear model for the *nitridation of graphite*, where three most uncertain parameters needed to be estimated. The experimental design was performed in a sequential manner where the possible experiments were fixed to 28 experiments beforehand. Additionally, the authors compare the results of their method with the results of using maximum entropy sampling on a small non-ODE model with additive and multiplicative noise and the results are comparable for the model with additive noise. However, maximum entropy sampling is not applicable to the model with multiplicative noise.

Huan and Marzouk [HM11] recently described a promising and interesting framework where a Bayesian experimental design is performed considering general distributions of model parameters of non-linear models and the expected Kullback-Leibler divergence between posterior and prior distribution is maximized. This maximization is done with stochastic optimization algorithms which are suited to problems where the objective function is available only approximately, namely as average over a function of the samples from the posterior and prior distribution. Additionally, to speed up the evaluation of their objective function, the authors used a *polynomial chaos* surrogate to obtain a smooth function depending on model parameters and design conditions. However, the authors showed the performance of their method only for a model where two parameters are to be estimated.

As a last remark, it is worth to mention that several methods for the performance of BED for the model identification purpose were proposed recently in the field of systems biology like the one seen above by Busetto *et al.* [BOB09], examined in the group of Michael Stumpf [TS10, BSSS11] and by Vyshemirsky and Girolami [VG08].

CHAPTER 7

Sequential Bayesian experimental design by means of maximum entropy sampling

> If your result needs a statistician then you should design a better experiment.
>
> *(Ernest Rutherford)*

Having developed in Part II a method for parameter estimation for GRNs using a Bayesian approach and obtaining distributions over parameters, we will use these distributions now to perform sequential Bayesian experimental design (BED).

7.1. Bayesian Learning Framework

For parameter estimation, we embed our nonlinear ordinary differential equations model as described in more detail in Section 5.1 into a Bayesian framework. For this purpose, we assume that the measured data $d_{i\tau}$, $i \in \{1, \ldots, n\}$, $\tau \in \{1, \ldots, T\}$, is corrupted by independent mean zero Gaussian noise with variance σ^2. Thus, by denoting with $\omega = (s, \gamma, \beta, k, m, \text{start})$ the vector which contains all model parameters, the likelihood looks like

$$p(D|\omega, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \prod_{i=1}^{n} \prod_{\tau=1}^{T} \mathrm{e}^{-\frac{1}{2\sigma^2}\left(d_{i\tau} - x_i(t,\omega)\right)^2}, \tag{7.1}$$

which is equivalent to a least squares objective function up to log-transformation and scaling. Note, that $x_i(t, \omega)$ is obtained by numerical integration of (5.4) and thus, additionally to the model parameters (s, γ, β, k, m), we need to consider also the initial values $\text{start} := \big(x_1(t_0), \ldots, x_n(t_0)\big)$ of the ODE system as parameters, since we do not know the initial values accurately because of noisy experimental data.

Differential equations were solved numerically using the methods for numerical integration of ordinary differential equations as implemented in Matlab. For a detailed description for

this implementation see [SR97, SGT03].

Since we are interested in the probability distribution of the model parameters ω and the variance σ^2 given the experimental data D, we use Bayes' theorem to obtain

$$p(\omega, \sigma^2|D) = \frac{p(D|\omega, \sigma^2)p(\omega, \sigma^2)}{p(D)}, \tag{7.2}$$

where $p(D|\omega, \sigma^2)$ is given by the likelihood (7.1), $p(\omega, \sigma^2)$ denotes the prior distribution on model parameters and $p(D)$ is a normalizing constant independent of ω and σ^2. Here, we set the parameter σ^2 in advance and do not estimate it.

7.2. Bayesian Experimental Design Procedure

7.2.1. General BED

The goal for experimental design is to specify an experiment that will provide new data which is best suitable for our purposes. In a Bayesian experimental design framework, the key ingredient for this purpose is the *predictive distribution* $p(d|D, M_e)$ for future data. Here we denote by the index e one of the possible experiments we are able to perform, by D the data we already have, by d the data we expect from experiment e and by M_e we denote the model with which we will generate data for given experiment e. What remains now is a decision problem which experiment to choose, since we are able to make data predictions with all experiments. What we need next is a *utility function* $U(d, e)$ which gives us a measure how useful an experiment e with given data d is. Since we do not know the data the experiment will give beforehand, we have to integrate over all possible data. Thus, the best experiment e_{best} is the one which maximizes the expected utility of the possible data, i.e.,

$$e_{\text{best}} = \arg\max_e EU(e) = \arg\max_e \int U(d, e)p(d|D, M_e)\, dd. \tag{7.3}$$

Two questions arise:

1. How do we calculate the predictive distribution $p(d|D, M_e)$?
2. What is a suitable utility function $U(d, e)$?

To answer the first question, let us remember that we look at models M_e which are parametrized by some parameters ω. With prior knowledge on model parameters, or learned parameter distributions according to given data D, we are able to calculate the predictive distribution of future data d for the experiment e as follows:

$$p(d|D, M_e) = \int p(d|\omega, M_e)p(\omega|D)\, d\omega. \tag{7.4}$$

In our case we consider as possible experiments e additional time points t_e where concentrations of all gene products are measured. Thus, the first term in (7.4) is obtained by sampling d_i from $\mathcal{N}\big(x_i(t_e, \omega), \sigma^2\big)$ for all $i \in \{1, \ldots, n\}$. From (7.2) we obtain the second term in (7.4).

To answer the second question, we take the *information* of the final posterior distribution for the model parameters ω as utility function, i.e.,

$$U(d, e) = \int p(\omega|d, D, M_e) \log p(\omega|d, D, M_e)\, d\omega =: \mathrm{I}(\omega|d, D, M_e). \tag{7.5}$$

This utility function was suggested by Lindley in 1956 [Lin56] and is often used in BED, if one wants to learn about the parameters ω (see also Section 6.3.3).

Plugging (7.4) and (7.5) in (7.3), we obtain

$$EU(e) = \int \left(\int \left(\int p(\omega'|d, D, M_e) \log p(\omega'|d, D, M_e)\, d\omega' \right) \cdot p(d|\omega, M_e)\, dd \right) p(\omega|D, M)\, d\omega.$$

7.2.2. Maximum Entropy Sampling (MES)

This computationally intractable triple integral can be rewritten with the information theory analog of Bayes' theorem as:

$$\begin{aligned} EU(e) &= \int p(d|D, M_e)\, \mathrm{I}(\omega|d, D, M_e)\, dd = \mathrm{I}(d, \omega|D, M_e) - \mathrm{I}(d|D, M_e) \\ &= \mathrm{I}(\omega|D, M_e) + \int p(\omega|D, M_e)\, \mathrm{I}(d|\omega, D, M_e)\, d\omega - \mathrm{I}(d|D, M_e). \end{aligned} \tag{7.6}$$

The first term in (7.6) is the information in the prior distribution $p(\omega|D, M)$ which is independent of the experiment e and thus constant. The second term gives the average information in the sampling distribution $p(d|\omega, D, M_e)$, which is also constant for our model, since $d_i \sim \mathcal{N}\big(x_i(t_e, \omega), \sigma^2\big)$ for all $i \in \{1, \ldots, n\}$ with σ^2 fixed and the information is translation invariant according to Theorem 2.2.11. The third term in (7.6) gives the information in the predictive distribution $p(d|D, M_e)$. Thus, our maximization problem (7.3) can be written as:

$$e_{\text{best}} = \arg\max_e EU(e) = \arg\max_e \big(-\mathrm{I}(d|D, M_e)\big) = \arg\max_e \mathrm{Ent}(d|D, M_e) \tag{7.7}$$

since the negative information of a distribution gives the *entropy* of a distribution. The entropy is a measure of the uncertainty in a distribution[1]. Thus, according to (7.7), the best experiment to choose is the one where the uncertainty in the predictive distribution is the highest. In other words:

> We perform an experiment where we know the least!

This procedure is called *maximum entropy sampling* and was first described by Sebastiani and Wynn [SW97, SW00]. Loredo [Lor04] gives an excellent overview over how maximum entropy sampling can be used.

It is essential to mention that the parameter σ^2 is fixed in advance, since otherwise the second term in (7.6) would not be constant anymore and thus MES would not be applicable anymore.

7.3. MCMC Sampling from the Posterior

We used the population-based MCMC (for an introduction into population-based MCMC algorithms see Section 3.1.4) approach by Hu and Tsui [HT10] with two small changes. Let us first describe the approach by Hu and Tsui. It is called the *Distributed Evolutionary Monte Carlo* (DGMC) algorithm and a pseudocode version is given in Algorithm 7.1. The

[1] compare to Section 2.2.1 and Definition 2.2.2

Algorithm 7.1 Distributed Evolutionary Monte Carlo (DGMC) [HT10]

Require: desired distribution $p(\cdot)$, size of chain population N, number of subpopulations k, starting values $\left(x_1^{(0)}, \ldots, x_N^{(0)}\right)$ for the N chains, number of Markov chain samples T, probability for migration p_{mig}, probability for mutation p_{mut}, proportion in every subpopulation $\text{prop}_{\text{cross}}$ to perform the crossover operator on

$t \leftarrow 0$
while $t < T$ **do**
  Sample u_1 from $\mathcal{U}(0,1)$
  if $u_1 < p_{\text{mig}}$ **then**
    Perform migration operator (see Algorithm 7.2)
  end if
  for $i = 1$ **to** k **do**
    Sample u_2 from $\mathcal{U}(0,1)$
    if $u_2 < p_{\text{mut}}$ **then**
      Perform mutation operator on subpopulation i (see Algorithm 7.3)
    else
      Perform crossover operator on subpopulation i (see Algorithm 7.4)
    end if
  end for
  $t \leftarrow t + 1$
end while
return Markov chains $\left(x_1^{(t)}, \ldots, x_N^{(t)}\right)_{t=0}^{T}$

idea of the DGMC algorithm is to use the *Distributed Genetic Algorithm* (DGA) [Tan89] and incorporate it into an MCMC framework. In the DGA the whole population is divided into several subpopulations and a genetic algorithm is performed on each of the subpopulations. Additionally, individuals from subpopulations migrate to other subpopulations, such that information is exchanged between the subpopulations. The DGA algorithm is able to prevent *premature convergence*, where the genetic algorithm converges to early and ends in a suboptimal result, because the population members are too similar. This behavior occurs in practice for single-population genetic algorithms [Rya96].

In the DGMC algorithm N Markov chains are generated, where all generate samples from the desired distribution $p(\cdot)$[2]. These N chains are divided into k subpopulations each containing m chains. To be more clear here, we start with starting values for every chain $\left(x_1^{(0)}, \ldots, x_N^{(0)}\right)$, divide these N values into k subpopulations where each subpopulation contains m values. And then we apply genetic operators on the values of each subpopulation. The genetic operators that are used by Hu and Tsui are *migration*[3], *mutation* and *crossover*. Pseudocode versions of them are given in Algorithms **??** - 7.4.

In the migration operator information between the subpopulations is exchanged. To do so, one first has to specify a scheme according to one knows which subpopulation gives information to another subpopulation. For this purpose, one permutes the vector $(1, \ldots, k)$ and obtains the vector P_{sub}. According to this we now let pass information from the subpopulation $P_{\text{sub}}(i)$

[2] it is straightforward to use Markov chains which generate samples from different distributions (compare to Section 3.1.4)

[3] this genetic operator was called the *exchange* operator in Section 3.1.4

Algorithm 7.2 Migration operator for the DGMC algorithm

Require: size of chain population N, number of subpopulations k, current samples $\left(x_1^{(t)}, \ldots, x_N^{(t)}\right)$ for the N chains

$\left(x_1^{(t+1)}, \ldots, x_N^{(t+1)}\right) \leftarrow \left(x_1^{(t)}, \ldots, x_N^{(t)}\right)$
$P_{\text{sub}} \leftarrow$ Permute the vector $(1, \ldots, k)$
for $i = 1$ **to** k **do**
    randomly pick one individual $x_*^{(t)}$ from subpopulation $P_{\text{sub}}(i)$ and store it in a new vector h as $h(i)$, additionally store the index of the individual in a new vector h_{index} as $h_{index}(i)$
end for
for $i = 1$ **to** k **do**
    $x_{h_{index}(i)}^{(t+1)} \leftarrow h(i-1)$ {in this loop we use the notation $0 \equiv k$}
end for
return Samples $\left(x_1^{(t+1)}, \ldots, x_N^{(t+1)}\right)$ for the N chains

Algorithm 7.3 Mutation operator for the DGMC algorithm

Require: desired distribution $p(\cdot)$, number m of individuals in the subpopulation i, current samples $\left(x_{i1}^{(t)}, \ldots, x_{im}^{(t)}\right)$ for the m chains, proposal distribution $q(\cdot|x^{(t)})$

$\left(x_{i1}^{(t+1)}, \ldots, x_{im}^{(t+1)}\right) \leftarrow \left(x_{i1}^{(t)}, \ldots, x_{im}^{(t)}\right)$
randomly pick one individual from subpopulation i with the index h_{index}
Sample $\tilde{x}$ from $q\left(\cdot|x_{h_{index}}^{(t)}\right)$
$\alpha \leftarrow \min\left\{1, \frac{p(\tilde{x})q\left(\tilde{x}|x_{h_{index}}^{(t)}\right)}{p\left(x_{h_{index}}^{(t)}\right)q\left(x_{h_{index}}^{(t)}|\tilde{x}\right)}\right\}$
Sample u from $\mathcal{U}(0,1)$
if $u < \alpha$ **then**
    $x_{h_{index}}^{(t+1)} \leftarrow \tilde{x}$
end if
return Samples $\left(x_{i1}^{(t+1)}, \ldots, x_{im}^{(t+1)}\right)$ for the m chains

to the subpopulation $P_{\text{sub}}(i+1)$. Here we note that subpopulation $k+1$ should be defined as subpopulation 1. An example for 4 subpopulations with $P_{\text{sub}} = (2, 1, 4, 3)$ is depicted in Figure 7.1. Having now the needed scheme, we randomly select one individual from every subpopulation and pass it over to the destined subpopulation according to the vector P_{sub}. However, can we now be sure that this migration operator holds the desired distribution $p(\cdot)$ invariant? The short answer to this question is: Yes! And for the long answer, i.e., the mathematical proof, we refer to the original paper [HT10] Section 3.3.

In the mutation operator one individual in subpopulation i is selected randomly and a single Metropolis-Hastings step is performed on it. The first change that we made in the usage of the DGMC algorithm is that we not only updated one individual in subpopulation i within the mutation operator step, but updated a proportion prop_{mut} of individuals in subpopulation i. This parameter is a user-defined parameter and encounters for the purpose of efficiency and thus not that many individuals are kept unchanged in one DGMC sampling step.

The crossover operator used by Hu and Tsui was proposed by the authors themselves. It works as follows:

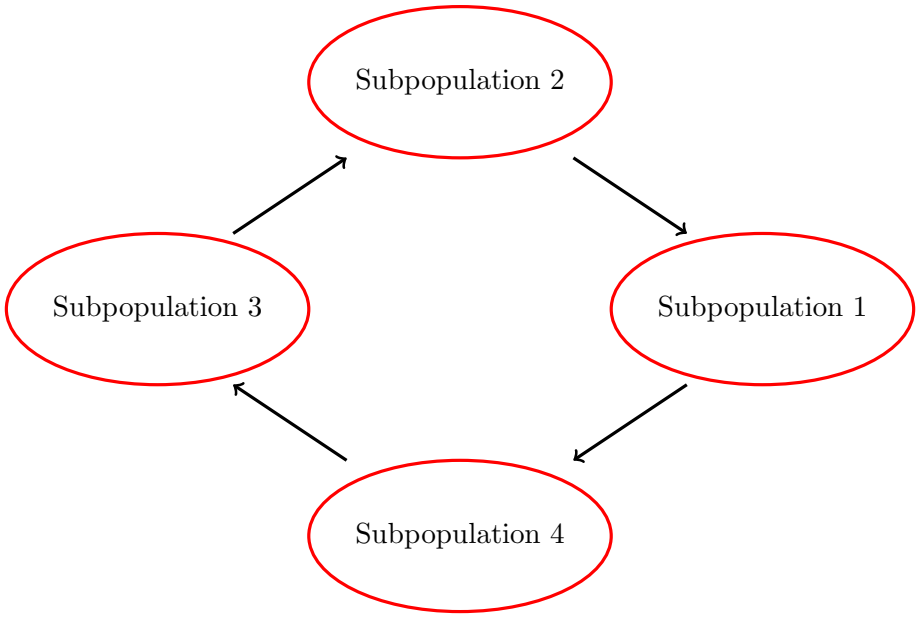


Figure 7.1.: *Exemplary exchange of information in the DGMC algorithm between* 4 *subpopulations with the vector* $P_{\text{sub}} = (2, 1, 4, 3)$*, i.e., subpopulation* 2 *passes over information to subpopulation* 1*, etc. This picture is adapted from [HT10].*

Algorithm 7.4 Crossover operator for the DGMC algorithm proposed by Hu and Tsui [HT10]

Require: desired distribution $p(\cdot)$, number m of individuals in the subpopulation i, current samples $\left(x_{i1}^{(t)}, \ldots, x_{im}^{(t)}\right)$ for the m chains, proportion $\text{prop}_{\text{cross}}$ in subpopulation i to perform the crossover operator on

1: $\left(x_{i1}^{(t+1)}, \ldots, x_{im}^{(t+1)}\right) \leftarrow \left(x_{i1}^{(t)}, \ldots, x_{im}^{(t)}\right)$
2: **for** $\ell = 1$ **to** $(\text{prop}_{\text{cross}} \cdot m)$ **do**
3: randomly pick one individual from subpopulation i with the index h_{index} {here one has to guarantee that the individual with the index h_{index} has not been used before in the former for-loop steps}
4: randomly pick two other individuals from subpopulation i with the indexes h_{index1} and h_{index2}
5: $\mathbf{e} \leftarrow \frac{x_{h_{index1}}^{(t+1)} - x_{h_{index2}}^{(t+1)}}{\left\| x_{h_{index1}}^{(t+1)} - x_{h_{index2}}^{(t+1)} \right\|}$
6: sample a scalar r from the density $g(r) = \frac{p\left(x_{h_{index}}^{(t)} + r\mathbf{e}\right)}{\int p\left(x_{h_{index}}^{(t)} + r'\mathbf{e}\right) dr'}$
7: $x_{h_{index}}^{(t+1)} \leftarrow x_{h_{index}}^{(t)} + r\mathbf{e}$
8: **end for**
9: **return** Samples $\left(x_{i1}^{(t+1)}, \ldots, x_{im}^{(t+1)}\right)$ for the m chains

1. randomly select one individual $x^{(t)}_{h_{index}}$ in subpopulation i
2. randomly select two other individuals $x^{(t+1)}_{h_{index1}}$ and $x^{(t+1)}_{h_{index2}}$ in subpopulation i
3. sample a scalar r from the density

$$g(r) = \frac{p\big(x^{(t)}_{h_{index}} + r\mathbf{e}\big)}{\int p\big(x^{(t)}_{h_{index}} + r'\mathbf{e}\big)\, dr'}, \qquad \text{where} \qquad \mathbf{e} = \frac{x^{(t+1)}_{h_{index1}} - x^{(t+1)}_{h_{index2}}}{\big\|x^{(t+1)}_{h_{index1}} - x^{(t+1)}_{h_{index2}}\big\|} \tag{7.8}$$

4. set $x^{(t+1)}_{h_{index}} = x^{(t)}_{h_{index}} + r\mathbf{e}$

This procedure is applied iteratively to $\text{prop}_{\text{cross}} \cdot m$ individuals in subpopulation i, where $\text{prop}_{\text{cross}}$ denotes the user-defined proportion of individuals that has to be updated with the crossover operator in one DGMC algorithm step. Two concerns arise immediately looking at this procedure:

1. Does this crossover operator leave the desired distribution invariant?
2. How do we always know the terms $x^{(t+1)}$ for future time $t+1$ needed for the calculation of the direction $\mathbf{e}$?

For the answer of the first question we refer to the original paper for a proof. This is a nice result also in terms that the algorithm will run more efficiently, since the proposed new individual $x^{(t)}_{h_{index}} + r\mathbf{e}$ is always accepted. However, the drawback is that it has to be easy to sample from the density $g(r)$, which is unfortunately not always the case. The authors suggest to use other sampling algorithms to obtain samples for the scalar r inside the DGMC algorithm. However, they also state that this leads to additional computational time and a single Metropolis-Hastings may be used at the cost of losing some accuracy. Since for our distribution of interest it is not easy to generate samples for the scalar r, we used a Metropolis-Hastings step in the crossover operator as shown as pseudocode version in Algorithm 7.5.

The answer to the second question is simple: We don't! However, the crossover operator is applied iteratively to the individuals of one subpopulation. Thus, if any of the individuals $x^{(t)}_{h_{index1}}$ and $x^{(t)}_{h_{index2}}$ was updated already with the crossover operator, we use the updated version of it and otherwise we just use the current individual. To guarantee that the notation we used is defined, we first set the values $x^{(t+1)}$ to the current states $x^{(t)}$ as a first step (compare to the first step in Algorithm 7.4 and Algorithm 7.5).

7.4. Entropy Calculation

We used the algorithm provided by Györfi and van den Meulen [GvdM87] to obtain estimates for the entropy of the predictive distribution $p(d|D, M_e)$ for different experiments e. The estimator the authors define is a histogram-based entropy estimator and they show its strong consistency (for a definition and theoretical background on estimators see Section 2.1.4).

We will describe in this section the estimator in more detail and give also a pseudocode version of how we calculate the needed estimates. Let us start with samples $(y_1, \ldots, y_n)$ from a density of a continuous random variable X, which are seen as realizations of random

Algorithm 7.5 Crossover operator how we use it for the DGMC algorithm

Require: desired distribution $p(\cdot)$, number m of individuals in the subpopulation i, current samples $\left(x_{i1}^{(t)}, \ldots, x_{im}^{(t)}\right)$ for the m chains, proportion $\text{prop}_{\text{cross}}$ in subpopulation i to perform the crossover operator on, shape parameter r and rate parameter a of gamma distribution Gamma(r, a), proposal distribution $q(\cdot|x^{(t)})$

$\left(x_{i1}^{(t+1)}, \ldots, x_{im}^{(t+1)}\right) \leftarrow \left(x_{i1}^{(t)}, \ldots, x_{im}^{(t)}\right)$
for $\ell = 1$ **to** $(\text{prop}_{\text{cross}} \cdot m)$ **do**
  randomly pick one individual from subpopulation i with the index h_{index} {here one has to guarantee that the individual with the index h_{index} has not been used before in the former for-loop steps}
  randomly pick two other individuals from subpopulation i with the indexes h_{index1} and h_{index2}
  $\mathbf{e} \leftarrow \frac{x_{h_{index1}}^{(t+1)} - x_{h_{index2}}^{(t+1)}}{\left\| x_{h_{index1}}^{(t+1)} - x_{h_{index2}}^{(t+1)} \right\|}$
  sample a scalar $\tilde{r}$ from the gamma distribution Gamma(r, a)
  $\tilde{x} \leftarrow x_{h_{index}}^{(t)} + \tilde{r}\mathbf{e}$
  $\alpha \leftarrow \min\left\{1, \frac{p(\tilde{x}) q\left(\tilde{x}|x_{h_{index}}^{(t)}\right)}{p\left(x_{h_{index}}^{(t)}\right) q\left(x_{h_{index}}^{(t)}|\tilde{x}\right)}\right\}$
  Sample u from $\mathcal{U}(0, 1)$
  if $u < \alpha$ **then**
    $x_{h_{index}}^{(t+1)} \leftarrow \tilde{x}$
  end if
end for
return Samples $\left(x_{i1}^{(t+1)}, \ldots, x_{im}^{(t+1)}\right)$ for the m chains

variables $(Y_1, \ldots, Y_n)$. Furthermore, let A_{ni} be cubes in $\mathbb{R}^k$ with *edge length* h_n, where k denotes the dimension of the data d. Then we have for the Lebesgue measure of A_{ni}

$$\lambda(A_{ni}) = h_n^k.$$

Additionally, we define the *empirical measure* for the cubes A_{ni} as

$$\mu_n(A_{ni}) = \frac{\text{number of } y_i\text{'s falling in } A_{ni}}{n}.$$

With these two definitions we are able to give the formula for the entropy estimator as

$$T_n^e = -\sum_{i \in \mathcal{F}_n} \mu_n(A_{ni}) \frac{\mu_n(A_{ni})}{\lambda(A_{ni})}. \tag{7.9}$$

One may now wonder, what the set $\mathcal{F}_n$ is, if it does not contain all possible indices i, and why we can neglect the other terms. To tackle the first concern, we give the formal definition of

$$\mathcal{F}_n := \{i | \mu_n(A_{ni}) \geq a_n h_n^k\},$$

with $0 < a_n < 1$ for all $n \in \mathbb{N}$ and

$$\lim_{n \to \infty} a_n = 0. \tag{7.10}$$

Algorithm 7.6 Algorithm to calculate entropy estimates

Require: realizations $(y_1, \ldots, y_n)$ of the density of interest, dimension k of data
1: $\alpha \leftarrow 1/(k+1)$
2: $\beta \leftarrow 1/(k+2)$
3: $h_n \leftarrow 1/(n^\alpha)$
4: $a_n \leftarrow n^{-\beta}$
5: specify cubes A_{ni} according to their edge length h_n
6: $T_n^e \leftarrow 0$
7: **for** i **do**
8: **if** $\mu_n(A_{ni}) \geq a_n h_n^k$ **then**
9: $T_n^e \leftarrow T_n^e + \mu_n(A_{ni}) \frac{\mu_n(A_{ni})}{h_n^k}$
10: **end if**
11: **end for**
12: $T_n^e \leftarrow -T_n^e$
13: **return** T_n^e as defined in (7.9)

We see immediately that because of (7.10) the set $\mathcal{F}_n$ contains more indices i for growing number of samples n. Basically, one can say that cubes A_{ni} that contain none or only very few samples y_i are neglected during the calculation of T_n^e.

To complete the survey of the estimator we give the the essential theorem for the reliability of its usage.

Theorem 7.4.1. *Assume that the condition* (7.10) *holds. Additionally, assume that the following condition holds for each* $c > 0$

$$\sum_{n=1}^{\infty} \frac{1}{a_n h_n^k} \exp(-cna_n h_n^k) < \infty$$

and $1/h_n$ *is an integer such that*

$$\lim_{n\to\infty} h_n = 0.$$

If $\mathrm{Ent}(X)$ *is finite, then we have*

$$T_n^e \xrightarrow{a.s.} \mathrm{Ent}(X).$$

Proof. See [GvdM87] Section 4. □

It remains to specify the constants a_n and h_n for practical use. Györfi and van den Meulen mentioned, that the assumptions of Theorem 7.4.1 are valid, if one uses $h_n = 1/(n^\alpha)$ and $a_n = n^{-\beta}$, where $k \cdot \alpha + \beta < 1$ and $a, b > 0$. We will use their suggestion and set $\alpha = 1/(k+1)$ and $\beta = 1/(k+2)$. A pseudocode version of the algorithm we used is shown in Algorithm 7.6.

7.5. Evaluation of obtained posterior distributions

To evaluate the results of the BED method described in the previous sections, we take a look at the posterior distributions of the model parameters after performing optimal experiments as proposed by the BED with MES framework. We estimated the information $\mathrm{I}(\omega)$ using the entropy estimator by Györfi and van den Meulen as described in the previous section.

Of course, since we want to obtain the *information* and not the *entropy*, we just take the negative value of the obtained entropy estimate, since we have by definition for a random variable X

$$\mathrm{I}(X) = -\operatorname{Ent}(X).$$

7.6. Implementation

We implemented our algorithm in Matlab, Release 2010b (The Mathworks), using the Statistics Toolbox. The Statistics Toolbox was needed to generate random numbers from the gamma distribution with the function `gamrnd`. We need these to propose the crossover operator parameter r and the standard deviation of the normal distribution which is used to propose new samples in the mutation operator.

The calculations were done on a Linux cluster with dual-processor 3.1 GHz XEON quadcore machines with 32 GB RAM.

To speed up the calculations of the population-based MCMC algorithm we parallelized the code with *pMatlab*, the Parallel Matlab Toolbox v2.0.1 [Kep09,KMK], which uses *MatlabMPI* as a Message Passing Interface basis. pMatlab and MatlabMPI follow the SPMD model and the distributed array model is used as a model to perform communication between processors (compare to Section 3.2).

7.7. Results

7.7.1. Simulated Data: Five Gene Network

We evaluate our approach on a 5-gene regulatory network defined by our ODE model (5.4). We simulated data with the model parameters described in Subsection 5.8.1. The data were simulated by numerical integration with the function `ode15s` in Matlab with the initial values $(1, 1, 1, 1, 1)$.

We did not use any prior distributions on the model parameters.

In the first run of the parameter estimation sampling we started with two data points, i.e., we used the data of all genes at the time points $t = 0$ and $t = 11$. As possible experiments e to perform we proposed to measure the concentration of all gene products at the 19 time points $(1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 12, 13, 14, 15, 16, 17, 18, 19, 20)$. Thus, altogether we need to perform 20 parameter estimation steps. We did so for our sequential BED procedure and estimated the information of the obtained posterior distribution $p(\omega|D_i)$, $i \in \{1, \ldots, 20\}$, where D_i denotes the data that we used in the i-th parameter estimation procedure to obtain the samples of the posterior distribution of the parameter vector ω.

Let us give an example to clarify this notation. We have always $D_1 = \big\{x(0, \omega), x(11, \omega)\big\}$ and let us assume that the next proposed experiment would be to measure the concentration of all 5 gene products at $t = 5$. Then we would have $D_2 = \big\{x(0, \omega), x(5, \omega), x(11, \omega)\big\}$, where we write $x(t, \omega) := \big(x_1(t, \omega), \ldots, x_5(t, \omega)\big)$.

To get a feeling how much the proposed experiments improve the parameter estimation procedure, we compared the information of the posterior distribution of the model parameters obtained with our BED procedure with the information of the posterior distribution of the model parameters, if we just perform sequentially one randomly chosen additional experiment. Furthermore, we compare the amount of information with the sequential experimental design

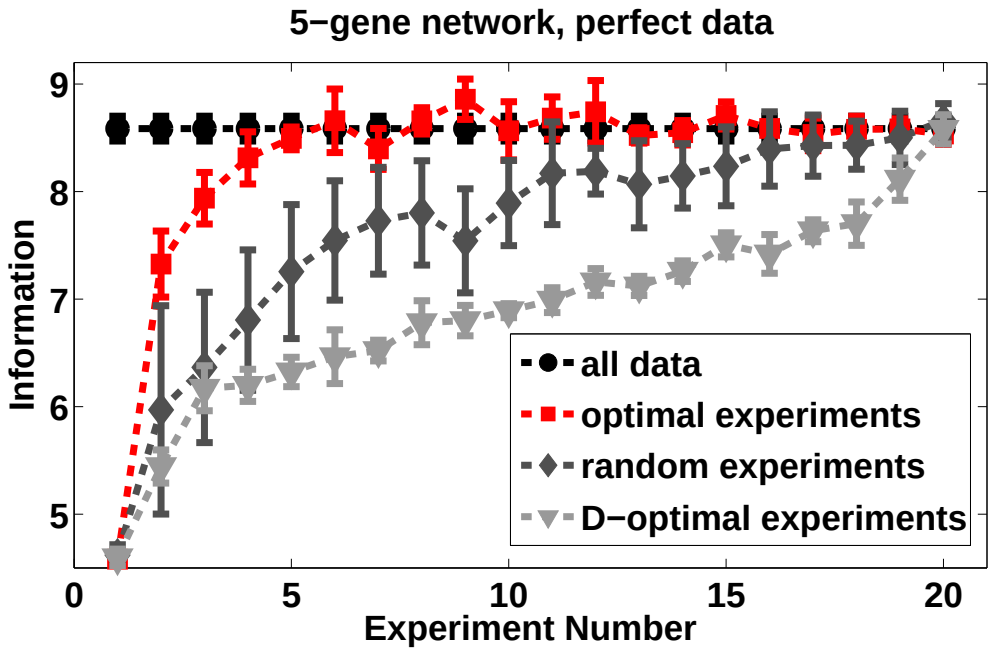


Figure 7.2.: *Results for the ODE model containing* 46 *parameters for gene network reconstruction with perfect data. For* 5 *independent runs the mean and the standard deviation of the information contained in the posterior distributions is depicted. The black line represents the runs with the maximal amount of data available. The dark grey line represents random experiments, the light grey line represents the results obtained with classical experimental design and the red line line illustrates optimal experiments.*

procedure with the amount of maximally available information that we will obtain, if we run the sampling procedure with D_{20}.

We generated samples from the posterior distributions $p(\omega|D_i)$, $i \in \{1, \ldots, 20\}$, with the population-based MCMC algorithm specified in Algorithm 7.1 with the crossover operator as in Algorithm 7.5. We ran 20 Markov chains subdivided into 4 subpopulations, where we generated with every chain $202,000$ samples from $p(\omega|D_i)$ with a burn-in of $2,000$ steps. The probability for migration was set to $p_{\text{mig}} = 0.005$, the probability for mutation was set to $p_{\text{mut}} = 0.5$ and the proportion prop_{mut} was set to 1, i.e., the mutation operator is performed on all individuals of one subpopulation. The proportion $\text{prop}_{\text{cross}}$ was also set to 1, i.e., the crossover operator is performed iteratively on all individuals of one subpopulation. The noise parameter is set to $\sigma = 0.01$ for simulated data used without noise and to $\sigma = 0.1$ for simulated data used with noise.

We parallelized the code in such a way that the samples to be generated in one subpopulation are generated on one processor. Doing so, we keep the overhead of the parallelized code small, since the processors only have to communicate, if the migration operator is performed. However, this is the case in only 0.5% of the samples generated, i.e., in our case only the processors have only to communicate on average 1010 times during the sampling procedure.

Results with perfect data

The obtained results for our experimental design procedure with simulated data without noise are depicted in Figure 7.2. We performed 5 independent runs of our approach and show the

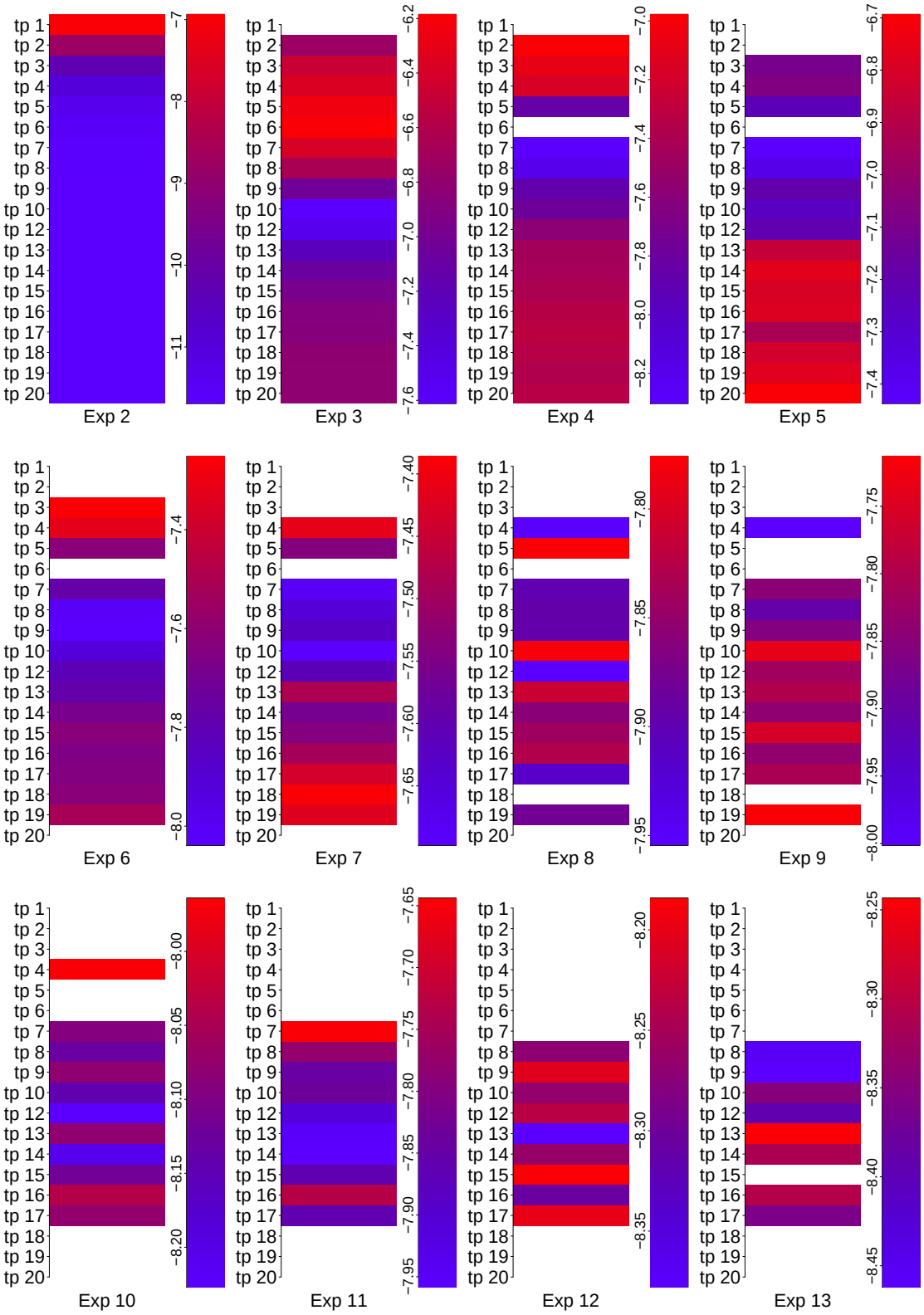


Figure 7.3.: *Entropy estimates for run* 1 *for perfect data (Part I)*

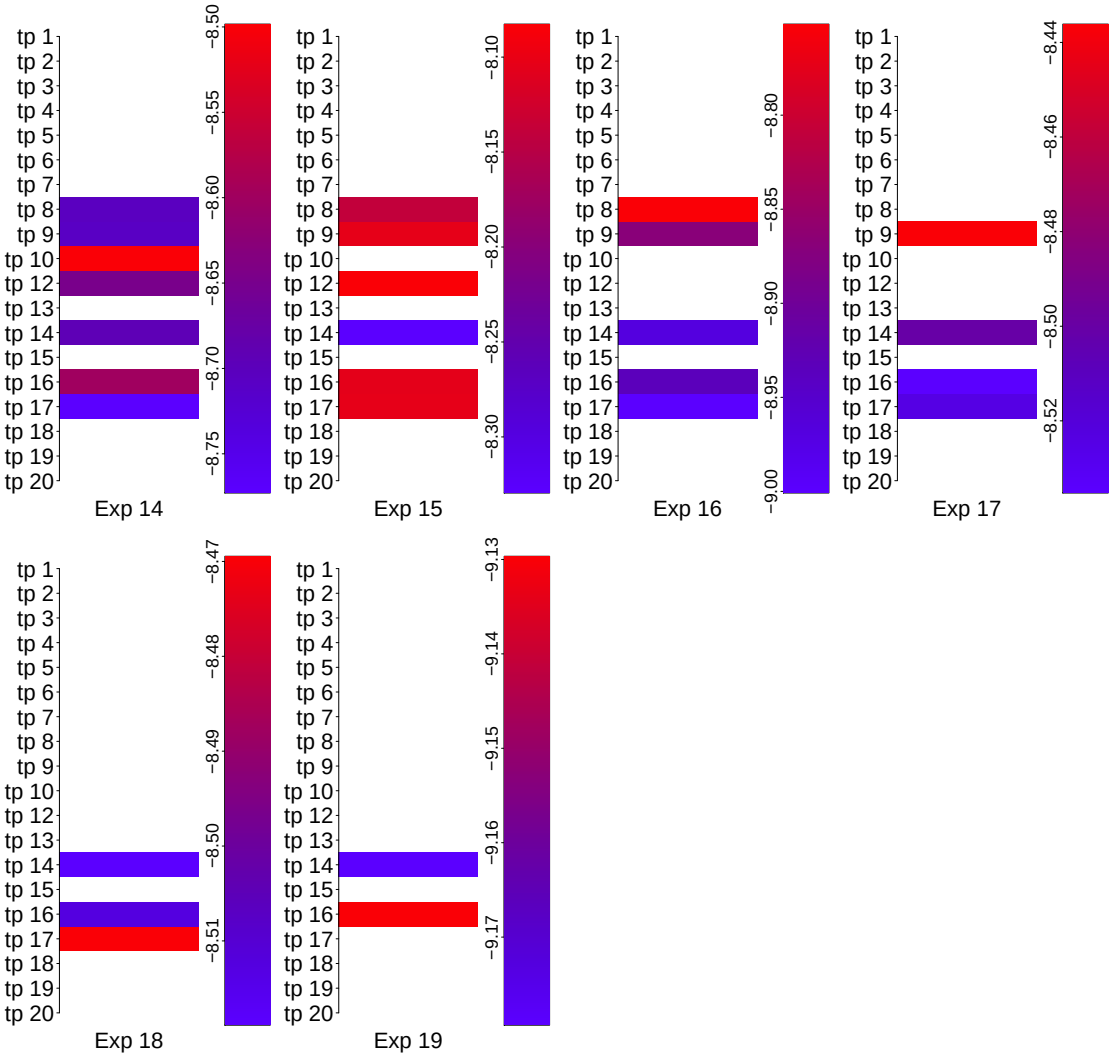


Figure 7.4.: *Entropy estimates for run* 1 *for perfect data (Part II)*

mean and the standard deviation of the obtained values for the information of the posterior distribution of the model parameters. We clearly see that our approach outperforms random experiments and after performing 4 additional experiments, the amount of information in the posterior distribution of the model parameters is the same as if we have used all available data.

It is worth to mention, that the amount of information in the posterior distribution of the model parameters is equal for experiment number 1. However, since we used for experiment #1 the same two data point, this is a desirable result. And it implies, that our MCMC approach really samples from the desired distribution and is robust, since the information content in the posterior distribution over the model parameters is very similar, i.e., the standard deviation is very low, for all independent 10^4 runs.

In Figures 7.3 and 7.4 the entropy estimates of the predictive distribution $p(d \mid D,\ M_e)$ for all possible experiments are depicted for the first independent run of our approach. The entropy estimates for the other four runs can be found in Appendix B in Figures B.1 – B.8. To be more precise, we show the values of the entropy estimates for all 19 time points, at which the concentration of all gene products may be measured as possible experiment. These estimates were used to specify the time point, where the entropy estimate is maximal and the obtained data was used in the corresponding experiment (the experiment number written under every subfigure). Of course, after performing an additional experiment and thus use the additional data in the parameter estimation step, no entropy estimate is given for that time point anymore. The notation of the number of the experiments is the same as in Figure 7.2: in experiment #1 no additional experiment was performed, in experiment #2 data measured at one additional time point is added, etc. Let us give a small example: in Figure 7.3 in the first subfigure for "Exp 2" the entropy estimates for all time points after finishing the sampling procedure for experiment #1 are shown and time point 1 (tp 1) shows the maximal entropy value. Thus, in experiment #2 the additional data obtained at time point 1 is added into the sampling procedure for parameter estimation. In the subfigure for "Exp 3" time point 1 is missing, since the data for is already included into the parameter estimation framework. The maximal entropy value in this subfigure is time point 6, which is not included anymore in the subfigure for "Exp 4", etc.

Results with noisy data

The results with simulated data with noise for our experimental design procedure are shown in Figure 7.5. The results look similar to the results obtained without noise, i.e., our approach outperforms random experiments. However, one has to perform 12 additional experiments to obtain the same amount of information in the posterior distribution of the model parameters as if we have used all available data. Nevertheless, the amount of information present in the posterior distribution after performing 5 experiments may be sufficient for practical issues.

It is worth to mention the fact that the expected result that the maximal amount of information contained in all available data points is smaller for noisy data compared to perfect data without noise.

In Figures 7.6 and 7.7 the entropy estimates for the first independent run of our approach of the predictive distribution $p(d \mid D,\ M_e)$ for all possible experiments are depicted, as was described in the previous subsection. The entropy estimates for the other four runs are shown in Appendix B in Figures B.9 – B.16.

[4] 5 runs with optimal experiments procedure and 5 runs with random experiments procedure

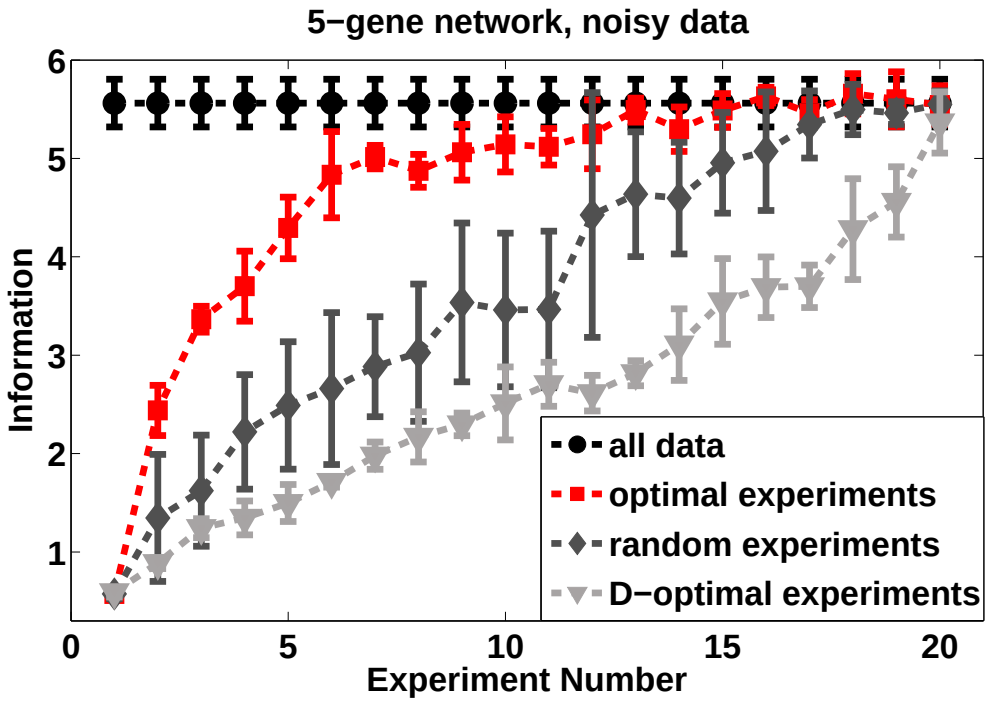


Figure 7.5.: *Results for the ODE model containing* 46 *parameters for gene network reconstruction with noisy data. For* 5 *independent runs the mean and the standard deviation of the information contained in the posterior distributions is depicted. The black line represents the runs with the maximal amount of data available. The dark grey line represents random experiments, the light grey line represents the results obtained with classical experimental design and the red line line illustrates optimal experiments.*

7.7.2. Experimental Data: The DREAM 2, Challenge #3 Dataset

We used the same data and preprocessed it in the same way as was done in Section 5.8.2. As for the case above with simulated data we did not use any prior distribution on the model parameters and generated samples from the posterior distributions $p(\omega|D_i)$, $i \in \{1, \ldots, 14\}$, using Algorithm 7.1 with the crossover operator as described in Algorithm 7.5 with the same parameters as we used it for the runs with simulated data.

In the first run of the parameter estimation sampling we started with two data points, i.e., we used the data of all genes at the time points $t = 0$ and $t = 8$. As possible experiments to perform we proposed to measure the concentration of all gene products at the 13 time points $(1, 2, 3, 4, 5, 6, 7, 9, 10, 11, 12, 13, 14)$. Thus, altogether we need to perform 14 parameter estimation steps.

Well, one parameter was different for the usage of real biological data, namely the noise parameter σ. We assumed here that the amount of noise in the data in the future experiments will be comparable to the amount of noise in the already available data. To be more precise, we calculated the median of the $5 \cdot (i+1)^5$ measurements available in the i-th parameter estimation step and set σ to this value.

The results with simulated data with noise for our experimental design procedure are shown in Figure 7.8. As for the simulated data, our experimental design approach outperforms random experiments. Moreover, after performing 6 experiments the amount of information in

[5] for experiment $\#i$ we use data at $(i+1)$ time points for all 5 genes

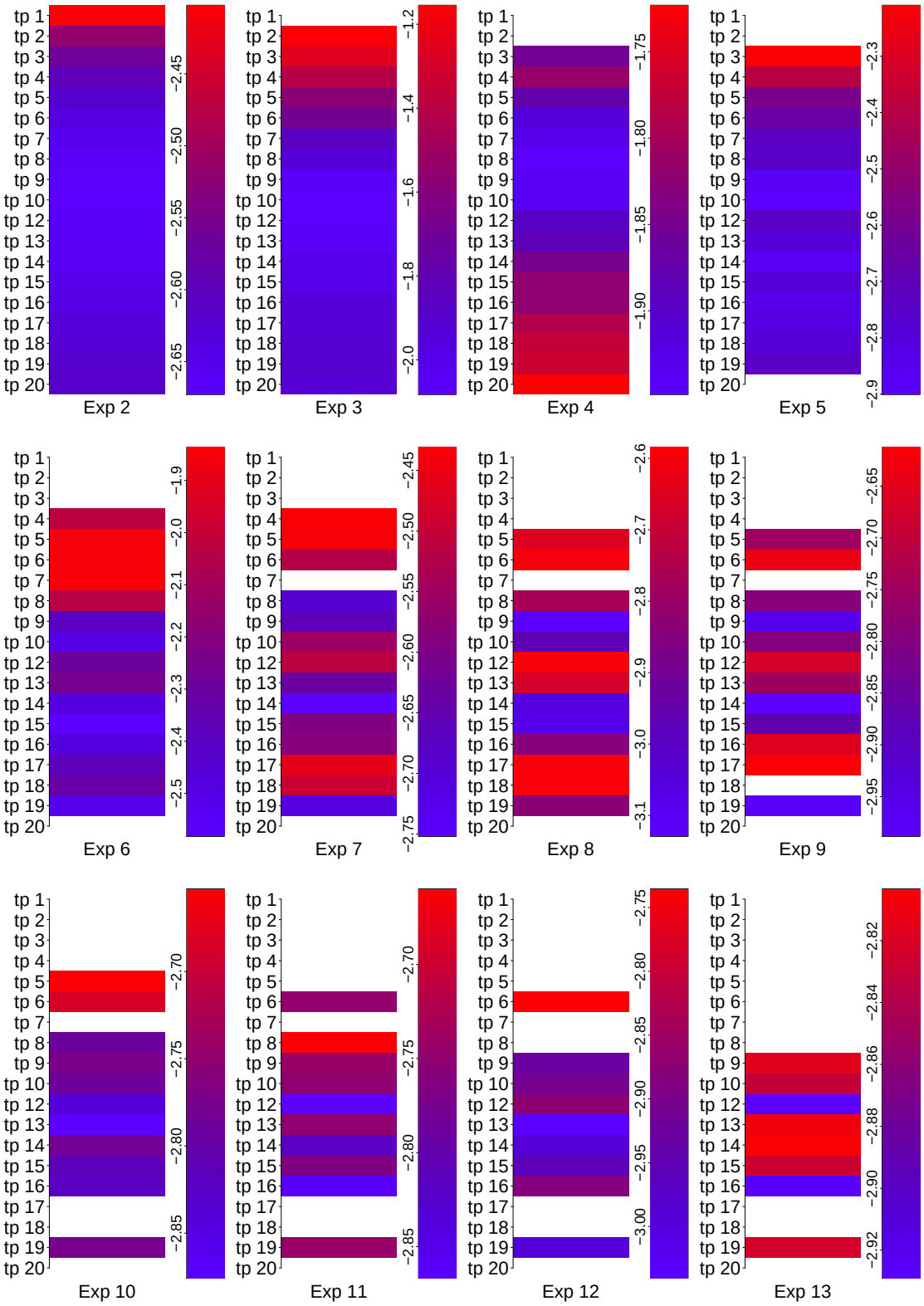


Figure 7.6.: *Entropy estimates for run* 1 *for noisy data (Part I)*

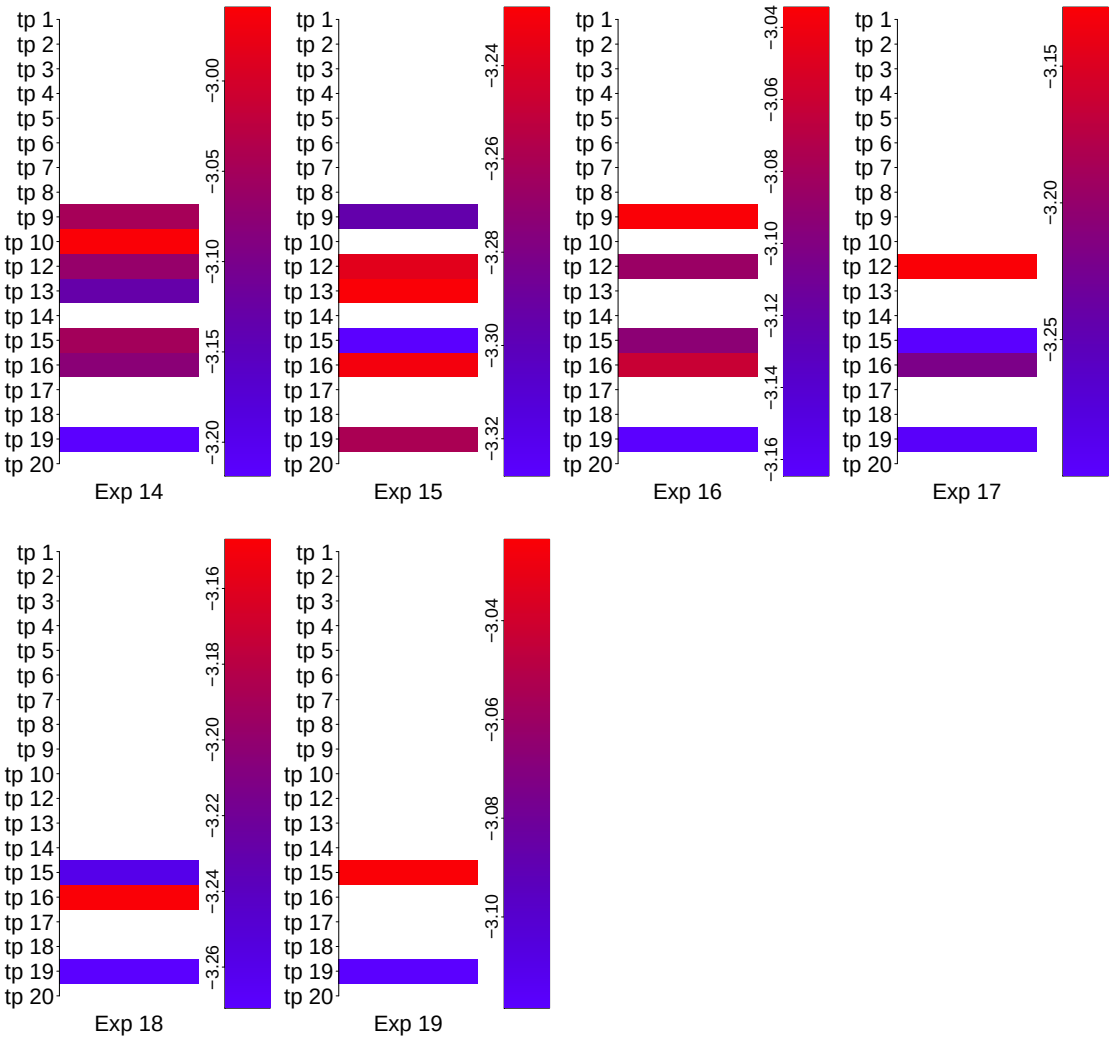


Figure 7.7.: *Entropy estimates for run* 1 *for noisy data (Part II)*

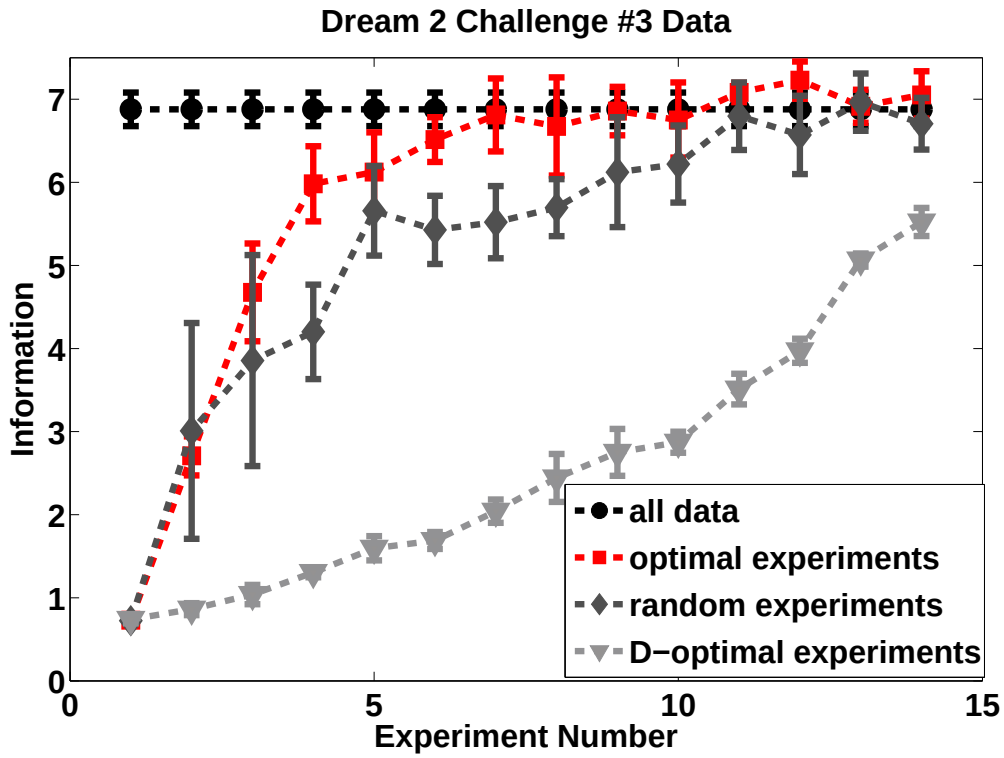


Figure 7.8.: *Results for the ODE model containing* 46 *parameters for gene network reconstruction with data from the DREAM* 2 *Challenge #*3 *initiative. For* 5 *independent runs the mean and the standard deviation of the information contained in the posterior distributions is depicted. The black line represents the runs with the maximal amount of data available. The dark grey line represents random experiments, the light grey line represents the results obtained with classical experimental design and the red line line illustrates optimal experiments.*

the posterior distribution of the model parameters is the same as if we have used all available data. Performing random experiments, the same amount of information is only obtained after the performance of 10 experiments.

In Figure 7.9 the entropy estimates for the first independent run of our approach of the predictive distribution $p(d \mid D,\ M_e)$ for all possible experiments are depicted, as was described in Section 7.7.1. The entropy estimates for the other four runs are shown in Appendix B in Figures B.17 – B.20.

7.7.3. Comparison with classical experimental design

You maybe already noticed that in Figures 7.2, 7.5 and 7.8 there is an additional curve that is annotated with "D-optimal experiments". Here we want to explain how the results generating this curve were calculated. To describe it first in a very general case, we can say that we performed classical experimental design where we linearized around a point estimate and minimized a scalar-valued function of the variance of this point estimate. We used the framework as described in Section 6.2.3 and in [ADT07] Chapter 17.8. We used the Matlab function `ode15s` to obtain the needed parameter sensitivities F_i, where $i \in \{1, \ldots, 5\}$ denotes the experimental outputs for the 5 genes present. The outputs were also called the *responses*

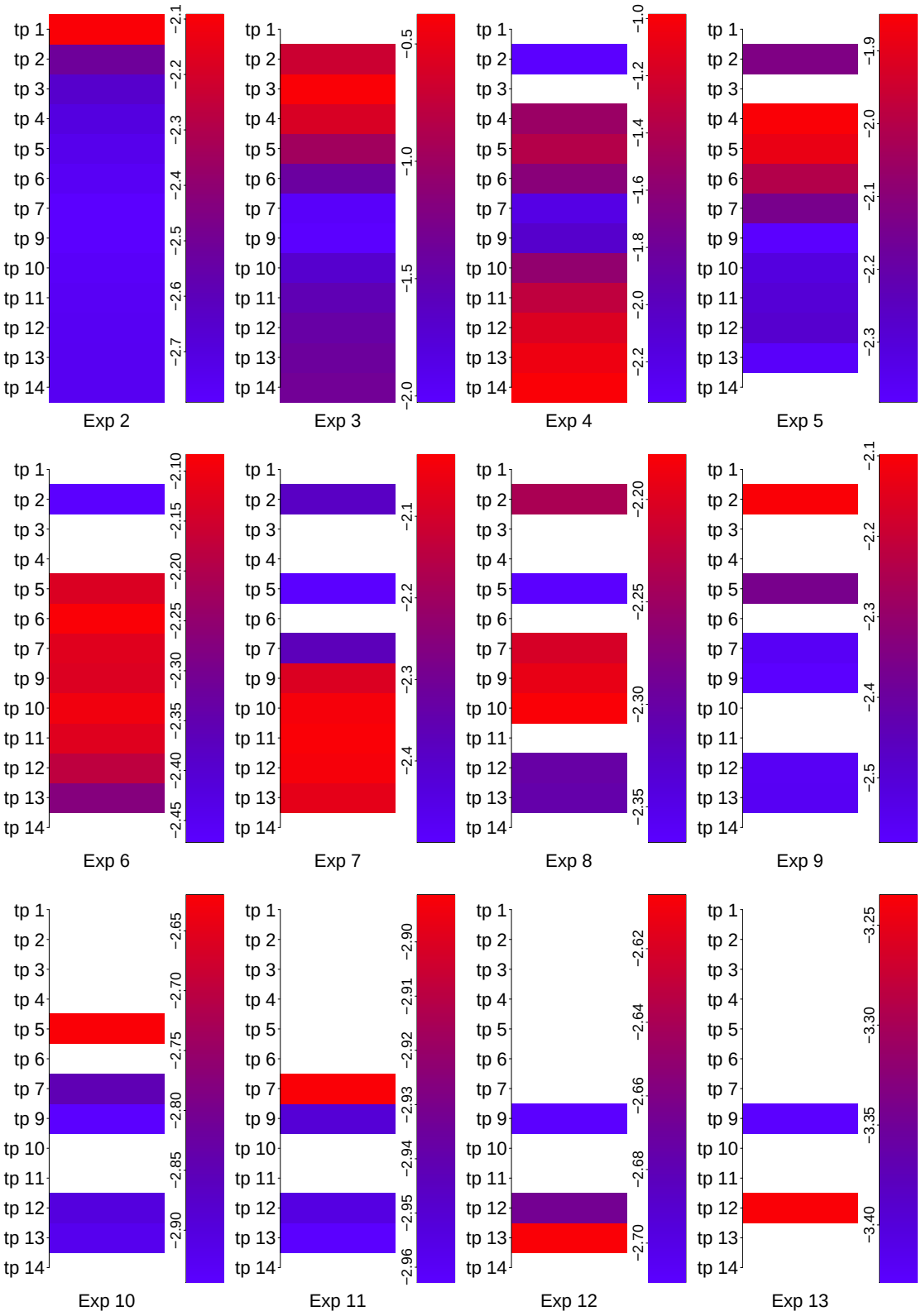


Figure 7.9.: *Entropy estimates for run* 1 *for the DREAM* 2 *Challenge #*3 *data*

in Chapter 6. Since we have five responses in our possible experiments, we use as information matrix the one that was introduced in (6.3). In our case, for a point estimate $\hat{p}$, we use the information matrix

$$M_{\hat{p}} = \sum_{i=1}^{5} \frac{1}{\sigma^2} (F_i)^T F_i,$$

since we assumed independent mean zero Gaussian noise with variance σ^2. As a point estimate $\hat{p}$ we use the set of parameters where the likelihood (7.1) is maximal for all the sampled parameter vectors.

The final decision which experiment is performed next is made by calculating for all possible time points where the gene products can be measured the determinant of the information matrix $M_{\hat{p}}$ and perform the experiment where $\det\big((M_{\hat{p}})^{-1}\big)$ was minimal, i.e., according to the definitions in Chapter 6 we use a D-optimality criteria. Since $M_{\hat{p}}$ in our case is underdetermined and thus singular, because we consider in our experimental design only one factor which gives five observations to specify 46 model parameters, we regularized it by adding a small multiple ϵ of the identity matrix Id, i.e., we used the information matrix

$$M_{\hat{p}}^{\epsilon} := M_{\hat{p}} + \epsilon \,\mathrm{Id}\,. \tag{7.11}$$

We set $\epsilon = 10^{-5}$. With this regularization the information matrix becomes invertible and we are able to calculate its determinant which now differs for different experiments. This type of regularization is described in [ADT07] in Chapter 10.3.

We see in all three cases, i.e., for the perfect data (see Figure 7.2), for the noisy data (see Figure 7.5) and also for the DREAM 2 Challenge #3 data (see Figure 7.8), that this classical experimental design procedure gives the worst results. It gives even worse results than the random choice of experiments. At first sight, these results seem strange, since a procedure for experimental design was performed. However, if we look in more detail at the experiments that were proposed with the classical experimental design, we see that for all five independent runs, independent of the used point estimate $\hat{p}$, the sequentially proposed experiments are to measure the gene product concentrations of all genes at the time points 20, 19, 18, 17, 16, 15, 14, 13, 12, 10, 9, 8, 7, 6, 5, 4, 3, 2 and 1 for the perfect and noisy simulated data and 14, 13, 12, 11, 10, 9, 7, 6, 5, 4, 3, 2 and 1 for the DREAM 2 Challenge #3 data. On the contrary, looking at the experiments proposed with our Bayesian experimental design procedure we can see in Figures 7.3, 7.4, 7.6, 7.7 and 7.9 in this chapter and in the Figures B.1 – B.20 in Appendix B that the most informative experiments to perform are, well not exclusively, the ones at early time points. And for the classical experimental design procedure the data obtained at these time points is added very late. In comparison, for random experiments, as the name implies, the experiments are chosen randomly, i.e., the probability to propose the measurement of all gene product concentrations at one specific time point is the same for all time points, and thus early time points will be chosen earlier in the sequential experimental design procedure.

7.8. Discussion

Relevance

The algorithmic approach used here for Bayesian experimental design shows that BED is possible for high-dimensional models based on non-linear ordinary differential equations, the

most often used type of models in the field of systems biology. In comparison to existing experimental design procedures, where the variance around a point estimate is minimized, the used method here considers the whole distribution of model parameters and suggests as next experiments the one where the information content in the whole posterior distribution will increase.

We have evaluated our approach on simulated and on real experimental data from a synthetic 5-gene regulatory network. We compared the amount of information present in the posterior distribution performing the experiments proposed with our experimental design procedure with the performance of random experiments and with the performance of experiments proposed with the classical experimental design procedure for ODE models described by Atkinson [ADT07] in Chapter 17.8[6]. For all tested datasets (perfect simulated data, noisy simulated data and real experimental data) we have shown that our experimental design procedure outperforms the usage of classical experimental design and the usage of random experiments to a high extent. Even more striking is the fact, that the used procedure to perform classical experimental design for ODE models is not really applicable to situations with this little amount of data and the information matrix used has to be regularized to become invertible and thus obtain a determinant different from zero.

The work presented here shows that with the simplification of maximum entropy sampling and the usage of population-based MCMC algorithms, Bayesian experimental design is possible even for general non-linear problems without the need of linearization around point estimates or the assumption of the distributions to follow a Gaussian distribution. Moreover, the DGMC algorithm enables the easy and efficient parallelization of the sampling procedure, since the overhead produced by the communication between the processors is intended to be small. Altogether, the technical limitations usually mentioned to perform general BED are solved and distributions over parameter values can be considered completely for experimental design purposes. And this is a desirable feature, because of the known problems of non-identifiabilities and sloppiness in parameter estimation tasks especially for problems tackled with non-linear ordinary differential equations models arising in systems biology [GWC+07, RKM+09].

Since maximum entropy sampling for Bayesian experimental design is, in principle, applicable to any kind of experiment[7] and to any ODE model, the algorithmical method proposed gives modelers and experimental biologists the possibility to accomplish the sequential task of mathematical modeling, experimental design and biological experiments as depicted in Figure 0.1 in the Introduction under consideration of distributions over model parameters and not only point estimates.

Limitations and Future Work

One clear limitation of this work is the long computational time it needs to obtain sufficient reliable samples from the distribution of interest. This is the usual unsolved problem of how to use Markov chain Monte Carlo algorithms in practice to obtain reliable samples from the desired distribution in an efficient manner. Issues concerning this problem where already argued in [MK11]. There are no general guidelines to deal with this problem and more theoretical and practical considerations have to be looked at in more detail. We argued in

[6] see also Section 6.2.3 in this thesis for a description

[7] perturbation experiments, knockdown experiments of certain components in the system under consideration, etc.

the previous section and in [MK11], that population-based MCMC algorithms are promising. As one step to be followed in the future will be to generate new efficient crossover steps. For example, one could propose new values $\tilde{x}$ using the gradients of the objective function and propose those values which are close to local minima of the objective function[8]. This needs more theoretical and algorithmical work to be performed, since *algorithmic differentiation*[9] has then to be used for general problems.

However, the known issues using MCMC algorithms is only half of the story. In this work, we additionally observed a lot of problems with the numerical integration. Although, we already used the function `ode15s` within Matlab which deals with stiff differential equations, we had to deal with numerical instabilities. Even worse, for some proposed parameter values in the DGMC algorithm the function `ode15s` was not able to integrate over the whole time interval. We assumed that these parameter values will be rejected anyway within the MCMC steps. We think that this is a reasonable assumption, since the solution of such stiff problems will not be accurate and will not describe the underlying dynamics anyway. However, even if this assumption is not valid, the cases where we do not have integrated values over the whole time interval are rare in the procedure, and tend to appear more often for noisy data, i.e., for the simulated data with noise and for the DREAM 2 Challenge #3 data. Furthermore, this problem appears much more often in the cases where only a few data points are available, i.e., for low experiment numbers. Altogether, we observed a lot of practical problems while using numerical integration approaches, which we already partially discussed in Subsection 2.4.3. To circumvent this, one possibility would be to avoid numerical integration and use an approach based on splines, as was done in Chapter 5, for parameter estimation. We point out here, that the numerical problems with our approach mainly originate from the numerical instabilities during numerical integration, which has to be performed in every single sampling step. Thus, this problem is the crucial one which has to be solved in the future, at least for high-dimensional models.

A further point to mention here is the fact, that we completely neglected the integration of prior knowledge into our BED method. Doing so will be a major point for future work, since a lot of biological knowledge is available and can and should be used to help with the underlying parameter estimation problem. However, doing so, one is interested in the Kullback-Leibler divergence between the prior distribution and the obtained posterior distribution and not in the information of the posterior distribution of the model parameters. Thus, the used utility function is no longer the information (7.5) but is the Kullback-Leibler divergence

$$U(d,e) = D_{\mathrm{KL}}\big(p(\omega|d,D,M_e) \parallel p(\omega)\big).$$

Thus, maximum entropy sampling seems not to be applicable directly to the case where prior knowledge is available. However, since it holds

$$\begin{aligned} D_{\mathrm{KL}}\big(p(\omega|d,D,M_e) \parallel p(\omega)\big) &= \int p(\omega|d,D,M_e) \log \frac{p(\omega|d,D,M_e)}{p(\omega)}\, d\omega \\ &= \mathrm{I}(\omega|d,D,M_e) - \int \log p(\omega)\, d\omega \end{aligned} \tag{7.12}$$

and the second term in (7.12) is independent of the experiment e, the same utility function as for the case without prior knowledge can be used which enables us to use maximum entropy

[8] compare to derivative-based optimization algorithms (see e.g. [NW06])
[9] see [GW08b]

sampling even in the case where prior knowledge is incorporated into the parameter estimation framework.

We compared our approach of Bayesian experimental design by usage of maximum entropy sampling only with the classical experimental design described by Atkinson [ADT07]. The recently proposed methods by Steinke *et al.* [SST07], Busetto and Buhmann [BOB09] and Kramer and Radde [KR10] use perturbation experiments to perform BED for non-linear ODE models. Thus, our method has to be adapted for perturbation experiments first to be able to see how our BED procedure compares with the existing methods using perturbation experiments. Moreover, the comparison with the very recent proposed general approaches for Bayesian experimental design by Terejanu *et al.* [TUM11] and by Huan and Marzouk [HM11] would be very interesting. However, both publications only show the applicability of their methods for a model with three resp. two parameters. It remains to be shown, if their approaches scale up to models with more parameters, like in our case to models with 46 parameters.

A more technical issue to consider will the impact of the entropy estimator used. Several other entropy estimators are available, i.e., [HS09,KHAR10], which where specially developed for the case where only a small amount of samples is available. It will be worth to examine, if the usage of another entropy estimator changes the proposed experiments in the BED procedure. Moreover, with these entropy estimators one could also examine the effect of generating less samples with the MCMC algorithm of the posterior distribution with our Bayesian experimental design procedure by means of maximum entropy sampling.

To decrease the amount of computational time needed several other possibilities from computer science have to be addressed. One point would be to parallelize the used code even more than was already done. The simplest possibility would be to run more Markov chains in parallel on more processors where each one produces less samples but together the number of generated samples from the posterior distribution remains the same. Another promising possibility is to run the code, instead on *central processor units* (CPUs), on *graphical processor units* (GPUs), which has already been done in computational problems arising in systems biology like *sequence alignment* [BL11,HKAA11] which leads to tremendous speedup of calculations in comparison to the usage of the same algorithms on CPUs.

As a last point it remains to mention, that although in this thesis the application is the inference of gene regulatory networks, the proposed sequential BED procedure can and should be applied to general high-dimensional ODE models, especially for the cases where the amount of data is limited and noisy. This is, of course, the case for modeling in systems biology but also for other complex models arising in nature like *climate modeling* or *economic modeling*. A further point would also be to extend the proposed method to be applicable to partial differential equations models, which not only need the temporal but also the spatial information to be modeled, like, for example, in *geologic modeling*.

Part IV.

Discussion and Outlook

CHAPTER 8

What we learned

> What is success? I think it is a mixture of having a flair for the thing that you are doing; knowing that it is not enough, that you have got to have hard work and a certain sense of purpose.
>
> *(Margaret Thatcher)*

8.1. Summary

This thesis addressed the challenging tasks of parameter estimation and experimental design for high-dimensional non-linear ODE models arising in systems biology and applied the obtained methods and algorithms to the inference of gene regulatory networks.

More precisely, Bayesian approaches for parameter estimation and experimental design were used and new algorithms were developed. Bayesian frameworks are especially suitable for problems arising in systems biology, as was discussed in the introduction and in [MK11], since the given data is limited and contains a lot of noise. And thus, one has to deal with technical problems like non-identifiabilities and sloppiness of parameters.

In this thesis, first an efficient method and algorithm for the parameter estimation task for ODE models describing gene regulatory networks is given in Chapter 5. It circumvents the need of time-consuming numerical integration in every sampling step of the parameters using the hybrid Monte Carlo algorithm. This is done using as objective function a least squares function that does not compare the measured data concentrations to the solution of the system of ordinary differential equation, but compares the derivative of the time-series data concentrations with the slopes predicted with the ODE model. The needed derivatives of the data concentrations are obtained by fitting a smoothing spline with a smoothing factor λ to the data. Since the results of this approach highly depend on the value for the derivatives of the experimental data and thus on the smoothing factor λ, an iterative approach for the sampling

of the model parameters and the smoothing factor is used. The usage of smoothing splines gives a tradeoff between accurate representation of the data and smoothing out the noise. Furthermore, the existence of a closed formula for the calculation of smoothing splines[1] makes the whole sampling procedure faster compared to the usage of numerical integration in every sampling step. The method was evaluated on both, simulated and real experimental data. For the simulated data, which consists of oscillations, we show that the developed method is able to reconstruct the underlying topology of the 5-gene regulatory network accurately, at least for low amounts of noise. However, for high amounts of noise and small number of data points the oscillations in the data break down and thus a successful reconstruction of the network is not possible anymore. As experimental data, the DREAM 2 Challenge #3 data was used. These data represent a synthetically designed 5-gene network with known topology in yeast and was generated by Cantone *et al.* [CMI+09] for the DREAM initiative. We outperform other approaches that were submitted to the DREAM 2 competition in the two categories DIRECTED-SIGNED-INHIBITORY and DIRECTED-UNSIGNED and obtained comparable results with the best submitted method for the DIRECTED-SIGNED-EXCITATORY category.

Second, in this thesis an efficient algorithm for the performance of Bayesian experimental design is given and applied to a high-dimensional non-linear ODE model describing gene regulatory networks in Chapter 7. We showed that with maximum entropy sampling it is possible to perform Bayesian experimental design for general distributions of model parameters and they do not have to follow a certain parametric form. Experiments are then selected such that the information in the posterior distribution will be maximally increased. Going the standard way to perform general BED, this would mean to have to solve triple integrals[2]. With the usage of maximum entropy sampling, this computationally non-tractable task reduces to the estimation of the entropy, being only one integral, of the distribution of the data that will be obtained after performance of this experiment. Then the experiment will be chosen where the entropy is maximal, i.e., an experiment will be performed where one knows the least. To obtain entropy estimates, one needs a high number of samples from the distribution of the model parameters. Reliable samples of the model parameters are obtained with the usage of a population-based Markov chain Monte Carlo algorithm which is parallelized to speed up the sampling procedure. The method was evaluated for simulated data as well for the DREAM 2 Challenge #3 data. It was compared to the choice of random experiments and the usage of the D-optimal classical experimental design for ODE models described in Atkinson [ADT07] Chapter 17.8. We show that the maximum entropy approach clearly outperforms the choice of random experiments as well as the classical experimental design method.

Altogether, this thesis provides two excellent approaches for the parameter estimation and the experimental design for high-dimensional non-linear ordinary differential equations models that arise in systems biology. The proposed Bayesian approaches consider the whole distribution over the model parameters and do not only concentrate on point estimates which was shown to be problematic because of non-identifiability and sloppiness problems for the obtained point estimates [GWC+07,RKM+09] and furthermore the possibility that the global optimum may not be reasonable from the biological point of view [SSC+10]. To show the figure of the sequential procedure for modeling biological processes in Figure 8.1 again, which was also shown in the introduction, we see that our method for the Bayesian parameter esti-

[1] see Section 2.5.2 for a reference for the closed formula
[2] see Section 7.2.1

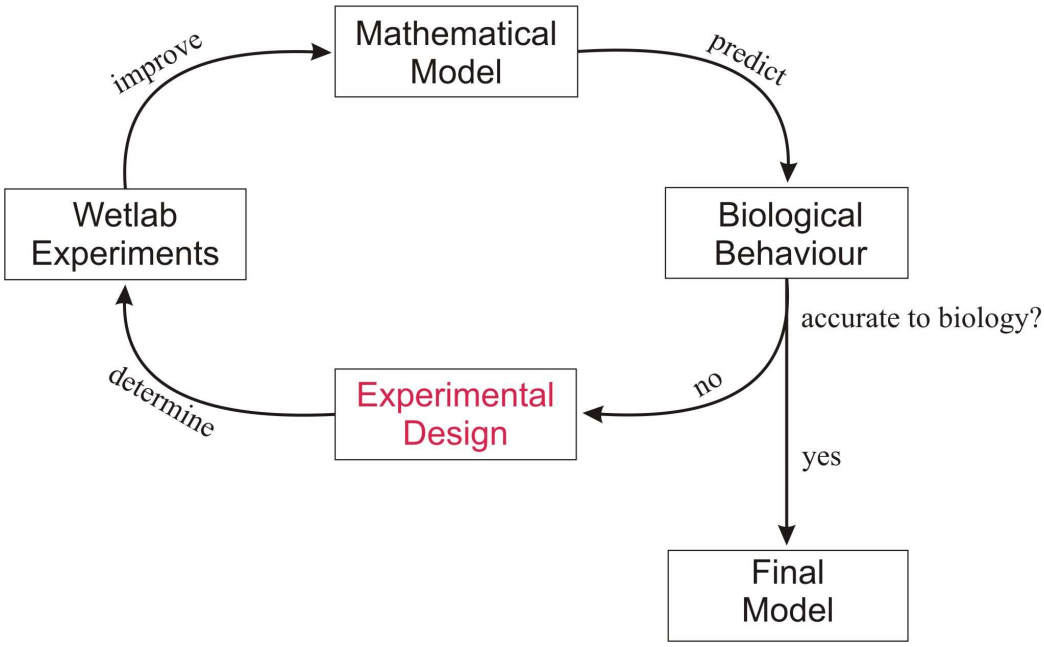


Figure 8.1.: *A sequential procedure for modeling biological processes.*

mation helps in the procedures indicated with the "improve" and "predict" arrows and our Bayesian experimental design approach describes the procedure in the "Experimental Design" box and helps in the work indicated with the "determine" arrow. Taken together, the methods presented here are involved in a large part of the sequential modeling process of biological systems.

8.2. Discussion

Having summarized the results and the impact of the research presented in this thesis, it remains to discuss the limitations and critical points about the presented work.

Let us start with the Bayesian experimental design procedure examined in Chapter 5. One striking fact was that the prior distribution for the smoothing factor has to be very strict, since otherwise the data will be overfitted and no noise will be smoothed out of the data. Also the priors for the threshold parameters have to be predefined with carefulness, since according to the model they are crucial to estimate the right directions, i.e., activation or inhibition, of the regulations between genes.

We observed also that especially the reconstruction from experimental data, although we outperformed other methods, is far from being perfect. This may be due to a dense population of local optima and picking a single optimal topology may be the wrong way to treat the obtained multi-modal distribution over model parameters.

For the Bayesian experimental design procedure there are several limitations that have to be pointed out. First, it cannot be applied for online experimental design, i.e., while a biological experiment is running, the optimal time point when to measure next is estimated. This is, of course, due to the long runtime of the sampling algorithms. The next big concern is the presence of numerical instabilities in the procedure, which is caused by the numerical

integration of the ODE system for little and noisy data. Furthermore, the sampling for a noise parameter in the model is not included, since an estimate for the amount of noise present in the used data is needed to apply maximum entropy sampling.

It seems also to be a drawback of the proposed method for experimental design that the set of experiments has to be fixed at the beginning and for every experiment the entropy of the predictive distribution has to be stored. Thus, the optimal experiment is found by *enumeration* over the whole search space which may of course lead for large search spaces to storage and runtime problems. However, for problems in systems biology there exist a lot of restrictions for the experiments that can be performed [KT09], e.g.,

1. only some of the species present in an ODE model can be measured
2. external perturbations that can be performed practically are limited

Altogether, the above mentioned point is not such a big concern, since the search space that is enumerated is reasonably small due to experimental constraints.

Compared to alphabetical classical experimental design our method is applicable and produces useful results, even if non-identifiabilities of the parameters occur in the model. In this situation the information matrix will be singular and regularization techniques has to be applied to obtain at least some results for the classical experimental design methods.

CHAPTER 9

What remains to be learned

> It is said that the present is pregnant with the future.
>
> *(Voltaire)*

Having discussed the relevance and the critical points of the scientific work presented in this thesis in the last chapter, this chapter will deal with the future directions of the presented methods and algorithms.

For the Bayesian parameter estimation procedure presented in Chapter 5 the future work should include the exhaustive study of the impact of the hyperparameters used for the prior distributions on the obtained results. Furthermore, the used iterative sampling of model parameters and the smoothing factor should be compared with the usage of smoothing splines as used by Poyton *et al.* [PVM$^+$06] and Ramsay *et al.* [RHCC07]. In their work the model predictions are fed back into the spline estimation step, whereas we only use as penalty term the second derivative of the spline. The investigation of this difference and its impact for the inference procedure is of great interest. In addition, the noise present in the data should be incorporated into the learning framework. Right now, this information is completely neglected and not used in the reverse engineering task.

The Bayesian experimental design framework presented in Chapter 7 offers a lot of directions for future research. One obvious direction will be to avoid numerical integration of the ODE system in every sampling step, since this causes a lot of numerical instabilities especially for the cases where only little data are available. One possibility to avoid numerical integration can be the usage of smoothing splines as for the Bayesian parameter estimation method in Chapter 5. This may solve the numerical instabilities and also speed up the experimental design procedure. Another direction of future research can the usage of another entropy estimator, especially those that are developed for the case where only a small number of samples for a distribution are available [HS09,KHAR10]. This step can also help with speeding up the Bayesian experimental design procedure. Finally, one last big topic for future work will be the adaptation of the BED approach to other experiments, like perturbation experiments. Thus,

our method could also be compared with other presented methods for Bayesian experimental design that all use perturbation experiments as potential experiments to be performed.

And since the results for Bayesian methods using Markov chain Monte Carlo algorithms can only be as good as the obtained samples for the distribution under investigation, a necessary and demanding goal for future research is to find more efficient sampling algorithms. And especially for the population-based MCMC algorithm used for the Bayesian experimental design approach new crossover operators, which are able to explore the search space more efficiently, would be of high interest.

In summary, the main point for future work, apart from the application to other high-dimensional ODE systems and other experiments, is the reduction of the computational runtime. Several theoretical and algorithmical mathematical possibilities how this can be done were suggested. However, more detailed knowledge about efficient programming practice and more efficient devices like the usage of GPUs has to be explored and used for speeding up the whole process of the proposed Bayesian experimental design procedure.

A more general topic to be considered in the future will be the question, how the huge generated data sets with both presented methods in this thesis, i.e., the samples generated of the posterior distribution and the data that had to be stored for the BED procedure to be able to calculate the desired entropies, can be further used and analyzed. Both data sets, the posterior distribution of the model parameters as well the distribution of the predictive distribution may be used to explore in more detail all modes of these distributions. The posterior distribution will then give us all possible topologies of the underlying gene regulatory network that are consistent with the data. Or, if used on other ODE models, the modes will give all parameter sets that are consistent with the data. The modes of the predictive distribution may be potentially used to analyze the dynamical system under consideration in more detail. These distributions can be used to get a feeling for which parameter ranges the dynamics of the system does not change at all, or for which parameter ranges the dynamics vary a lot even for small parameter changes. To formulate this in other words: one may perform a *sensitivity analysis* of the system under consideration. Altogether, how this can be formulated precisely and proven mathematically is an interesting topic for future work.

> Now this is not the end. It is not even the beginning of the end. But it is, [...], the end of the beginning.
>
> Winston Churchill

Part V.

Appendices

APPENDIX **A**

Proof of value for random guessing for three-class ROC

I enjoy learning technical details.

(Ken Follett)

In classical two class receiver operator characteristics analysis the area under curve value gives a measure of how well the two classes are separated. In our case this approach is not suitable, since we want to distinguish three classes and evaluate how well they are predicted. We now describe the confusion matrix underlying our analysis as well as our method for the generation of ROC points and the calculation of the AUC value. After recalling what the AUC value is for guessing in the two class ROC analysis, we calculate the AUC value for guessing in our three class ROC analysis for sensitivity vs. 1−specificity and precision vs. recall. For more information concerning receiver operator characteristics, see [Faw06].

A.1. Confusion matrix

In Table A.1 the confusion matrix of our three class problem is denoted. An edge is denoted as true positive (TP), if it is a positive or negative link and predicted also as a positive or negative link, respectively. False positives (FP) are all predicted positive or negative links which are not correctly predicted, i.e., either they are non-existent or they have another sign in the reference network. As true negatives (TN) we denote correctly predicted non-existent edges and as false negatives (FN) falsely predicted non-existent edges are defined, i.e., an edge is predicted to be non-existent, but it is a positive or a negative link in the reference network.

		predicted		
		positive link	negative link	non-existent link
actual	positive link	TP	FP	FN
	negative link	FP	TP	FN
	non-existent link	FP	FP	TN

Table A.1.: *Mapping of three-class classification problem (no edge present, positive regulation, negative regulation) onto two-class* ROC / PR *evaluation.*

A.2. ROC point generation and AUC Calculation

Let B be the set of the predicted interaction parameters, i.e., the edge weights. Further denote by $\beta_{\max}$ the maximum of the absolute values of B. Now we choose a precision factor p and calculate the step size $s = \beta_{\max}/p$. Then, for a varying threshold from zero to $\beta_{\max}$ by adding the step size s we obtain p thresholds. For each of these thresholds we set the predicted interaction parameters to zero for values inside the interval [- threshold, threshold]. Then we calculate sensitivity, specificity and precision according to the confusion matrix denoted in Table A.1 for each of the p thresholds. Then the AUC value is calculated as the integral under the curve described by these points.

A.3. AUC value for guessing for two class problem

For the two class problem the AUC value for guessing for sensitivity vs. 1−specificity is known to be 0.5. Assume that one has randomly guessed the edges of a network. If you now have a classifier which gives you sensitivity of 0.8, i.e., 80 percent of true existing edges are found with our classifier, then you also expect to have 1−specificity of 0.8, i.e., 80 percent of the predicted existing edges are not present in the true network.

The AUC value for guessing for precision vs. recall is the ratio of present edges in the gold standard divided by the possible edges in the gold standard for varying recall between zero and 1. (See also Section 3.3.1.)

A.4. AUC value for guessing for three class problems

First, we introduce as notations $p(+_L)$ for the probability of a positive learned edge, $p(-_L)$ for the probability of a negative learned edge and $p(0_L)$ for the probability of a non-existent learned edge. Furthermore, we denote by ratio(+), ratio(−) and ratio(0) the ratio between the positive, negative resp. zero edges in the true network. For example, if in the true network there are one third positive edges, one third negative edges and one third zero edges, then ratio(+) = ratio(−) = ratio(0) = 1. If two third of the edges are zero edges and one third of the edges are positive, then ratio(0) = 2, ratio(+) = 1 and ratio(−) = 0. Note, that we always have

$$\text{ratio}(0) + \text{ratio}(+) + \text{ratio}(-) = 3 \tag{A.1}$$

in our three class problem.

Remark A.4.1. With the method for the ROC point generation described in Section A.2 we have two properties of the above defined probabilities:

1. Since we take a symmetric threshold variation, we want to have for randomly generated edge weights
$$p(+_L) = p(-_L). \tag{A.2}$$
2. The aim is now to give a function which depends on $p(+_L)$, $p(-_L)$, $p(0_L)$ and ratio(0), ratio(+), ratio(−), which gives us the points in the ROC graph by varying $p(0_L)$ from zero to one or analogously with (A.2) by varying $p(+_L)$ or $p(-_L)$ from zero to one half.

Furthermore, denote by $p(\mathrm{TP}\,|+_L)$ the conditional probability for a TP under the condition, that an edge was learned as positive. Similarly, for all other combinations of TP, FP, TN, FN and $+_L$, $-_L$, 0_L. Now assume that one generates randomly with the uniform distribution weights on the edges. Thereby we assume, that we have infinite many edges to ensure that $p(+_L) = p(-_L)$ and therefore with Table A.1 and our assumptions we have

$$\begin{aligned}
p(\mathrm{TP}\,|+_L) &= \frac{\mathrm{ratio}(+)}{3} \cdot p(+_L) \\
p(\mathrm{TP}\,|-_L) &= \frac{\mathrm{ratio}(-)}{3} \cdot p(+_L) \\
p(\mathrm{FP}\,|+_L) &= \frac{\mathrm{ratio}(-) + \mathrm{ratio}(0)}{3} \cdot p(+_L) \\
p(\mathrm{FP}\,|-_L) &= \frac{\mathrm{ratio}(+) + \mathrm{ratio}(0)}{3} \cdot p(+_L) \\
p(\mathrm{FN}\,|0_L) &= \frac{\mathrm{ratio}(+) + \mathrm{ratio}(-)}{3} \cdot p(0_L) \\
p(\mathrm{TN}\,|0_L) &= \frac{\mathrm{ratio}(0)}{3} \cdot p(0_L).
\end{aligned} \tag{A.3}$$

Furthermore, we have

$$\begin{aligned}
0 &= p(\mathrm{TP}\,|0_L) = p(\mathrm{FP}\,|0_L) \\
&= p(\mathrm{TN}\,|+_L) = p(\mathrm{FN}\,|+_L) \\
&= p(\mathrm{TN}\,|-_L) = p(\mathrm{FN}\,|-_L)
\end{aligned} \tag{A.4}$$

and with the classification in Table A.1 and together with (A.4) we obtain

$$\begin{aligned}
p(\mathrm{TP}) &= p(\mathrm{TP}\,|+_L) + p(\mathrm{TP}\,|-_L) \\
p(\mathrm{FP}) &= p(\mathrm{FP}\,|+_L) + p(\mathrm{FP}\,|-_L) \\
p(\mathrm{TN}) &= p(\mathrm{TN}\,|0_L) \\
p(\mathrm{FN}) &= p(\mathrm{FN}\,|0_L).
\end{aligned} \tag{A.5}$$

With (A.2) we have

$$\begin{aligned}
1 &= p(+_L) + p(-_L) + p(0_L) \\
&= 2p(+_L) + p(0_L).
\end{aligned} \tag{A.6}$$

A. Proof of value for random guessing for three-class ROC

To calculate sensitivity, specificity and precision we need to know the probabilities for true and false positives as well as for true and false negatives. With (A.3), (A.5) and (A.6) we get

$$\begin{aligned} p(\mathrm{TP}) &= \frac{\mathrm{ratio}(+) + \mathrm{ratio}(-)}{3} \cdot p(+_L) \\ p(\mathrm{FP}) &= \frac{2\,\mathrm{ratio}(0) + \mathrm{ratio}(+) + \mathrm{ratio}(-)}{3} \cdot p(+_L) \\ p(\mathrm{FN}) &= \frac{\mathrm{ratio}(+) + \mathrm{ratio}(-)}{3} \cdot p(0_L) = \frac{\mathrm{ratio}(+) + \mathrm{ratio}(-)}{3} \cdot \big(1 - 2p(+_L)\big) \\ p(\mathrm{TN}) &= \frac{\mathrm{ratio}(0)}{3} \cdot p(0_L) = \frac{\mathrm{ratio}(0)}{3} \cdot \big(1 - 2p(+_L)\big). \end{aligned} \tag{A.7}$$

For sensitivity we now obtain

$$\text{sensitivity} \;= \frac{p(+_L)}{1 - p(+_L)},$$

analogously, for 1−specificity with (A.1)

$$1 - \text{specificity} \;= \frac{\big(\mathrm{ratio}(0) + 3\big) \cdot p(+_L)}{\big(3 - \mathrm{ratio}(0)\big) \cdot p(+_L) + \mathrm{ratio}(0)}$$

and for precision

$$\text{precision} \;= \frac{\mathrm{ratio}(+) + \mathrm{ratio}(-)}{6}. \tag{A.8}$$

We now obtain directly that the AUC value for guessing for the precision vs. recall curve equals the right hand side of (A.8), since it does not depend on $p(+_L)$.

Remark A.4.2.

1. The AUC value for guessing for precision vs. recall curve varies between zero and one half. It is zero, if the true network only contains zero edges, i.e., $\mathrm{ratio}(0) = 3$, and it is one half, if the true network does not contain any zero edges, i.e., $\mathrm{ratio}(0) = 0$.
2. The AUC value for guessing for precision vs. recall only depends on the ratio for the zero edges because of (A.1) and *not* on the proportion of the ratios for the negative and positive edges.

To obtain the AUC value for guessing for the sensitivity vs. 1−specificity curve we have to calculate the area under the curve whose graph G can be written as

$$\begin{aligned} G &= \left\{ \left(\frac{\big(\mathrm{ratio}(0) + 3\big) \cdot a}{\big(3 - \mathrm{ratio}(0)\big) \cdot a + \mathrm{ratio}(0)}, \frac{a}{1 - a} \right) \;;\; a \in \left[0, \frac{1}{2}\right] \right\} \\ &= \left\{ \left(x, \frac{\mathrm{ratio}(0) \cdot x}{\mathrm{ratio}(0) + 3 - 3x} \right) \;;\; x \in [0, 1] \right\}. \end{aligned}$$

With integration by parts we now obtain

$$\begin{aligned} \int_0^1 \frac{\mathrm{ratio}(0) \cdot x}{\mathrm{ratio}(0) + 3 - 3x} dx = \frac{\mathrm{ratio}(0)}{\mathrm{ratio}(0) + 3} \Bigg(& -\frac{1}{9} \ln\Big(\frac{\mathrm{ratio}(0)}{\mathrm{ratio}(0) + 3}\Big) \cdot \mathrm{ratio}(0)^2 \\ & - \frac{2}{3} \ln\Big(\frac{\mathrm{ratio}(0)}{\mathrm{ratio}(0) + 3}\Big) \cdot \mathrm{ratio}(0) \\ & - \ln\Big(\frac{\mathrm{ratio}(0)}{\mathrm{ratio}(0) + 3}\Big) - \frac{\mathrm{ratio}(0)}{3} - 1 \Bigg). \end{aligned}$$

Remark A.4.3. 1. The AUC value for guessing for sensitivity vs. 1−specificity curve varies between zero and approximately 0.39. It is zero, if the true network does not contain any zero edges, i.e., ratio(0) = 0, and it is $2\ln 2 - 1 \approx 0.39$, if the true network only contains zero edges, i.e., ratio(0) = 3.

2. As in the case for guessing for precision vs. recall, the AUC value for guessing for sensitivity vs. 1−specificity only depends on the ratio for the zero edges and *not* on the proportion of the ratios for the negative and positive edges.

Remark A.4.4. By using our notation we will now consider the classical two-class problem. By $p(+_L)$ we now denote the probability of a learned positive and with $p(0_L)$ we denote the probability of a learned negative. Furthermore, we denote by ratio(+) the ratio for the positives in the true network and with ratio(0) the ratio for the negatives in the true network. Note that we here have

$$\text{ratio}(+) + \text{ratio}(0) = 2.$$

Then we have for random generated edges

$$\begin{aligned} p(\text{TP}) &= \frac{\text{ratio}(+)}{2} \cdot p(+_L), \\ p(\text{FP}) &= \frac{\text{ratio}(0)}{2} \cdot p(+_L), \\ p(\text{TN}) &= \frac{\text{ratio}(0)}{2} \cdot p(0_L), \\ p(\text{FN}) &= \frac{\text{ratio}(+)}{2} \cdot p(0_L). \end{aligned}$$

As in Section A.3 we obtain sensitivity = $p(+_L)$, 1−specificity = $p(+_L)$ and precision = 0.5. Thus, we also get the AUC values of 0.5 for sensitivity vs. 1−specificity and ratio(+)/2 for precision. It is important to note here, that, different from our three class ROC analysis, the AUC value for sensitivity vs. $1 -$ specificity does not depend neither on ratio(+) nor on ratio(0).

APPENDIX B

Evaluation of Bayesian experimental design for 5-gene network

A good picture is equivalent to a good deed.

(Vincent van Gogh)

B.1. Simulated perfect data

In this section the evaluation pictures for the runs 2 - 5 performed for the Bayesian experimental design scheme presented in Chapter 7 for the perfect simulated data are shown in Figures B.1 – B.8. Depicted are the entropy estimates as was described in Chapter 7.

B.2. Simulated noisy data

In this section the evaluation pictures for the runs 2 - 5 performed for the Bayesian experimental design scheme presented in Chapter 7 for the noisy simulated data are shown in Figures B.9 – B.16. Depicted are the entropy estimates as was described in Chapter 7.

B.3. DREAM 2 Challenge #3 data

In this section the evaluation pictures for the runs 2 - 5 performed for the Bayesian experimental design scheme presented in Chapter 7 for the DREAM 2 Challenge #3 data are shown in Figures B.17 – B.20. Depicted are the entropy estimates as was described in Chapter 7.

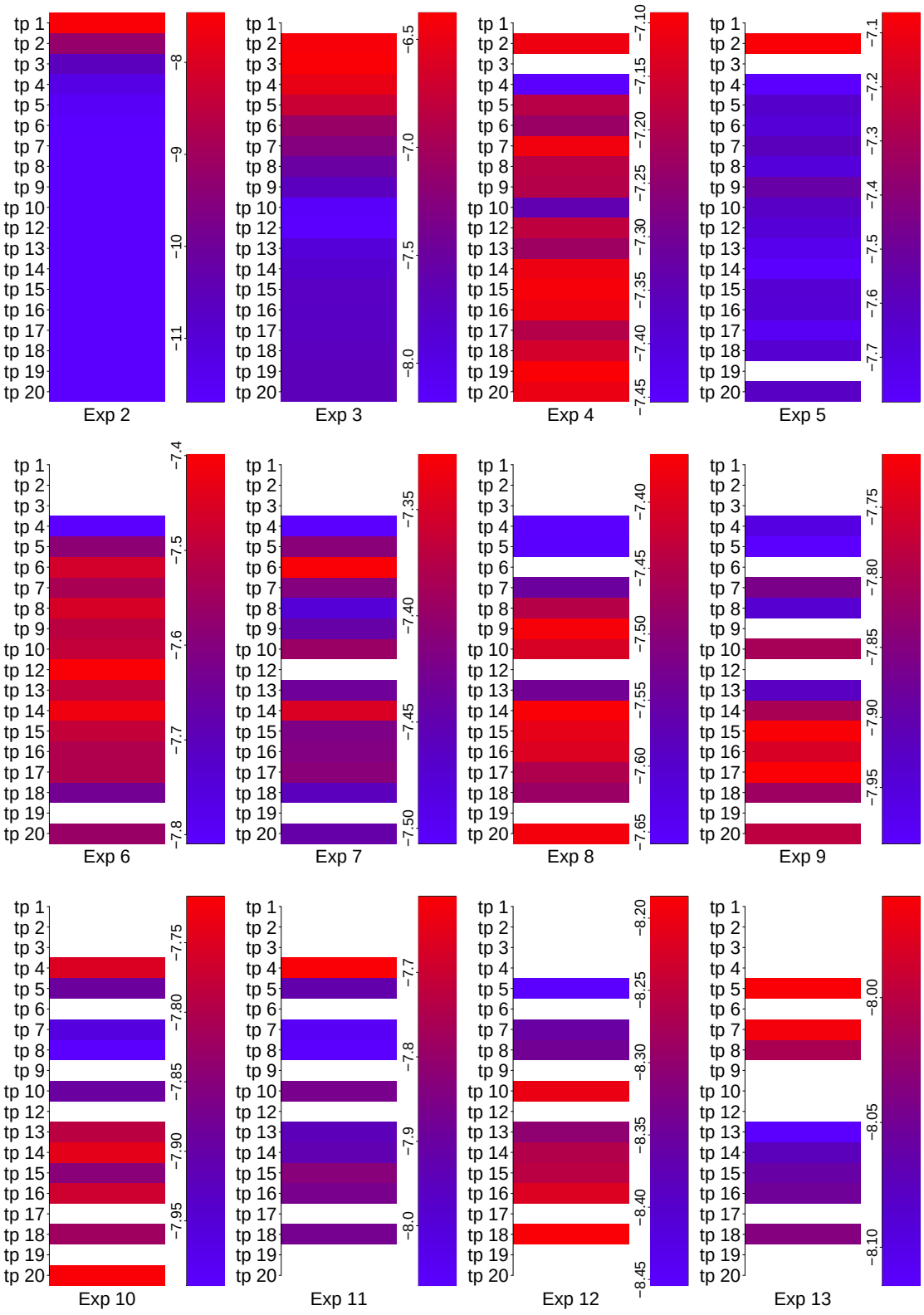


Figure B.1.: *Entropy estimates for run 2 for perfect data (Part I)*

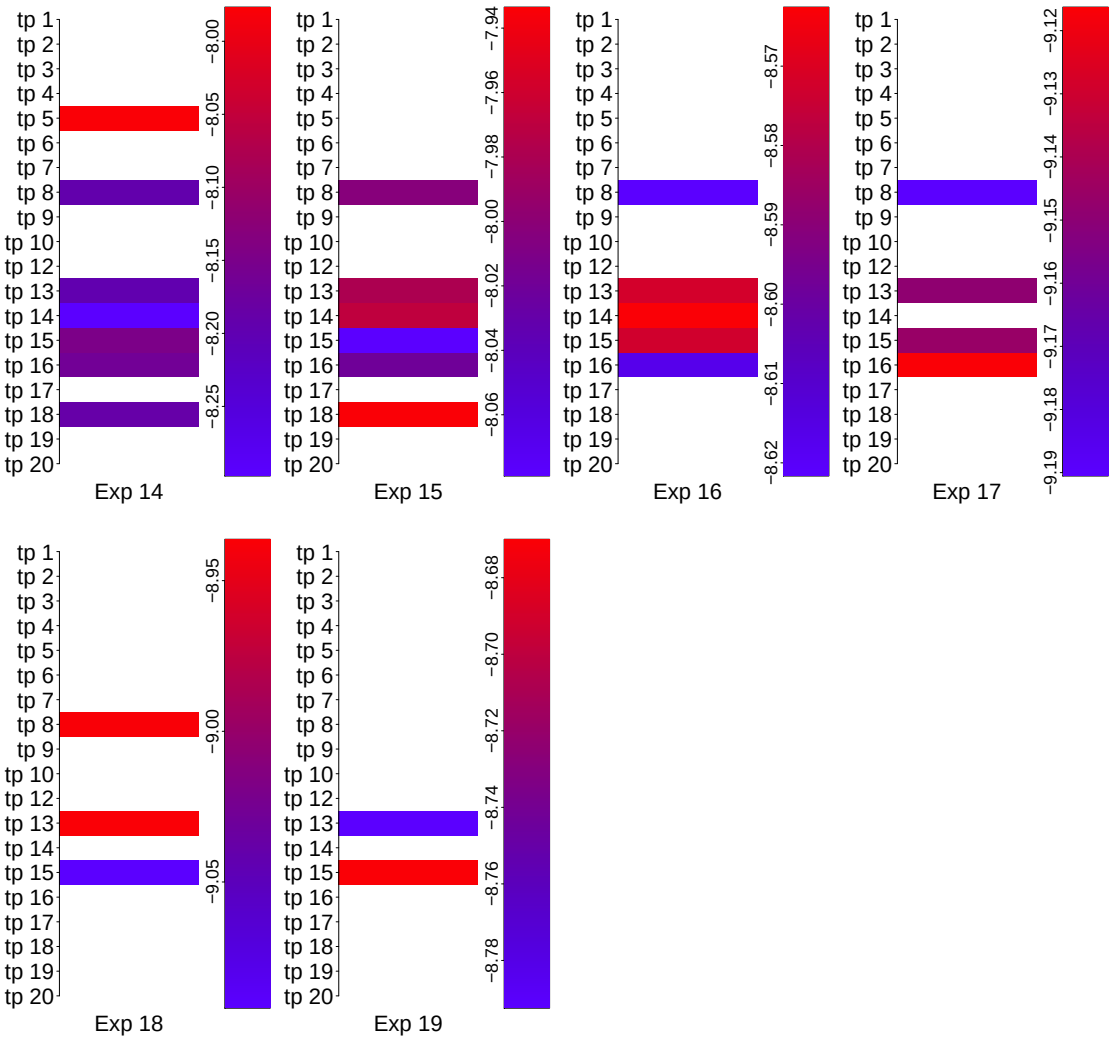


Figure B.2.: *Entropy estimates for run 2 for perfect data (Part II)*

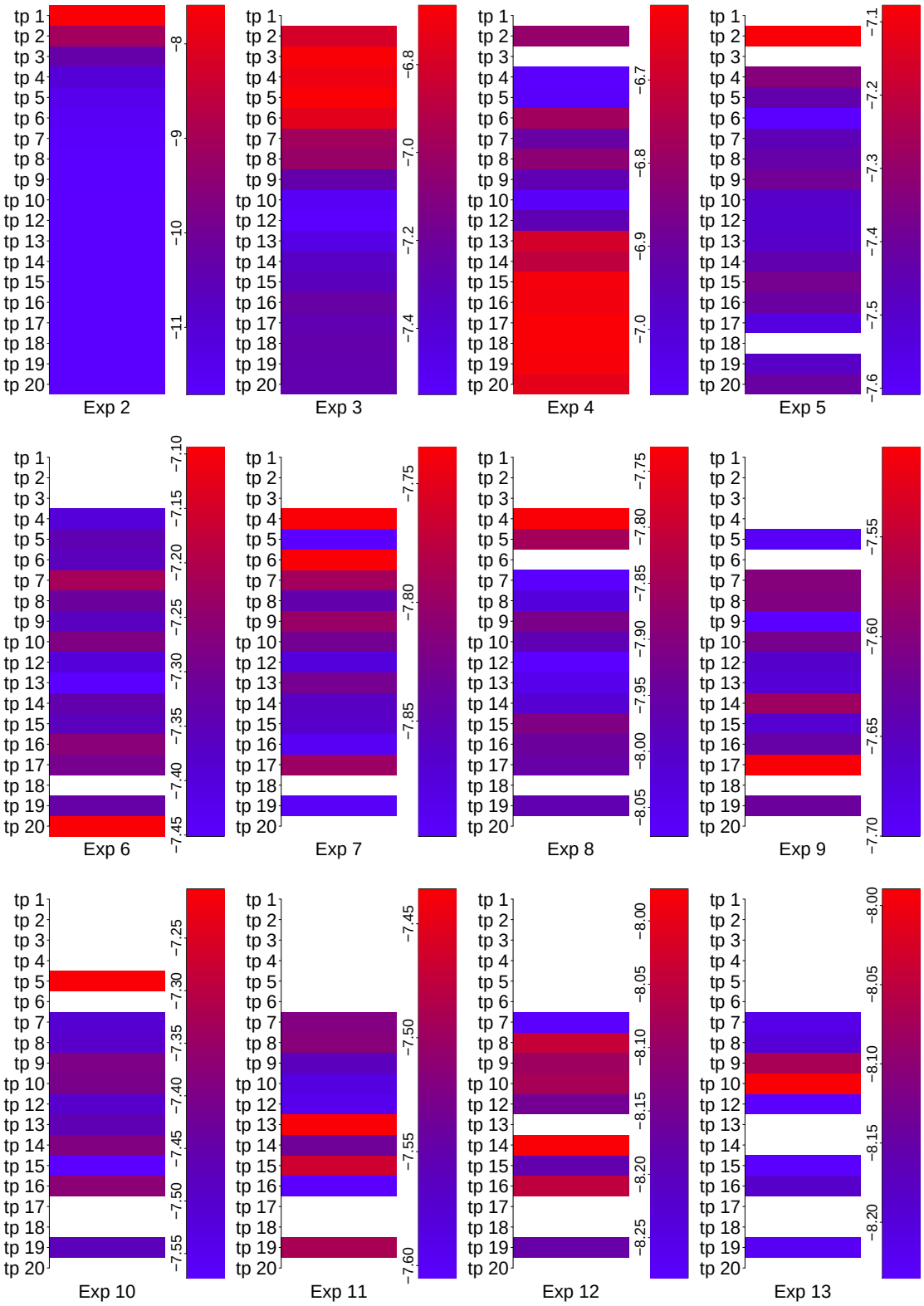


Figure B.3.: *Entropy estimates for run 3 for perfect data (Part I)*

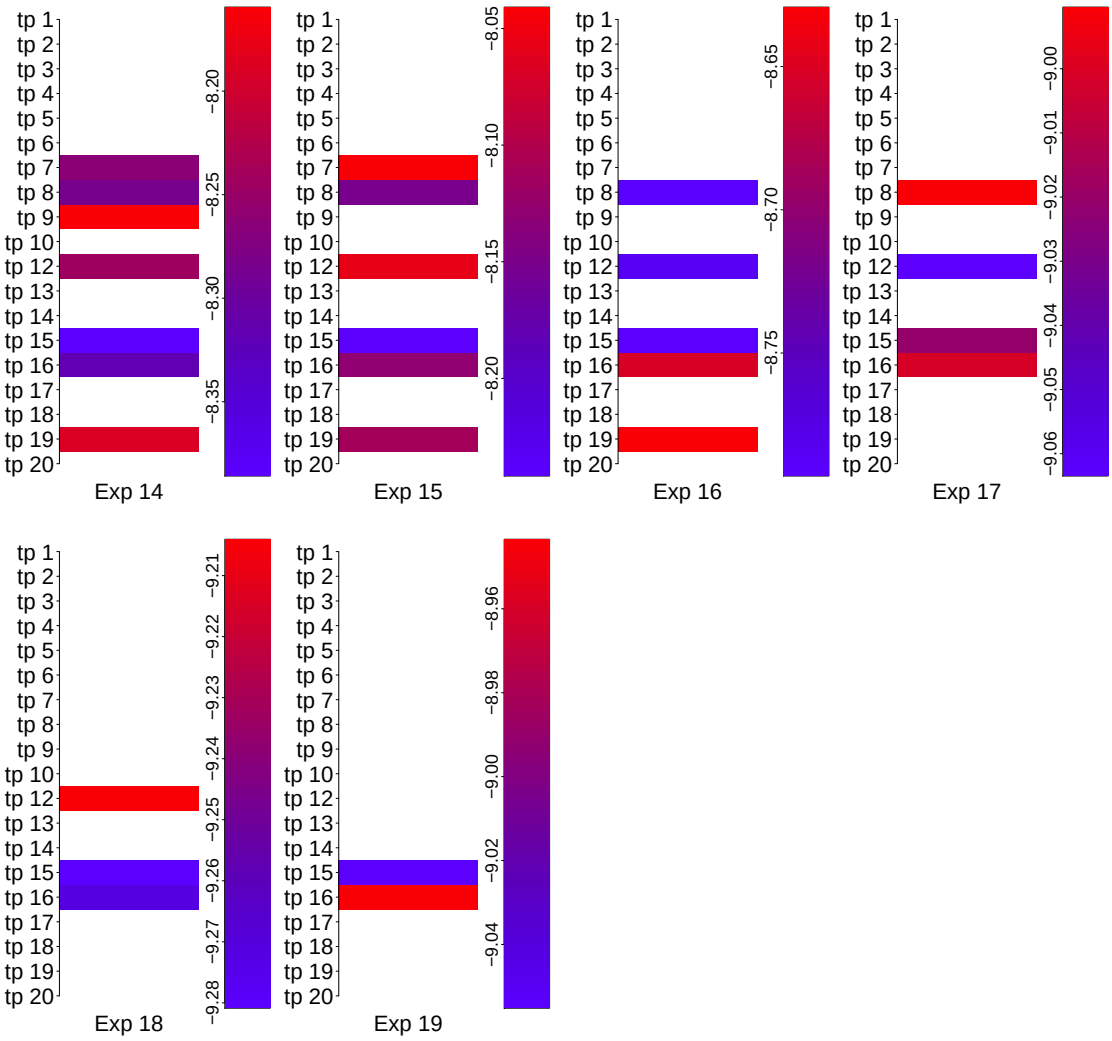


Figure B.4.: *Entropy estimates for run 3 for perfect data (Part II)*

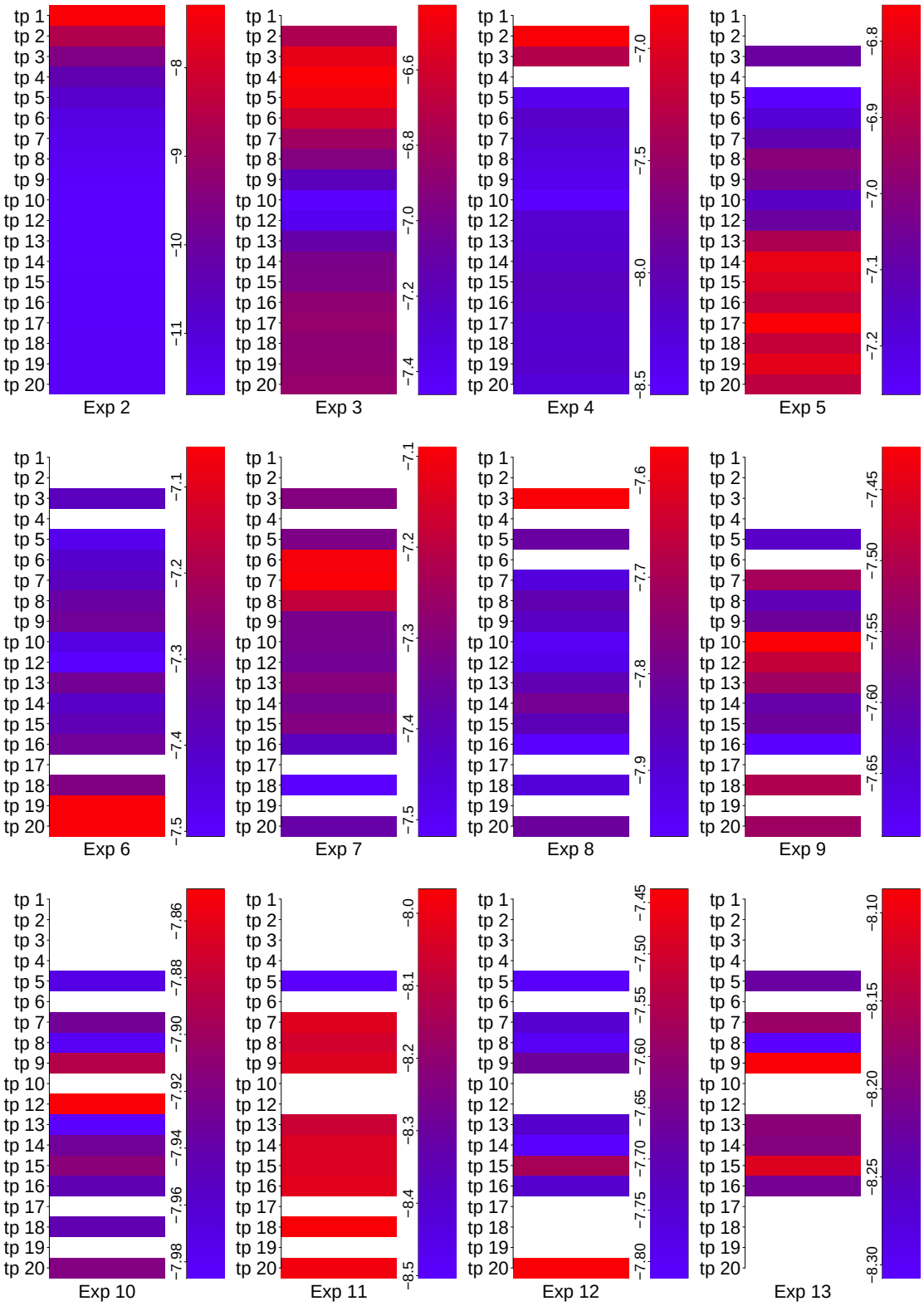


Figure B.5.: *Entropy estimates for run 4 for perfect data (Part I)*

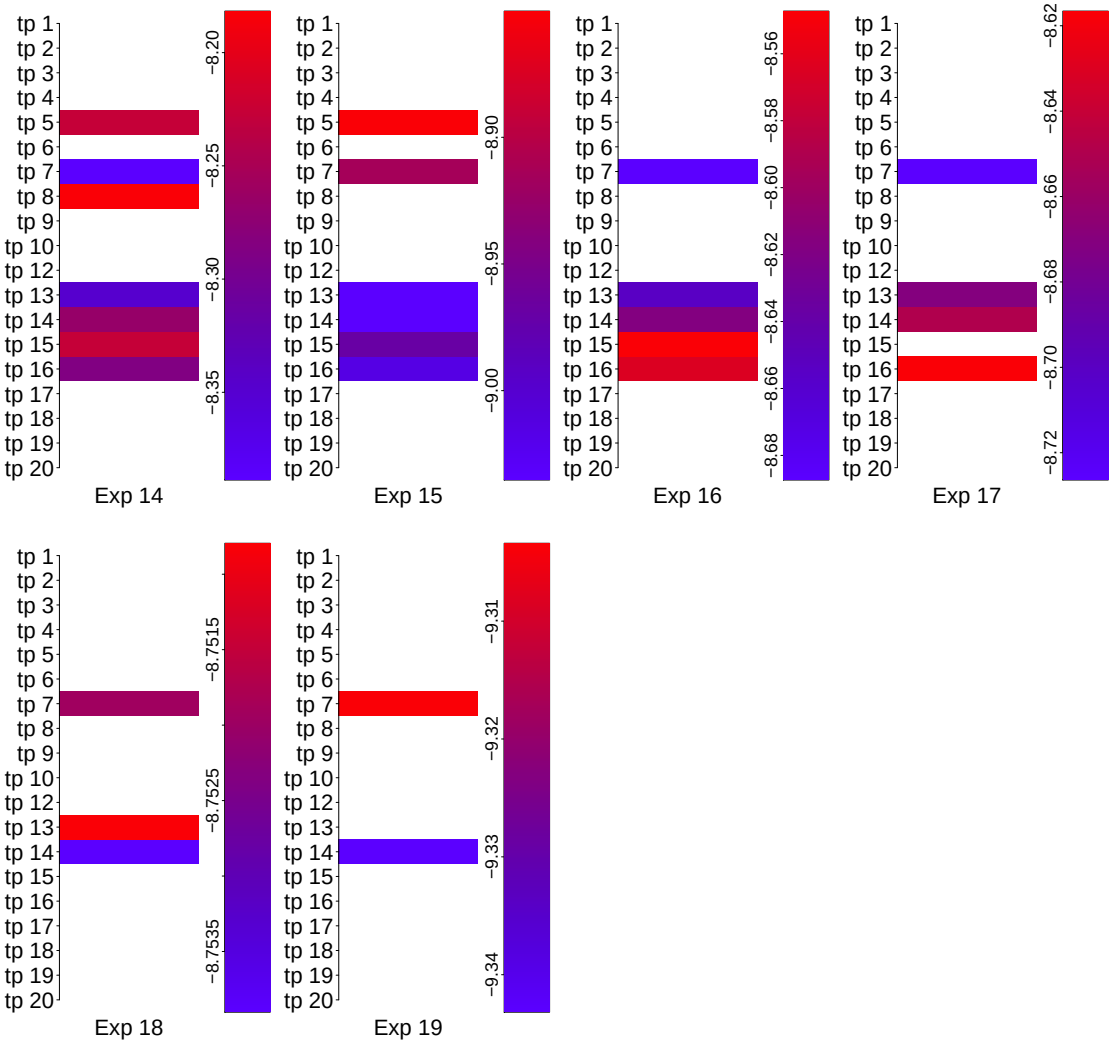


Figure B.6.: *Entropy estimates for run 4 for perfect data (Part II)*

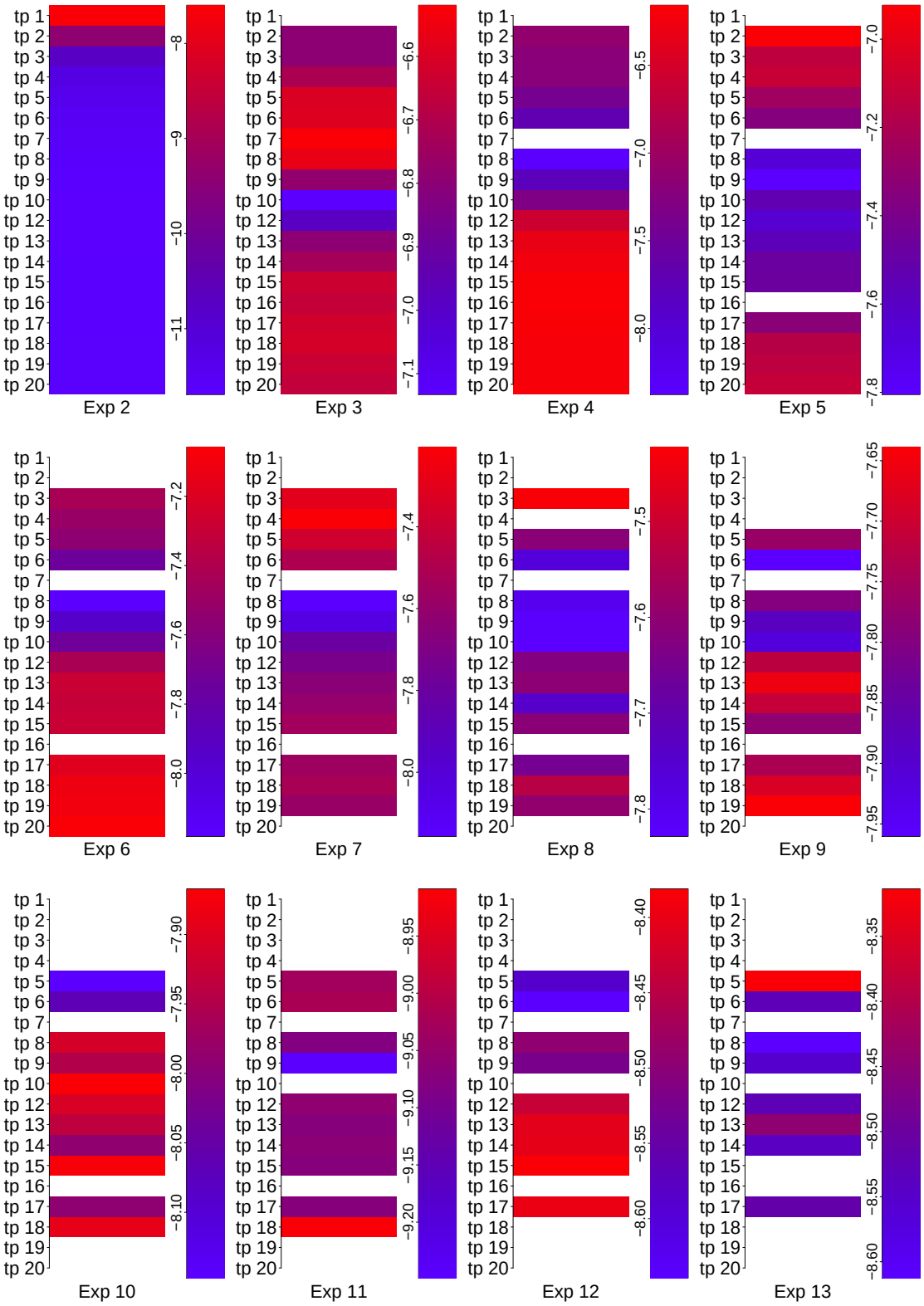


Figure B.7.: *Entropy estimates for run 5 for perfect data (Part I)*

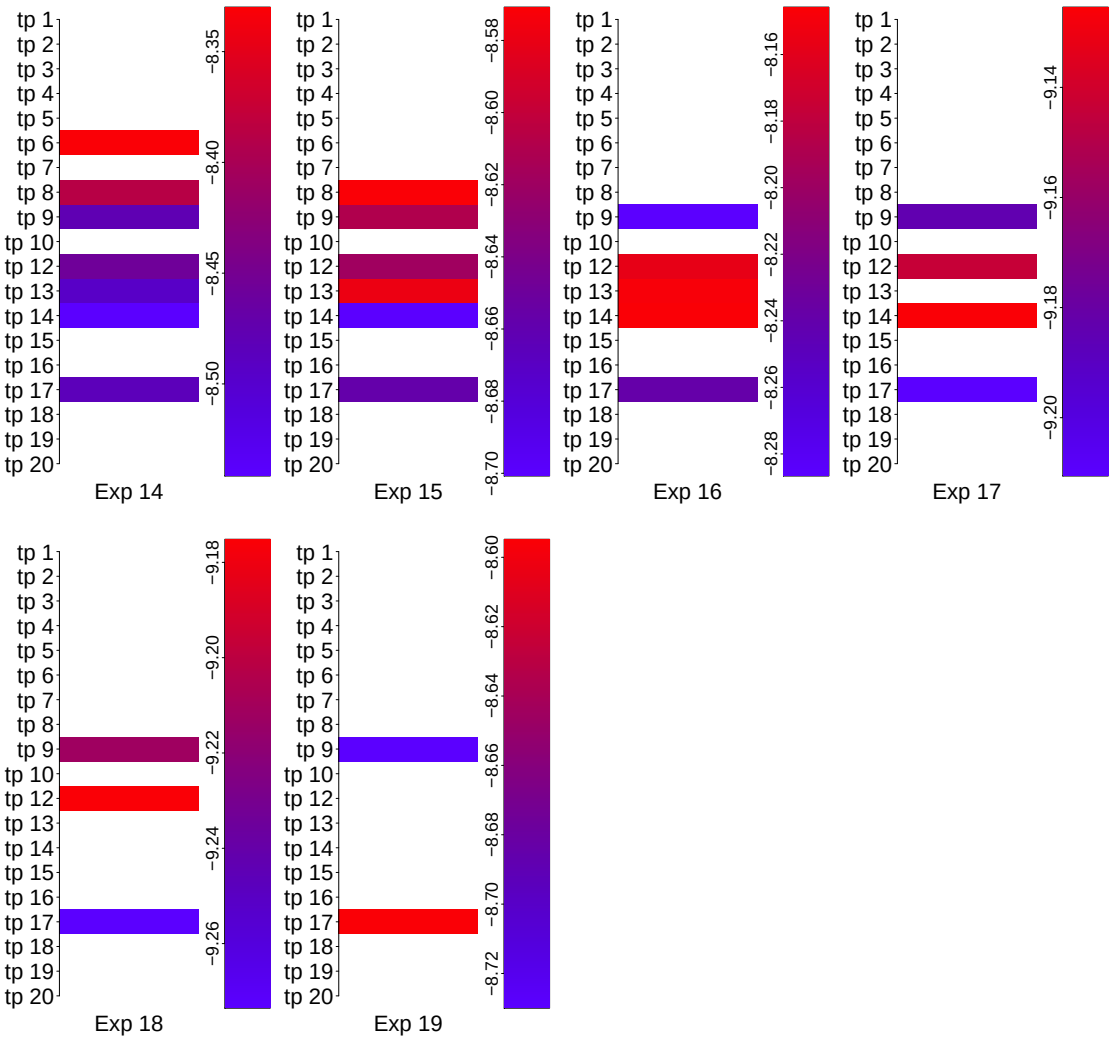


Figure B.8.: *Entropy estimates for run 5 for perfect data (Part II)*

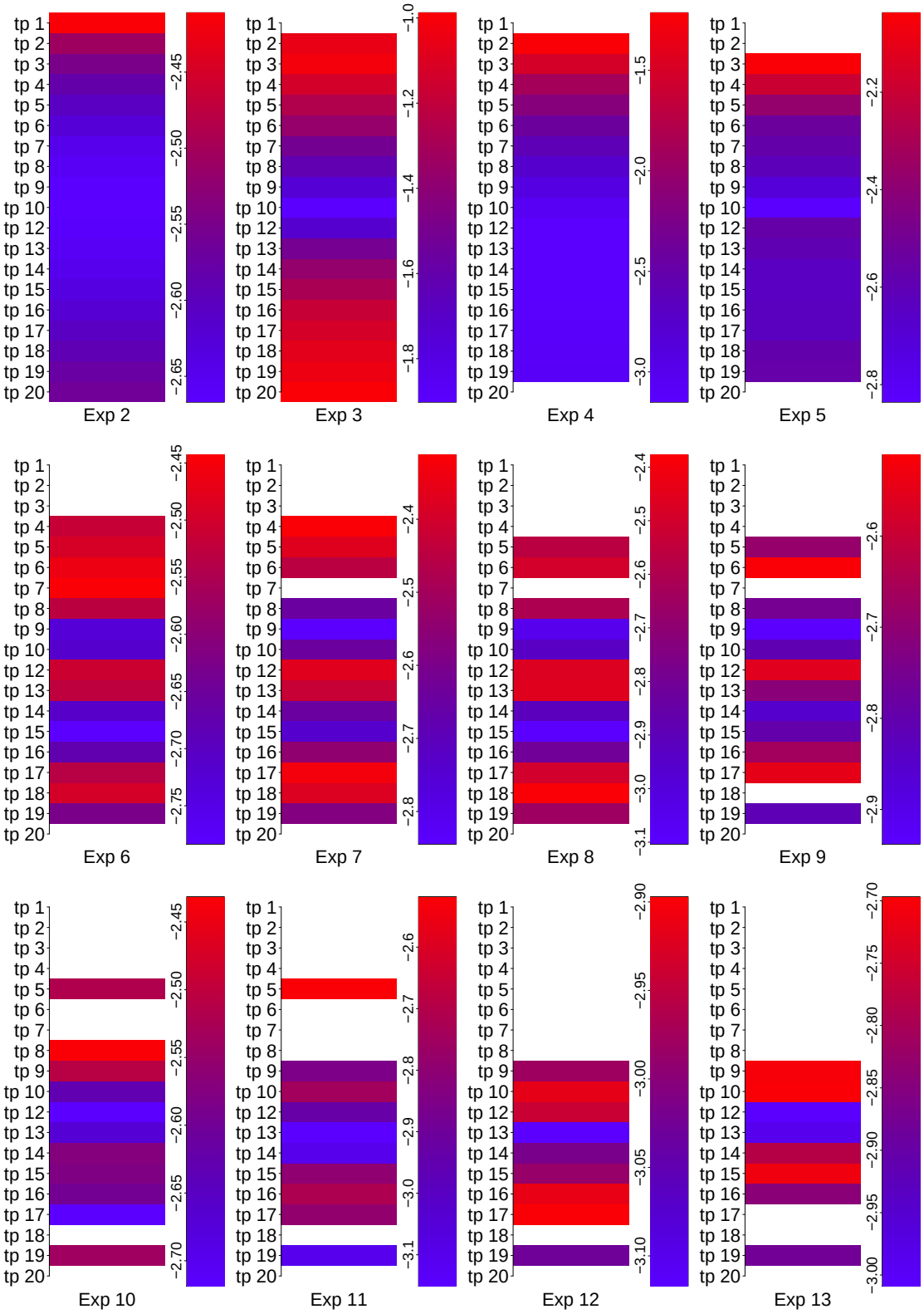


Figure B.9.: *Entropy estimates for run 2 for noisy data (Part I)*

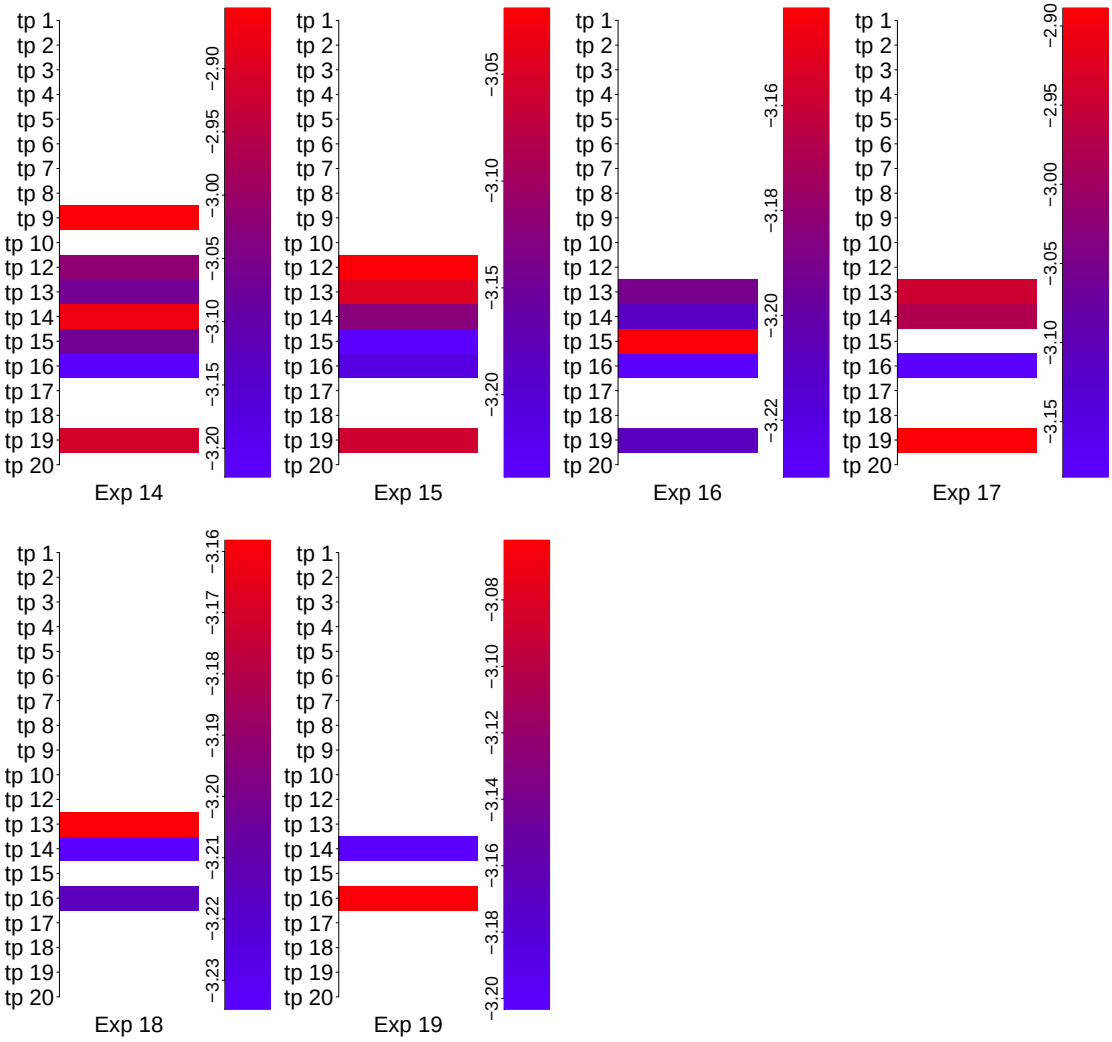


Figure B.10.: *Entropy estimates for run 2 for noisy data (Part II)*

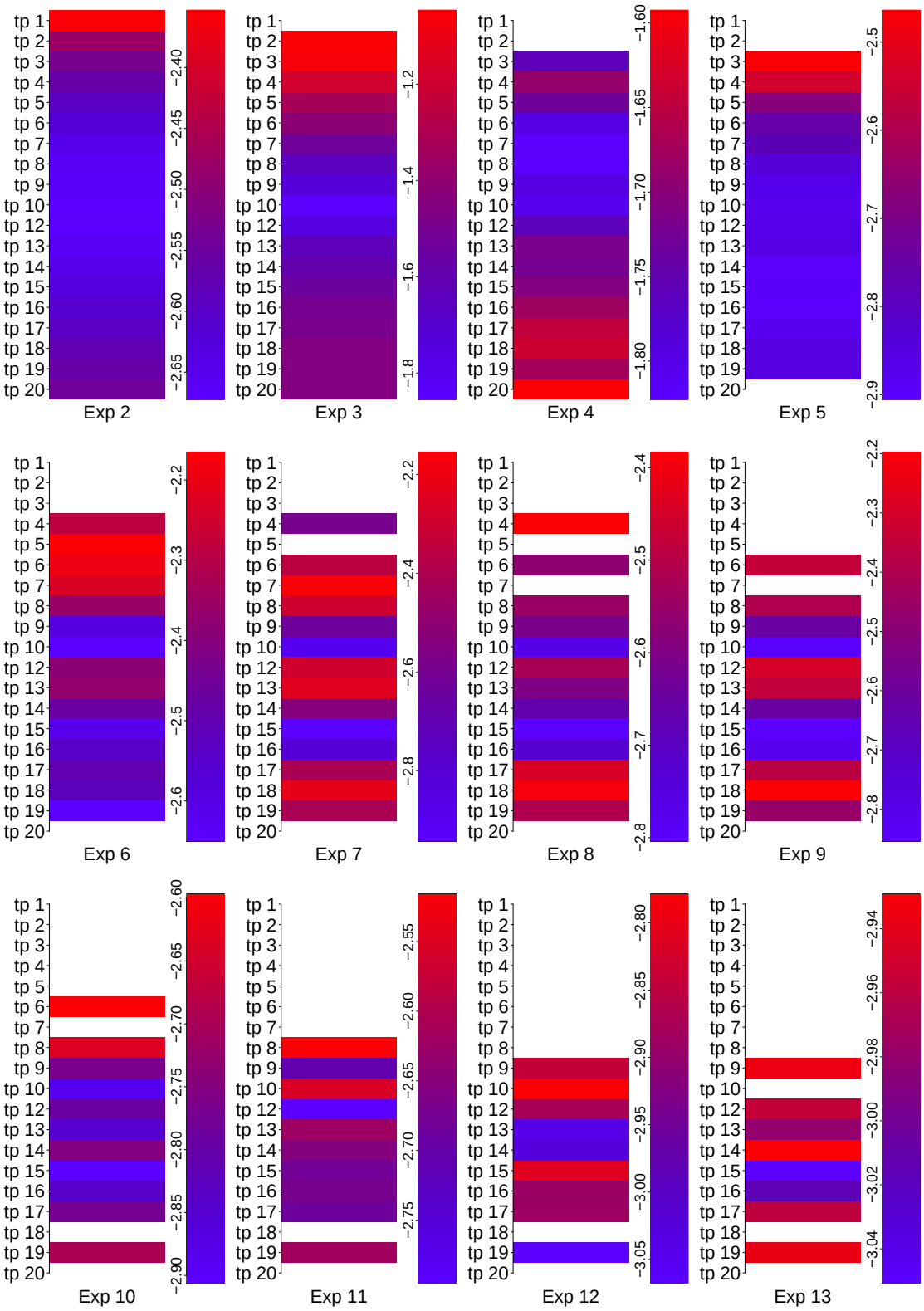


Figure B.11.: *Entropy estimates for run 3 for noisy data (Part I)*

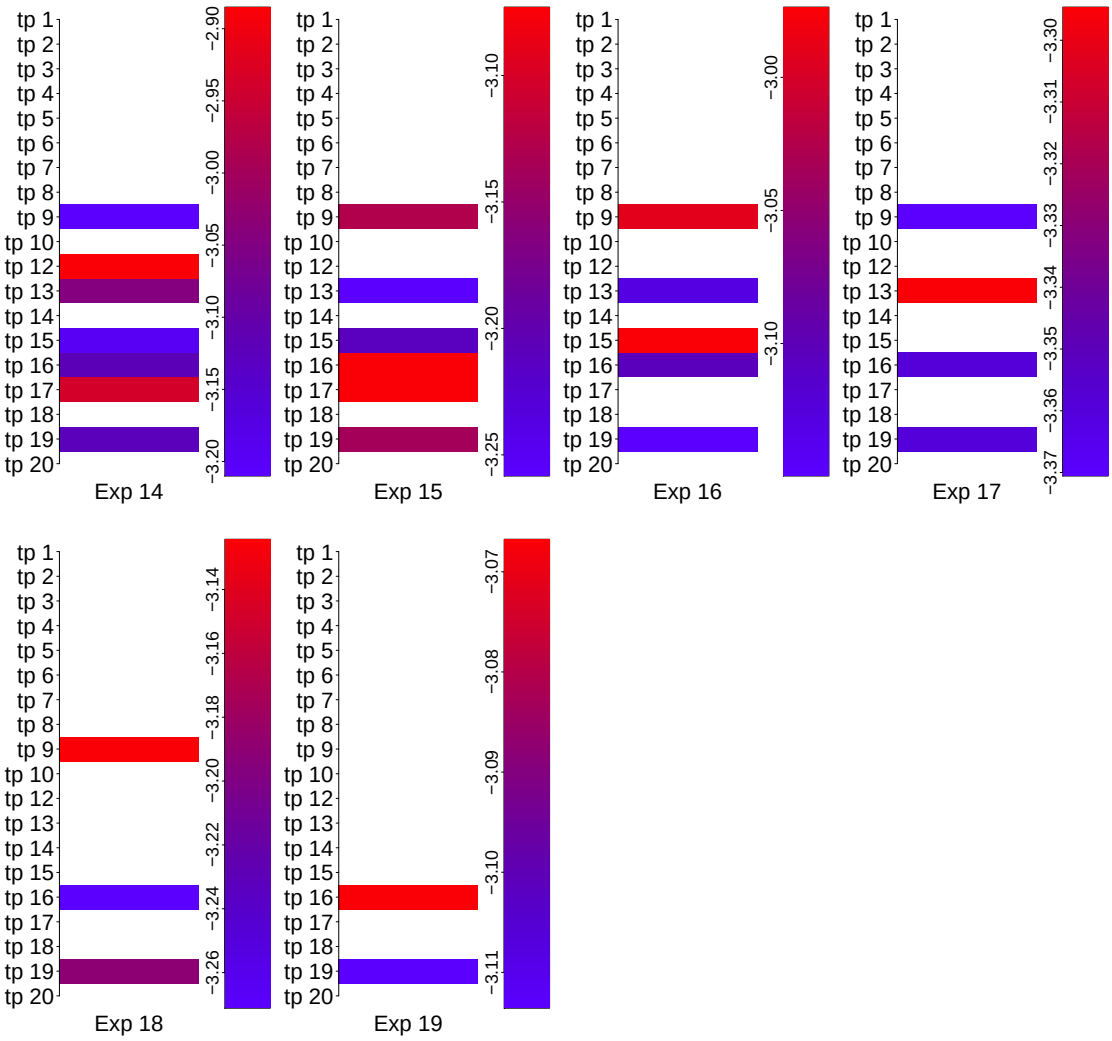


Figure B.12.: *Entropy estimates for run* 3 *for noisy data (Part II)*

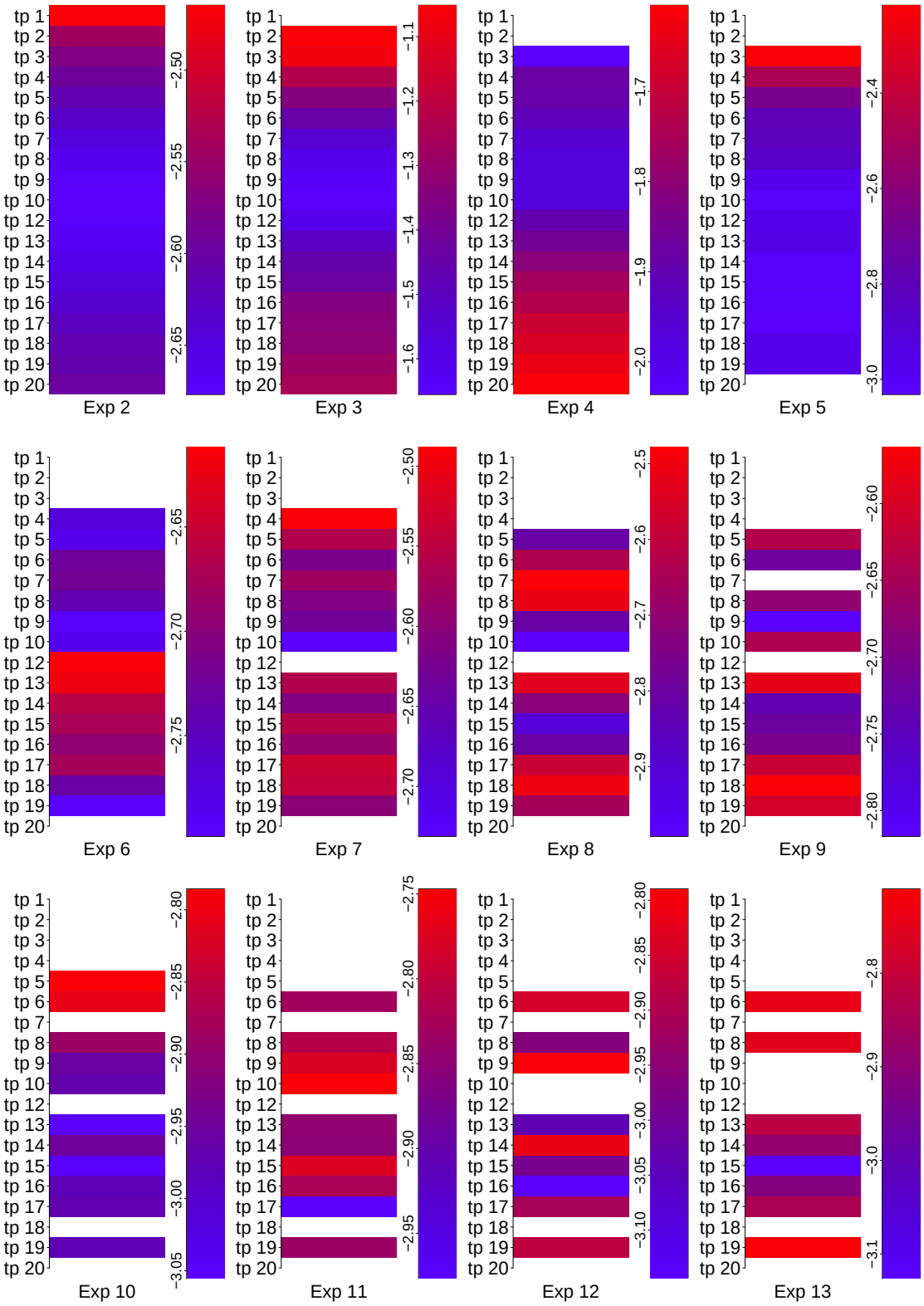


Figure B.13.: *Entropy estimates for run 4 for noisy data (Part I)*

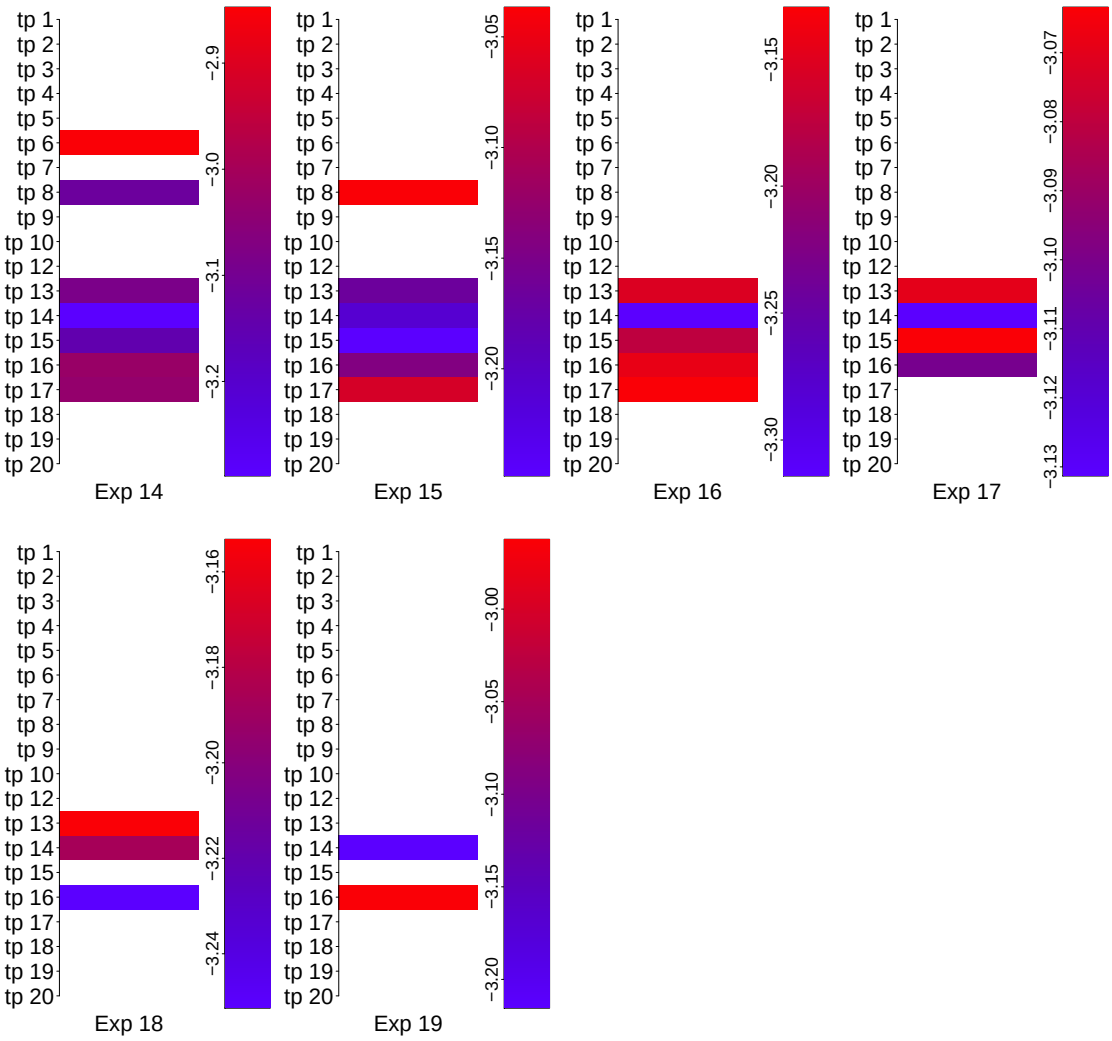


Figure B.14.: *Entropy estimates for run 4 for noisy data (Part II)*

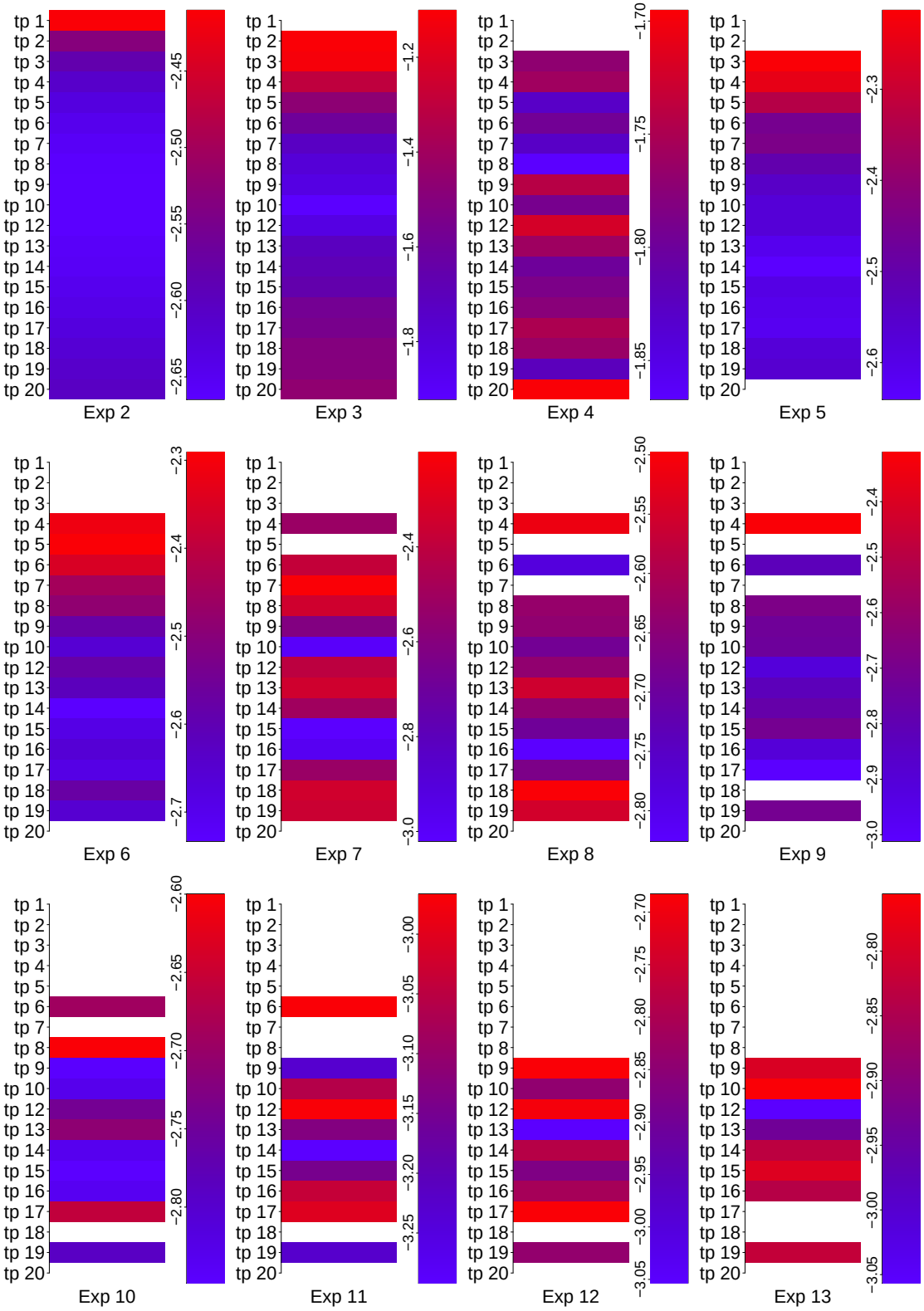


Figure B.15.: *Entropy estimates for run 5 for noisy data (Part I)*

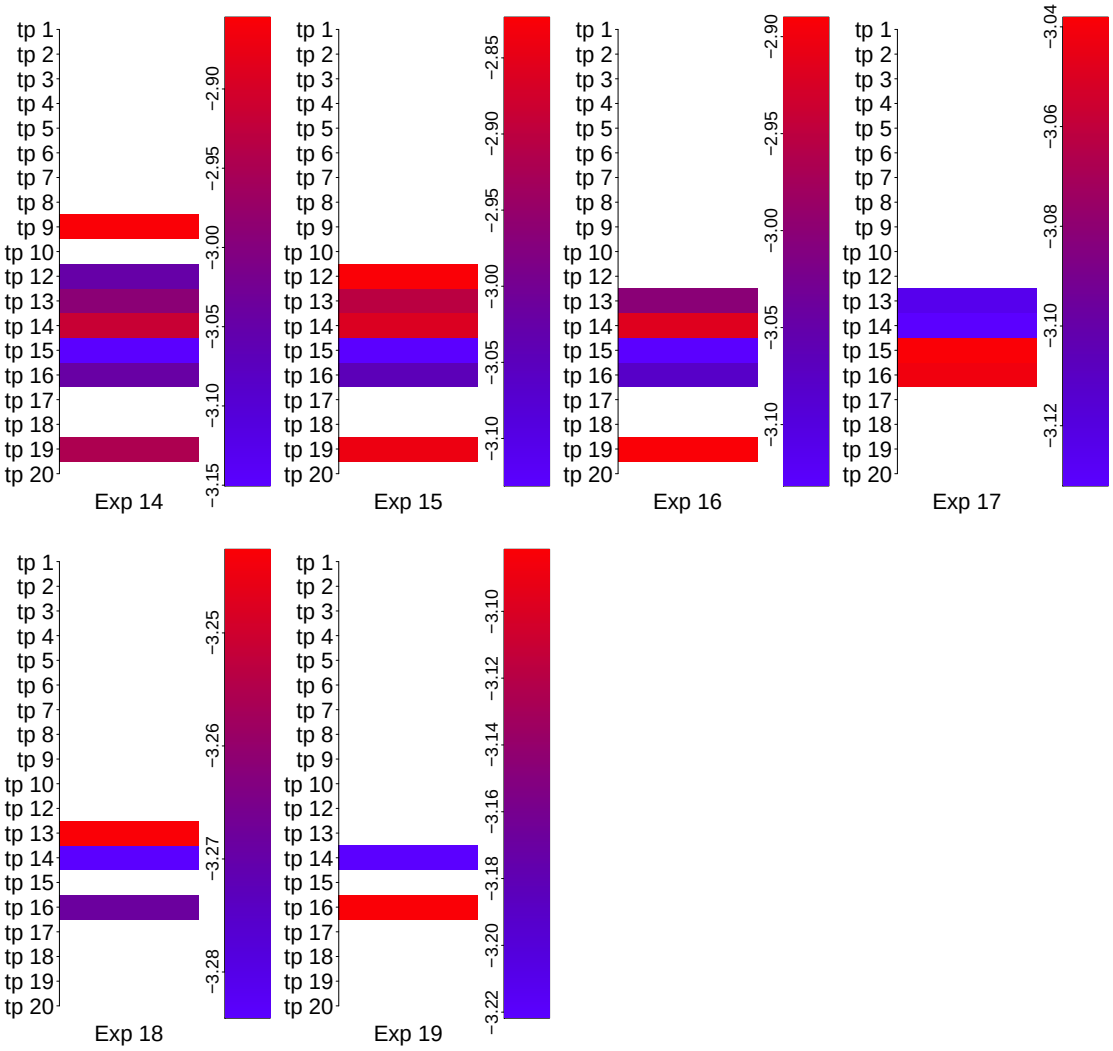


Figure B.16.: *Entropy estimates for run* 5 *for noisy data (Part II)*

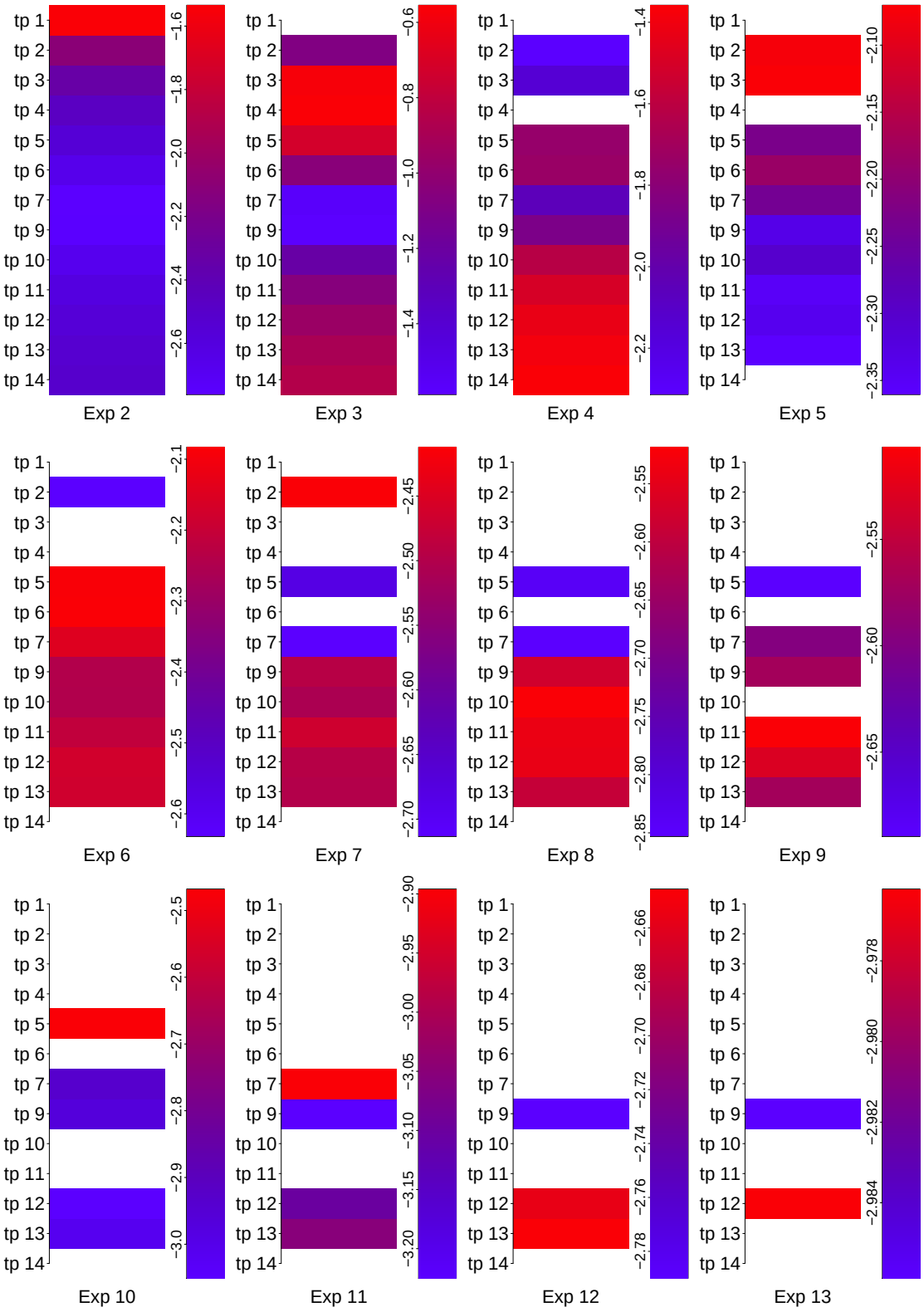


Figure B.17.: *Entropy estimates for run 2 for the DREAM 2 Challenge #3 data*

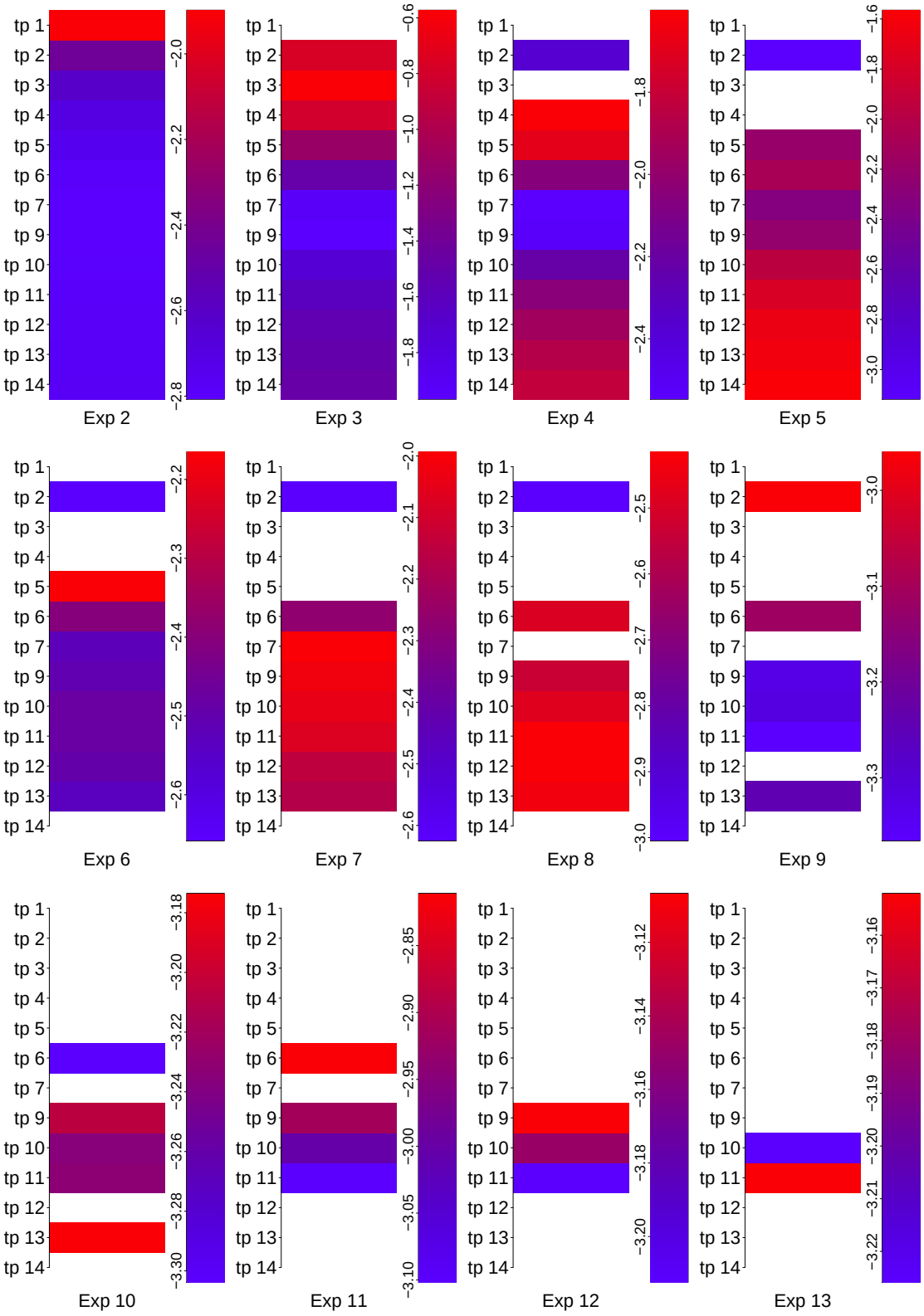


Figure B.18.: *Entropy estimates for run 3 for the DREAM 2 Challenge #3 data*

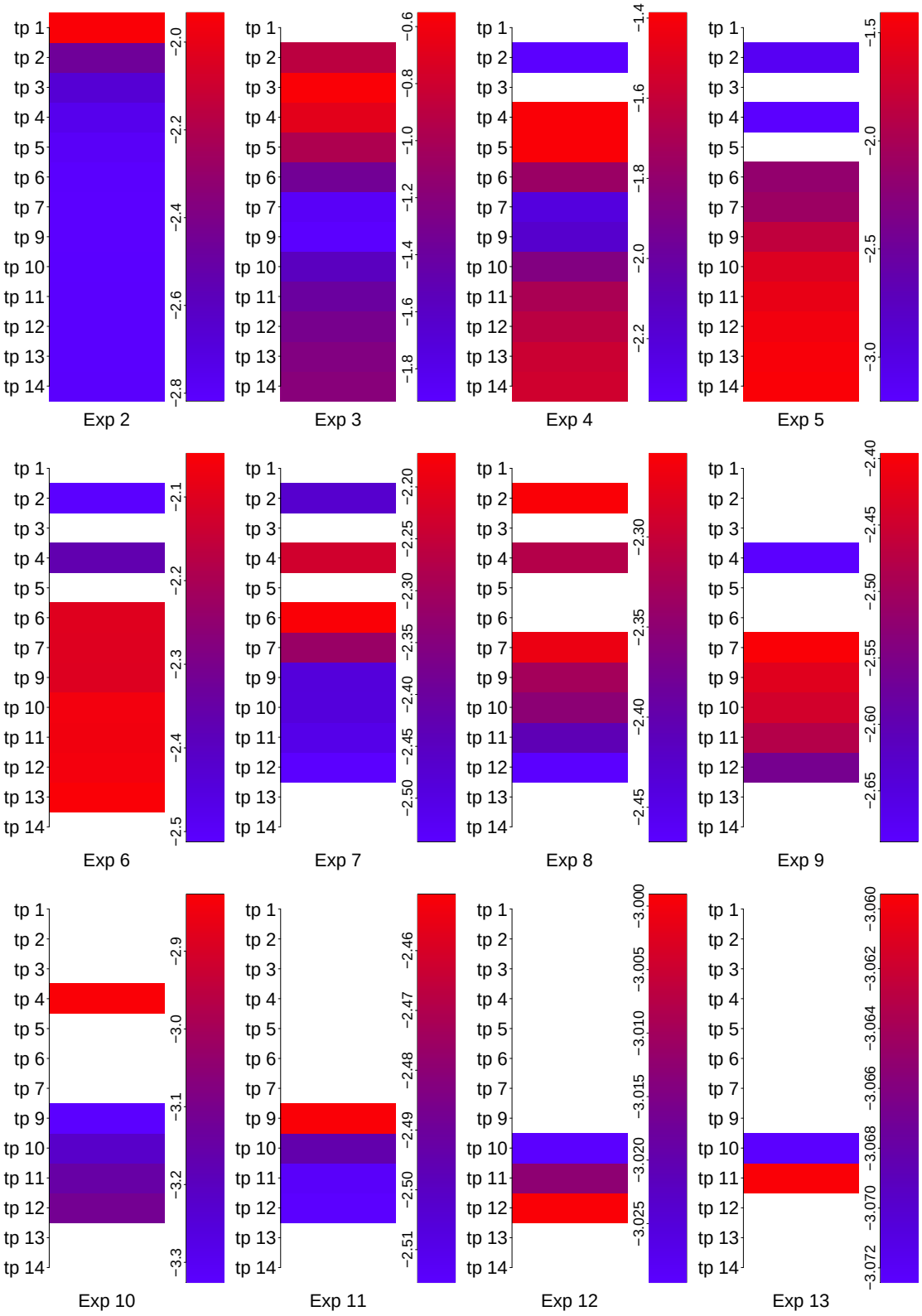


Figure B.19.: *Entropy estimates for run 4 for the DREAM 2 Challenge #3 data*

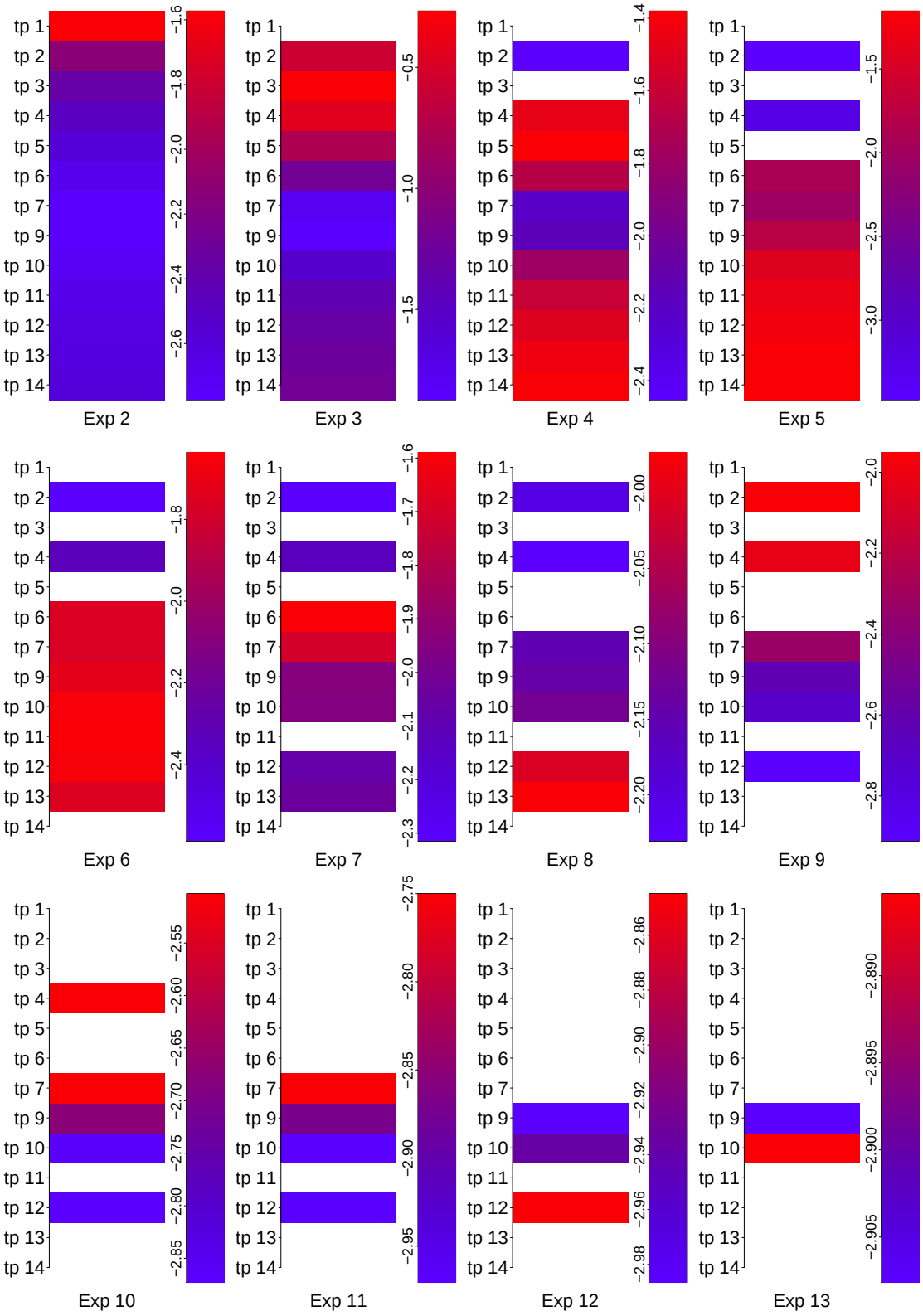


Figure B.20.: *Entropy estimates for run 5 for the DREAM 2 Challenge #3 data*

Index

D

E

F

G

H

I

N

S

List of Symbols and Acronyms

A ..... abbr. for *adenine*, page 9
$A(\cdot,\cdot)$ ..... acceptance probability for MCMC algorithms, see equation (3.1), page 49
a_j ..... propensity function of a chemical reaction R_j needed for Gillespie's stochastic simulation algorithm, page 82
ARACNE ..... abbr. for *Algorithm for the Reconstruction of Accurate Cellular Networks* [MNB+06], page 70
ATP ..... abbr. for *adenosine triphosphate*, the source of chemical energy for cells, page 7
[a.u.] ..... abbr. for *arbitrary unit*, page 20
AUC ..... abbr. for *area under curve*, page 62
$\mathrm{AUC}_{\mathrm{P2R}}$ ..... area under curve for the precision vs. recall ROC curve, page 65
$\mathrm{AUC}_{\mathrm{ROC}}$ ..... area under curve for the sensitivity vs. $1-$specificity ROC curve, page 65
AUG ..... start codon for protein synthesis, page 14
$\mathcal{B}$ ..... Borel sets, i.e., the σ-algebra on $\mathbb{R}$, page 26
BED ..... abbr. for *Bayesian experimental design*
$\mathrm{B}(\cdot,\cdot)$ ..... beta function, see equation (2.5), page 27
$\mathrm{Beta}(\alpha,\beta)$ ..... beta distribution with *shape parameters* α and β, page 27
β_{ij} ..... regulation strengths parameters in the ODE model for GRNs, page 86
BIC ..... abbr. for *Bayesian Information Criterion*, see equation (4.5), page 73
BN ..... abbr. for *Bayesian network*, page 72
bool_{x_0} ..... sequence of states in a Boolean network with initial network x_0, page 74
C ..... abbr. for *cytosine*, page 9
cDNA ..... abbr. for *complementary DNA*, page 16
c_j ..... constant describing the probability that molecules will react as needed to specify propensity functions in the SSA, page 82
CO_2 ..... abbr. for *carbon dioxide*, page 7
$\mathrm{Corr}(X_i,X_j)$ ... correlation coefficient between two random variables X_i and X_j, see equation (2.9), page 28
$\mathrm{Cov}(X_i,X_j)$ .... covariance between two random variables X_i and X_j, see equation (2.8), page 28
CPU ..... abbr. for *central processor unit*, page 147
C_T ..... abbr. for *crossing threshold* in qPCR, page 18

d ..... dimension of the parameter space to be sampled from, page 91
DAG ..... abbr. for *directed acyclic graph*, page 72
DBN ..... abbr. for *Dynamic Bayesian Network*, page 74
Δ ..... finite sequence of points in an interval defining the knots in a cubic spline S_Δ, page 44
δ ..... Euler's delta function, page 49
$\det(M)$ ..... determinant of a matrix M, page 144
DGA ..... abbr. for *Distributed Genetic Algorithm*, page 128
DGMC ..... abbr. for *Distributed Evolutionary Monte Carlo* by Hu and Tsui [HT10], page 127
$D_{\mathrm{KL}}(\cdot \parallel \cdot)$ ..... Kullback-Leibler divergence, see equation (2.17), page 33
DNA ..... abbr. for *deoxyribonucleic acid*, page 9
DREAM ..... abbr. for *Dialogue on Reverse Engineering Assessment and Methods*, page 96
$\mathbf{e}$ ..... direction of a proposed crossover step in a population-based MCMC algorithm, page 54
$\mathcal{E}$ ..... directed edges in a directed graph $G = (V, \mathcal{E})$, page 69
e ..... experiment to be estimated with experimental design, page 119
e_{best} ..... optimal experiment obtained in experimental design, page 119
EM ..... abbr. for *Expectation-Maximization*, page 73
$\mathrm{Ent}(\cdot \mid \cdot)$ ..... conditional differential entropy, see equation (2.21), page 35
$\mathrm{Ent}(\cdot)$ ..... differential entropy, see equation (2.20), page 35
ε ..... time step used in the leapfrog discretization of the HCM algorithm, page 51
ER ..... abbr. for *endoplasmic reticulum*, page 9
$E(X)$ ..... mean of a random variable X, see equation (2.6), page 27
$E(x)$ ..... potential energy of a physical state x, page 50
$\mathbb{F}$ ..... finite field, page 77
$\mathcal{F}$ ..... σ-algebra in a probability space $(\Omega, \mathcal{F}, p)$, page 25
F ..... distribution function of a random variable X, page 26
f ..... density function of a continuous random variable X, page 27
$\mathrm{FIM}(\theta, e)$ ..... Fisher information matrix for a model with parameters θ and experiment e, see equation (6.7), page 122
FN ..... abbr. for *false negatives*, page 61
FP ..... abbr. for *false positives*, page 61
fpr ..... abbr. for *false positive rate*, page 60
$\mathrm{frac}_{\mathrm{serial}}$ ..... fraction of time spent on serial operations in parallel programs, see equation (3.12), page 59
$f(x, p)$ ..... parametrized model used in experimental design dependent on *factors* x and *model parameters* p, page 112
G ..... abbr. for *guanine*, page 9
$g(\cdot)$ ..... density to sample from for the crossover operator proposed by [HT10], see equation (7.8), page 131
$\mathrm{Gamma}(r, a)$ ... gamma distribution with *shape parameter* r and *rate parameter* a, page 27
$\Gamma(\cdot)$ ..... gamma function, see equation (2.3), page 27
γ_i ..... degradation rate of the ODE model for GRNs for one gene, see equation (5.1), page 85
GB ..... abbr. for *gigabyte*, page 95

GGM ..... abbr. for *Gaussian graphical model*, page 70
GHz ..... abbr. for *gigahertz*, page 95
GO ..... abbr. for *Gene Ontology*, page 70
G_{p_j} ..... parameter sensitivities for parameter p_j needed for classical experimental design, page 117
GPU ..... abbr. for *graphical processor unit*, page 147
GRN ..... abbr. for *gene regulatory network*, page 69
$G(V, \mathcal{E})$ ..... topology of a gene regulatory network with nodes V and interactions $\mathcal{E}$, page 69
$H(\cdot)$ ..... Shannon entropy, see equation (2.15), page 32
$H(\cdot \mid \cdot)$ ..... conditional entropy, see equation (2.19), page 34
$h(\cdot \mid \cdot)$ ..... conditional differential entropy, see equation (2.21), page 35
$h(\cdot)$ ..... differential entropy, see equation (2.20), page 35
H_2 ..... abbr. for *hydrogen*, page 77
H_2O ..... abbr. for *water*, page 77
HMC ..... abbr. for the *hybrid Monte Carlo algorithm*, page 50
$H(x, \rho)$ ..... Hamiltonian of the phase space (x, ρ), see equation (3.4), page 51
$\mathrm{I}(\cdot)$ ..... Shannon information, see equation (2.16), page 32
Id ..... identity matrix, page 144
i.i.d. ..... abbr. for *independent and identically distributed* random variables, page 29
iPDA ..... abbr. for *iterative principal differential analysis*, page 89
K ..... precision matrix, page 71
$\hat{K}$ ..... estimate for the precision matrix K, page 71
$K(\rho)$ ..... kinetic energy of the momentum variables ρ, see equation (3.3), page 51
$L(\cdot; \cdot)$ ..... likelihood function, see equation (2.14), page 31
λ ..... smoothing factor in a smoothing spline, page 89
L_C ..... Lipschitz constant, see equation (2.30), page 42
$\lambda_{\mathbb{R}}$ ..... Lebesgue measure, i.e., the measure on $\mathbb{R}$, page 26
M_0 ..... initial marking of a Petri net, page 77
MAPK ..... abbr. for *mitogen-activated protein kinase*, page 72
MAQC ..... abbr. for the *MicroArray Quality Control* project, page 23
MCMC ..... abbr. for *Markov Chain Monte Carlo*, page 48
M_e ..... model used in BED to generate data with the underlying experiment e nomrefpage
MES ..... abbr. for *Maximum Entropy Sampling*, page 127
$\mathrm{MI}(\cdot; \cdot)$ ..... mutual information, see equation (2.18), page 34
m_{ij} ..... Hill coefficients of the ODE model for GRNs, see equation (5.2), page 86
ML ..... abbr. for *Maximum Likelihood*, page 31
$M_{\hat{p}}$ ..... information matrix for the parameter estimate $\hat{p}$, page 113
$M_{\hat{p}}^{\epsilon}$ ..... regularized information matrix for the parameter estimate $\hat{p}$, see equation (7.11), page 144
MPI ..... abbr. for *Message Passing Interface*, page 57
MPMD ..... abbr. for *multiple-program multiple-data*, page 58
mRNA ..... abbr. for *messenger RNA*, page 12
μ ..... mean of a random variable X, page 27
$\mathbb{N}$ ..... natural numbers, page 28
nm ..... abbr. for the unit *nano meter*, page 7

$\mathcal{N}(\mu, \sigma^2)$ ..... normal distribution with mean μ and variance σ^2, page 27
ν ..... number specifying the convergence of homogeneous Markov chain, see equation (2.23), page 39
ν_j ..... state-change vector as needed for the SSA, page 82
O_2 ..... abbr. for *oxygen*, page 77
ODE ..... abbr. for *ordinary differential equation*, page 40
Ω ..... set of outcomes in a probability space $(\Omega, \mathcal{F}, p)$, page 25
$(\Omega, \mathcal{F}, p)$ ..... probability space, page 25
Ω_V ..... volume chemical reactions occur in, page 82
p ..... probability measure in a probability space $(\Omega, \mathcal{F}, p)$, page 26
pa_i ..... parent nodes of random variable X_i, page 72
PBN ..... abbr. for *probabilistic Boolean network*, page 76
PCR ..... abbr. for *polymerase chain reaction*, page 18
$p(d \mid D, M_e)$ .... predictive distribution used in BED dependent on available data D and the model M_e for experiment e, page 126
$\hat{p}$ ..... estimator for the model parameters p as used in experimental design, page 113
π ..... invariant distribution, page 39
$p_i(\cdot)$ ..... probabilities for the states of random variables X_i in a Markov chain, page 37
P_{ID} ..... unique identifier for processors in parallel programs, page 57
$\mathcal{P}_j\big(a_j(\mathbf{x})\tau\big)$ ..... Poisson distributed random variable with mean $a_j(\mathbf{x})\tau$ as used for the SSA, see equation (4.13), page 84
p_{mig} ..... probability for migration in the DGMC algorithm [HT10], page 128
p_{mut} ..... probability for mutation in the DGMC algorithm [HT10], page 128
$\mathrm{prop}_{\mathrm{cross}}$ ..... proportion in every subpopulation to perform the crossover operator on in the DGMC algorithm [HT10], page 128
$\mathrm{prop}_{\mathrm{mut}}$ ..... proportion in every subpopulation to perform the mutation operator on in the DGMC algorithm [HT10], page 129
P_{sub} ..... vector according to individuals of subpopulations in the DGMC algorithms are passed over to other subpopulations in the migration operator, page 128
$q(\cdot \mid \cdot)$ ..... proposal distribution in MCMC algorithms, page 49
qPCR ..... abbr. for *real-time quantitative polymerase chain reaction*, page 17
$\mathbb{R}$ ..... real numbers, page 26
R ..... rules in a population-based MCMC algorithm, how to use the genetic operators, page 55
RAM ..... abbr. for *random-access memory*, page 95
ρ ..... momentum variables $(\rho_1, \ldots, \rho_n)$ of a closed system as used in the hybrid Monte Carlo algorithm, page 50
ρ_{ij} ..... correlation coefficient between two random variables X_i and X_j, page 28
R_i ..... chemical reactions as used in the SSA, page 82
RNA ..... abbr. for *ribonucleic acid*, page 12
RNAi ..... abbr. for *RNA interference*, page 76
ROC ..... abbr. for *Receiver Operating Characteristics*, page 59
rRNA ..... abbr. for *ribosomal RNA*, page 11
RT ..... abbr. for *reverse transcription*, page 22
S ..... directed acyclic graph structure in a Bayesian network, page 72

S_Δ ..... cubic spline corresponding to the sequence Δ, page 44
S_i ..... chemical species as used in the SSA, page 82
s_i ..... synthesis rate of the ODE model for GRNs for one gene, see equation (5.1), page 85
Σ ..... covariance matrix, page 28
σ^2 ..... variance of a random variable X, page 27
$\hat{\Sigma}$ ..... sample covariance matrix, see equation (4.2), page 71
S_p ..... speedup of a parallel program, see equation (3.11), page 59
SPMD ..... abbr. for *single-program multiple-data*, page 58
SSA ..... abbr. for *Gillespie's stochastic simulation algorithm*, page 82
T ..... abbr. for *thymine*, page 9
T ..... transition probability for a homogeneous Markov chain, page 38
$\tau(\cdot)$ ..... function of a parameter of a distribution function, page 30
τ ..... time parameter for the Hamiltonian dynamics, see equation (3.6), page 51
θ_{ij} ..... threshold parameters of the ODE model for GRNs, see equation (5.2), page 86
TN ..... abbr. for *true negatives*, page 61
T_n ..... transition probability for a Markov chain, page 37
T_n^e ..... estimator function, see equation (2.12), page 30
TP ..... abbr. for *true positives*, page 61
tRNA ..... abbr. for *transfer RNA*, page 11
U ..... abbr. for *uracil*, page 12
$U(d, e)$ ..... utility function used in BED dependent on future data d and the experiments e, page 126
$U(d, \theta, e, Y)$ .... general utility function needed for BED as proposed by Lindley [Lin72]
V ..... vertices in a directed graph $G = (V, \mathcal{E})$, page 69
$\mathrm{Var}(X)$ ..... variance of a random variable X, see equation (2.7), page 28
$\mathbf{X}$ ..... random vector of random variables $X_1, \ldots, X_n$, page 28
X ..... random variable, page 26

Bibliography

[AA06] Christophe Andrieu and Y. F. Atchadé. On the efficiency of adaptive MCMC algorithms. In *valuetools '06: Proceedings of the 1st international conference on Performance evaluation methodologies and tools*, page 43, New York, NY, USA, 2006. ACM.

[ABL+94] Bruce Alberts, Dennis Bray, Julian Lewis, Martin Raff, Keith Roberts, and James D. Watson. *Molecular Biology of the Cell*. Taylor & Francis, New York, third edition, 1994.

[AD97] Maria I. Arnone and Eric H. Davidson. The hardwiring of development: organization and function of genomic regulatory systems. *Development*, 124:1851–1864, 1997.

[ADT07] A. C. Atkinson, A. N. Donev, and R. D. Tobias. *Optimum Experimental Designs, with SAS*, volume 34 of *Oxford Statistical Science Series*. Oxford University Press, New York, USA, first edition, 2007.

[AFDJ03] Christophe Andrieu, Nando De Freitas, Arnaud Doucet, and Michael I. Jordan. An introduction to MCMC for machine learning. *Machine Learning*, 50:5–43, 2003.

[AJCG11] Wassim Abou-Jaoudé, Madalena Chaves, and Jean-Luc Gouzé. A theoretical exploration of birhythmicity in the p53-Mdm2 network. *PLoS ONE*, 6(2):e17075, 2011.

[AJL+08] Bruce Alberts, Alexander Johnson, Julian Lewis, Martin Raff, Keith Roberts, and Peter Walter. *Molecular Biology of the Cell*. Garland Science, Taylor & Francis Group, New York, fifth edition, 2008.

[AKB+10] Anil Aswani, Soile V. E. Keränen, James Brown, Charless C. Fowlkes, David W. Knowles, Mark D. Biggin, Peter Bickel, and Claire J. Tomlin. Nonparametric identification of regulatory interactions from spatial and temporal gene expression data. *BMC Bioinformatics*, 11:413, 2010.

[ÄL09] Tarmo Äijö and Harri Lähdesmäki. Learning gene regulatory networks from gene expression measurements using non-parametric molecular kinetics. *Bioinformatics*, 25(22):2937–2944, 2009.

[Alo06] Uri Alon. *An Introduction to Systems Biology: Design Principles of Biological Circuits*. Chapman & Hall/CRC, Boca Raton, FL, USA, July 2006.

[AM02] S. P. Asprey and S. Macchietto. Designing robust optimal dynamic experiments.

Journal of Process Control, 12:545–556, 2002.

[Amd67] Gene M. Amdahl. Validity of the single processor approach to achieving large scale computing capabilities. In *Proceedings of the April 18-20, 1967, spring joint computer conference*, AFIPS '67 (Spring), pages 483–485, New York, NY, USA, 1967. ACM.

[AMML11] Anthony Almudevar, Matthew N. McCall, Helene McMurray, and Hartmut Land. Fitting Boolean networks from steady state perturbation data. *Statistical Applications in Genetics and Molecular Biology*, 10(1):Article 47, 2011.

[AR05] Yves F. Atchadé and Jeffrey S. Rosenthal. On adaptive Markov chain Monte Carlo algorithms. *Bernoulli*, 11(5):815–828, 2005.

[ARM98] Adam Arkin, John Ross, and Harley H. McAdams. Stochastic kinetic analysis of developmental pathway bifurcation in phage λ-infected *Escherichia coli* cells. *Genetics*, 149:1633–1648, 1998.

[AWWT10] Joshua F. Apgar, David K. Witmer, Forest M. White, and Bruce Tidor. Sloppy models, parameter uncertainty, and the role of experimental design. *Molecular BioSystems*, 6:1890–1900, 2010.

[Bau92] Heinz Bauer. *Maß- und Integrationstheorie*. de Gruyter, Berlin, Germany, second edition, 1992.

[Bau01] Heinz Bauer. *Wahrscheinlichkeitstheorie*. de Gruyter, Berlin, Germany, fifth edition, 2001.

[BB09] Alberto Giovanni Busetto and Joachim M. Buhmann. Structure identification by optimized interventions. *Journal of Machine Learning Research - Proceedings Track*, 5:57–64, 2009.

[BC09] Nicolas E. Buchler and Frederick R. Cross. Protein sequestration generates a flexible ultrasensitive response in a genetic network. *Molecular Systems Biology*, 5:272, 2009.

[BCTR+08] Hauke Busch, David Camacho-Trullio, Zbigniew Rogon, Kai Breuhahn, Peter Angel, Roland Eils, and Axel Shabowski. Gene network dynamics controlling keratinocyte migration. *Molecular Systems Biology*, 4:Article number 199, 2008.

[Beh00] Ehrhard Behrends. *Introduction to Markov Chains (with Special Emphasis on Rapid Mixing)*. Vieweg (Advanced Lectures in Mathematics), 2000.

[BGdB06] Mukesh Bansal, Giusy Della Gatta, and Diego di Bernardo. Inference of gene regulatory networks and compound mode of action from time course gene expression profiles. *Bioinformatics*, 22(7):815–822, 2006.

[BHT89] Andreas Buja, Trevor Hastie, and Robert Tibshirani. Linear smoothers and additive models. *The Annals of Statistics*, 17(2):453–510, 1989.

[BK00] A. J. Butte and I. S. Kohane. Mutual information relevance networks: functional genomic clustering using pairwise entropy measurements. *Pacific Symposium on Biocomputing*, 5:415–426, 2000.

[BL59] G. E. P. Box and H. L. Lucas. Design of experiments in non-linear situations. *Biometrika*, 46(1–2):77–90, 1959.

[BL11] Plamenka Borovska and Milena Lazarova. Parallel models for sequence alignment on CPU and GPU. In *Proceedings of the 12th International Conference on Computer Systems and Technologies*, CompSysTech '11, pages 210–215, New York, NY, USA, 2011. ACM.

[BOB09] Alberto Giovanni Busetto, Cheng Soon Ong, and Joachim M. Buhmann. Optimized expected information gain for nonlinear dynamical systems. In Andrea Po-

horeckyj Danyluk, Léon Bottou, and Michael L. Littman, editors, *ICML*, volume 382 of *ACM International Conference Proceeding Series*, page 13. ACM, 2009.

[Böc08] Matthias Böck. *Bayesian learning of Boolean regulatory networks derived from expression data*. Diploma thesis, Technical University of Munich, Germany, 2008.

[Bre96] Leo Breiman. Bagging predictors. *Machine Learning*, 24:123–140, 1996.

[BSE+09] Samuel Bandara, Johannes P. Schlöder, Roland Eils, Hans Georg Bock, and Tobias Meyer. Optimal experimental design for parameter estimation of a cell signaling model. *PLoS Computational Biology*, 5(11):e1000558, 2009.

[BSSS11] Chris P. Barnes, Daniel Silk, Xia Sheng, and Michael P. H. Stumpf. Bayesian design of synthetic biological systems. *Proceedings of the National Academy of Sciences*, 2011.

[BVI00] Kristel Bernaerts, Karina J. Versyck, and Jan F. Van Impe. On the design of optimal dynamic experiments for parameter estimation of a Ratkowsky-type growth kinetics at suboptimal temperatures. *International Journal of Food Microbiology*, 54:27–38, 2000.

[BVI02] Julio R. Banga, Karina J. Versyck, and Jan F. Van Impe. Computation and optimal identification experiments for nonlinear dynamic process models: a stochastic global optimization approach. *Industrial & Engineering Chemistry Research*, 41(10):2425–2430, 2002.

[BWK08] R. J. Boys, D. J. Wilkinson, and T. B. L. Kirkwood. Bayesian inference for a discretely observed stochastic kinetic model. *Statistics and Computing*, 18(2):125–135, 2008.

[CAR+10] Y. Ann Chen, Jonas S. Almeida, Adam J. Richards, Peter Müller, Raymond J. Carroll, and Baerbel Rohrer. A nonparametric approach to detect nonlinear correlation in gene expression. *Journal of Computational and Graphical Statistics*, 19(3):552–568, 2010.

[CBF+07] Fergal P. Casey, Dan Baird, Qiyu Feng, Ryan N. Gutenkunst, Joshua J. Waterfall, Christopher R. Myers, Kevin S. Brown, Richard A. Cerione, and James P. Sethna. Optimal experimental design in an epidermal growth factor receptor signalling and down-regulation model. *IET Systems Biology*, 1(3):190–202, 2007.

[CdJG06] Richard Casey, Hidde de Jong, and Jean-Luc Gouzé. Piecewise-linear models of genetic regulatory networks: Equilibria and their stability. *Journal of Mathematical Biology*, 52(1):27–56, 2006.

[CFX06] Long Cai, Nir Friedman, and X. Sunney Xie. Stochastic protein expression in individual cells at the single molecule level. *Nature*, 440:358–362, 2006.

[CG08] Ben Calderhead and Mark Girolami. Sloppy parameters in oscillatory systems with unobserved species. In Miika Ahdesmäki, Korbinian Strimmer, Nicole Radde, Jörg Rahnenführer, Konstantin Klemm, Harri Lähdesmäki, and Olli Yii-Harja, editors, *Proceedings of the Fifth International Workshop on Computational Systems Biology*, pages 21–24. TICSP Series #41, 2008.

[CGP05] Yang Cao, Daniel T. Gillespie, and Linda R. Petzold. The slow-scale stochastic simulation algorithm. *Journal of Chemical Physics*, 122(1):014116, 2005.

[CGP06] Yang Cao, Daniel T. Gillespie, and Linda R. Petzold. Efficient step size selection for the tau-leaping simulation method. *Journal of Chemical Physics*, 124(4):044109, 2006.

[CGP+10] Vijender Chaitankar, Preetam Ghosh, Edward J. Perkins, Ping Gong, Youping

Deng, and Chaoyang Zhang. A novel gene network inference algorithm using predictive minimum description length approach. *BMC Systems Biology*, 4(Suppl 1):S7, 2010.

[CHC99] Ting Chen, Hongyu L. He, and George M. Church. Modeling gene expression with differential equations. In *Pacific Symposium on Biocomputing*, pages 29–40, 1999.

[Chi96] David Maxwell Chickering. Learning Bayesian networks is NP-complete. In D. Fisher and H. J. Lenz, editors, *Learning from Data: Artificial Intelligence and Statistics V*, pages 121–130. Springer, 1996.

[CHM04] David Maxwell Chickering, David Heckerman, and Christopher Meek. Large-sample learning of Bayesian networks is NP-hard. *Journal of Machine Learning Research*, 5:1287–1330, 2004.

[Cly01] Merlise A. Clyde. Experimental design: A Bayesian perspective. *International Encyclopedia of Social and Behavioral Sciences*, 2001.

[CMI+09] Irene Cantone, Lucia Marucci, Francesco Iorio, Maria Aurelia Ricci, Vincenzo Belcastro, Mukesh Bansal, Stefania Santini, Mario di Bernardo, Diego di Bernardo, and Maria Pia Cosma. A yeast synthetic network for in vivo assessment of reverse-engineering and modeling approaches. *Cell*, 137(1):172–181, 2009.

[Con06] MAQC Consortium. The MicroArray Quality Control (MAQC) project shows inter- and intraplatform reproducibility of gene expression measurements. *Nature Biotechnology*, 24(9):1151–1161, 2006.

[Con10] MAQC Consortium. The MicroArray Quality Control (MAQC)-II study of common practices for the development and validation of microarray-based predictive models. *Nature Biotechnology*, 28(8):827–838, 2010.

[CR09] Jiguo Cao and James O. Ramsay. Generalized profiling estimation for global and adaptive penalized spline smoothing. *Computational Statistics & Data Analysis*, 53:2550–2562, 2009.

[CRRT04] Claudine Chaouiya, Elisabeth Remy, Paul Ruet, and Denis Thieffry. Qualitative modelling of genetic networks: From logical regulatory graphs to standard petri nets. In Jordi Cortadella and Wolfgang Reisig, editors, *ICATPN*, volume 3099 of *Lecture Notes in Computer Science*, pages 137–156. Springer, 2004.

[CSKW03] Kwang-Hyun Cho, Sung-Young Shin, Walter Kolch, and Olaf Wolkenhauer. Experimental design in systems biology, based on parameter sensitivity analysis using a Monte Carlo method: A case study for the TNFα-mediated NF-κB signal transduction pathway. *Simulation*, 79(12):726–739, 2003.

[CT06] Thomas M. Cover and Joy A. Thomas. *Elements of information theory*. Wiley-Interscience, New York, NY, USA, second edition, 2006.

[CTS11] Ricky Chachra, Mark K. Transtrum, and James P. Sethna. Comment on "sloppy models, parameter uncertainty, and the role of experimental design". *Molecular BioSystems*, 7:2522, 2011.

[CV95] Kathryn Chaloner and Isabella Verdinelli. Bayesian experimental design: A review. *Statistical Science*, 10(3):273–304, 1995.

[dB01] Carl de Boor. *A Practical Guide to Splines*, volume 27 of *Applied Mathematical Sciences*. Springer, New York, USA, revised edition, 2001.

[DCS+08] Bryan C. Daniels, Yan-Jiun Chen, James P. Sethna, Ryan N. Gutenkunst, and Christopher R. Myers. Sloppiness, robustness, and evolvability in systems biol-

ogy. *Current Opinion in Biotechnology*, 19(4):389–395, 2008.

[DEO+09] Rónán Daly, Kieron D. Edwards, John S. O'Neill, Stuart Aitken, Andrew J. Millar, and Mark Girolami. Using higher-order dynamic Bayesian networks to model periodic data from the circadian clock of *Arabidopsis thaliana*. In *PRIB '09: Proceedings of the 4th IAPR International Conference on Pattern Recognition in Bioinformatics*, pages 67–78, Berlin, Heidelberg, 2009. Springer.

[DGM+06] Norbert Dojer, Anna Gambin, Andrzej Mizera, Bartek Wilczyński, and Jerzy Tiuryn. Applying dynamic Bayesian networks to perturbed gene expression data. *BMC Bioinformatics*, 7:249, 2006.

[dJGH+04] Hidde de Jong, Jean-Luc Gouzé, Céline Hermandez, Michel Page, Tewfik Sari, and Johannes Geiselmann. Qualitative simulation of genetic regulatory networks using piecewise-linear models. *Bulletin of Mathematical Biology*, 66:301–340, 2004.

[dJP00] Hidde de Jong and Michel Page. Qualitative simulation of large and complex genetic regulatory systems. In W. Horn, editor, *Proceedings of the Fourteenth European Conference on Artificial Intelligence, ECAI 2000*, pages 141–145, Amsterdam, Netherlands, 2000. IOS Press.

[DKPR87] D. Duane, A. D. Kennedy, B. J. Pendleton, and D. Roweth. Hybrid Monte Carlo. *Physics Letters B*, 195:216–222, 1987.

[dlFBHM04] Alberto de la Fuente, Nan Bing, Ina Hoeschele, and Pedro Mendes. Discovery of meaningful associations in genomic data using partial correlation coefficients. *Bioinformatics*, 20(18):3565–3574, 2004.

[DLR77] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):1–38, 1977.

[Doo53] J. L. Doob. *Stochastic Processes*. John Wiley & Sons, New York, USA, 1953.

[Dur05] Richard Durrett. *Probability: Theory and Examples*. Brooks / Cole, Belmont, CA, USA, third edition, 2005.

[DWM11] Markus Durzinsky, Annegret Wagler, and Wolfgang Marwan. Reconstruction of extended Petri nets from time series data and its application to signal transduction and to gene regulatory networks. *BMC Systems Biology*, 5:113, 2011.

[DWWM08] Markus Durzinsky, Annegret Wagler, Robert Weismantel, and Wolfgang Marwan. Automatic reconstruction of molecular and genetic networks from discrete time series data. *Biosystems*, 93(3):181–190, 2008.

[edi10] MAQC-II: analyze that! *Nature Biotechnology*, 28(8):761, 2010.

[Edw00] David Edwards. *Introduction to Graphical Modelling*. Springer, NY, USA, second edition, 2000.

[EG00] R. Edwards and L. Glass. Combinatorial explosion in model gene networks. *Chaos*, 10(3):691, 2000.

[Faw06] Tom Fawcett. An introduction to ROC analysis. *Pattern Recognition Letters*, 27:861–874, 2006.

[Fer82] Thomas S. Ferguson. An inconsistent maximum likelihood estimate. *Journal of the American Statistical Association*, 77(380):831–834, 1982.

[FGP+09] Thomas Fournier, Jean-Pierre Gabriel, Jerôme Pasquier, Christian Mazza, José Galbete, and Nicolas Mermod. Stochastic models and numerical algorithms for a class of regulatory networks. *Bulletin of Mathematical Biology*, 71:1394–1431, 2009.

[FKT03] D. Faller, U. Klingmüller, and J. Timmer. Simulation methods for optimal experimental design in systems biology. *Simulation*, 79(12):717–725, 2003.

[FLNP00] Nir Friedman, Michal Linial, Iftach Nachman, and Dana Pe'er. Using Bayesian networks to analyze expression data. *Journal of Computational Biology*, 7(3/4):601–620, 2000.

[FMR98] Nir Friedman, Kevin Murphy, and Stuart Russell. Learning the structure of dynamic probabilistic networks. In *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*, pages 139–147. Morgan Kaufmann, 1998.

[Fri89] Jerome H. Friedman. Regularized discriminant analysis. *Journal of the American Statistical Association*, 84(405):165–175, 1989.

[Fri97] Nir Friedman. Learning belief networks in the presence of missing values and hidden variables. In *Proceedings of the Fourteenth International Conference on Machine Learning*, pages 125–133. Morgan Kaufmann, 1997.

[Fri98] Nir Friedman. The Bayesian structural EM algorithm. In *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*, pages 129–138, San Francisco, CA, USA, 1998. Morgan Kaufmann.

[GCSR04] Andrew Gelman, John B. Carlin, Hal S. Stern, and Donald B. Rubin. *Bayesian Data Analysis*. Chapman & Hall/CRC, Boca Raton, FL, second edition, 2004.

[GG06] Didier Gonze and Albert Goldbeter. Circadian rhythms and molecular noise. *Chaos*, 16(2):026110, 2006.

[GG10] Clare E. Giacomantonio and Geoffrey J. Goodhill. A Boolean model of the gene regulatory network underlying mammalian cortical area development. *PLoS Computational Biology*, 6(9), 2010.

[GGI05] Kapil G. Gadkar, Rudiyanto Gunawan, and Francis J. Doyle III. Iterative approach to model identification of biological networks. *BMC Bioinformatics*, 6:155, 2005.

[GHL05] Mikael Gustafsson, Michael Hornquist, and Anna Lombardi. Constructing and analyzing a large-scale gene-to-gene regulatory network-Lasso-constrained inference and biological validation. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 2(3):254–261, 2005.

[Gil76] Daniel T. Gillespie. A general method for numerically simulating the stochastic time evolution of coupled chemical reactions. *Journal of Computational Physics*, 22(4):403–434, 1976.

[Gil77] Daniel T. Gillespie. Exact stochastic simulation of coupled chemical reactions. *The Journal of Physical Chemistry*, 81(25):2340–2361, 1977.

[Gil01] Daniel T. Gillespie. Approximate accelerated stochastic simulation of chemically reacting systems. *Journal of Chemical Physics*, 115(4):1716–1733, 2001.

[Gil07] Daniel T. Gillespie. Stochastic simulation of chemical kinetics. *Annual Review of Physical Chemistry*, 58:35–55, 2007.

[GK73] Leon Glass and Stuart A. Kauffman. The logical analysis of continuous, non-linear biochemical control networks. *Journal of Theoretical Biology*, 39(1):103–129, 1973.

[GKKG03] Ananth Grama, George Karypis, Vipin Kumar, and Anshul Gupta. *Introduction to Parallel Computing*. Addison Wesley, Harlow, England, second edition, 2003.

[GL07] Gopi Goswami and Jun S. Liu. On learning strategies for evolutionary Monte Carlo. *Statistics and Computing*, 17(1):23–38, 2007.

[GLS99] William Gropp, Ewing Lusk, and Anthony Skjellum. *Using MPI: Portable Par-*

allel Programming with the Message Passing Interface. MIT press, Cambridge, MA, second edition, 1999.

[Gol89] David E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1989.

[GR08] Elaine Garbarine and Gail Rosen. An information theoretic method of microarray probe design for genome classification. Presented at the conference: *IEEE Engineering in Medicine and Biology Society Conference (EMBC)* in Vancouver, Canada, 2008.

[GRS98] Walter R. Gilks, S. Richardson, and D.J. Spiegelhalter. *Markov chain Monte Carlo in practice*. Chapman & Hall/CRC, 1998.

[GRW07] J. Gebert, N. Radde, and G.-W. Weber. Modeling gene regulatory networks with piecewise linear differential equations. *European Journal of Operational Research*, 181(3):1148–1165, September 2007.

[GvdM87] László Györfi and Edward C. van der Meulen. Density-free convergence properties of various estimators of entropy. *Computational Statistics & Data Analysis*, 5:425–436, 1987.

[GW05] A. Golightly and D. J. Wilkinson. Bayesian inference for stochastic kinetic models using a diffusion approximation. *Biometrics*, 61:781–788, 2005.

[GW08a] A. Golightly and D. J. Wilkinson. Bayesian inference for nonlinear multivariate diffusion models observed with error. *Computational Statistics & Data Analysis*, 52(3):1674–1693, 2008.

[GW08b] Andreas Griewank and Andrea Walther. *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation*. Society for Industrial and Applied Mathematics, Philadelphia, second edition, 2008.

[GWC+07] Ryan N. Gutenkunst, Josua J. Waterfall, Fergal P. Casey, Kevin S. Brown, Christopher R. Myers, and James P. Sethna. Universally sloppy parameter sensitivities in systems biology models. *PLoS Computational Biology*, 3(10):e189, 2007.

[HAP10] Raed I. Hamed, S. I. Ahson, and R. Parveen. A new approach for modelling gene regulatory networks using fuzzy Petri nets. *Journal of Integrative Bioinformatics*, 7(1):113, 2010.

[Has70] W. K. Hastings. Monte Carlo sampling methods using Markov chains and their applications. *Biometrika*, 57(1):97–109, 1970.

[Hec96] David Heckerman. A tutorial on learning with Bayesian networks. Technical report, Microsoft Research, Advanced Technology Division, 1996.

[Heu06] Harro Heuser. *Gewöhnliche Differentialgleichungen*. Teubner, Wiesbaden, Germany, 2006.

[HFR07] Elizabeth A. Heron, Bärbel Finkenstädt, and David A. Rand. Bayesian inference for dynamic transcriptional regulation; the Hes1 system as a case study. *Bioinformatics*, 23(19):2596–2603, 2007.

[HGC95] David Heckerman, Dan Geiger, and David M. Chickering. Learning Bayesian networks: The combination of knowledge and statistical data. *Machine Learning*, 20(3):197–243, 1995.

[Hil10] A. V. Hill. The possible effects of the aggregation of the molecules of haemoglobin on its dissociation curves. *Journal of Physiology*, 40 (Suppl.):iv–vii, 1910.

[HKAA11] L Hasan, Kentie, and Z. Al-Ars. GPU-accelerated protein sequence alignment.

In *33rd Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC '11)*, pages 2442–2446, August 2011.

[HLT+09] Michael Hecker, Sandro Lambeck, Susanne Toepfer, Eugene van Someren, and Reinhard Guthke. Gene regulatory network inference: Data integration in dynamic models - A review. *BioSystems*, 96:86–103, 2009.

[HM11] X. Huan and Y. M. Marzouk. Simulation-based optimal Bayesian experimental design for nonlinear systems. *ArXiv e-prints*, August 2011. `arXiv:1108.4146v1 [stat.ML]`.

[HS09] Jean Hausser and Korbinian Strimmer. Entropy inference and the James-Stein estimator, with application to nonlinear gene association networks. *Journal of Machine Learning Research*, 10:1469–1484, 2009.

[HST01] Heikki Haario, Eero Saksman, and Johanna Tamminen. An adaptive Metropolis algorithm. *Bernoulli*, 7(2):223–242, 2001.

[HT10] Bo Hu and Kam-Wah Tsui. Distributed evolutionary Monte Carlo for Bayesian computing. *Computational Statistics & Data Analysis*, 54(3):688–697, 2010.

[Hun10] Margaret Hunt. Real time PCR. `http://pathmicro.med.sc.edu/pcr/realtime-home.htm`, July 2010.

[HYB10] Fei He, Hong Yue, and Martin Brown. Investigating Bayesian robust experimental design with principles of global sensitivity analysis. In Mayuresh Kothare, Moses Tade, Alain Vande Wouwer, and Ilse Smets, editors, *Proceedings of the 9th International Symposium on Dynamics and Control of Process Systems, DYCOPS 2010, Leuven, Belgium, July 5–7, 2010*, pages 563–568, 2010.

[JB09] William P. Janzen and Paul Bernasconi, editors. *High Throughput Screening: Methods and Protocols*, volume 565 of *Methods in Molecular Biology*. Humana Press, Totowa, New Jersey, USA, second edition, 2009.

[JM61] F. Jacob and J. Monod. Genetic regulatory mechanisms in the synthesis of proteins. *Journal of Molecular Biology*, 3:318–356, 1961.

[Jol02] I. T. Jolliffe. *Principal Component Analysis*. Springer Series in Statistics. Springer, New York, USA, second edition, 2002.

[Jos07] Jürgen Jost. *Partial Differential Equations*, volume 214 of *Graduate Texts in Mathematics*. Springer, New York, USA, second edition, 2007.

[JSH07] Ajay Jasra, David A. Stephens, and Christopher C. Holmes. On population-based simulation for static inference. *Statistics and Computing*, 17:263–279, 2007.

[Kau69] S. A. Kauffman. Metabolic stability and epigenesis in randomly constructed genetic nets. *Journal of Theoretical Biology*, 22(4):437–467, 1969.

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

[KCW04] Zoltán Kutalik, Kwang-Hyun Cho, and Olaf Wolkenhauer. Optimal sampling time selection for parameter estimation in dynamic pathway modeling. *BioSystems*, 75:43–55, 2004.

[KDZ+09] Lars Kaderali, Eva Dazert, Ulf Zeuge, Michael Frese, and Ralf Bartenschlager. Reconstructing signaling pathways from RNAi data using probabilistic Boolean threshold networks. *Bioinformatics*, 25(17):2229–2235, 2009.

[Kep09] Jeremy Kepner. *Parallel MATLAB for Multicore and Multinode Computers*. Society for Industrial and Applied Mathematics, Philadelphia, 2009.

[KGS+10] Christoph Kaleta, Anna Göhler, Stefan Schuster, Knut Jahreis, Reinhard

Guthke, and Swetlana Nikolajewa. Integrative inference of gene-regulatory networks in *Escherichia coli* using information theoretic concepts and sequence analysis. *BMC Systems Biology*, 4:116, 2010.

[KHAR10] A. Kramer, J. Hasenauer, F. Allgöwer, and N. Radde. Computation of the posterior entropy in a Bayesian framework for parameter estimation in biological networks. In *IEEE International Conference on Control Applications (CCA), 2010*, pages 493–498, September 2010.

[KIK+05] Shuhei Kimura, Kaori Ide, Aiko Kashihara, Makoto Kano, Mariko Hatakeyama, Ryoji Masui, Noriko Nakagawa, Shigeyuki Yokoyama, Seiki Kuramitsu, and Akihiko Konagaya. Inference of S-system models of genetic networks using a cooperative coevolutionary algorithm. *Bioinformatics*, 21(7):1154–1163, 2005.

[Kit02] Hiroaki Kitano. Computational systems biology. *Nature*, 420:206–210, 2002.

[KKB+02] Boris N. Kholodenko, Anatoly Kiyatkin, Frank J. Bruggeman, Eduardo Sontag, Hans V. Westerhoff, and Jan B. Hoek. Untangling the wires: A strategy to trace functional interactions in signaling and gene networks. *Proceedings of the National Academy of Sciences*, 99(20):12841–12846, 2002.

[KMK] Hahn Kim, Julia Mullen, and Jeremy Kepner. Introduction to parallel programming and pMatlab v2.0. `www.ll.mit.edu/mission/isr/pmatlab/pMatlab_intro.pdf`.

[Kna06] Andreas Knauf. Vorlesung Gewöhnliche Differentialgleichungen. `www.mathematik.uni-erlangen.de/~knauf/Skripte/ode.pdf`, 2006. lectures notes from the Universität Erlangen-Nürnberg.

[Kon09] Kevin Kontos. *Gaussian Graphical Model Selection for Gene Regulatory Network Reverse Engineering and Function Prediction*. PhD thesis, Université Libre de Bruxelles, Belgium, 2009.

[Kör02] Stefan Körkel. *Numerische Methoden für Optimale Versuchsplanungsprobleme bei nichtlinearen DAE-Modellen*. PhD thesis, Ruprecht-Karls-Universität Heidelberg, Germany, 2002.

[KPST03] Stuart Kauffman, Carsten Peterson, Björn Samuelsson, and Carl Toein. Random Boolean network models and the yeast transcriptional network. *Proceedings of the National Academy of Sciences*, 100(25):14796–14799, 2003.

[KR08] Lars Kaderali and Nicole Radde. Inferring gene regulatory networks from expression data. In Arpad Kelemen, Ajith Abraham, and Yuehui Chen, editors, *Computational Intelligence in Bioinformatics*, volume 94 of *Studies in Computational Intelligence*, pages 33–74. Springer, Berlin, 2008.

[KR10] Andrei Kramer and Nicole Radde. Towards experimental design using a Bayesian framework for parameter identification in dynamic intracellular network models. *Procedia Computer Science*, 1(1):1639–1647, 2010.

[KS08] Guy Karlebach and Ron Shamir. Modelling and analysis of gene regulatory networks. *Nature Reviews Molecular Cell Biology*, 9:770–780, 2008.

[KT09] Clemens Kreutz and Jens Timmer. Systems biology: experimental design. *FEBS Journal*, 276:923–942, 2009.

[KTA+03] Shinichi Kikuchi, Daisuke Tominaga, Masanori Arita, Katsutoshi Takahashi, and Masaru Tomita. Dynamic modeling of genetic networks using genetic algorithm and S-system. *Bioinformatics*, 19(5):643–650, 2003.

[KZF+06] Lars Kaderali, Thomas Zander, Ulrich Faigle, Jürgen Wolf, Joachim L. Schultze, and Rainer Schrader. CASPAR: a hierarchical Bayesian approach to predict

survival times in cancer from gene expression data. *Bioinformatics*, 22(12):1495–1502, 2006.

[KZRZ09] Younhee Ko, ChengXiang Zhai, and Sandra Rodriguez-Zas. Inference of gene pathways using mixture Bayesian networks. *BMC Systems Biology*, 3:54, 2009.

[LBK+07] Harvey Lodish, Arnold Berk, Chris A. Kaiser, Monty Krieger, Matthew P. Scott, and Anthony Bretscher. *Molecular Cell Biology*. Palgrave Macmillan, sixth edition, 2007.

[LFS98] Shoudan Liang, Stefanie Fuhrman, and Roland Somogyi. REVEAL, a general reverse engineering algorithm for inference of genetic network architectures. *Pacific Symposium on Biocomputing*, (3):18–29, 1998.

[Lin56] D. V. Lindley. On a measure of the information provided by an experiment. *The Annals of Mathematical Statistics*, 27(4):986–1005, 1956.

[Lin72] Dennis Victor Lindley. *Bayesian Statistics, A Review*. Society of Industrial and Applied Mathematics, Philadelphia, USA, 1972.

[Lip76] R. J. Lipton. The reachability problem requires exponential space. Research report 62, New Haven, Connecticut: Yale University, Department of Computer Science, 1976.

[LLL+04] Fangting Li, Tao Long, Ying Lu, Qi Ouyang, and Chao Tang. The yeast cell-cycle network is robustly designed. *Proceedings of the National Academy of Sciences*, 101(14):4781–4786, 2004.

[LLT+09] Bing Liu, Jiuyong Li, Anna Tsykin, Lin Liu, Arti B. Gaur, and Gregory J. Goodall. Exploring complex miRNA-mRNA interactions with Bayesian networks by splitting-averaging strategy. *BMC Bioinformatics*, 10:408, 2009.

[LN96] Rudolf Lidl and Harald Niederreiter. *Finite Fields*, volume 20 of *Encyclopedia of Mathematics and its Applications*. Cambridge University Press, second edition, 1996.

[Lor04] Thomas J. Loredo. Bayesian adaptive exploration. *ArXiv e-prints*, September 2004. `arXiv:astro-ph/0409386v1`.

[LS01] Kenneth J. Livak and Thomas D. Schmittgen. Analysis of relative gene expression data using real-time quantitative PCR and the $2^{-\Delta\Delta C_T}$ method. *Methods*, 26:402–408, 2001.

[LS04] Reinhard Laubenbacher and Brandilyn Stigler. A computational algebra approach to the reverse engineering of gene regulatory networks. *Journal of Theoretical Biology*, 229(4):523–537, 2004.

[LS11] Harri Lähdesmäki and Ilya Shmulevich. BN/PBN Matlab Toolbox. `http://personal.systemsbiology.net/ilya/PBN/PBN.htm`, November 2011.

[LSYH03] Harri Lähdesmäki, Ilya Shmulevich, and Olli Yli-Harja. On learning gene regulatory networks under the Boolean network model. *Machine Learning*, 52:147–167, 2003.

[LW00] Jürgen Lehn and Helmut Wegmann. *Einführung in die Statistik*. Teubner, Stuttgart, third edition, 2000.

[LW01] Faming Ling and Wing Hung Wong. Real-parameter evolutionary Monte Carlo with applications to Bayesian mixture models. *Journal of the American Statistical Association*, 96(454):653–666, 2001.

[LWZ11] Li-Zhi Liu, Fang-Xiang Wu, and W.J. Zhang. Inference of biological S-system using separable estimation method and genetic algorithm. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 99(PrePrints), 2011.

[LZP+07] Peng Li, Chaoyang Zhang, Edward J. Perkins, Ping Gong, and Youping Deng. Comparison of probabilistic Boolean network and dynamic Bayesian network approaches for inferring gene regulatory networks. *BMC Bioinformatics*, 8(Suppl7):S13, 2007.

[LZPA07] Yufeng Liu, Hao Helen Zhang, Cheolwoo Park, and Jeongyoun Ahn. Support vector machines with adaptive Lq penalty. *Computational Statistics & Data Analysis*, 51(12):6380–6394, 2007.

[MA99] Harley H. McAdams and Adam Arkin. It's a noisy business! Genetic regulation at the nanomolar scale. *Trends in Genetics*, 15(2):65–69, 1999.

[Mar05] Florian Markowetz. *Probabilistic Models for Gene Silencing Data*. PhD thesis, Freie Universität Berlin, Germany, 2005.

[MCS05] Arianne J. Matlin, Francis Clark, and Chritopher W. J. Smith. Understanding alternative splicing: Towards a cellular code. *Nature Reviews Molecular Cell Biology*, 6:386–398, 2005.

[MDBA11] Daniel Madar, Erez Dekel, Anat Bren, and Uri Alon. Negative auto-regulation increases the input dynamic-range of the arabinose system of *Escherichia coli*. *BMC Systems Biology*, 5:111, 2011.

[MDNM00] Hiroshi Matsuno, Atsushi Doi, Masao Nagasaki, and Satoru Miyano. Hybrid Petri net representation of gene regulatory network. *Pacific Symposium on Biocomputing*, 5:338–349, 2000.

[MGB07] Shisong Ma, Qingqiu Gong, and Hans J. Bohnert. An *Arabidopsis* gene network based on the graphical Gaussian model. *Genome Research*, 17(11):1614–1625, 2007.

[MHDD09] Linyong Mao, John L. Van Hemert, Sudhansu Dash, and Julie A. Dickerson. *Arabidopsis* gene co-expression network and its functional modules. *BMC Bioinformatics*, 10:346, 2009.

[MHK10] Christoph Müssel, Martin Hopfensitz, and Hans A. Kestler. BoolNet - an R package for generation, reconstruction and analysis of Boolean networks. *Bioinformatics*, 26(10):1378–1380, 2010.

[MK11] Johanna Mazur and Lars Kaderali. The importance and challenges of Bayesian parameter learning in systems biology. In H. G. Bock, T. Carraro, W. Jäger, S. Körkel, R. Rannacher, and J. P. Schlöder, editors, *Model Based Parameter Estimation: Theory and Applications*, volume 3 of *Contributions in Mathematical and Computational Sciences*. Springer, 2011. in press.

[MKKK11] Markus Maucher, Barbara Kracher, Michael Kühl, and Hans A. Kestler. Inferring Boolean network structure via correlation. *Bioinformatics*, 27(11):1529–1536, 2011.

[MM99] Kevin Murphy and Saira Mian. Modelling gene expression data using dynamic Bayesian networks. Technical report, Computer Science Division, University of California, 1999.

[MNB+06] Adam A. Margolin, Ilya Nemenman, Katia Basso, Chris Wiggins, Gustavo Stolovitzky, Riccardo Della Favera, and Andrea Califano. ARACNE: An algorithm for the reconstruction of gene regulatory networks in a mammalian cellular context. *BMC Bioinformatics*, 7(Suppl 1):S7, 2006.

[MP95] Peter Müller and Giovanni Parmigiani. Optimal design via curve fitting of Monte Carlo experiments. *Journal of the American Statistical Association*, 90(432):1322–1330, 1995.

[MPO95] Thomas Mestl, Erik Plahte, and Stig W. Omholt. A mathematical framework for describing and analysing gene regulatory networks. *Journal of Theoretical Biology*, 176(2):291–300, 1995.

[MRRK09] Johanna Mazur, Daniel Ritter, Gerhard Reinelt, and Lars Kaderali. Reconstructing nonlinear dynamic models of gene regulation using stochastic sampling. *BMC Bioinformatics*, 10:448, 2009.

[MRRT53] Nicholas Metropolis, Arianna W. Rosenbluth, Marshall N. Rosenbluth, and Augusta H. Teller. Equations of state calculations by fast computing machines. *Journal of Chemical Physics*, 21(6):1087–1092, 1953.

[MS07] Florian Markowetz and Rainer Spang. Inferring cellular networks - a review. *BMC Bioinformatics*, 8(Suppl 6):S5, 2007.

[MT09] Sean Meyn and Richard L. Tweedie. *Markov Chains and Stochastic Stability*. Cambridge University Press, Cambridge, UK, second edition, 2009.

[Mül99] Peter Müller. Simulation-based optimal design. In José M. Bernardo, James O. Berger, A. P. Dawid, and Adrian F. M. Smith, editors, *Bayesian Statistics 6*, pages 459–474. Oxford University Press, 1999.

[Mur89] Tadao Murata. Petri nets: Properties, analysis and applications. *Proceedings of the IEEE*, 77(4):541–580, 1989.

[Nea93] Radford M. Neal. *Probabilistic Inference Using Markov Chain Monte Carlo Methods*. Technical Report, 1993.

[Nel01] O. Nelles. *Nonlinear System Identification*. Springer, Berlin, Germany, 2001.

[NHB06] Tania Nolan, Rebecca E. Hands, and Stephen A. Bustin. Quantification of mRNA using real-time RT-PCR. *Nature Protocols*, 1(3):1559–1582, 2006.

[NK09] Lan K. Nguyen and Don Kulasiri. On the functional diversity of dynamical behaviour in genetic and metabolic feedback systems. *BMC Systems Biology*, 3:51, 2009.

[NMB+09] Chris J. Needham, Iain W. Manfield, Andrew J. Bulpitt, Philip M. Gilmartin, and David R. Westhead. From gene expression to gene regulatory networks in *Arabidopsis thaliana*. *BMC Systems Biology*, 3:85, 2009.

[NNL+05] Olli Niemitalo, Antje Neubauer, Ulf Liebal, Johanna Myllyharju, André H. Juffer, and Peter Neubauer. Modelling of translation of human protein disulfide isomerase in *Escherichia coli* - A case study of gene optimisation. *Journal of Biotechnology*, 120(1):11–24, 2005.

[NRF04] I. Nachman, A. Regev, and N. Friedman. Inferring quantitative models of regulatory networks from expression data. *Bioinformatics*, 20(Suppl.1):i248–i256, 2004.

[NW06] Jorge Nocedal and Stephen J. Wright. *Numerical Optimization*. Springer Series in Operations Research and Financial Engineering. Springer, New York, USA, second edition, 2006.

[ORS07] Rainer Opgen-Rhein and Korbinian Strimmer. From correlation to causation networks: a simple approximate learning algorithm and its application to high-dimensional plant gene expression data. *BMC Systems Biology*, 1:37, 2007.

[OTK+02] Ertugrul M. Ozbudak, Mukund Thattai, Iren Kurtser, Alan D. Grossman, and Alexander van Oudenaarden. Regulation of noise in the expression of a single gene. *Nature Genetics*, 31:69–73, 2002.

[PBT05] J. M. Peña, J. Björkegren, and J. Tegnér. Growing Bayesian network models of gene networks from seed genes. *Bioinformatics*, 21(Suppl.2):ii224–ii229, 2005.

[Pea00] Judea Pearl. *Causality: Models, Reasoning, and Inference*. Cambridge University Press, March 2000.

[Pen55] R. Penrose. A generalized inverse for matrices. *Mathematical Proceedings of the Cambridge Philosophical Society*, 51:406–413, 1955.

[Pet62] Carl Adam Petri. *Kommunikation mit Automaten*. PhD thesis, Technische Hochschule Darmstadt, Germany, 1962.

[PHdB09] A. Polynikis, S. J. Hogan, and M. di Bernardo. Comparing different ODE modelling approaches for gene regulatory networks. *Journal of Theoretical Biology*, 261(4):511–530, 2009.

[PJRG06] Theodore J. Perkins, Johannes Jaeger, John Reinitz, and Leon Glass. Reverse engineering the Gap gene network of *Drosophila melanogaster*. *PLoS Computational Biology*, 2(5):e51, 2006.

[PK05] Erik Plahte and Sissel Kjøglum. Analysis and generic properties of gene regulatory networks with graded response functions. *Physica D: Nonlinear Phenomena*, 201(1–2):150–176, 2005.

[PK08] Mario Pineda-Krch. GillespieSSA: Implementing the Gillespie stochastic simulation algorithm in R. *Journal of Statistical Software*, 25(12):1–18, 2008.

[PRA05] Mor Peleg, Daniel Rubin, and Russ B. Altman. Using Petri net tools to study properties and dynamics of biological systems. *Journal of the American Medical Informatics Association*, 12(2):181–199, 2005.

[Pro11a] The DREAM Project. Dream6. `http://wiki.c2b2.columbia.edu/dream/index.php/The_DREAM_Project`, June 2011.

[Pro11b] The DREAM Project. Synthetic five-gene network inference (dream2, challenge 3). `http://wiki.c2b2.columbia.edu/dream/index.php/D2c3`, July 2011.

[Puk06] Friedrich Pukelsheim. *Optimal Design of Experiments*, volume 50 of *Classics in Applied Mathematics*. Society for Industrial and Applied Mathematics, Philadelphia, 2006.

[PVM+06] A. A. Poyton, M. S. Varziri, K. B. McAuley, P. J. McLellan, and J. O. Ramsay. Parameter estimation in continuous-time dynamic models using principal differential analysis. *Computers and Chemical Engineering*, 30:698–708, 2006.

[QBd07] Minh Quach, Nicolas Brunel, and Florence d'Alché-Buc. Estimating parameters and hidden variables in non-linear state-space models based on ODEs for biological networks inference. *Bioinformatics*, 23(23):3209–3216, 2007.

[QGID10] Xiaoning Qian, Noushin Ghaffari, Ivan Ivanov, and Edward R. Dougherty. State reduction for network intervention in probabilistic Boolean networks. *Bioinformatics*, 26(24):3098–3104, 2010.

[Rad07] Nicole Radde. *Non-Linear Dynamic Phenomena in Biochemical Networks*. PhD thesis, Universität zu Köln, Germany, 2007.

[Ram96] J. O. Ramsay. Principal differential analysis: Data reduction by differential operators. *Journal of the Royal Statistical Society. Series B (Methodological)*, 58(3):495–508, 1996.

[RAT06] S. Reinker, R. M. Altman, and J. Timmer. Parameter estimation in stochastic biochemical reactions. *IEE Proceedings Systems Biology*, 153(4):168–178, 2006.

[RBKT10] A. Raue, V. Becker, U. Klingmüller, and J. Timmer. Identifiability and observability analysis for experimental design in nonlinear dynamical models. *Chaos*, 20:045105, 2010.

[RCLP07] Andre S. Ribeiro, Daniel A. Charlebois, and Jason Lloyd-Price. *CellLine*, a stochastic cell lineage simulator. *Bioinformatics*, 23(24):3409–3411, 2007.

[Rei67] Christian H. Reinsch. Smoothing by spline functions. *Numerische Mathematik*, 10:177–183, 1967.

[RGG97] Gareth O. Roberts, A. Gelman, and Walter R. Gilks. Weak convergence and optimal scaling of random walk Metropolis algorithm. *Annals of Applied Probability*, 7:110–120, 1997.

[RHCC07] J. O. Ramsay, G. Hooker, D. Campbell, and J. Cao. Parameter estimation for differential equations: a generalized smoothing approach. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 69(5):741–796, 2007.

[RHMP05] Sébastien Rimour, David Hill, Cécile Militon, and Pierre Peyret. GoArrays: highly dynamic and efficient microarray probe design. *Bioinformatics*, 21:1094–1103, 2005.

[Rib10] Andre S. Ribeiro. Stochastic and delayed stochastic models of gene expression and regulation. *Mathematical Biosciences*, 223(1):1–11, 2010.

[Rit08] Daniel Ritter. *Machine Network Learning - Bayesian Inference of Gene Regulatory Networks with Differential Equations using Stochastic Simulation*. Master thesis, University of Heidelberg, Germany, 2008.

[RK07] Nicole Radde and Lars Kaderali. Bayesian inference of gene regulatory networks using gene expression time series data. In Sepp Hochreiter and Roland Wagner, editors, *BIRD*, volume 4414 of *Lecture Notes in Computer Science*, pages 1–15. Springer, 2007.

[RK09] Nicole Radde and Lars Kaderali. Inference of an oscillating model for the yeast cell cycle. *Discrete Applied Mathematics*, 157(10):2285–2295, 2009.

[RKM+09] A. Raue, C. Kreutz, T. Maiwald, J. Bachmann, M. Schilling, U. Klingmüller, and J. Timmer. Structural and practical identifiability analysis of partially observed dynamical models by exploiting the profile likelihood. *Bioinformatics*, 25(15):1923–1929, 2009.

[RKM+11] A. Raue, C. Kreutz, T. Maiwald, U. Klingmüller, and J. Timmer. Addressing parameter identifiability by mode-based experimentation. *IET Systems Biology*, 5(2):120–130, 2011.

[RPCG03] Muruhan Rathinam, Linda R. Petzold, Yang Cao, and Daniel T. Gillespie. Stiffness in stochastic chemically reacting systems: The implicit tau-leaping method. *Journal of Chemical Physics*, 119(24):12784, 2003.

[RPT+06] Arjun Raj, Charles S. Peskin, Daniel Tranchina, Diana Y. Vargas, and Sanjay Tyagi. Stochastic mRNA synthesis in mammalian cells. *PLoS Biology*, 4(10):e309, 2006.

[RR01] Gareth O. Roberts and Jeffrey. S. Rosenthal. Optimal scaling of various Metropolis-Hastings algorithms. *Statistical Science*, 16(4):351–367, 2001.

[RRK06] Grzegorz A. Rempala, Kenneth S. Ramos, and Ted Kalbfleisch. A stochastic model of gene transcription: An application to L1 retrotransposition events. *Journal of Theoretical Biology*, 242(1):101–116, 2006.

[Rya96] Conor Ryan. *Reducing Premature Convergence in Evolutionary Algorithms*. PhD thesis, University College Cork, Ireland, 1996.

[RZ06] Marc R. Roussel and Rui Zhu. Validation of an algorithm for delay stochastic simulation of transcription and translation in prokaryotic gene expression. *Physical Biology*, 3:274–284, 2006.

[RZK06] Andre Ribeiro, Rui Zhu, and Stuart A. Kauffman. A general modeling strategy for gene regulatory networks with stochastic dynamics. *Journal of Computational Biology*, 13(9):1630–1639, 2006.

[Sav70] M. A. Savageau. Biochemical systems analysis: III. Dynamic solutions using a power-law approximation. *Journal of Theoretical Biology*, 26(2):215–226, 1970.

[Sav91] M. A. Savageau. Biochemical systems theory: Operational differences among variant representations and their significance. *Journal of Theoretical Biology*, 151(4):509–530, 1991.

[SB05] Josef Stoer and Roland Bulirsch. *Numerische Mathematik 2*. Springer, Heidelberg, Germany, fifth edition, 2005.

[SB07] Thomas Schlitt and Alvis Brazma. Current approaches to gene regulatory network modelling. *BMC Bioinformatics*, 8(Suppl 6):S9, 2007.

[SBSW07] L. Jason Steggles, Richard Banks, Oliver Shaw, and Anil Wipat. Qualitatively modelling and analysing genetic regulatory networks: a Petri net approach. *Bioinformatics*, 23(3):336–343, 2007.

[SCJ05] Henning Schmidt, Kwang-Hyun Cho, and Elling W. Jacobsen. Identification of small scale biochemical networks based on general type system perturbations. *The FEBS Journal*, 272(9):2141–2151, 2005.

[SD01] Almut Schulze and Julian Downward. Navigating gene expression using microarrays - a technology review. *Nature Cell Biology*, 3:E190–E195, 2001.

[SD10] Ilya Shmulevich and Edward R. Dougherty. *Probabilistic Boolean Networks: The Modeling and Control of Gene Regulatory Networks*. Society for Industrial and Applied Mathematics, Philadelphia, USA, 2010.

[SDKZ02] Ilya Shmulevich, Edward R. Dougherty, Seungchan Kim, and Wei Zhang. Probabilistic Boolean networks: a rule-based uncertainty model for gene regulatory networks. *Bioinformatics*, 18(2):261–274, 2002.

[SGH+03] Ilya Shmulevich, Ilya Gluhovsky, Ronaldo F. Hashimoto, Edward R. Dougherty, and Wei Zhang. Steady-state analysis of genetic regulatory networks modelled by probabilistic Boolean networks. *Comparative and Functional Genomics*, 4:601–608, 2003.

[SGS00] Peter Spirtes, Clark Glymour, and Richard Scheines. *Causation, Prediction, and Search*. MIT Press, Cambridge, MA, second edition, 2000.

[SGT03] Lawrence F. Shampine, I. Gladwell, and S. Thompson. *Solving ODEs with MATLAB*. Cambridge University Press, New York, USA, 2003.

[Sha48] C. E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27:379–423,623–656, 1948.

[SHS06] Christian Spieth, Nadine Hassis, and Felix Streichert. Comparing mathematical models on the problem of network inference. In *Proceedings of the 8th annual conference on Genetic and evolutionary computation*, GECCO '06, pages 279–286, New York, NY, USA, 2006. ACM.

[SJ06] Chiara Sabatti and Gareth M James. Bayesian sparse hidden components analysis for transcription regulation networks. *Bioinformatics*, 22(6):739–746, 2006.

[SJOW07] Daniel Schultz, Eshel Ben Jacob, José N. Onuchic, and Peter G. Wolynes. Molecular level stochastic model for competence cycles in *Bacillus subtilis*. *Proceedings of the National Academy of Sciences*, 104(45):17582–17587, 2007.

[SKK04] Eduardo Sontag, Anatoly Kiyatkin, and Boris N. Kholodenko. Inferring dynamic architecture of cellular networks using time series of gene expression, protein and

metabolite data. *Bioinformatics*, 20(12):1877–1886, 2004.

[SMC07] Gustavo Stolovitzky, Don Monroe, and Andrea Califano. Dialogue on reverse engineering assessment and methods: The DREAM of high throughput pathway inference. *Annals of the New York Academy of Sciences*, 1115:1–22, 2007.

[SME+09] Sylvia Streit, Christoph W. Michalski, Mert Erkan, Jörg Kleeff, and Helmut Friess. Northern blot analysis for detection and quantification of RNA in pancreatic cancer cells and tissues. *Nature Protocols*, 4:37–43, 2009.

[SORS06] Juliane Schäfer, Rainer Opgen-Rhein, and Korbinian Strimmer. Reverse engineering genetic networks using the GeneNet package. *R News*, 6/5:50–53, 2006.

[Sou75] E. M. Southern. Detection of specific sequences among DNA fragments separated by gel electrophoresis. *Journal of Molecular Biology*, 98(3):503–517, 1975.

[SR97] Lawrence F. Shampine and Mark W. Reichelt. The MATLAB ODE suite. *SIAM Journal on Scientific Computing*, 18(1):1–22, 1997.

[SS05a] Juliane Schäfer and Korbinian Strimmer. An empirical Bayes approach to inferring large-scale gene association networks. *Bioinformatics*, 21(5):754–764, 2005.

[SS05b] Juliane Schäfer and Korbinian Strimmer. Learning large-scale graphical Gaussian models from genomic data. *AIP Conference Proceedings*, 776(1):263–276, 2005.

[SSC+10] Diego Fernández Slezak, Cecilia Suárez, Guillermo A. Cecchi, Guillermo Marshall, and Gustavo Stolovitzky. When the optimal is not the best: Parameter estimation in complex biological models. *PLoS ONE*, 5(10):e13283, 2010.

[SSDB95] Mark Schena, Dari Shalon, Ronald W. Davis, and Patrick O. Brown. Quantitative monitoring of gene expression patterns with a complementary DNA microarray. *Science*, 270(5235):467–470, 1995.

[SSDK03] Joshua M. Stuart, Eran Segal, Daphne Doller, and Stuart K. Kim. A gene-coexpression network for global discovery of conserved genetic modules. *Science*, 302, 2003.

[SSSZ04] Christian Spieth, Felix Streichert, Nora Speer, and Andreas Zell. A memetic inference method for gene regulatory networks based on S-systems. In *Proceedings of the IEEE Congress on Evolutionary Computation*, pages 152–157, 2004.

[SSSZ05] Christian Spieth, Felix Streichert, Nora Speer, and Andreas Zell. Multiobjective model optimization for inferring gene regulatory networks. In *Proceedings of the Conference on Evolutionary Multi-Criterion Optimization*, volume 3410 of *Lecture Notes on Computer Science*, pages 607–620, 2005.

[SST07] Florian Steinke, Matthias Seeger, and Koji Tsuda. Experimental design for efficient identification of gene regulatory networks using sparse Bayesian models. *BMC Systems Biology*, 1:51, 2007.

[SSZ+98] Paul T. Spellman, Gavin Sherlock, Michael Q. Zhang, Vishwanath R. Iyer, Kirk Anders, Michael B. Eisen, Patrick O. Brown, David Botstein, and Bruce Futcher. Comprehensive identification of cell cycle-regulated genes of the yeast *saccharomyces cerevisiae* by microarray hybridization. *Molecular Biology of the Cell*, 9(12):3273–3297, 1998.

[Sto05] Roland B. Stoughton. Applications of DNA microarrays in biology. *Annual Review of Biochemistry*, 74:53–82, 2005.

[SW97] Paola Sebastiani and Henry P. Wynn. Bayesian experimental design and Shannon information. In *1997 Proceedings of the Section on Bayesian Statistical Science*, pages 176–181, 1997.

[SW00] Paola Sebastiani and Henry P. Wynn. Maximum entropy sampling and optimal Bayesian experimental design. *Journal of the Royal Statistical Society. Series B (Methodological)*, 62(1):145–157, 2000.

[SZK+08] Ravi Vijaya Satya, Nela Zavaljevski, Kamal Kumar, Elizabeth Bode, Susana Padilla, leonard Wasieloski, Jeanne Geyer, and Jacques Reifman. *In silico* microarray probe design for diagnosis of multiple pathogens. *BMC Genomics*, 9:496, 2008.

[Tan89] Reiko Tanese. Distributed genetic algorithms. In J. David Schaffer, editor, *Proceedings of the third international conference on Genetic algorithms*, pages 434–439, San Francisco, CA, USA, 1989. Morgan Kaufmann Publishers Inc.

[tB06] Cajo ter Braak. A Markov chain Monte Carlo version of the genetic algorithm differential evolution: easy Bayesian computing for real parameter spaces. *Statistics and Computing*, 16(3):239–249, 2006.

[TH02] Hiroyuki Toh and Katsuhisa Horimoto. Inference of a genetic network by a combined approach of cluster analysis and graphical Gaussian modeling. *Bioinformatics*, 18(2):287–297, 2002.

[TMIY01] Masaru Taniguchi, Katsuyuki Miura, Hiroshi Iwao, and Shinya Yamanaka. Quantitative assessment of DNA microarrays - comparison with northern blot analyses. *Genomics*, 71(1):34–39, 2001.

[TMS10] Mark K. Transtrum, Bejamin B. Machta, and James P. Sethna. Why are nonlinear fits to data so challenging? *Physical Review Letters*, 104:060201, 2010.

[Tra96] Paul Trayhurn. Northern blotting. *Proceedings of the Nutrition Society*, 55:583–589, 1996.

[TS10] Tina Toni and Michael P. H. Stumpf. Simulation-based model selection for dynamical systems in systems and population biology. *Bioinformatics*, 26(1):104–110, 2010.

[TSG79] Harry Towbin, Theophil Staehelin, and Julian Gordon. Electrophoretic transfer of proteins from polyacrylamide gels to nitrocellulose sheets: Procedure and some applications. *PNAS*, 76(9):4350–4354, 1979.

[TUM11] Gabriel Terejanu, Rochan R. Upadhyay, and Kenji Miki. Bayesian experimental design for the active nitridation of graphite by atomic nitrogen. *ArXiv e-prints*, July 2011. `arXiv:1107.1445v1 [physics.data-an]`.

[TXGB07] Tianhai Tian, Songlin Xu, Junbin Gao, and Kevin Burrage. Simulated maximum likelihood method for estimating kinetic rates in gene expression. *Bioinformatics*, 23(1):84–91, 2007.

[Var82] J. M. Varah. A spline least squares method for numerical parameter estimation in differential equations. *SIAM Journal on Scientific & Statistical Computing*, 3(1):28–46, 1982.

[VCJL10] Alan Veliz-Cuba, Abdul Salam Jarrah, and Reinhard Laubenbacher. Polynomial algebra of discrete models in systems biology. *Bioinformatics*, 26(13):1637–1643, 2010.

[VCV+08] Marco Vilela, I-Chun Chou, Susana Vinga, Ana TR. Vasconcelos, Eberhard O. Voit, and Jonas S. Almeida. Parameter optimization in S-system models. *BMC Systems Biology*, 2:35, 2008.

[Vej10] Tomas Vejchodsky. GillespieSSA for Matlab. `http://www.math.cas.cz/~vejchod/gillespiessa/gillespiessa.html`, March 2010.

[VG08] Vladislav Vyshemirsky and Mark Girolami. Bayesian ranking of biochemical

system models. *Bioinformatics*, 24(6):833–839, 2008.

[VP90] Thomas Verma and Judea Pearl. Equivalence and synthesis of causal models. In Piero P. Bonissone, Max Henrion, Laveen N. Kanal, and John F. Lemmer, editors, *UAI*, pages 255–270. Elsevier, 1990.

[vSWR00] Eugene P. van Someren, Lodewyk F. A. Wessels, and Marcel J. T. Reinders. Linear modeling of genetic networks from experimental data. In Philip E. Bourne, Michael Gribskov, Russ B. Altman, Nancy Jensen, Debra A. Hope, Thomas Lengauer, Julie C. Mitchell, Eric D. Scheeff, Chris Smith, Shawn Strande, and Helge Weissig, editors, *ISMB*, pages 355–366. AAAI, 2000.

[VVF08] Heather D. VanGuilder, Kent E. Vrana, and Willard M. Freeman. Twenty-five years of quantitative PCR for gene expression analysis. *BioTechniques*, 44(5):619–626, 2008.

[Wal49] Abraham Wald. Note on the consistency of the maximum likelihood estimate. *Annals of Mathematical Statistics*, 20(4):595–601, 1949.

[WBT+05] Leor S. Weinberger, John C. Burnett, Jared E. Toettcher, Adam P. Arkin, and David V. Schaffer. Stochastic gene expression in a lentiviral positive-feedback loop: HIV-1 tat fluctuations drive phenotypic diversity. *Cell*, 122(2):169–182, 2005.

[WBT+09] Dominik M. Wittmann, Florian Blöchl, Dietrich Trümbach, Wolfgang Wurst, Nilima Prakash, and Fabian J. Theis. Spatial analysis of expression patterns predicts genetic interactions at the mid-hindbrain boundary. *PLoS Computational Biology*, 5(11):e1000569, 2009.

[WCD05] Junbai Wang, Leo Wang-Kit Cheung, and Jan Delabie. New probabilistic graphical models for genetic regulatory networks studies. *Journal of Biomedical Informatics*, 38(6):443–455, 2005.

[WEG08] Ethan P. White, Brian J. Enquist, and Jessica L. Green. On estimating the exponent of power-law frequency distributions. *Ecology*, 89(4):905–912, 2008.

[WH08] Moira S. Walker and Thomas A. Hughes. Messenger RNA expression profiling using DNA microarray technology: Diagnostic tool, scientific analysis or uninterpretable data? (Review). *International Journal of Molecular Medicine*, 21:13–17, 2008.

[Wil06] Darren J. Wilkinson. *Stochastic Modelling for Systems Biology*. Chapman & Hall/CRC, Boca Raton, FL, USA, 2006.

[Wil09] Darren J. Wilkinson. Stochastic modelling for quantitative description of heterogeneous biological systems. *Nature Reviews Genetics*, 10:122–133, 2009.

[WKB05] Cecily J. Wolfe, Isaac S. Kohane, and Atul J. Butte. Systematic survey reveals general applicability of "guilt-by-association" within gene coexpression networks. *BMC Bioinformatics*, 6:227, 2005.

[WKSR+09] Dominik M. Wittmann, Jan Krumsiek, Julio Saez-Rodriguez, Douglas A. Lauffenburger, Steffen Klamt, and Fabian J. Theis. Transforming Boolean models to continuous models: methodology and application to T-cell receptor signaling. *BMC Systems Biology*, 3:98, 2009.

[Wor11] Petri Nets World. Petri nets tool database. `http://www.informatik.uni-hamburg.de/TGI/PetriNets/tools/db.html`, November 2011.

[WZV+04] Anja Wille, Philip Zimmermann, Eva Vranová, Andreas Fürholz, Oliver Laule, Stefan Bleuler, Lars Hennig, Amela Prelić, Peter von Rohr, Lothar Thiele, Eckart Zitzler, Wilhelm Gruissem, and Peter Bühlmann. Sparse graphical Gaus-

sian modeling of the isoprenoid gene network in *Arabidopsis thaliana*. *Genome Biology*, 5:R92, 2004.

[XCP+10] Dan Xie, Chieh-Chun Chen, Leon M. Ptaszek, Shu Xiao, Xiaoyi Cao, Fang Fang, Huck H. Ng, Harris A. Lewin, Chad Cowan, and Sheng Zhong. Rewirable gene regulatory networks in the preimplantation embryonic development of three mammalian species. *Genome Research*, 20(6):804–815, 2010.

[YTC02] M K. Stephen Yeung, Jesper Tegnér, and James J. Collins. Reverse engineering gene networks using singular value decomposition and robust regression. *Proceedings of the National Academy of Sciences*, 99(9):6163–6168, 2002.

[YXR+06] Ji Yu, Jie Xiao, Xiaojia Ren, Kaiqin Lao, and X. Sunney Xie. Probing gene expression in live cells, one protein molecule at a time. *Science*, 311(5767):1600–1603, 2006.

[YY71] Gad Yagil and Ezra Yagil. On the relation between effector concentration and the rate of induced enzyme synthesis. *Biophysical Journal*, 11:11–27, 1971.

[ZDJJ06] Yu Zhang, Zhidong Deng, Hongshan Jiang, and Peifa Jia. Gene regulatory network construction using dynamic Bayesian network (DBN) with structure expectation maximization (SEM). In Guoying Wang, James Peters, Andrzej Skowron, and Yiyu Yao, editors, *Rough Sets and Knowledge Technology*, volume 4062 of *Lecture Notes in Computer Science*, pages 402–407. Springer Berlin / Heidelberg, 2006.

[ZMC10] Pietro Zoppoli, Sandro Morganella, and Michele Ceccarelli. TimeDelay-ARACNE: Reverse engineering of gene networks from time-course data by an information theoretic approach. *BMC Bioinformatics*, 11:154, 2010.

Publications

Parts of the work presented in this thesis has been published before. Here is a list of these publications.

Articles

Johanna Mazur and Lars Kaderali. Bayesian experimental design for the inference of gene regulatory networks. In Stefan Kramer and Neil Lawrence, editors, *Proceedings of the Fifth International Workshop on Machine Learning in Systems Biology, Vienna, Austria, July 20–21, 2011*, pages 54–58

Johanna Mazur and Lars Kaderali. The importance and challenges of Bayesian parameter learning in systems biology. In H. G. Bock, T. Carraro, W. Jäger, S. Körkel, R. Rannacher and J. P. Schlöder, editors, *Model Based Parameter Estimation: Theory and Applications*, volume 3 of *Contributions in Mathematical and Computational Sciences*. Springer, 2011, in press.

Johanna Mazur, Daniel Ritter, Gerhard Reinelt and Lars Kaderali. Reconstructing non-linear dynamic models of gene regulation using stochastic sampling. *BMC Bioinformatics*, 10:448, 2009.

Johanna Mazur and Lars Kaderali. Bayesian experimental design for parameter estimation with nonlinear dynamic models in systems biology. *In preparation for the submission to Bioinformatics.*

Talks

Johanna Mazur and Lars Kaderali. *Bayesian Experimental Design for the Inference of Gene Regulatory Networks.* International Workshop on Machine Learning in Systems Biology (MLSB) 2011, Vienna, Austria, July 20–21, 2011

Johanna Mazur. *Bayesian experimental design for the inference of ordinary differential equations models in systems biology.* HGS MathComp Annual Colloquium 2010, Heidelberg, Germany, November 19, 2010

Johanna Mazur and Lars Kaderali. *Rekonstruktion molekularer Netzwerke mit regularisierten Schätzverfahren.* GMDS Annual Meeting 2010, Mannheim, Germany, September 5–9, 2010

Johanna Mazur, Daniel Ritter, Gerhard Reinelt and Lars Kaderali. *Reconstructing Nonlinear Dynamic Models of Gene Regulation using Stochastic Sampling.* Model Based Parameter Estimation - Theory and Applications (MPTA), Interdisciplinary Center for Scientific Computing (IWR), Heidelberg, Germany, July 15–17, 2009

Posters

Johanna Mazur and Lars Kaderali. *Optimum Bayesian Experimental Design for the Inference of Ordinary Differential Equations Models in Systems Biology.* International Conference on Systems Biology (ICSB) 2011, Heidelberg/Mannheim, Germany, August 28 – September 1, 2011

Johanna Mazur, Daniel Ritter, Gerhard Reinelt and Lars Kaderali. *Reverse Engineering of Gene Regulatory Networks using Nonlinear Dynamics and Stochastic Sampling.* International Conference on Systems Biology of Human Disease (SBHD) 2010, Boston, USA, June 16–18, 2010

Johanna Mazur, Daniel Ritter, Gerhard Reinelt and Lars Kaderali. *Reverse Engineering of Gene Regulatory Networks with a Nonlinear ODE-Model Embedded into a Bayesian Framework.* German Conference on Bioinformatics (GCB) 2009, Halle (Saale), Germany, September 28–30, 2009

Johanna Mazur, Daniel Ritter, Gerhard Reinelt and Lars Kaderali. *Reverse Engineering of Gene Regulatory Networks with a Nonlinear ODE-Model by means of Stochastic Sampling.* German Symposium on Systems Biology, Heidelberg, Germany, May 12–15, 2009

Bettina Knapp, Johanna Mazur and Lars Kaderali. *Inferring Networks from High-Throughput Data.* 2^{nd} ViroQuant retreat, Asselheim/Grünstadt, Germany, February 16–17, 2009

Lars Kaderali, Johanna Mazur and Daniel Ritter. *Statistical inference of biochemical networks with delay differential equations.* International Conference on Systems Biology (ICSB) 2008, Gothenburg, Sweden, August 22–28, 2008

Daniel Ritter, Johanna Mazur, Gerhard Reinelt and Lars Kaderali. *Reconstructing Gene Regulatory Networks using Differential Equations in a Stochastic Framework.* Conference on Systems Biology of Mammalian Cells (SBMC) 2008, Dresden, Germany, May 22–24, 2008

Johanna Mazur, Daniel Ritter and Lars Kaderali. *Reverse Engineering of Gene Regulatory Networks using Bayes' regularized Differential Equations.* 1^{st} ViroQuant retreat, Asselheim/Grünstadt, Germany, April 21–22, 2008