TECHNISCHE UNIVERSITÄT MÜNCHEN

Lehrstuhl für Nachrichtentechnik

Communication theory applied to selected problems in computational genetics

Janis Dingel

Vollständiger Abdruck der von der Fakultät für Elektrotechnik und Informationstechnik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktor–Ingenieurs

genehmigten Dissertation.

Vorsitzender: Univ.–Prof. Dr.–Ing. J. Eberspächer

Prüfer der Dissertation:

1. Univ.–Prof. Dr.–Ing. Dr.–Ing. E.h. J. Hagenauer (i.R.)
2. Prof. O. Milenkovic, Ph.D.,
   University of Illinois at Urbana Champaign, USA

Die Dissertation wurde am 25.06.2009 bei der Technischen Universität München eingereicht und durch die Fakultät für Elektro- und Informationstechnik am 06.10.2009 angenommen.

DEDICATED TO
LONI ELISABETH DINGEL
*23.05.1950 - †23.01.2010

Preface

This thesis is the result of my work as a research and teaching assistant at the Institute for Communications Engineering of the Technische Universität München. There are a number of people I would like to thank.

First of all, I would like to thank Professor Dr.–Ing. Dr.–Ing. E.h. Joachim Hagenauer for giving me the opportunity to work in his group and for his continuous support throughout the course of my research. His interest and advice have been essential to this work.

Interdisciplinary, collaborative work can only be successful with strong partners from the other discipline involved. Therefore, I am very grateful to PD Dr. Jakob Müller, Dr. Vladimir Kuryshev, and Dr. Jürgen Zech for their invaluable comments and our fruitful discussions during our regular ComInGen group meetings, and their patience in explaining biology to me.

Chapter 7 of this work is the result of my collaboration with Professor Olgica Milenkovic from the University of Illinois at Urbana Champaign. My visit at the Coordinated Science Laboratory was an unforgettable experience from which I have benefited a lot. I would like to thank Professor Milenkovic for this invitation, her advice, and for serving as the co-examiner of this thesis.

I am very thankful to all my former colleagues and students for the enjoyable atmosphere and the inspiring working environment at the LNT. I thank Pavol Hanus and Dr. Johanna Weindl for our scientific discussions and for proofreading this work. Special thanks to Stehphan Hellerbrand, Manfred Danzer, Martin Kontny and Professor Dr.-Ing. habil. Günter Söder for making the work in the system administration group enjoyable. I would also like to thank Professor Dr.–Ing. Norbert Hanik for the possibility to finish my work at the Institute for Communications Engineering.

Finally I would like to thank my family, my friends, and especially Laura for her love and support.

Munich, February 2010 Janis Dingel

Contents

Zusammenfassung

Kommunikations- und Informationstheorie bilden heute die Grundlage für den Entwurf digitaler Übertragungssysteme. Interdisziplinäre Zusammenarbeit zwischen Kommunikationsingenieuren und Biologen baut darauf auf, dass die zelluläre Informationsverarbeitung auf digitalen Signalen (DNA) basiert. In diesem Zusammenhang stellt unsere Arbeit kommunikationstheoretische Ansätze für ausgewählte Probleme der Genetik vor. Wir entwickeln Algorithmen zur Detektion konservierter Bereiche in multiplen, ausgerichteten DNA-Sequenzen und zur Rekonstruktion von Faltungscodes in konservierten DNA Sequenzen und entwerfen auf Listen-Decodierung basierende Methoden für die Inferenz von Gennetzwerkmodellen. Unsere Algorithmen werden zunächst mittels Simulationen evaluiert und schließlich auf genbiologische Daten aus Datenbanken angewandt. Diese Arbeit zeigt, dass der kommunikationstheoretische Ansatz wichtige Erkenntnisse für biologische Problemstellungen liefern kann.

Abstract

Communication and information theory provide the framework for the design of digital systems. Recent collaboration between communication engineers and biologists is based on the observation that information processing in living cells has a digital basis (DNA). Within this framework, our thesis approaches reverse engineering problems arising in computational genetics from a communication theoretic point of view. We outline algorithms for unbiased detection of conserved regions in multiple DNA sequence alignments and the reverse engineering of convolutional codes suitable to detect coding structure in DNA sequences. We use list-decoding of Reed-Muller codes for the inference of dynamics in stochastic gene network models. The algorithms are evaluated using simulations and subsequently applied to genomic data obtained from databases. Our work shows how insights to biological problems can be gained by approaching them from a communication theoretic perspective.

List of Figures

1

Introduction

Deoxyribonucleic acid (DNA) is the primary carrier of the information encoding the development and functioning of *all* living organisms. In 1953, Watson and Crick (with help of others [Bar03]) discovered the molecular structure of the DNA [CW53]. They showed that the DNA consists of two strands, composed of four different molecules called *nucleotides*, that are joined by hydrogen bonds, and twisted in the shape of a double helix. The key observation motivating this thesis is that the genetic information of any organism is encoded as a long sequence of only four possible nucleotides (Adenine, Cytosine, Guanine, and Thymine). More abstract, DNA can thus be represented as a discrete-time, discrete-valued signal defined over a quarternary alphabet. Therefore, cellular information processing has a *digital basis*.

In 2001, almost fifty years after the structure of DNA had been discovered, the *Human Genome Project* (HGP) announced the first release of the complete sequence of the human genome [LLB$^+$01]. It was expected that the availability of the whole genome sequence and identification of all genes would bring biologists close to understanding the complex nature of organism development. However, quite the opposite was the case: the outcomes of the project raised many further questions. Unexpected results included the relatively small number of genes in the human genome compared to other organisms, long repeat sequences of DNA, and the large fraction of non-genic DNA (at this time wrongly dismissed as *junk DNA*). The lesson learned from the HGP was that merely identifying all genes would not be sufficient to understand development of organisms and functioning of cells.

Meanwhile, rapid advances in DNA sequence technology enabled whole genome sequencing of more and more species allowing for a comparative analysis of sequence data. Surprisingly, it was revealed that there was a significant amount of *junk DNA* in the human genome that was almost identical in evolutionary very distant species (such as mouse, rat, and fish) [DRA05]. This did not conform with the *junk* hypothesis since sequence conservation is a strong indicator for natural selection and thus functionality. The function of most of these conserved regions is not yet understood. Mattick et al. suggested that they form a critical hidden layer of gene regulation in complex organisms [Mat07,MM06]. Once the paradigm of non-coding DNA (=*junk*) vs. coding DNA (=functional) was no longer valid, it was clear that further great efforts would be required to obtain a functional map of the genome.

The ENCODE (**ENC**yclopedia **O**f **D**NA **E**lements) project, launched in 2003 by *The National Human Genome Research Institute*, aims at identifying all functional elements in the human genome sequence. In the pilot phase, the ENCODE consortium tested and compared existing methods to rigorously analyze a defined portion of 1% of the human genome sequence [The07, MCA+07]. Again, the project led to surprises [Che07]: previously, it was believed that only the information encoded in genes would be read out from the DNA (*transcribed* in terminology of biologists). Results from the ENCODE project, however, suggest that in fact almost 80% of the whole DNA is transcribed [Che07]. Although it is not known whether all these *transcripts* are functional, there is hence strong indication that the traditional gene-centric perspective is invalid for higher organisms [Pen07].

Even identifying all functional elements in DNA will not suffice to understand life. Individual components do often not itself perform any function but interact with other components and molecules, forming a highly complex regulatory network that is ultimately responsible for an organism's form and function [Alo07]. All these findings suggest that we are just beginning to understand the complexity of life at the molecular scale, and that approaches beyond those developed in traditional biology may be required to decipher the principles of how cells process the digital information represented by DNA.

Today, enormous amounts of biological data (such as DNA sequences and whole genomes) are made available through public databases. Analytical and computational approaches for biological data analysis are now well established. Over the last decades, molecular biology has evolved into a highly interdisciplinary science. *Bioinformatics* is the field of science in which biology, computer science, mathematics and statistics merge into a single discipline. Recently, *systems biology* emerged as the analysis of components and the interactions among them all as a part of a *system* instead of studying their specific function, thereby collaborating with scientists from other disciplines such as mathematics, physics and engineering.

In 1948, Claude E. Shannon's publication of *A mathematical theory of communication* [Sha48] laid the foundation of modern digital communication theory. Based on the theoretical framework termed *information theory* provided by Shannon, communication engineers were very successful in developing optimal methods and systems to faithfully process and transmit digital information [CHIW98]. Since cellular information processing has a digital basis, methods from digital communication and information theory have the potential to lead to major breakthroughs in the field of molecular biology. On the other hand, new insights on the design of communication systems is expected to be gained from analyzing biological systems, that achieve astonishingly robust storage and transmission of digital information in a highly noisy and fragile environment.

Despite these facts, communication engineers and information theorists only recently began to foster cooperation with biologists. To name a few recent contributions: Hagenauer et al. modeled the relationship between certain positions in DNA and diseases like Parkinson as a communication channel and applied information theoretic measures to infer their relationships [DGH+06, DSJM05, GDHM05]. The same group showed how to apply data compression schemes to infer the relationship of species [DHHM05, KH08]. Szpankowski et al. extended these methods to detect *alternative splice sites* in genes [AKL+07]. Milenkovic introduced coding theoretic approaches to analyze *gene networks* and *RNA secondary structures* [Mil06, MV04]. Rissanen's *minimum description length principle* was applied recently in numerous biological modeling frameworks [DTA08, RGH+07].

Within this framework, this thesis shows how to approach certain *reverse engineering* problems, arising in molecular biology, from a communication theoretic view. In a typical bioinformatics problem setting, one is given a set of measured observations from a biological system and aims to infer the biological *systematics* explaining the data. In a first step, an abstract model of the system needs to be built based on prior biological knowledge, obtained from theoretical considerations or experimental evidence. In a second step, the model needs to be theoretically analyzed, important system parameters must be specified, and identification algorithms developed. Finally, results must be biologically interpreted and compared to those obtained with other methods or models. We refer to such an approach as *reverse engineering*.

Briefly, the outline of this thesis is as follows: since we operate at the interface of two disciplines, Chapters 2-4 provide the necessary general theoretical background, in a way that, hopefully, should be understandable to both communities. Chapters 5-7 present the reverse engineering frameworks, methods, and obtained results. A brief introduction to the specific theoretical background is given at the beginning of each chapter. Finally, Chapter 8 presents the conclusion. The remainder of this chapter describes the outline in more detail:

Chapter 2 introduces the basic notions of information and coding theory, and the statistical methods applied in this thesis. Examples are given throughout the chapter, relating to both, information theory and DNA sequence analysis.

Chapter 3 can be regarded as an introduction to molecular biology for communication engineers. Prior knowledge of the reader is not expected. We review how genetic information is processed in the cell. Similar to a communication system, we split a cellular information processing system into layers: on the physical layer, information is encoded into DNA, and a set of protocols defines how to process this molecule at a high fidelity rate. On a functional level, the DNA is organized into distinct sections - the genes, carrying the fundamental information. Finally, genes exchange messages, and this set of interactions allows for the formation of an information network layer. However, we shall also emphasize fundamental differences that exist between technical and cellular systems.

Chapter 4 provides important tools from Bioinformatics which shall mainly be used in Chapter 5. The main concepts of single and multiple sequence analysis are presented from an information theoretic perspective. Interestingly, a main result in DNA sequence analysis is found to be related to the *channel coding theorem*.

In **Chapter 5**, we first present likelihood methods for reconstructing models of multiple DNA sequence evolution, referred to as *phylogenetic systems*. Subsequently, we focus on the identification of conserved regions in genomes of multiple species. Such regions are candidates for functional regions since conservation implies natural selection. We develop a new approach based on maximum likelihood and the Kullback-Leibler divergence and compare our algorithm to state-of-the art methods. An ENCODE region is analyzed using our method and results are compared to those obtained from the ENCODE project.

Recently, Battail argued, based on theoretical considerations, that such highly conserved regions could only be explained by an error correcting code in the genome [Bat08]. **Chapter 6** is an attempt to systematically approach this hypothesis. We review previous work and critically discuss Battail's claim. Then we focus on the technical code reverse engineering problem. The main contribution is the presentation of probabilistic algorithms for the inference of convolutional encoders from a noisy data stream. We also apply our method to conserved DNA sequences and discuss why application to sequence data is difficult.

Code reverse engineering problems also arise when inferring the dynamics of gene networks under algebraic expression models. In **Chapter 7**, we show how this problem relates to coding theory and, in particular, to the decoding of generalized Reed-Muller codes. We present a list-decoding approach to address reverse engineering in the noisy data and small sample size setting. Our algorithm is applied to the gene network of the bacterium *E. coli*.

Chapter 8 summarizes the main contributions of this thesis. Directions of future research are given at the end of Chapters 5,6 and 7.

Parts of this thesis were published in journals and conference proceedings as listed in Appendix A.

2

Information theory and statistical methods

We briefly introduce the basic concepts, and statistical methods and models applied in this thesis: Section 2.1 provides the guidelines for the mathematical notation used. Sections 2.2 and 2.3 review basic definitions and principles of information and coding theory, referred to throughout this work. Markov chains are introduced in Section 2.4. We present continuous Markov processes in more detail since they are important models in DNA sequence analysis (cf. Chapters 4 and 5) yet rarely covered in the bioinformatics literature in a coherent way. Maximum likelihood estimation, hidden Markov models, and expectation maximization (applied in Chapters 5, 4 and 6) are reviewed in Sections 2.5, 2.4.5, and 2.6, respectively.

2.1 Mathematical notation

- ⋆ $\mathbb{R}$ and $\mathbb{R}^+$ denote the set of real numbers and the set of non-negative real numbers, while $\mathbb{Z}$ and $\mathbb{Z}^+$ denote the set of integers and positive integers, respectively.
- ⋆ $\mathbb{F}_q$ denotes the finite field of order q. Addition in $\mathbb{F}_2$ is denoted by the symbol $\oplus$.
- ⋆ Calligraphic capitals $\mathcal{A}$ denote sets and alphabets, $|\mathcal{A}|$ denotes cardinality of a set.
- ⋆ $\mathbb{F}^N$ is the N-dimensional vector field over $\mathbb{F}$, $\mathcal{A}^N$ denotes the set of sequences with elements from $\mathcal{A}$ of length N.
- ⋆ The probability for the event that the realization of random variable X is x is denoted by $P(\mathsf{X} = x)$ or simply $P(x)$.
- ⋆ Sans serif capitals X indicate random variables (r. v.) with corresponding realizations x. A probability density function (pdf) is denoted as $p_{\mathsf{X}}(x)$. If X is a discrete r. v., $p_{\mathsf{X}}(x)$ denotes the probability mass (pmf) function. We shall often simply write $p(x)$ or p_{X}.
- ⋆ The notation $\mathsf{X} \sim p(x)$ or $x \sim p(x)$ means that the random variable X is distributed according to $p(x)$.
- ⋆ A distribution $p(x;\theta)$ depends on a deterministic parameter θ, opposed to $p(x|\theta)$ which denotes a conditional pdf in case θ is the realization of a random variable.
- ⋆ $\{\mathsf{X}_n\}_{n\geq 0}$ denotes a random process indexed by natural numbers n, $\{\mathsf{X}_t\}_{t\geq 0}$ denotes

a continuous random process with $t \in \mathbb{R}^+$.
- ⋆ $\hat{x}$ is the estimate of x.
- ⋆ Vectors $\boldsymbol{x}$ and matrices $\boldsymbol{X}$ are denoted by bold small and capital letters, respectively.
- ⋆ $[\boldsymbol{x}]_i$ denotes the ith entry of $\boldsymbol{x}$ and $[\boldsymbol{X}]_{ij}$ denotes the entry in the ith row and jth column of matrix $\boldsymbol{X}$.
- ⋆ A joint sub- and superscript $\boldsymbol{x}_m^n$ denotes the sub-vector $[x_m, x_{m+1}, ..., x_n]$ of $\boldsymbol{x} = [x_1, .., x_N]$, where $0 \leq m \leq n \leq N$.
- ⋆ The abbreviations *iff* and *iid* mean *if and only if* and *independently and identically distributed*, respectively.

2.2 Information theory

Information theory developed by Shannon in his seminal paper [Sha48] explores the fundamental limits of information processing. In Shannon's framework, a communications system consists of essentially five parts (cf. Figure 2.1):

1. An *information source* which produces a message or sequence of messages.
2. A *transmitter* which processes the signal in some way suitable for transmission over the *channel*.
3. The *channel* is the medium used to transmit the signal.
4. *The receiver* performs the inverse operation of that done by the transmitter.
5. The *destination*.

In his paper, Shannon demonstrated how to represent the various elements of a communication system as mathematical entities *"..suitably idealized from their physical counterparts."* [Sha48]. This approach allows the schematic depicted in Figure 2.1 to be applied to a broad range of scenarios arising in science and engineering and to consider very generic problems involving information transmission and processing. This was the brilliant achievement of Shannon.

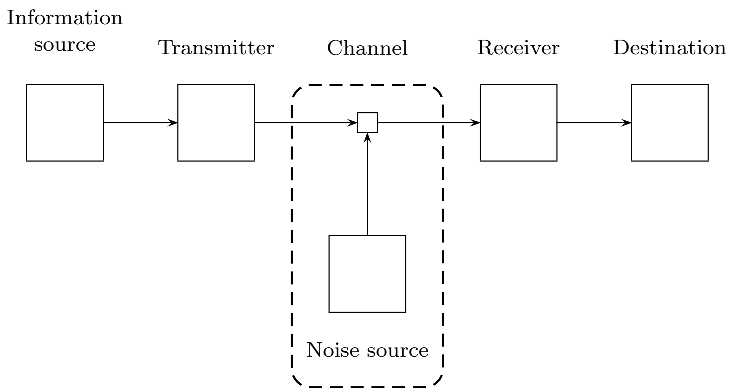


Figure 2.1: A communication system according to Shannon (modified from Fig. 1 in [Sha48]).

Information theory had the most decisive impact on the field of digital communications. However, due to its universality, it also found important applications in various research areas such as economics, statistics and, more recently, biology [CT91, MV04, DGH+06, AKL+07, Yoc05, SSGE86]. In Shannon's universal approach, information is assumed to be represented entirely by the stochasticity of the signal, and it is irrelevant whether the underlying physical medium carrying the signal is an electromagnetic wave, a time series of stock values, or biochemichal molecules. We shall briefly review Shannon's fundamental definitions that finally lead to a surprising result, the channel coding theorem, stating that error free communication is possible over noisy channels up to a certain transmission rate. Following this statement, we shall give a short introduction to coding theory and error correcting codes that are able to practically approach the rate promised by Shannon.

2.2.1 Entropy, divergence and mutual information

Let $\mathcal{X}$ and $\mathcal{Y}$ be two finite alphabets and let X and Y be discrete random variables with corresponding realizations $x \in \mathcal{X}$ and $y \in \mathcal{Y}$, respectively. If not stated otherwise, X, Y have distribution $p(x, y)$ with marginals $p(x)$ and $p(y)$. Shannon defines *information* as follows:

Definition 2.2.1 (Shannon information) The *information* that an event $x \in \mathcal{X}$ reveals is given by

$$h(x) = -\log(P(\mathsf{X} = x)). \tag{2.1}$$

If the logarithm is taken to base 2 then $h(x)$ has the unit *bits*[1]. □

Information is a measure of how surprising it is to observe a particular realization of X. The expected information revealed by the realizations of X is called the *entropy* of X:

Definition 2.2.2 (Entropy) The *entropy* of a discrete random variable X, denoted by $H(\mathsf{X})$, is defined as

$$H(\mathsf{X}) = -\sum_{x \in \mathcal{X}} p(x) \log(p(x)). \tag{2.2}$$

□

Entropy is best interpreted as a measure of average uncertainty about the outcome of X. Entropy is always non-negative and is upper bounded by $\log |\mathcal{X}|$:

$$0 \leq H(\mathsf{X}) \leq \log |\mathcal{X}|, \tag{2.3}$$

where $|\mathcal{X}|$ denotes the cardinality of the set $\mathcal{X}$. Equality with the upper bound holds if and only if X is uniformly distributed.

Example 2.2.1 (Entropy of a binary Bernoulli random variable) *Consider the binary r. v.* X *with* $x \in \{0, 1\}$ *and pdf defined by*

$$P(x = 0) = p_0, \quad P(x = 1) = p_1 = 1 - p_0.$$

[1]Throughout this section, if not stated otherwise, logarithms are taken to base 2.

The entropy of X *is given by*

$$H(\mathsf{X}) = -p_0 \log(p_0) - (1-p_0)\log(1-p_0).$$

Figure 2.2 shows the entropy of X *over* p_0*. As expected,* $H(\mathsf{X})$ *reaches a maximum of* $\log(|\mathcal{X}|) = 1$ *bit at* $p_0 = 0.5$ *and is zero in the deterministic cases* $p_0 = 0$ *and* $p_0 = 1$*.* □

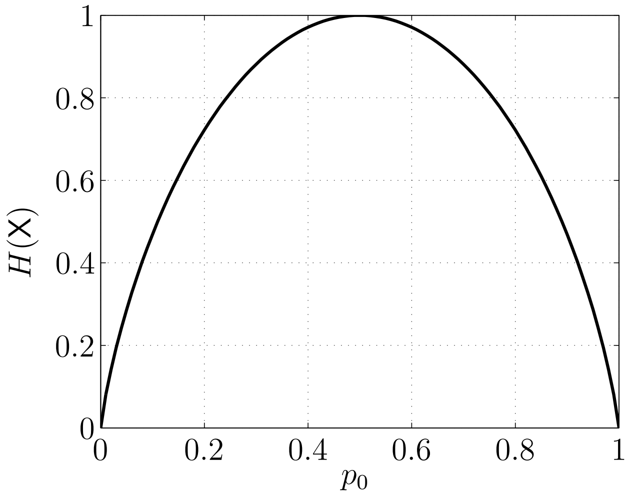


Figure 2.2: Entropy of a binary Bernoulli random variable X over $P(\mathsf{X}=0) = p_0$.

Simply by following Definition 2.2.2, we find that the entropy of the joint random variable $(\mathsf{X},\mathsf{Y}) \in \mathcal{X} \times \mathcal{Y}$ is

$$H(\mathsf{X},\mathsf{Y}) = -\sum_{x\in\mathcal{X}}\sum_{y\in\mathcal{Y}} p(x,y)\log(p(x,y)). \tag{2.4}$$

$H(\mathsf{X},\mathsf{Y})$ is called the *joint entropy* of X and Y. Another definition that we shall need is *conditional entropy* $H(\mathsf{X}|\mathsf{Y})$:

Definition 2.2.3 (Conditional entropy) The conditional entropy of the random variable X given Y is defined as

$$\begin{aligned}
H(\mathsf{X}|\mathsf{Y}) &= -\sum_{x\in\mathcal{Y}}\sum_{y\in\mathcal{Y}} p(x,y)\log\left(p(x|y)\right) && \text{(2.5a)}\\
&= -\sum_{x\in\mathcal{Y}}\sum_{y\in\mathcal{Y}} p(x,y)\log\left(\frac{p(x,y)}{p(y)}\right) && \text{(2.5b)}\\
&= -\sum_{x\in\mathcal{Y}}\sum_{y\in\mathcal{Y}} p(x,y)\log\left(p(x,y)\right) + \sum_{y\in\mathcal{Y}} p(y)\log\left(p(y)\right) && \text{(2.5c)}\\
&= H(\mathsf{X},\mathsf{Y}) - H(\mathsf{Y}). && \text{(2.5d)}
\end{aligned}$$

□

Conditional entropy tells how much the uncertainty about the joint (X,Y) is reduced when gaining knowledge about Y. The definition is intuitive since the joint entropy $H(\mathsf{X},\mathsf{Y})$ is reduced by the entropy (uncertainty) of Y when Y is given.

Mutual information is a measure of how much two random variables tell about each other:

Definition 2.2.4 (Mutual information (MI)) The mutual information between random variables X and Y is defined as

$$I(\mathsf{X};\mathsf{Y}) = H(\mathsf{X}) - H(\mathsf{X}|\mathsf{Y}). \tag{2.6}$$

□

A few intuitive arguments are necessary to derive the properties of MI: knowing Y can only decrease our uncertainty about X; so $H(\mathsf{X}) \geq H(\mathsf{X}|\mathsf{Y})$, with equality if and only if X and Y are statistically independent. By a similar argument, if Y determines X, we have zero uncertainty about X and $H(\mathsf{X}|\mathsf{Y}) = 0$. Furthermore,

$$\begin{aligned} I(\mathsf{X};\mathsf{Y}) &= H(\mathsf{X}) - H(\mathsf{X}|\mathsf{Y}) && \text{(2.7a)} \\ &\overset{(2.5d)}{=} H(\mathsf{X}) - H(\mathsf{X},\mathsf{Y}) + H(\mathsf{Y}) && \text{(2.7b)} \\ &= H(\mathsf{Y}) - H(\mathsf{Y}|\mathsf{X}) && \text{(2.7c)} \\ &= I(\mathsf{Y};\mathsf{X}), && \text{(2.7d)} \end{aligned}$$

i.e., X tells us as much about Y as Y about X. In summary, $I(\mathsf{X};\mathsf{Y})$ has the following properties:

$$\begin{aligned} &\text{(i)} \quad 0 \leq I(\mathsf{X};\mathsf{Y}) \leq H(\mathsf{X}) && \text{(2.8a)} \\ &\text{(ii)} \quad I(\mathsf{X};\mathsf{Y}) = I(\mathsf{Y};\mathsf{X}). && \text{(2.8b)} \end{aligned}$$

Mutual information is a special case of the *Kullback-Leibler* (KL) divergence, which is also often termed *relative entropy.* The KL divergence has important applications in statistics as a measure of divergence between two probability densities:

Definition 2.2.5 (Kullback-Leibler divergence) Let X and Y have densities p_{X} and p_{Y} respectively, then the Kullback-Leibler (KL) divergence $\mathcal{D}(\cdot||\cdot)$ is defined as

$$\mathcal{D}(p_{\mathsf{X}}||p_{\mathsf{Y}}) = \sum_{x\in\mathcal{X}} p_{\mathsf{X}} \log\left(\frac{p_{\mathsf{X}}}{p_{\mathsf{Y}}}\right), \tag{2.9}$$

with the convention $0\log(\frac{0}{p_{\mathsf{Y}}}) = 0$. □

It is well known that $\log(x) \leq x - 1$ with equality iff $x = 1$. It follows that

$$\begin{aligned} -\mathcal{D}(p_{\mathsf{X}}||p_{\mathsf{Y}}) &= \sum_{x\in\mathcal{X}} p_{\mathsf{X}} \log\left(\frac{p_{\mathsf{Y}}}{p_{\mathsf{X}}}\right) && \text{(2.10a)} \\ &\leq \sum_{x\in\mathcal{X}} p_{\mathsf{X}} \left(\frac{p_{\mathsf{Y}}}{p_{\mathsf{X}}} - 1\right) && \text{(2.10b)} \\ &= 0. && \text{(2.10c)} \end{aligned}$$

Hence, $\mathcal{D}(p_{\mathsf{X}}||p_{\mathsf{Y}})$ is always non-negative and is zero iff $p(x) = p(y)$. Note that $\mathcal{D}(p_{\mathsf{X}}||p_{\mathsf{Y}})$ is not a distance as it does not satisfy the triangle inequality and is not symmetric, i.e., $\mathcal{D}(p_{\mathsf{X}}||p_{\mathsf{Y}}) \neq \mathcal{D}(p_{\mathsf{Y}}||p_{\mathsf{X}})$ in general. We can now easily show that the MI is a special case of the KL divergence:

Theorem 1 *Let* X *and* Y *have joint density* $p(x, y)$ *and marginal densities* $p(x)$ *and* $p(y)$ *respectively, then the Mutual information* $I(\mathsf{Y};\mathsf{X})$ *is the relative entropy between the joint and the product density* $p(x)p(y)$ *of* X *and* Y*, i.e.,*

$$\mathcal{D}(p(x,y)||p(x)p(y)) = I(\mathsf{X};\mathsf{Y}). \tag{2.11}$$

□

Proof

$$\begin{aligned}
\mathcal{D}(p(x,y)||p(x)p(y)) &= \sum_{x\in\mathcal{X}}\sum_{y\in\mathcal{Y}} p(x,y)\log\left(\frac{p(x,y)}{p(x)p(y)}\right) && \text{(2.12a)}\\
&= \sum_{x\in\mathcal{X}}\sum_{y\in\mathcal{Y}} p(x,y)\left(\log\left(p(x,y)\right) - \log\left(p(x)p(y)\right)\right) && \text{(2.12b)}\\
&= -H(\mathsf{X},\mathsf{Y}) - \sum_{x\in\mathcal{X}} p(x)\log(p(x)) - \sum_{y\in\mathcal{Y}} p(y)\log(p(y)) && \text{(2.12c)}\\
&= H(\mathsf{X}) - (H(\mathsf{X},\mathsf{Y}) - H(\mathsf{Y})) && \text{(2.12d)}\\
&\overset{(2.5\text{d})}{=} H(\mathsf{X}) - H(\mathsf{X}|\mathsf{Y}) && \text{(2.12e)}\\
&= I(\mathsf{X};\mathsf{Y}), && \text{(2.12f)}
\end{aligned}$$

where we used the definition of conditional entropy from Eq. (2.5d). ■

Mutual information is therefore a *measure of statistical dependence* between random variables and is zero iff the random variables are statistically independent.

2.2.2 Transmission channels

Using the definitions established above, it is now possible to make important statements about the information processing capabilities of the system depicted in Figure 2.1: a source, modeled by a random variable $\mathbf{X} \sim p(\boldsymbol{x})$, emits a message $\boldsymbol{x} \in \mathcal{X}^N$ that is transmitted to a receiver. Transmission is assumed to be noisy, i.e., the sent message may be changed during transmission. The received vector $\boldsymbol{y} \in \mathcal{Y}^N$ is then modeled as a random variable $\mathbf{Y}$, and each input sequence $\boldsymbol{x}$ induces a probability distribution on the output sequences $\boldsymbol{y}$. The *transmission channel* is therefore formally defined as follows:

Definition 2.2.6 A *discrete channel*, denoted by $(\mathcal{X}^N, p(\boldsymbol{y}|\boldsymbol{x}), \mathcal{Y}^N)$, consists of two finite sets $\mathcal{X}^N$ and $\mathcal{Y}^N$ and a collection of probability mass functions $p(\boldsymbol{y}|\boldsymbol{x})$, one for each $\boldsymbol{x} \in \mathcal{X}^N$, such that for every $\boldsymbol{x}$ and $\boldsymbol{y}$, $p(\boldsymbol{y}|\boldsymbol{x}) \geq 0$, and for every $\boldsymbol{x}$, $\sum_{\boldsymbol{x}} p(\boldsymbol{y}|\boldsymbol{x}) = 1$, with the interpretation that $\mathbf{X}$ is the input and $\mathbf{Y}$ the output of the channel.
A channel that is used without feedback, i.e.,, the input symbols do not depend on the past output symbols, and whose ith output symbol only depends on the ith input symbol, is called a *discrete memoryless channel* (DMC), and the channel transition functions factor as

$$p(\boldsymbol{y}|\boldsymbol{x}) = \prod_{n=1}^{N} p(y_n|x_n). \tag{2.13}$$

□

It is often convenient to describe a channel by its *transition matrix* $\boldsymbol{P}$. Here, the entry in the ith row and jth column denotes the conditional probability $p(\boldsymbol{y} = j|\boldsymbol{x} = i)$, where j and i represent the jth and ith element of the ordered sets $\mathcal{X}^N$ and $\mathcal{Y}^N$, respectively.

Definition 2.2.7 A transition matrix (channel) is said to be *symmetric* if the rows of the channel transition matrix $p(\boldsymbol{y}|\boldsymbol{x})$ are permutations of each other, and the columns are permutations of each other. A channel is *weakly symmetric* if every row $p(\cdot|\boldsymbol{x})$ is a permutation of every other row, and all the column sums $\sum_x p(\boldsymbol{y}|\boldsymbol{x})$ are equal. If a channel is symmetric it is also weakly symmetric. □

We are now ready for a fundamental result in information theory that we state for the transmission over a discrete memoryless channel. The *channel coding theorem* - that we are about to sketch - states that the highest transmission rate that can be sent with *arbitrary low probability of error at the receiver* over a channel is given by

$$C = \max_{p(x)} I(\mathsf{X};\mathsf{Y}), \tag{2.14}$$

where the maximum is taken over all possible input distributions $p(x)$. This quantity is the famous *channel capacity*. At first glance, this statement seems to express a paradox, because it seems impossible to achieve noise free transmission when errors are actually introduced in the sent messages. The idea of proving this claim is based on the definitions of entropy and mutual information [CT91]: suppose an iid source $\mathsf{X} \sim p(x)$ produces a *long* sequence $\boldsymbol{x} = [x_1, x_2, .., x_N]$, the message, which is to be sent over the channel $(\mathcal{X}^N, p(\boldsymbol{y}|\boldsymbol{x}), \mathcal{Y}^N)$. The possible messages that the source generates fall into two groups: a high probability group which can be shown to contain approximately $2^{NH(\mathsf{X})}$ sequences and the remaining sequences which will occur with vanishingly small probability. Similarly, the received sequences will fall into a high probability group with approximately $2^{NH(\mathsf{Y})}$ members and the remaining sequences forming a low probability group. Each input $\boldsymbol{x}$ can potentially produce any sequence $\boldsymbol{y}$ because of the noise. However, it can be shown that the size of the set containing the high probability sequences produced by any $\boldsymbol{x}$ is given by $2^{NH(\mathsf{Y}|\mathsf{X})}$ and all other outputs outside of this set have vanishingly small probabilities. We wish to ensure that no two messages $\boldsymbol{x}$,$\boldsymbol{x}'$ produce the same output $\boldsymbol{y}$: we therefore divide the set of high probability receive sequences in disjoint sets of size $2^{NH(\mathsf{Y}|\mathsf{X})}$ and associate each of these with a unique message $\boldsymbol{x}$. The total number of disjoint sets is less than or equal to $2^{NH(\mathsf{Y})-NH(\mathsf{Y}|\mathsf{X})} = 2^{NI(\mathsf{X};\mathsf{Y})}$, hence we can send at most $2^{NI(\mathsf{X};\mathsf{Y})}$ distinguishable sequences of length N. The distinguishable sequences can be represented by $NI(\mathsf{X};\mathsf{Y})$ bits, which results in a maximum rate of $I(\mathsf{X};\mathsf{Y})$ information bits per message.

While the derivation outlined above gives an upper bound on the capacity, it can also be shown that this bound can be achieved [CT91]. However, it is very important to emphasize that the theorem only holds for large N. The following examples show how to calculate the capacity for symmetric channels, and two types of channels with unknown capacity.

We give an example of a quarternary alphabet since it is encountered in genetics:

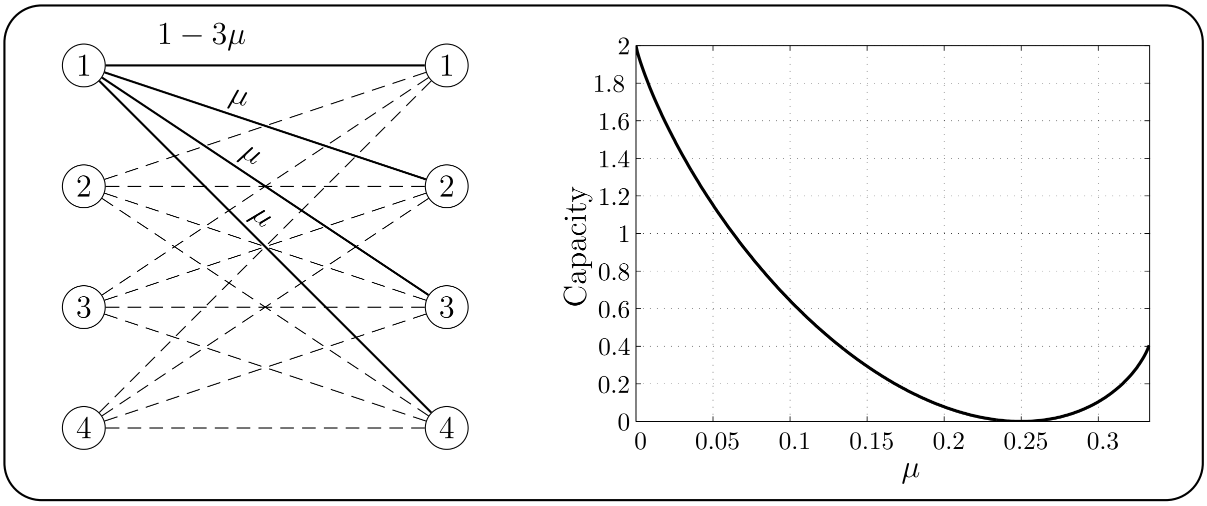


Figure 2.3: *Left: The transition diagram of the quarternary symmetric channel with transition probability 3μ. Right: Capacity of quarternary symmetric channel over transition probability μ.*

Example 2.2.2 (Quarternary Symmetric Channel) *Consider the channel $(\mathcal{X}, \boldsymbol{P}, \mathcal{Y})$ with $\mathcal{X} = \mathcal{Y} = \{1, 2, 3, 4\}$ given by the transition matrix*

$$\boldsymbol{P}_{QSC} = \begin{pmatrix} 1-3\mu & \mu & \mu & \mu \\ \mu & 1-3\mu & \mu & \mu \\ \mu & \mu & 1-3\mu & \mu \\ \mu & \mu & \mu & 1-3\mu \end{pmatrix}, \quad 0 \le \mu \le \frac{1}{3}.$$

The channel transition diagram is shown in Figure 2.3. We have

$$\begin{aligned} I(\mathsf{X};\mathsf{Y}) &= H(\mathsf{Y}) - H(\mathsf{Y}|\mathsf{X}) \\ &\le \log(|\mathcal{Y}|) - H(\mathsf{Y}|\mathsf{X}) \\ &= 2 - H(\mathsf{Y}|\mathsf{X}), \end{aligned}$$

with equality for a uniform output distribution. Setting $p(x) = \frac{1}{|\mathcal{X}|}$ yields

$$p(y) = \sum_{x\in\mathcal{X}} p(y|x)p(x) = \frac{1}{|\mathcal{X}|}\sum_{x} p(y|x) = c\frac{1}{|\mathcal{X}|} = \frac{1}{|\mathcal{Y}|},$$

where c denotes the column sum of $\boldsymbol{P}$, which is constant because the columns are permutations of each other.

This proves that choosing a uniform distribution maximizes the mutual information and the capacity of the channel given by $\boldsymbol{P}$ is therefore

$$C_{\boldsymbol{P}_{QSC}} = \max_{p(x)} I(\mathsf{X};\mathsf{Y}) = 2 - H(\mathsf{Y}|\mathsf{X}),$$

with

$$H(\mathsf{Y}|\mathsf{X}) = \sum_{x,y} p(x)p(y|x)\log\left(\frac{1}{p(y|x)}\right) = (1-3\mu)\log\left(\frac{1}{1-3\mu}\right) + 3\mu\log\left(\frac{1}{\mu}\right),$$

which follows from the property of $\boldsymbol{P}$ that the rows are permutations of each other. The capacity of the QSC over μ is shown in Figure 2.3 . In the noiseless case ($\mu = 0$), 2bits *can be transmitted and with increasing μ, the capacity decreases exponentially reaching* 0 *at* $\mu = 1/4$. *The capacity increases again when μ increases beyond* $1/4$, *up to* $2 - \log(3)$ *bit for the extremal case* $\mu = 1/3$. *The explanation for this is as follows: suppose* $\mu = 1/3$ *and assume that the symbol* 3 *is received. Then we know that* 3 *was* not *the symbol which had been sent and this additional information can be used to encode messages such that transmission rate* $2 - \log(3)$ *bit can be achieved.* □

When calculating the capacity in the example above, we made use of the facts that the column sums of $\boldsymbol{P}$ were constant and that the rows were permutations of one another. These are exactly the properties of a weakly symmetric channel, and an analytical expression for any channel from that class can be derived as shown.
If a channel is not weakly symmetric, there is no analytical expression for the capacity in general.

2.2.3 Example: insertion and deletion channels

An *iid binary deletion channel* (BDC) has binary $\{0, 1\}$ input X and binary output Y: with probability $1 - d$, the channel reveals $y = x$ at the receiver. With probability d, however, the channel does not output anything. Such a channel takes N transmitted bits and outputs a random subsequence of the input, where the subsequence is obtained by deleting each bit independently with probability d. Note that this is substantially different from an erasure channel [CT91], where the receiver knows at which position an erasure has occurred. The channel model for a single bit is shown in Figure 2.4. The *iid*

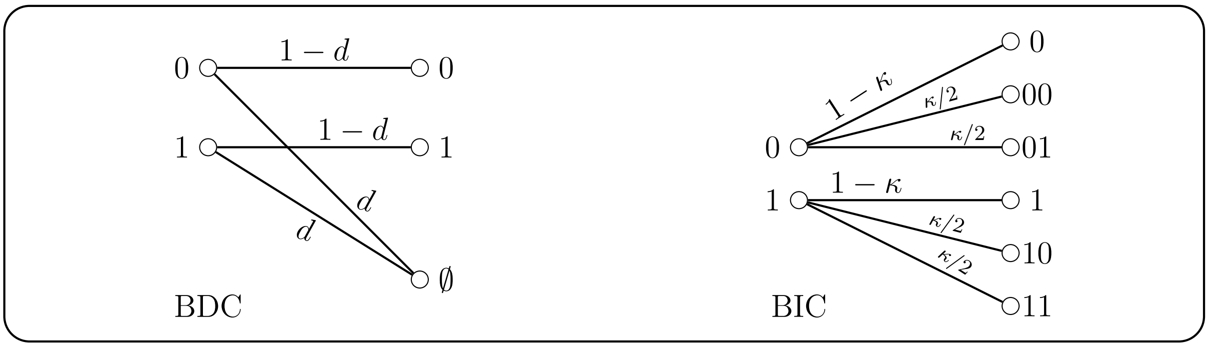


Figure 2.4: Left: Transition diagram of a binary deletion channel model with deletion probability d. A deletion is represented by the symbol $\emptyset$. Right: Transition diagram for a binary insertion channel model with insertion probability κ.

binary insertion channel (BIC), also shown in Figure 2.4, works exactly in the opposite way: the BIC takes a binary symbol x and reveals $y = x$ at the output with probability $1 - \kappa$. However, with probability $\kappa/2$, the channel emits $x1$, and with probability $\kappa/2$ it emits $x0$, inserting a random, non-transmitted symbol after the transmitted one. The iid BIC is thus a $(\{0, 1\}, p(y|x), \{0, 1, 00, 01, 10, 11\})$ channel with $p(y|x) = 1 - \kappa$ if $x = y$ and $\kappa/2$ otherwise.

The BDC and the BIC are simple stochastic models of channels with synchronization errors. Insertion and deletion channels also arise in biological information processing as discussed in Chapter 3, Section 3.2. The capacities for deletion and insertion channels are unknown in the general case and were studied, for example, in [DM07]. Simulation approaches try to approximate the transmission rates [HDE08]. Coding strategies for these channels were presented, for example, in [Slo00, DML00, LM07].

2.3 Coding theory and error correction

The channel coding theorem promises the existence of codes that achieve capacity. Error free transmission is achieved by dividing the set of all possible messages into distinguishable sequences such that a received vector $\boldsymbol{y}$ can be uniquely assigned to a sent message with very high probability. The maximum rate at which this can happen is given by the mutual information. However, the theorem is not constructive, i.e., does not devise strategies how to design practically good codes. Therefore, since the appearance of Shannon's original paper, people have searched for practically useful codes that achieve the promised performance, and the field of *coding theory* emerged.

In 1993, Berrou et al. presented the first class of capacity achieving codes having practical decoding complexity, the so-called Turbo-Codes [HOP96]. These codes are now an integral part of many modern technical communication systems. Coding theory is the *art of error correction*, and we will outline the basic principles in the following. For a deeper treatment of the subject, the reader is referred to [RU07, LC83].

2.3.1 Basic principles

We consider the transmission scenario depicted in Figure 2.5: a source emits a message $\boldsymbol{x} = [x_1, .., x_K]$, and an encoder maps the message to a codeword $\boldsymbol{c}$ subsequently transmitted over the channel. The channel introduces noise, and a distorted version of $\boldsymbol{c}$ is received. The decoder performs two tasks: it first estimates which codeword has been sent and subsequently performs the inverse operation of the encoder. The vector $\hat{\boldsymbol{x}}$ is then presented to the information sink as the resulting estimate for the original message $\boldsymbol{x}$. We shall only consider binary codes in this section, i.e., $x_i \in \mathbb{F}_2$, where $\mathbb{F}_2$ denotes the field of order two.

Definition 2.3.1 (Hamming weight and distance) Given two vectors of equal length $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{F}_2^N$, *the Hamming weight* $w_H(\boldsymbol{x})$ is defined as

$$w_H(\boldsymbol{x}) = |\{i : x_i \neq 0\}|, \tag{2.15}$$

where 0 denotes the zero element of the field $\mathbb{F}_2$, and the *Hamming distance* $d_H(\boldsymbol{x}, \boldsymbol{y})$ is defined as

$$d_H(\boldsymbol{x}, \boldsymbol{y}) = w_H(\boldsymbol{x} - \boldsymbol{y}), \tag{2.16}$$

i.e., the number of positions where $\boldsymbol{x}$ and $\boldsymbol{y}$ differ. □

Definition 2.3.2 (Block code (N, K, d_{min})) A binary block code (N, K, d_{min}) is a K-dimensional subspace $\mathcal{C}$ of the N-dimensional binary vector space

$$\mathcal{C} \subseteq \mathbb{F}_2^N, \tag{2.17}$$

such that any two vectors in $\mathcal{C}$ have Hamming distance at least d_{min}:

$$d_{min} = \min_{\substack{\boldsymbol{c},\boldsymbol{c}' \in \mathcal{C} \\ \boldsymbol{c} \neq \boldsymbol{c}'}} \{d_H(\boldsymbol{c}, \boldsymbol{c'})\}. \tag{2.18}$$

We say that the code has length N and dimension K, and the parameter d_{min} is called the *minimum Hamming distance* of the code. The *code rate* $R = \frac{K}{N}$ is the fraction of information bits per symbol x_i that can be sent over the channel, and the coding theorem requires that

$$R < C.$$

□

An *encoder*

$$\mathbb{F}_2^K \to \mathcal{C},\ \operatorname{enc}(\boldsymbol{x}) \mapsto \boldsymbol{c}$$

takes one of 2^K possible messages $\boldsymbol{x}$ and maps it uniquely to a codeword $\boldsymbol{c}$ of length $N > K$. This codeword is transmitted over the channel. In the following, we will consider a simple transmission scenario: suppose that upon sending the codeword $\boldsymbol{c}$ over a channel we receive the vector

$$\boldsymbol{y} = \boldsymbol{c} + \boldsymbol{\varepsilon},\ \boldsymbol{\varepsilon} \in \mathbb{F}_2^N, \tag{2.19}$$

where $\boldsymbol{\varepsilon} \in \mathbb{F}_2^N$ is a binary, random noise vector. The message $\boldsymbol{x}$ and the noisy receive vector $\boldsymbol{y}$ differ in $w_H(\boldsymbol{\varepsilon})$ positions, and our goal is to reconstruct $\boldsymbol{x}$ from $\boldsymbol{y}$ without knowing $\boldsymbol{\varepsilon}$.

The decoding strategy is to look for the codeword $\boldsymbol{c}$ that looks "closest" to $\boldsymbol{y}$ in some sense. It can be shown that the optimal (maximum likelihood (ML)) decision rule for reconstructing $\boldsymbol{c}$ is given by

$$\hat{\boldsymbol{c}} = \arg\max_{\boldsymbol{c} \in \mathcal{C}} \{p(\boldsymbol{y}|\boldsymbol{c})\}. \tag{2.20}$$

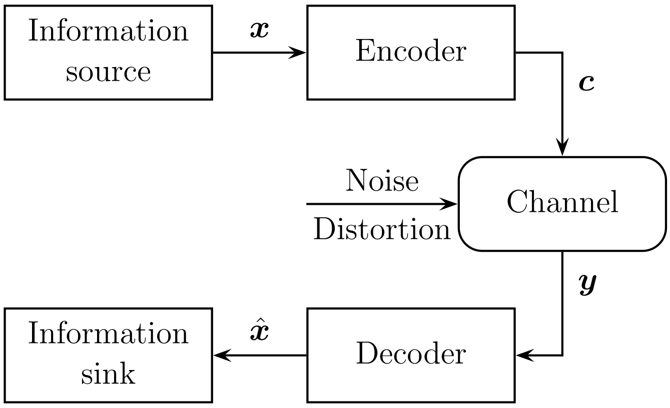


Figure 2.5: Simplified model of a coded transmission system.

However, this is infeasible since Eq. (2.20) implies the comparison of the received word with all possible codewords in $\mathcal{C}$, in number exponentially growing with N, and the channel coding theorem requires large N. A suboptimal, practically much more feasible, decision rule is to look for a codeword that lies within a radius of $d_{min}/2$ around the received word. As the code, by construction, has minimum distance d_{min}, this decision rule is guaranteed to uniquely recover the correct codeword if $w_H(\boldsymbol{\varepsilon}) < d_{min}/2$. If no codeword can be found within the radius we declare a decoding error. This strategy is known as *bounded minimum distance (BMD) decoding.* A graphical comparison between BMD and the ML decision rule is shown in Figure 2.6. In Chapter 7, a third decoding strategy shall be discussed.

Many algebraic constructions for block codes were developed in the past 60 years [Bla08, MS77]. It was shown that a certain class of block codes, so-called LDPC codes, can approach the channel capacity with a low decoding complexity [CFR⁺01]. We shall consider two examples of codes: the following example introduces a trivial class of codes, repetition codes. The next section introduces a practical class of codes, convolutional codes, that exhibit low encoding and decoding complexity, and a good error correcting performance:

Example 2.3.1 (Repetition code) *A repetition code is a* $(1, N, N)$ *block code that is formed by repeating a single bit* N *times. We can think of it as simply sending each source bit* N *times over the channel. The code has maximum minimum distance but it consists of only two codewords:*

$$c_1 = [0\ 0\ ...\ 0],$$

$$c_2 = [1\ 1\ ...\ 1].$$

Decoding is done simply by majority voting. A repetition code cannot achieve the channel capacity: as N *goes to infinity, the error probability goes to zero (unless the channel introduces errors with probability* $1/2$*), but at the same time the transmission rate goes to zero.* □

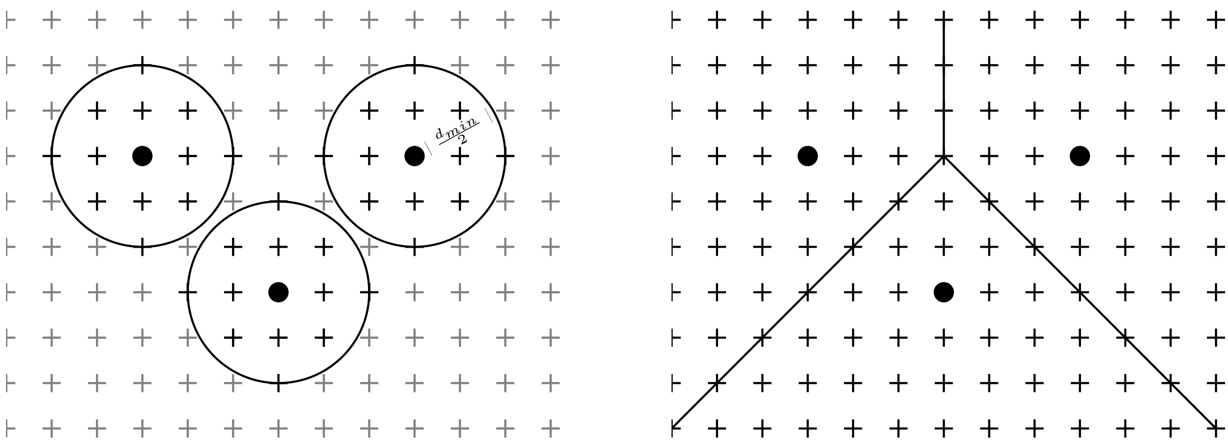


Figure 2.6: Decision regions resulting from different decoding strategies. Black circles mark codewords, crosses mark elements from $\mathbb{F}_2^N$. Left: BMD decoding, Right: ML decoding.

2.3.2 Example: convolutional codes

Convolutional codes, introduced by Elias [Eli55], are a certain class of highly structured codes, allowing a simple implementation and good performance even for small N. A long message $\boldsymbol{x}$ of length K is split into small blocks of length K_s

$$\boldsymbol{x}_i = [x_{(i-1)K_s+1}, .., x_{(i-1)K_s+K_s}], \quad i = 1, .., K/K_s$$

that enter the encoder, which maps them to output blocks of length N_s

$$\boldsymbol{c}_i = [c_{(i-1)N_s+1}, .., c_{(i-1)N_s+N_s}], \quad i = 1, .., N/N_s$$

that form the codeword $\boldsymbol{c} = [\boldsymbol{c}_1, ..., \boldsymbol{c}_{N/N_s}]$. The encoder possesses memory, i.e., the encoding rule for $\boldsymbol{c}_i$ depends on the actual and the previous inputs $\boldsymbol{x}_i, \boldsymbol{x}_{i-1}, ..., \boldsymbol{x}_{i-M}$ up to some constant M. The rate of the code is $R = K_s/N_s = K/N$. Convolutional codes are linear, i.e., $\forall \boldsymbol{c}, \boldsymbol{c}' \in \mathcal{C} : \boldsymbol{c} + \boldsymbol{c}' \in \mathcal{C}$.

It can then be shown that any encoder can be represented by an *encoder circuit*. An encoder with parameters $K_s = 1$, $N_s = 2$ and $M = 2$ is shown in Figure 2.7: at any

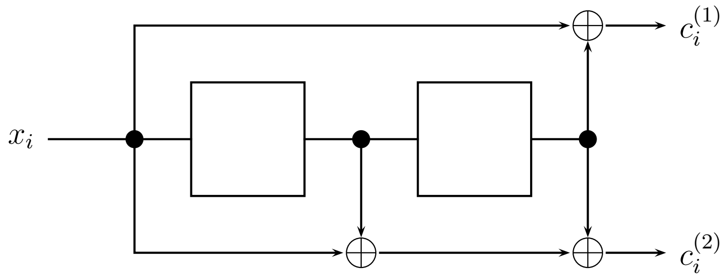


Figure 2.7: A simple encoder circuit of a $R = 1/2$ convolutional code. Boxes represent *memory elements*. A bit enters a memory element at time i, which holds it for one time step, releasing it at time $i + 1$.

clock cycle i, a single bit x_i is shifted into the circuit, the bits in the memory elements are released and the encoder output $\boldsymbol{c}_i = [c_i^{(1)}, c_i^{(2)}]$ is formed. Hence, the codebits are given by the equations

$$\begin{aligned} c_i^{(1)} &= x_i \oplus x_{i-2}, \\ c_i^{(2)} &= x_i \oplus x_{i-1} \oplus x_{i-2}. \end{aligned}$$

The contents of the memory elements at any given time are called the state of the encoder denoted by $\boldsymbol{s}_i = [s_i^{(1)} s_i^{(2)}]$. The code bit equations can then be rewritten as

$$\begin{aligned} c_i^{(1)} &= x_i \oplus s_i^{(2)}, \\ c_i^{(2)} &= x_i \oplus s_i^{(1)} \oplus s_i^{(2)}. \end{aligned}$$

There are four possible states which determine the encoder output at time i, depending on the current input. The encoder can be represented by a graphical representation called

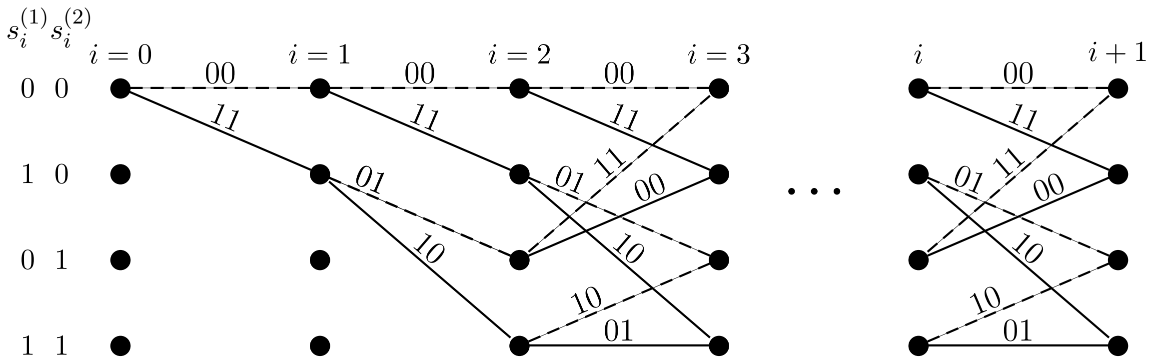


Figure 2.8: Trellis representation of the Rate 1/2, $M = 2$ convolutional encoder.

trellis diagram as shown in Figure 2.8. The nodes in the trellis graph represent the 2^M different possible states $\boldsymbol{s}_i$ of the encoder at time steps $i = 0, 1, ..$ from left to right. The nodes are arranged such that different rows represent different states (shown on the left of each row), and different columns represent different time steps i (shown on top). The branches denote possible state transitions depending on the input to the encoder at time i. Dashed branches denote that a 0 has entered the decoder, solid branches denote $x_i = 1$. At $i = 0$ the encoder state is assumed to be all zero, therefore the branches start from the node corresponding to state 0 0 at the left most trellis section. The codewords can now directly be read off from the trellis representation. As an example, the message

$$\boldsymbol{x} = [00110] \quad \text{yields the codeword} \quad \boldsymbol{c} = [0000111010].$$

The corresponding state sequence reads as

$$[\boldsymbol{s}_0, \boldsymbol{s}_1, \boldsymbol{s}_2, \boldsymbol{s}_3, \boldsymbol{s}_4] = [00,\ 00,\ 10,\ 11,\ 01].$$

2.4 Markov processes

Sequence evolution of DNA sequences is modeled by continuous-time Markov processes (Chapter 4, Section 4.1.2, Chapter 5 and Chapter 6). We therefore introduce the basic theory of Markov processes. We stress some theory for the continuous case as this aspect is rarely covered in depth by books found in the bioinformatics literature. Discrete Markov processes are also building blocks of hidden Markov processes which are used in (Chapters 5 and 6) for modeling DNA sequences. Hidden Markov processes are briefly discussed in Section 2.4.5.

2.4.1 Discrete time Markov process (DTMP)

We shall first consider processes in discrete time, using time index

$$n \in \mathbb{Z}^+ = \{0, 1, 2, 3, ..\},$$

and we write $\{\mathsf{X}_n\}_{n\geq 0}$ to denote a *discrete time random process*. Let $\mathcal{S}$ be a finite set with an enumeration $\mathcal{S} = \{s_0, s_1, .., s_{|\mathcal{S}|-1}\}$ where the elements in $\mathcal{S}$ represent the possible realizations of the random variables X_n. The elements $s_i \in \mathcal{S}$ are called the *states*, and $\mathcal{S}$ is called the *state space* of the process. We need the following definitions:

Definition 2.4.1 (Stochastic matrix) A matrix $\boldsymbol{P}$, $[\boldsymbol{P}]_{ij} = p_{ij}$, is *stochastic* if every row of $\boldsymbol{P}$ is a distribution, i.e., $\boldsymbol{P}$ satisfies the following conditions:

(i) $0 \leq p_{ij} \leq 1 \quad$ for all i, j,
(ii) $\sum_j p_{ij} = 1 \quad$ for all i.

We also refer to a matrix satisfying these properties as *transition matrix*. □

Let $\boldsymbol{\lambda}$ denote a distribution over $\mathcal{S}$, i.e.:

$$\boldsymbol{\lambda} = [\lambda_{s_0}, .., , \lambda_{s_{|\mathcal{S}|}}],$$

with $s_i \in \mathcal{S}$, $0 \leq \lambda_{s_i} \leq 1$, $\sum_{s_i \in \mathcal{S}} \lambda_{s_i} = 1$ and $\lambda_{s_i} = P(x = s_i)$.

Definition 2.4.2 (Markov chain) A stochastic process $\{\mathsf{X}_n\}_{n\geq 0}$ is a first order *Markov chain* with initial distribution $\boldsymbol{\lambda}$ iff

(i) X_0 has distribution $\boldsymbol{\lambda}$,
(ii) for $n \geq 0$, X_{n+1} depends only on X_n and has distribution $p(x_{n+1}|x_n)$.

The properties of a Markov chain are summarized more formally as

$$P(x_0 = s_{i_0}) = \lambda_{s_{i_0}}, \tag{2.21a}$$
$$P(x_{n+1} = s_{i_{n+1}}|x_n = s_{i_n}, .., x_0 = s_{i_0}) = P(x_{n+1} = s_{i_{n+1}}|x_n = s_{i_n}), \tag{2.21b}$$

for all states $s_{i_{n+1}}, s_{i_n}, .., s_{i_0}$. □

Definition 2.4.3 (Homogeneity) A Markov chain is *homogenous* if the transition probabilities are independent of the time index n and given by the *transition matrix* $\boldsymbol{P}$, i.e.,

$$P(x_{n+1} = s_j|x_n = s_i) = P(x_1 = s_j|x_0 = s_i) \doteq p_{s_i s_j}, \quad \forall (s_i, s_j) \in \mathcal{S} \times \mathcal{S},\ \forall n \geq 0. \tag{2.22}$$

We refer to such a process as $\texttt{Markov}(\boldsymbol{\lambda}, \boldsymbol{P})$ with $[\boldsymbol{P}]_{ij} = p_{s_i s_j}$. □

A process $\{\mathsf{X}_n\}_{0\leq n\leq N}$ that is $\texttt{Markov}(\boldsymbol{\lambda}, \boldsymbol{P})$ has distribution

$$p(x_0, x_1, ..., x_N) = p(x_0) \prod_{n=1}^{N} p(x_n|x_{n-1}), \tag{2.23a}$$

with probabilities given by

$$P(x_0 = s_{i_0}, x_1 = s_{i_1}, .., x_N = s_{i_N}) = P(x_0 = s_{i_0}) P(x_1 = s_{i_1}|x_0 = s_{i_0}) \tag{2.24a}$$
$$\ldots\ P(x_N = s_{i_N}|x_{N-1} = s_{i_{N-1}}) \tag{2.24b}$$
$$= \lambda_{s_{i_0}} p_{s_{i_0} s_{i_1}} p_{s_{i_1} s_{i_2}} ... p_{s_{i_{N-1}} s_{i_N}}. \tag{2.24c}$$

Let the initial distribution $\boldsymbol{\lambda}$ be the row vector indexed by $\mathcal{S}$, i.e., $[\boldsymbol{\lambda}]_i = P(x_0 = s_i)$ denotes the ith entry in that vector. Further, let $[\boldsymbol{P}]_{ij} = p_{s_i s_j}$, and further define the *state vector* $\boldsymbol{p}_n$ with $[\boldsymbol{p}_n]_i = P(x_n = s_i)$. Then, for all $n, m \geq 0$, a $\texttt{Markov}(\boldsymbol{\lambda}, \boldsymbol{P})$ process has

$$P(x_1 = s_i) = \sum_j P(x_0 = s_j) P(x_1 = s_i | x_0 = s_j) \tag{2.25a}$$

$$= \sum_j \lambda_{s_j} p_{s_j s_i} = [\boldsymbol{\lambda P}]_i, \tag{2.25b}$$

$$P(x_2 = s_i) = \sum_j P(x_1 = s_j) P(x_2 = s_i | x_1 = s_j) \tag{2.25c}$$

$$= \sum_j [\boldsymbol{\lambda P}]_j p_{s_j s_i} = \sum_{j,k} \lambda_{s_k} p_{s_k s_j} p_{s_j s_i} = [\boldsymbol{\lambda P P}]_i \tag{2.25d}$$

$$= [\boldsymbol{\lambda P}^2]_i. \tag{2.25e}$$

It follows that the state vector of the process at any time n is given by

$$\boldsymbol{p}_n = \boldsymbol{\lambda P}^n. \tag{2.26}$$

In order to make statements about the convergence of the process, we need to introduce a distribution with a special property:

Definition 2.4.4 (Stationary distribution) A distribution $\boldsymbol{\pi}$ is *stationary* if

$$\boldsymbol{\pi P} = \boldsymbol{\pi}. \tag{2.27}$$

The terms *equilibrium* and *invariant* distribution are also frequently used. □

Example 2.4.1 (Stationary distribution) *Consider the following transition matrix of a two-state Markov process*

$$\boldsymbol{P} = \begin{pmatrix} 1/3 & 2/3 \\ 4/9 & 5/9 \end{pmatrix}.$$

Solving the stationarity condition in Eq. (2.27) *for* $\boldsymbol{\pi}$ *with the additional constraint* $\pi_1 + \pi_2 = 1$*, we find that the stationary distribution for* $\boldsymbol{P}$ *is given by:*

$$\boldsymbol{\pi} = [2/5 \;\; 3/5].$$ □

Note that $\boldsymbol{\pi}$ is the left eigenvector of $\boldsymbol{P}$ corresponding to eigenvalue 1.
We assume throughout this work that $p_{s_i s_j} > 0$, $\forall s_i, s_j$. In this case, it can be shown that $\boldsymbol{P}$ has a stationary distribution (the process is ergodic). The following theorem shows that the state vector converges to this distribution for large n.

Theorem 2 (Convergence) *Suppose* $\boldsymbol{P}$ *has stationary distribution* $\boldsymbol{\pi}$ *and let* $\boldsymbol{\lambda}$ *be an arbitrary distribution. Then, for a* $\texttt{Markov}(\boldsymbol{\lambda}, \boldsymbol{P})$ *process* $\{\mathsf{X}_n\}_{n\geq 0}$

$$\boldsymbol{p}_\infty = \lim_{n\to\infty} \boldsymbol{p}_n \to \boldsymbol{\pi}, \tag{2.28}$$

which is independent of the starting distribution $\boldsymbol{\lambda}$*.* □

PROOF See Appendix B. ■

2.4.2 Continuous time Markov process (CTMP)

Building upon the theory established for discrete times, we now consider stochastic processes $\{\mathsf{X}_t\}_{t\geq 0}$ with finite state space $\mathcal{S}$ but in continuous time

$$t \in \mathbb{R}^+ = [0, \infty).$$

The main difficulty will be to define a stochastic matrix $\boldsymbol{P}$ such that probabilities $P(x_{t+h} = s_j | x_t = s_i)$ can be derived for all $t, h \in \mathbb{R}^+$. After having established this, the theory of continuous-time Markov processes (CTMP) will follow the same lines as for discrete-time Markov processes (DTMP).

We shall first introduce rate matrices and matrix exponentials, that play an important role in the definition of CTMP:

Definition 2.4.5 A *rate matrix* on $\mathcal{S}$ is a matrix $\boldsymbol{R}$, $[\boldsymbol{R}]_{ij} = r_{s_i s_j}$, $s_i, s_j \in \mathcal{S}$, satisfying the following properties:

$$\begin{array}{rlll} \text{(i)} & 0 \leq r_{s_i s_j} < \infty & \forall i \neq j \\ \text{(ii)} & 0 \leq -r_{s_i s_i} < \infty & \forall i \\ \text{(iii)} & \sum_j r_{s_i s_j} = 0 & \forall i \end{array} \qquad \square$$

Note that the entries in the off-diagonals are arbitrary positive values and that the diagonal entries are strictly negative such that the row sums are all zero. We interprete the off-diagonals $r_{s_i s_j}$ as rates of going from state s_i to s_j and a diagonal element $r_{s_i s_i}$ as the rate of leaving the state s_i.

We saw in Eq. (2.26) that the state vector at time index n of a $\texttt{Markov}(\boldsymbol{\lambda}, \boldsymbol{P})$ process is determined by $\boldsymbol{P}^n$. For scalar $p \in \mathbb{R}^+$, a natural way to interpolate the sequence $p^n, n = 0, 1, 2, ...$ is by the function e^{tr}, $t \geq 0$, with $r = \log(p)$. We seek a way to interpolate the series $\boldsymbol{P}^n$, $n = 0, 1, ...,$ and this is done by defining the *matrix exponential* of $\boldsymbol{R}$

$$\boldsymbol{P} \doteq e^{\boldsymbol{R}}. \tag{2.29}$$

By expanding the definition in Eq. (2.29) into a Taylor series, we find the following expression for the matrix exponential:

$$e^{\boldsymbol{R}} = \boldsymbol{I} + \boldsymbol{R} + \frac{1}{2!}\boldsymbol{R}\boldsymbol{R} + \ldots = \sum_{k=0}^{\infty} \frac{\boldsymbol{R}^k}{k!}, \tag{2.30}$$

where $\boldsymbol{I}$ denotes the identity matrix with ones in the diagonals and zeros otherwise. We shall find it often convenient to consider the eigenvalue decomposition of the rate matrix that exists if $\boldsymbol{R}$ has distinct eigenvalues[2]. Suppose $\boldsymbol{R}$ can be decomposed as

$$\boldsymbol{R} = \boldsymbol{U}\boldsymbol{\Phi}\boldsymbol{U}^{-1}, \tag{2.31}$$

[2]It is shown later that this always holds true for the class of CTMP considered in this work.

where $\boldsymbol{U} = [\boldsymbol{u}_1^T, .., \boldsymbol{u}_{|\mathcal{S}|}^T]$ denotes the matrix of eigenvectors and

$$\boldsymbol{\Phi} = \operatorname{diag}(\phi_1, .., \phi_{|\mathcal{S}|}) = \begin{pmatrix} \phi_1 & & \boldsymbol{0} \\ & \ddots & \\ \boldsymbol{0} & & \phi_{|\mathcal{S}|} \end{pmatrix}$$

the matrix with the eigenvalues of $\boldsymbol{R}$ in the diagonal. Then,

$$\boldsymbol{P}(t) = e^{t\boldsymbol{R}} = \sum_{k=0}^{\infty} \frac{(t\boldsymbol{R})^k}{k!} = \sum_{k=0}^{\infty} \frac{(t\boldsymbol{U}\boldsymbol{\Phi}\boldsymbol{U}^{-1})^k}{k!} \tag{2.32a}$$

$$\stackrel{(a)}{=} \boldsymbol{U}\left(\sum_{k=0}^{\infty} \frac{t^k \boldsymbol{\Phi}^k}{k!}\right)\boldsymbol{U}^{-1} \tag{2.32b}$$

$$= \boldsymbol{U} e^{t\boldsymbol{\Phi}} \boldsymbol{U}^{-1}, \tag{2.32c}$$

with

$$e^{t\boldsymbol{\Phi}} = \begin{pmatrix} e^{t\phi_1} & & \boldsymbol{0} \\ & \ddots & \\ \boldsymbol{0} & & e^{t\phi_{|\mathcal{S}|}} \end{pmatrix}. \tag{2.33}$$

In (a) we used the well known property of the decomposition:

$$(\boldsymbol{U}\boldsymbol{\Phi}\boldsymbol{U}^{-1})^k = (\boldsymbol{U}\boldsymbol{\Phi}\underbrace{\boldsymbol{U}^{-1}\boldsymbol{U}}_{=\boldsymbol{I}}\boldsymbol{\Phi}\boldsymbol{U}^{-1}...\boldsymbol{U}\boldsymbol{\Phi}\boldsymbol{U}^{-1}) = \boldsymbol{U}\boldsymbol{\Phi}^k\boldsymbol{U}^{-1}. \tag{2.34}$$

If we can find such a matrix $\boldsymbol{R}$ with $e^{\boldsymbol{R}} = \boldsymbol{P}$ and $\boldsymbol{P}$ a stochastic matrix, then we can interpolate the series

$$\boldsymbol{P}^n = e^{n\boldsymbol{R}}.$$

We can then think of making the discrete steps n arbitrarily small and eventually end up with a continuous process. It should therefore not surprise us that a continuous time Markov process is defined by a rate matrix $\boldsymbol{R}$ and transition probabilities $\boldsymbol{P}(t) = e^{t\boldsymbol{R}}$. We shall next establish important properties of the matrix exponential and show that the exponential of a rate matrix is a stochastic matrix. This result then allows us to give the fundamental definition of a CTMP.

Theorem 3 (General matrix exponential) *Given a matrix $\boldsymbol{R}$ on a finite set (not necessarily a rate matrix), set $\boldsymbol{P}(t) = e^{t\boldsymbol{R}}$, $t \geq 0$. Then, $\boldsymbol{P}(t)$ has the following properties:*

(i) $\boldsymbol{P}(h+t) = \boldsymbol{P}(h)\boldsymbol{P}(t)$, *for all* s, t
(ii) $\boldsymbol{P}(t)$ *and* $\boldsymbol{R}$ *commute for all* t, *i.e.,* $\boldsymbol{P}(t)\boldsymbol{R} = \boldsymbol{R}\boldsymbol{P}(t)$,
(iii) $\boldsymbol{P}(t)$ *is the unique solution to the* forward equation

$$\frac{d}{dt}\boldsymbol{P}(t) = \boldsymbol{P}(t)\boldsymbol{R}, \quad \boldsymbol{P}(0) = \boldsymbol{I}, \tag{2.35}$$

(iv) $\boldsymbol{P}(t)$ *is the unique solution to the* backward equation

$$\frac{d}{dt}\boldsymbol{P}(t) = \boldsymbol{R}\boldsymbol{P}(t), \quad \boldsymbol{P}(0) = \boldsymbol{I}. \tag{2.36}$$

□

Proof If we suppose that $\boldsymbol{R}$ can be decomposed into $\boldsymbol{R} = \boldsymbol{U}\boldsymbol{\Phi}\boldsymbol{U}^{-1}$, the proof is fairly simple but lengthy, and was moved to Appendix B. It is also possible to prove (i)-(iv) without assuming the existence of the eigenvalue decomposition of $\boldsymbol{R}$ (cf. [Nor97, pp. 62]), however, we shall see that for all CTMP considered in this work, $\boldsymbol{R}$ is diagonalizable in the required form. ■

Theorem 4 (Rate matrix exponential) *A matrix $\boldsymbol{R}$ is a rate matrix iff $\boldsymbol{P}(t) = e^{t\boldsymbol{R}}$ is a stochastic matrix for all $t \geq 0$.* □

Proof For small t we have (Eq. (2.30)):

$$\boldsymbol{P}(t) \approx \boldsymbol{I} + t\boldsymbol{R} + O(t^2), \tag{2.37}$$

where $O(t)$ means that there is a t_0 such that $O(t)/t \leq C$ for all $t < t_0$, for some constant C. It follows that $r_{s_is_j} \geq 0$ for $s_i \neq s_j$ iff $p_{s_is_j} \geq 0$ for small enough t. Since $\boldsymbol{P}(t) = \boldsymbol{P}(t/n)^n$ for any n, we can make t/n arbitrarily small, and it follows that $r_{s_is_j} \geq 0$ for $s_i \neq s_j$ if and only if $p_{s_is_j} \geq 0$ for *all* t.

Let $\boldsymbol{e}$ denote the all-one vector $\boldsymbol{e} = [1, 1, .., 1]^T$. If $\boldsymbol{R}$ has zero row sum $\boldsymbol{Re} = \boldsymbol{0}$, so does $\boldsymbol{R}^n$: $\boldsymbol{R}^n\boldsymbol{e} = \boldsymbol{R}^{n-1}\boldsymbol{0} = \boldsymbol{0}$. It follows that $\boldsymbol{P}$ has row sum 1 for all t:

$$\boldsymbol{P}(t)\boldsymbol{e} = e^{t\boldsymbol{R}}\boldsymbol{e} \overset{(2.30)}{=} \boldsymbol{Ie} + t\boldsymbol{Re} + \frac{1}{2!}t\boldsymbol{R}^2\boldsymbol{e} + ... = \boldsymbol{Ie} = \boldsymbol{e}. \tag{2.38}$$

If, on the other hand, $\boldsymbol{P}(t)\boldsymbol{e} = \boldsymbol{e}$, then

$$\frac{d}{dt}(\boldsymbol{P}(t)\boldsymbol{e}) = \frac{d\boldsymbol{P}(t)}{dt}\boldsymbol{e} \overset{(a)}{=} \boldsymbol{RP}(\boldsymbol{t})\boldsymbol{e} = \boldsymbol{Re} = \boldsymbol{0}, \tag{2.39}$$

where (a) is due to Theorem 3. This proves the claim since we showed that $r_{s_is_j} \geq 0$ iff $p_{s_is_j} \geq 0$ for all t and $\boldsymbol{Pe} = \boldsymbol{e}$ iff $\boldsymbol{Re} = \boldsymbol{0}$. ■

The following theorem states the key results for CTMP:

Theorem 5 (Continuous time Markov process) *Let $\{\mathsf{X}_t\}_{t\geq 0}$ be a continuous process with state space $\mathcal{S}$ that satisfies the Markov property, i.e., for all $h, t \geq 0$, X_{t+h} is conditionally independent of $(\mathsf{X}_s : s \leq t)$ given X_t. Let $\boldsymbol{R}$ be a rate matrix. Then, for all $n = 0, 1, 2, ..$ and all times $t_0 \leq t_1 \leq ... \leq t_{n+1}$ and all states $s_{i_0}, s_{i_1}, ..., s_{i_{n+1}}$, we have*

$$P(x_{t_{n+1}} = s_{i_{n+1}} | x_{t_n} = s_{i_n}, ..., x_{t_0} = s_{i_0}) = p_{s_{i_n}s_{i_{n+1}}}(t_{n+1} - t_n), \tag{2.40}$$

□

where $p_{s_is_j}(t)$ is given by $\boldsymbol{P}(t) = e^{t\boldsymbol{R}}$, with $[\boldsymbol{P}(t)]_{ij} = p_{s_is_j}(t)$.

Proof See Appendix B. ■

We call such a process $\{\mathsf{X}_t\}_{t\geq 0}$ a continuous time Markov process with rate matrix $\boldsymbol{R}$ which is sometimes also referred to as *generator matrix*. So far, we showed that a Markov process $\{\mathsf{X}_t\}_{t\geq 0}$ is defined by a rate matrix $\boldsymbol{R}$ and that transition probabilities at any t are given by the matrix exponential $\boldsymbol{P}(t) = e^{t\boldsymbol{R}}$. The remainder of this section defines homogeneity, stationarity, convergence and reversibility for this process:

2.4.3 Properties of continuous time Markov processes

Definition 2.4.6 (Homogeneity) A continuous time Markov process is *homogenous* if the transition probabilities are independent of t, i.e., for all $h \geq 0$,

$$P(x_{t+h} = s_j | x_h = s_i) = P(x_t = s_j | x_0 = s_i). \tag{2.41}$$

□

We say that a homogenous CTMP $\{\mathsf{X}_t\}_{t\geq 0}$ is $\texttt{Markov}(\boldsymbol{\lambda}, \boldsymbol{R})$, where $\boldsymbol{\lambda}$ is the distribution of X_0 and $\boldsymbol{R}$ the rate matrix (generator) of the process. As for DTMP the state vector $\boldsymbol{p}(t)$ is defined as $[\boldsymbol{p}(t)]_i = P(x_t = s_i)$ with $\boldsymbol{p}(0) = \boldsymbol{\lambda}$ and for any $t, h \geq 0$

$$\boldsymbol{p}(t) = \boldsymbol{p}(0)\boldsymbol{P}(t) \tag{2.42a}$$

$$\boldsymbol{p}(t+h) = \boldsymbol{p}(0)\boldsymbol{P}(t+h) \tag{2.42b}$$

$$= \boldsymbol{p}(t)\boldsymbol{P}(h). \tag{2.42c}$$

In the following, we assume that all states *communicate*, i.e., $r_{s_i s_j} > 0$ for all $s_i \neq s_j$. By [Nor97, Theorem 3.2.1], this implies that also $p_{s_i s_j} > 0$ for all s_i, s_j. As discussed above for DTMP, the process generated by $\boldsymbol{R}$ then converges to a stationary distribution. We stated in Definition 2.4.4 that $\boldsymbol{\pi}\boldsymbol{P} = \boldsymbol{\pi}$ holds for the stationary distribution of a DTMP. Now we extend the notion of stationarity to CTMPs:

Theorem 6 (Stationarity) *Let* $\{\mathsf{X}_t\}_{t\geq 0}$ *be* $\texttt{Markov}(\lambda, \boldsymbol{R})$*, the following is equivalent:*

(i) $\boldsymbol{\pi}\boldsymbol{R} = \boldsymbol{0}$.
(ii) $\boldsymbol{\pi}\boldsymbol{P}(t) = \boldsymbol{\pi}$. □

Proof Let $\boldsymbol{R} = \boldsymbol{U}\boldsymbol{\Phi}\boldsymbol{U}^{-1}$, then $\boldsymbol{R}$ has eigenvalues $\phi_1, \phi_2, ..., \phi_{\mathcal{S}}$ and $\boldsymbol{P}(t)$ has eigenvalues $e^{t\phi_1}, e^{t\phi_2}, ..., e^{t\phi_{\mathcal{S}}}$ with the same corresponding eigenvectors according to Eq. (2.32c). If there is a stationary distribution $\boldsymbol{\pi}$ that satisfies $\boldsymbol{\pi}\boldsymbol{P}(t) = \boldsymbol{\pi}$ for all t, then $\boldsymbol{\pi}$ is the left eigenvector of $\boldsymbol{P}(t)$ with corresponding eigenvalue $e^{t\phi} = 1$. So ϕ must be zero and the corresponding eigenvector $\boldsymbol{\pi}$ satisfies $\boldsymbol{\pi}\boldsymbol{R} = \boldsymbol{0}$ ■

Definition 2.4.7 (Stationary process) A process $\{\mathsf{X}_t\}_{t\geq 0}$ is called *stationary* if it is $\texttt{Markov}(\boldsymbol{\pi}, \boldsymbol{R})$ with $\boldsymbol{\pi}\boldsymbol{R} = \boldsymbol{0}$, i.e., $\{\mathsf{X}_t\}_{t\geq 0}$ has stationary distribution $\boldsymbol{\pi}$ and initial distribution $\boldsymbol{\pi}$. □

Theorem 7 *Let* $\{\mathsf{X}_t\}_{t\geq 0}$ *be* $\texttt{Markov}(\boldsymbol{\pi}, \boldsymbol{R})$ *with stationary distribution* $\boldsymbol{\pi}$*, then* $\{\mathsf{X}_{h+t}\}_{t\geq 0}$ *is also* $\texttt{Markov}(\boldsymbol{\pi}, \boldsymbol{R})$. □

Proof From the property of stationarity follows that $\boldsymbol{p}(h) = \boldsymbol{\pi}\boldsymbol{P}(h) = \boldsymbol{\pi}$, the claim is then clear from the Markov and homogeneity property of the process. ■

The convergence properties are derived from the theory established for DTMP: according to Theorem 2, the *discrete time process* with transition probability matrix $\boldsymbol{P}(t)$ converges to $\boldsymbol{\pi}$. So for any t fixed:

$$\lim_{n\to\infty} \boldsymbol{\lambda}(\boldsymbol{P}(t))^n \to \boldsymbol{\pi}, \quad \text{for any distribution } \boldsymbol{\lambda}. \tag{2.43}$$

Since $\boldsymbol{P}(t)^n = \boldsymbol{P}(nt)$, it can also be easily shown that the continuous-time process generated by $\boldsymbol{R}$ converges to $\boldsymbol{\pi}$ [Nor97, pp.123]:

$$\lim_{t\to\infty} \boldsymbol{\lambda}\boldsymbol{P}(t) \to \boldsymbol{\pi} \quad \text{for any distribution } \boldsymbol{\lambda}. \tag{2.44}$$

In this work we shall only consider processes that are stationary and *reversible*, defined below:

Theorem 8 *Let* $\{\mathsf{X}_t\}_{0\le t\le T}$ *be* $\texttt{Markov}(\boldsymbol{\pi}, \boldsymbol{R})$ *with stationary distribution* $\boldsymbol{\pi}$*. Then, the process* $\{\mathsf{X}_{T-t}\}_{0\le t\le T}$ *is* $\texttt{Markov}(\boldsymbol{\pi}, \hat{\boldsymbol{R}})$ *with*

$$\pi_{s_j}\hat{r}_{s_j s_i} = \pi_{s_i} r_{s_i s_j},$$

Proof cf. [Nor97, pp.124] [Kel79]. ■

Definition 2.4.8 (Reversibility) A process is called *reversible* if $\boldsymbol{R}$ and $\boldsymbol{\pi}$ are in *detailed balance*

$$\pi_{s_j} r_{s_j s_i} = \pi_{s_i} r_{s_i s_j}, \quad \text{for all } s_i, s_j. \tag{2.45}$$

□

Due to Theorem 8, if reversibility holds the processes $\{\mathsf{X}_t\}_{0\le t\le T}$ and $\{\mathsf{X}_{T-t}\}_{0\le t\le T}$ are both $\texttt{Markov}(\boldsymbol{\pi}, \boldsymbol{R})$. Hence, for any t, the discrete time processes $\{\mathsf{X}_{nt}\}_{0\le n\le N}$ and $\{\mathsf{X}_{(N-n)t}\}_{0\le n\le N}$ are $\texttt{Markov}(\boldsymbol{\pi}, \boldsymbol{P}(t))$, i.e., reversible. Therefore, reversibility implies that for any t

$$\pi_{s_j} p_{s_j s_i}(t) = \pi_{s_i} p_{s_i s_j}(t), \quad \text{for all } s_i, s_j \in \mathcal{S}. \tag{2.46}$$

Reversibility says that the expected amount of changes from state s_j to s_i in steady state equals the amount of changes from s_i to s_j.

Reversibility further implies that $\boldsymbol{R}$ can be decomposed. This is shown next since we assumed this property of $\boldsymbol{R}$ throughout this section: let $\boldsymbol{\Pi} = \operatorname{diag}(\boldsymbol{\pi})$, then the detailed balance equality Eq. (2.45) can be written as

$$\boldsymbol{\Pi}\boldsymbol{R} = \boldsymbol{R}^T\boldsymbol{\Pi}. \tag{2.47}$$

Consider the matrix $\boldsymbol{\Pi}^{1/2}\boldsymbol{R}\boldsymbol{\Pi}^{-1/2}$, we find that it is symmetric if reversibility is satisfied:

$$\begin{aligned}
(\boldsymbol{\Pi}^{1/2}\boldsymbol{R}\boldsymbol{\Pi}^{-1/2})^T &= \boldsymbol{\Pi}^{-1/2}(\boldsymbol{R}^T\boldsymbol{\Pi})\boldsymbol{\Pi}^{-1/2} && \text{(2.48a)}\\
&\overset{(2.47)}{=} \boldsymbol{\Pi}^{-1/2}(\boldsymbol{\Pi}\boldsymbol{R})\boldsymbol{\Pi}^{-1/2} && \text{(2.48b)}\\
&= \boldsymbol{\Pi}^{1/2}\boldsymbol{R}\boldsymbol{\Pi}^{-1/2}, && \text{(2.48c)}
\end{aligned}$$

and therefore, there exists a real orthogonal matrix $\boldsymbol{U}'$ such that

$$\boldsymbol{\Pi}^{1/2}\boldsymbol{R}\boldsymbol{\Pi}^{-1/2} = \boldsymbol{U}'\boldsymbol{\Phi}\boldsymbol{U}'^T, \tag{2.49}$$

where $\boldsymbol{\Phi}$ denotes the diagonal matrix of eigenvalues of $\boldsymbol{R}$. It follows that

$$\begin{aligned}
\boldsymbol{R} &= \boldsymbol{\Pi}^{-1/2}\boldsymbol{U}'\boldsymbol{\Phi}\boldsymbol{U}'^T\boldsymbol{\Pi}^{1/2} && \text{(2.50a)}\\
&= \boldsymbol{U}\boldsymbol{\Phi}\boldsymbol{U}^{-1}, && \text{(2.50b)}
\end{aligned}$$

with[3] $\boldsymbol{U} \doteq \boldsymbol{\Pi}^{-1/2}\boldsymbol{U}'$. Hence, the decomposition always exists for reversible rate matrices.

[3]Recall that $(\boldsymbol{\Pi}^{-1/2}\boldsymbol{U}')^{-1} = \boldsymbol{U}'^{-1}\boldsymbol{\Pi}^{1/2}$ and $\boldsymbol{U}'^{-1} = \boldsymbol{U}'^T$

2.4.4 Example: continuous evolution of a quarternary symbol

Consider the CTMP defined over a quaternary state space $\mathcal{S} = \{1, 2, 3, 4\}$ with generator matrix

$$\boldsymbol{R} = \begin{pmatrix} -3\alpha & \alpha & \alpha & \alpha \\ \alpha & -3\alpha & \alpha & \alpha \\ \alpha & \alpha & -3\alpha & \alpha \\ \alpha & \alpha & \alpha & -3\alpha \end{pmatrix}.$$

By solving the forward equations (Theorem 3):

$$\begin{pmatrix} \dot{p}_{11}(t) & ... & \dot{p}_{14}(t) \\ \vdots & \ddots & \\ \dot{p}_{41}(t) & ... & \dot{p}_{44}(t) \end{pmatrix} = \begin{pmatrix} p_{11}(t) & ... & p_{14}(t) \\ \vdots & \ddots & \\ p_{41}(t) & ... & p_{44}(t) \end{pmatrix} \begin{pmatrix} -3\alpha & ... & \alpha \\ \vdots & \ddots & \\ \alpha & ... & -3\alpha \end{pmatrix}$$

with constraints

$$\begin{pmatrix} p_{11}(0) & ... & p_{14}(0) \\ \vdots & \ddots & \\ p_{41}(0) & ... & p_{44}(0) \end{pmatrix} = \begin{pmatrix} 1 & & \mathbf{0} \\ & \ddots & \\ \mathbf{0} & ... & 1 \end{pmatrix}, \tag{2.52}$$

we find that the transition probabilities are given by

$$p_{ij} = \frac{1}{4} \begin{cases} 1 + 3e^{-4\alpha t} & \text{for } i = j \\ 1 - e^{-4\alpha t} & \text{for } i \neq j \end{cases}. \tag{2.53}$$

Note that such a process models a quarternary symmetric channel (QSC)[4] as presented in Example 2.2.2 with $\mu = \frac{1}{4}\left(1 - e^{-4\alpha t}\right)$ for a fixed αt.

Since $r_{ij} > 0$, $\forall i \neq j$ in $\boldsymbol{R}$, all states communicate, and as shown above the process then converges to a stationary distribution $\boldsymbol{\pi}$. From Theorem 6 we find that the stationary distribution is uniform

$$\boldsymbol{\pi}\boldsymbol{R} \overset{!}{=} \mathbf{0} \Rightarrow \boldsymbol{\pi} = [1/4, ..., 1/4],$$

and hence, the process is reversible according to Definition 2.4.8. From Eq. (2.53) we can further derive the limiting behavior of the transition probabilities

$$\lim_{t\to\infty} p_{ij}(t) \to \frac{1}{4} \quad \forall i, j. \tag{2.54}$$

We analyze how fast the process converges by calculating the state vector $\boldsymbol{p}(t)$ for $\alpha t = 0.1, 1, 10$ with initial distribution $\boldsymbol{\lambda} = [1, 0, 0, 0]$:

$$\begin{aligned} \boldsymbol{p}(0.1/\alpha) &= [0.7528, 0.0824, 0.0824, 0.0824] \\ \boldsymbol{p}(1/\alpha) &= [0.2638, 0.2454, 0.2454, 0.2454] \\ \boldsymbol{p}(10/\alpha) &= [0.2500, 0.2500, 0.2500, 0.2500]. \end{aligned}$$

[4]However, not all QSCs can be modeled by this process but only those with $0 \leq \mu \leq \frac{1}{4}$ while a general QSC can have $0 \leq \mu \leq \frac{1}{3}$. This implies that all channels with $p_{ii} < \frac{1}{4}$ cannot be modeled via this process.

2.4.5 Hidden Markov processes

Hidden Markov processes are used in Chapter 5 as models of molecular evolution and to detect potentially functional DNA regions, and in Chapter 6 as a model for noisy data streams encoded with a convolutional code.

Markov processes are frequently used system models in engineering and science applications. Often, one is interested in controlling or tracking the state of a given system over time. However, in practice the system state may not be directly observable, and one can only measure variables that depend on the state. Hidden Markov processes (HMP) are an extension of classical Markov processes, allowing to model systems with unobservable state and observable variables. In the following, we define HMPs and show how to efficiently estimate the system state from such a model.

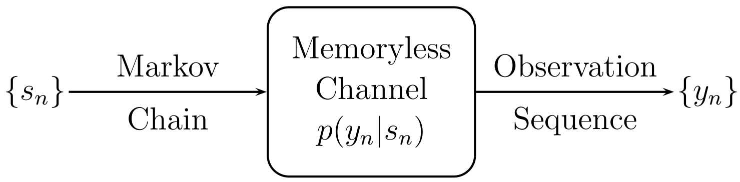


Figure 2.9: A hidden Markov process can be interpreted as a Markov chain state sequence observed through a discrete memoryless channel.

From an information theoretic perspective, an HMP is a DTMP with state space $\mathcal{S}$ that is observed through a memoryless channel $(\mathcal{S}, p(y_n|s_n), \mathcal{Y})$ as shown in Figure 2.9. At each time step n, the HMP emits a symbol $y_n \in \mathcal{Y}$ from an observation space $\mathcal{Y}$, where the y_n is emitted according to the distribution $p(y_n|s_n)$. The output of the HMP is the sequence $y_0, y_1, y_2, ...$, while the state sequence $s_0, s_1, ..$ cannot be observed. We can therefore define an HMP as follows:

Definition 2.4.9 (Hidden Markov process) Let $\{\mathsf{S}_n\}_{0 \le n \le N}$, $s_n \in \mathcal{S}$ be $\texttt{Markov}(\boldsymbol{\lambda}, \boldsymbol{P})$ as defined in Section 2.4 and let $\mathsf{Y}_n,\ y_n \in \mathcal{Y}$, be a random variable that depends on S_n through $p(y_n|s_n)$. The variable S_n is called the *hidden state* at time n, and Y_n is the observable output of the HMP at time n and is called *observation*. An HMP is therefore defined by the following set of parameters $\{\boldsymbol{\lambda}, \boldsymbol{P}, \mathcal{S}, p(y_n|s_n), \mathcal{Y}\}$ which represent initial distribution, transition, state space of the Markov chain S_n, emission probabilities, and observation space, respectively. □

From the equations for memoryless channels (2.13) and for the distribution of a DTMP (cf. Eq. (2.23a)), we can derive the joint distribution of observations and states for an

HMP:

$$p(y_0, y_1, ...y_N, s_0, s_1, ..., s_N) = p(y_0, y_1, ...y_N|s_0, s_1, ..., s_N)p(s_0, s_1, ..., s_N) \tag{2.55a}$$

$$= \left(\prod_{n=0}^{N} p(y_n|s_n)\right)\left(p(s_0)\prod_{n=1}^{N} p(s_n|s_{n-1})\right) \tag{2.55b}$$

$$= p(y_0, s_0)\prod_{n=1}^{N} p(y_n|s_n)p(s_n|s_{n-1}) \tag{2.55c}$$

$$= p(y_0, s_0)\prod_{n=1}^{N} p(y_n, s_n|s_{n-1}), \tag{2.55d}$$

where $p(y_0, s_0)$ and $p(y_n, s_n|s_{n-1})$ are determined from the parameters of the HMP as follows:

$$p(y_0, s_0) = \lambda_{s_0} p(y_0|s_0) \tag{2.56a}$$

$$p(y_n, s_n|s_{n-1}) = p_{s_{n-1},s_n}\ p(y_n|s_n), \quad n = 1, 2, ..., N. \tag{2.56b}$$

We next present efficient algorithms for state estimation of an HMP given a sequence of observations $y_0, y_1, .., y_N$. The following notation is used: a single subscript y_n denotes a single observation (realization of the r. v.) at time index n. The superscript notation $\boldsymbol{y}^n$ is used to denote all observations up to time index n, i.e., $y_0, y_1, .., y_n$, and a joint sub- and superscript $\boldsymbol{y}_{n+1}^N$ denotes the sequence $y_{n+1}, ..., y_N$.

We are interested in finding the *maximum a posteriori (MAP)* state probability

$$\hat{s}_n = \arg\max_{s_n}\{p(s_n|\boldsymbol{y}^N)\}. \tag{2.57}$$

The key idea is to split the calculation of $p(s_n, \boldsymbol{y}^N)$ into a *forward density* $p(s_n, \boldsymbol{y}^n)$ and a *backward* density $p(\boldsymbol{y}_{n+1}^N|s_n)$:

$$p(s_n, \boldsymbol{y}^N) = p(s_n, \boldsymbol{y}^n, \boldsymbol{y}_{n+1}^N) \tag{2.58a}$$

$$= p(\boldsymbol{y}_{n+1}^N|s_n, \boldsymbol{y}^n)p(s_n, \boldsymbol{y}^n) \tag{2.58b}$$

$$\overset{(a)}{=} p(\boldsymbol{y}_{n+1}^N|s_n)p(s_n, \boldsymbol{y}^n), \tag{2.58c}$$

where (a) holds because of the memoryless property of the channel. To see why this is helpful, we shall calculate the forward density for a few n:
for $n = 0$:

$$p(s_0, y_0) = p(y_0|s_0)\lambda_{s_0}, \tag{2.59}$$

for $n = 1$:

$$p(s_1, y_0, y_1) = \sum_{s_0} p(s_1, s_0, y_0, y_1) \tag{2.60a}$$

$$= \sum_{s_0} p(y_0|s_1, s_0, y_1)p(s_1, s_0, y_1) \tag{2.60b}$$

$$= \sum_{s_0} p(s_0, y_0)p(y_1|s_1)p(s_1|s_0), \tag{2.60c}$$

where $p(s_0, y_0)$ is given from the calculation for $n = 0$.
For general n:

$$p(s_n, \boldsymbol{y}^n) = \sum_{s_{n-1}} p(s_{n-1}, s_n, \boldsymbol{y}^n) \tag{2.61a}$$

$$= \sum_{s_{n-1}} p(\boldsymbol{y}^{n-1}|s_n, s_{n-1}, y_n)p(s_n, s_{n-1}, y_n) \tag{2.61b}$$

$$\stackrel{(a)}{=} \sum_{s_{n-1}} p(s_{n-1}, \boldsymbol{y}^{n-1})p(y_n|s_n)p(s_n|s_{n-1}), \tag{2.61c}$$

where in (a) it is used that $\boldsymbol{y}^{n-1}$ is independent of y_n and s_n given s_{n-1}, and y_n is independent of s_{n-1} given s_n. The forward densities can therefore be calculated in a recursive manner, and the backward density can be calculated similarly. In Summary, the forward and backward densities satisfy the following recursions [EM02]:

$$p(s_n, \boldsymbol{y}^n) = \begin{cases} \lambda_{s_0} p(y_0|s_0) & n = 0 \\ p(y_n|s_n) \sum_{s_{n-1}} p(s_{n-1}, \boldsymbol{y}^{n-1}) p_{s_{n-1}s_n} & n = 1, 2, .., N \end{cases}, \tag{2.62a}$$

$$p(\boldsymbol{y}_{n+1}^N|s_n) = \begin{cases} 1 & n = N \\ \sum_{s_{n+1}} p(y_{n+2}^N|s_{n+1}) p_{s_n s_{n+1}} p(y_{n+1}|s_{n+1}) & n = N-1, .., 0 \end{cases}. \tag{2.62b}$$

The MAP estimate $\hat{s}_n$ is then given as

$$\hat{s}_n = \arg\max_{s_n} \left\{ \frac{p(s_n, \boldsymbol{y}^n)p(\boldsymbol{y}_{n+1}^N|s_n)}{\sum_{s_n} p(s_n, \boldsymbol{y}^n)p(\boldsymbol{y}_{n+1}^N|s_n)} \right\}, \tag{2.63}$$

where we substituted Eq. (2.58c) into $p(s_n|\boldsymbol{y}^N) = \frac{p(s_n, \boldsymbol{y}^N)}{\sum_{s_n} p(s_n, \boldsymbol{y}^N)}$. In the context of error correcting codes, a variant of this algorithm is known as the BCJR algorithm, used for maximum a posteriori symbol decoding of convolutional codes [JZ99].

2.5 Maximum likelihood estimation

A frequently encountered situation in statistics is the problem of accurately estimating model parameters from data generated by a system under study. Let the random variable $\mathsf{X} = [\mathsf{X}_1, \mathsf{X}_2, ..]$ model the observable data with a parameter dependable distribution

$$p_{\mathsf{X}}(\boldsymbol{x}; \theta) \tag{2.64}$$

specifying the system. The system is characterized by a variable parameter[5] θ taking values in some parameter space Θ. Data is used to make statements about a proposed system model, and the goal is to infer the system state represented by the parameter θ. Likelihood is a central concept in statistical inference and leads to an important method: the maximum likelihood (ML) estimation, reviewed in this section.

[5] We will concentrate on the case where θ is a scalar value. All results can be generalized to the vector case.

2.5.1 Basic principle

Assume we observe a sequence of observations $\boldsymbol{x} = [x_1, ..., x_N]$, then the *likelihood function* of $\theta \in \Theta$ for the data $\boldsymbol{x}$ is defined as

$$\Theta \to [0,1],\ \theta \mapsto l_{\boldsymbol{x}}(\theta) \sim P(\boldsymbol{x};\theta). \tag{2.65}$$

For a *fixed* data vector $\boldsymbol{x}$, the likelihood function is proportional to the probability for observing $\boldsymbol{x}$ as a *function of* θ. Note the difference to a pdf $p(\boldsymbol{x};\theta)$ that is viewed as varying with $\boldsymbol{x}$ at fixed θ. Therefore, the likelihood is *not* a probability density. It is often convenient to consider the log of the likelihood, and we therefore introduce the notation

$$ll_{\boldsymbol{x}}(\theta) \doteq \log\left(l_{\boldsymbol{x}}(\theta)\right). \tag{2.66}$$

Example 2.5.1 *(Bernoulli distributed binary random variable) Let* X_i *be binary iid distributed random variables with* $P(x_i = 1) = \theta$ *and* $P(x_i = 0) = 1 - \theta$, $\theta \in [0,1]$, *then the likelihood function for a vector* $\boldsymbol{x} = [x_1, ..., x_N]$ *is given by*

$$l_{\boldsymbol{x}}(\theta) = \prod_{i=1}^{N} x_i\theta + (1 - x_i)(1 - \theta) = (1 - \theta)^{(N - w_H(\boldsymbol{x}))}\theta^{w_H(\boldsymbol{x})},$$

with $w_H(\boldsymbol{x})$ *denoting the Hamming weight (number of non-zero elements) of* $\boldsymbol{x}$. *The likelihood function over* θ *for a sample realization of* $\boldsymbol{x}$ *with* $\theta = 0.2$ *and* $N = 20$ *is shown in Figure 2.10.* □

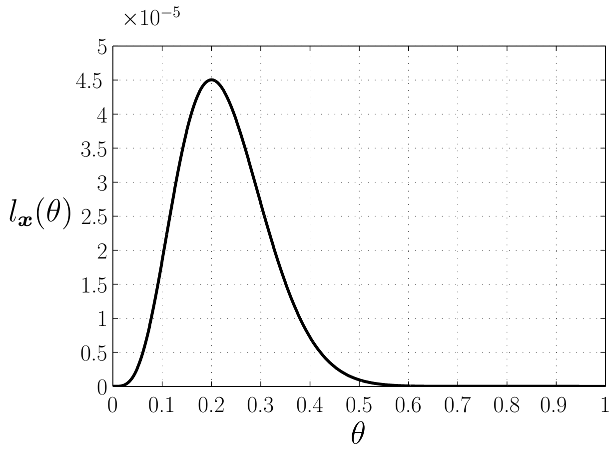


Figure 2.10: Likelihood function for a sample realization of an iid Bernoulli random variable.

In Figure 2.10, it can be observed that the likelihood function has a maximum around $\theta = 0.2$, which was actually the true value used to generate the data. The likelihood rapidly decreases when moving away from 0.2. It seems therefore intuitive to choose an estimator for θ that *maximizes* the likelihood of the observed data:

Definition 2.5.1 (Maximum likelihood estimator) Let X_i be a discrete random variable and x_i take values in a finite set $\mathcal{X}$, and let $\mathbf{X} = [\mathsf{X}_1, \mathsf{X}_2, .., \mathsf{X}_N]$ have distribution $p_{\mathbf{X}}(\boldsymbol{x};\theta)$ with model parameter $\theta \in \Theta$. Given the observation $\boldsymbol{x} = [x_1, x_2, ..., x_N]$, the maximum likelihood estimator (MLE) for θ is defined as

$$\hat{\theta}_{\mathrm{MLE}} = \arg\max_{\theta\in\Theta}\{l_{\boldsymbol{x}}(\theta)\}. \tag{2.67}$$

□

The MLE chooses the parameter θ that provides the best explanation, in the likelihood sense, of the observed data $\boldsymbol{x}$ under the probability model (2.64).

2.5.2 Example: ML distance of temporally diverged sequences

Let $\mathsf{X}_0^{(i)}$ and $\mathsf{X}_t^{(i)}$, $i = 1, .., N$ be two sequences of random variables of the $\mathtt{Markov}(\boldsymbol{\pi}, \boldsymbol{R})$ CTMP $\{\mathsf{X}_t^{(i)}\}_{t\geq 0}$ with quarternary state space $\mathcal{S} = \{1, 2, 3, 4\}$ as given in Example 2.4.4. Assume we observe the two sequences $\boldsymbol{x}_0 = [x_0^{(1)}, x_0^{(2)}, .., x_0^{(N)}]$ and $\boldsymbol{x}_t = [x_t^{(1)}, x_t^{(2)}, .., x_t^{(N)}]$ after fixed but unknown time t. The pairs $(\mathsf{X}_0^{(i)}, \mathsf{X}_t^{(i)})$, $i = 1, .., N$ are iid according to (cf. Example 2.4.4):

$$p(x_0^{(i)}, x_t^{(i)}; \alpha t) = p(x_t^{(i)}|x_0^{(i)}; \alpha t)p(x_0^{(i)}) = \pi_{s_0} p_{s_0 s_t} = \frac{1}{4} p_{s_0 s_t} = \frac{1}{4}\begin{cases} \frac{3}{4} + \frac{1}{4}e^{-4\alpha t} & s_0 = s_t \\ \frac{1}{4} - \frac{1}{4}e^{-4\alpha t} & s_0 \neq s_t \end{cases}.$$

The parameter of interest is the temporal distance αt of the sequences. Define $p = P(x_0^{(i)} \neq x_t^{(i)}) = \sum_{x_0^{(i)} \neq x_t^{(i)}} p(x_0^{(i)}, x_t^{(i)}; \alpha t)$, then the likelihood function for αt given the two sequences $(\boldsymbol{x}_0, \boldsymbol{x}_t)$ is

$$l_{(\boldsymbol{x}_0,\boldsymbol{x}_t)}(\alpha t) = P(\boldsymbol{x}_0, \boldsymbol{x}_t; \alpha t) = (1-p)^{N - d_H(\boldsymbol{x}_0,\boldsymbol{x}_t)} p^{d_H(\boldsymbol{x}_0,\boldsymbol{x}_t)},$$

where $d_H(\boldsymbol{x}_0, \boldsymbol{x}_t)$ denotes the Hamming distance between $\boldsymbol{x}_0$ and $\boldsymbol{x}_t$ (number of positions in which they differ). As the logarithm is a monotonically increasing function, we can as well maximize $ll_{(\boldsymbol{x}_0,\boldsymbol{x}_t)}(\alpha t)$. We find the MLE for αt by setting the derivative of $ll_{(\boldsymbol{x}_0,\boldsymbol{x}_t)}(\alpha t)$ with respect to p to zero:

$$\begin{aligned} ll_{(\boldsymbol{x}_0,\boldsymbol{x}_t)}(\alpha t) &= (N - d_H(\boldsymbol{x}_0, \boldsymbol{x}_t)) \log(1-p) + d_H(\boldsymbol{x}_0, \boldsymbol{x}_t) \log(p), && (2.68a) \\ \frac{\mathrm{d}ll_{(\boldsymbol{x}_0,\boldsymbol{x}_t)}(\alpha t)}{\mathrm{d}p} &= -(N - d_H(\boldsymbol{x}_0, \boldsymbol{x}_t))\frac{1}{(1-p)} + d_H(\boldsymbol{x}_0, \boldsymbol{x}_t)\frac{1}{p} && (2.68b) \\ &= 0. && (2.68c) \end{aligned}$$

This gives a very intuitive estimate for p:

$$\hat{p} = \frac{d_H(\boldsymbol{x}_0, \boldsymbol{x}_t)}{N}. \tag{2.69}$$

The exact probability of $x_0^{(i)} \neq x_t^{(i)}$ is easily calculated as $p = \frac{3}{4}(1 - e^{-4\alpha t})$. Substituting p in Eq. (2.69) finally yields

$$\hat{\alpha t} = -\frac{1}{4}\log\left(1 - \frac{4}{3}\frac{d_H(\boldsymbol{x}_0, \boldsymbol{x}_t)}{N}\right). \tag{2.70}$$

In genetics, this quantity is famously known as the Felsenstein correction [EM02, EG05] for an estimate of evolutionary distance of DNA sequences.

2.5.3 Properties of the MLE

To justify the choice of the MLE, we briefly state the main results in its analysis showing that it is an optimal estimator, at least if we have a lot of observations. An important measure in the analysis of any estimator is the *Fisher information* [Kay93]:

Definition 2.5.2 (Fisher information) Suppose X has distribution $p(\boldsymbol{x};\theta)$, the Fisher information is defined to be

$$I(\theta) = \sum_{\boldsymbol{x}} p(\boldsymbol{x};\theta)\left(\frac{\mathrm{d}ll_{\boldsymbol{x}}(\theta)}{\mathrm{d}\theta}\right)^2 = E_{\boldsymbol{x}}\left\{\left(\frac{\mathrm{d}ll_{\boldsymbol{x}}(\theta)}{\mathrm{d}\theta}\right)^2\right\}. \tag{2.71}$$

□

The Fisher information is a measure for the expected curvature of the log-likelihood function. For independent random variables X_i, the Fisher information is given as

$$I(\theta) = \sum_i I_i(\theta), \tag{2.72}$$

and for iid random variables, $I_i(\theta)$ is constant for all i, and the amount of information in a sample of size N is exactly N times larger than that contained in a single observation

$$I(\theta) = N I_1(\theta). \tag{2.73}$$

The following is a fundamental result in estimation theory:

Theorem 9 (Cramér-Rao lower bound (CLRB)) *Let $\boldsymbol{x}$ be a sample drawn from the distribution $p(\boldsymbol{x};\theta)$ and let $\hat{\theta}$ be an unbiased estimator for θ, i.e., $E\{\hat{\theta}\} = \theta$, then the variance of the estimate $\hat{\theta}$ is lower bounded by the inverse of the Fisher information*

$$E\{(\theta - \hat{\theta})^2\} \geq I(\theta)^{-1}. \tag{2.74}$$

□

Proof cf. [SG04, pp. 138-142] ■

Intuitively, the Cramér-Rao lower bound states that the steeper the likelihood function around the true value, the lower the variance that an optimal estimator can attain.
One of the main reasons for the popularity of the ML estimator is that it can be shown that the MLE is asymptotically optimal, reaching the Cramér-Rao lower bound:

Theorem 10 (Asymptotic properties of the MLE) *Let $\boldsymbol{x} = [x_1, ..., x_N]$ be a sample drawn from the distribution $p(\boldsymbol{x};\theta)$, and let $p(\boldsymbol{x};\theta)$ satisfy some regularity conditions (given in [SG04, pp.147-148]), then the MLE $\hat{\theta}_{MLE}$ converges in distribution for large N to*

$$\hat{\theta}_{MLE} \overset{a}{\sim} \mathcal{N}(\theta, I(\theta)^{-1}), \tag{2.75}$$

where $\mathcal{N}(\mu,\sigma^2)$ denotes the normal distribution with mean μ and variance σ, $I(\theta)$ denotes the Fisher information evaluated at the true value θ, and $\overset{a}{\sim}$ means "asymptotically distributed according to". The MLE is hence asymptotically unbiased and optimal (attains the CRLB). □

PROOF cf. [SG04, pp.148-151] [Kay93, pp.211-213] ∎

This result forms the basis for claiming optimality of the MLE. However, in practice it is often unknown how large N has to grow until the MLE reaches optimality. An analytical expression for the pdf of the MLE is usually impossible to derive.
Finding the MLE involves the maximization of the function $l_{\boldsymbol{x}}(\theta)$. Analytical expressions for $l_{\boldsymbol{x}}(\theta)$ can only be found in very simple cases. Hence, it is often necessary to use tools from numerical optimization theory. Local maxima can be found by iterative methods such as steepest descent or Newton-Raphson type algorithms that are guaranteed to converge to the global maximum if the likelihood function is convex. If the likelihood function is more complicated, one often has to use global optimization methods such as evolutionary algorithms or simulated annealing [SH97]. These algorithms are used in Chapter 5 and 6 to solve maximum likelihood problems. However, a detailed discussion of these methods is beyond the scope of this thesis.

A special problem arises if some of the data is unobserved (such as in an HMP scenario where the observer has only access to the outputs but can not observe the states). This issue will be addressed next by means of the expectation maximization algorithm.

2.6 Expectation maximization algorithm

We shall apply the expectation maximization (EM) algorithm in Chapter 6 to estimate the parameters of a convolutional code from a received data stream (or DNA sequence) by using a probabilistic tap model.

Let a system with input $\boldsymbol{x} \in \mathcal{X}^n$ and output $\boldsymbol{y} \in \mathcal{Y}^n$ be specified by the distribution $p(\boldsymbol{x}, \boldsymbol{y}; \theta)$ with (unknown) parameter $\theta \in \Theta$. In practice, we often face the situation that the system can only be partially observed, i.e., we are only given access to the output of the system $\boldsymbol{y}$. Estimating the system state θ from $\boldsymbol{y}$ via MLE therefore requires the maximization of the expression

$$\hat{\theta} = \arg\max_{\theta\in\Theta} \left\{ \sum_{\boldsymbol{x}\in\mathcal{X}^n} p(\boldsymbol{x}, \boldsymbol{y}; \theta) \right\}, \tag{2.76}$$

which is often difficult to solve. The expectation-maximization (EM) algorithm developed by Dempster et al. [DLR77] iteratively approaches the objective in Eq. (2.76) by alternating between a maximization and evaluation step. The algorithm is numerically stable and guaranteed to converge to a local maximum of the likelihood function even though convergence can be slow [Moo96]. In many situations, it greatly reduces the computational complexity. The remainder of this section outlines the basic steps in the derivation, and the algorithm is stated at the end.

In the EM framework, $\boldsymbol{x}$ is often referred to as *missing data* and $\boldsymbol{y}$ as the *observed data*. Recall that $l_{\boldsymbol{x}}(\theta)$ denotes the likelihood function of θ given the data $\boldsymbol{x}$ and $ll_{\boldsymbol{x}}(\theta) = \log(l_{\boldsymbol{x}}(\theta))$. We start with the following identity

$$ll_{\boldsymbol{y}}(\theta) = \log(p(\boldsymbol{y}; \theta)) = \log(p(\boldsymbol{x}, \boldsymbol{y}; \theta)) - \log(p(\boldsymbol{x}|\boldsymbol{y}; \theta)). \tag{2.77}$$

Suppose we are given the observed data $\boldsymbol{y}$ and assume that we have a guess $\theta^{[k]}$ about our parameter, then as the left hand side of Eq. (2.77) does not depend on $\boldsymbol{x}$, taking the expectation with respect to $p(\boldsymbol{x}|\boldsymbol{y};\theta^{[k]})$ on the right and left side of Eq. (2.77) yields

$$ll_{\boldsymbol{y}}(\theta) = \sum_{\boldsymbol{x}} p(\boldsymbol{x}|\boldsymbol{y};\theta^{[k]}) \log(p(\boldsymbol{y},\boldsymbol{x};\theta)) - \sum_{\boldsymbol{x}} p(\boldsymbol{x}|\boldsymbol{y};\theta^{[k]}) \log(p(\boldsymbol{x}|\boldsymbol{y};\theta)). \tag{2.78}$$

It is important to emphasize that $\theta^{[k]}$ is a *fixed* and *known* value whereas θ is an unknown variable parameter. Define the function

$$Q(\theta,\theta^{[k]}) \doteq \sum_{\boldsymbol{x}} p(\boldsymbol{x}|\boldsymbol{y};\theta^{[k]}) \log(p(\boldsymbol{y},\boldsymbol{x};\theta)), \tag{2.79}$$

which we identify as the first term in Eq. (2.78) and which we refer to as the *Q-function*. The following statement is the key to the EM algorithm

Theorem 11 *Suppose* $\boldsymbol{x} \in \mathcal{X}^m$, $\boldsymbol{y} \in \mathcal{Y}^n$, $(\boldsymbol{x},\boldsymbol{y}) \sim p(\boldsymbol{x},\boldsymbol{y};\theta)$ *with parameter* $\theta \in \Theta$. *Given the observed data* $\boldsymbol{y}$ *and the Q-function as defined in Eq.* (2.79),

$$Q(\theta^{[k^*]},\theta^{[k]}) \geq Q(\theta^{[k]},\theta^{[k]})$$

implies

$$ll_{\boldsymbol{y}}(\theta^{[k^*]}) \geq ll_{\boldsymbol{y}}(\theta^{[k]}),$$

for any $\theta^{[k]},\theta^{[k^*]} \in \Theta$. □

Proof

$$\begin{aligned}
\delta_{ll} &= ll_{\boldsymbol{y}}(\theta) - ll_{\boldsymbol{y}}(\theta^{[k]}) && (2.80a)\\
&\overset{(2.78)}{=} \sum_{\boldsymbol{x}} p(\boldsymbol{x}|\boldsymbol{y};\theta^{[k]}) \left(\log(p(\boldsymbol{y},\boldsymbol{x};\theta)) - \log(p(\boldsymbol{y},\boldsymbol{x};\theta^{[k]})) + \log\left(\frac{p(\boldsymbol{x}|\boldsymbol{y};\theta^{[k]})}{p(\boldsymbol{x}|\boldsymbol{y};\theta)}\right)\right)\\
&\overset{(2.9)}{=} Q(\theta,\theta^{[k]}) - Q(\theta^{[k]},\theta^{[k]}) + \mathcal{D}(p(\boldsymbol{x}|\boldsymbol{y};\theta^{[k]})||p(\boldsymbol{x}|\boldsymbol{y};\theta)) && (2.80b)\\
&\geq Q(\theta,\theta^{[k]}) - Q(\theta^{[k]},\theta^{[k]}). && (2.80c)
\end{aligned}$$

As shown in Section 2.2.1, the Kullback-Leibler distance $\mathcal{D}(\cdot||\cdot)$ is strictly positive. This property is used in Eq. (2.80c) to show that any $\theta^{[k^*]}$ that increases $Q(\theta,\theta^{[k]})$ with respect to $Q(\theta^{[k]},\theta^{[k]})$ increases the log likelihood $ll_{\boldsymbol{y}}(\theta^{[k^*]})$ with respect to $ll_{\boldsymbol{y}}(\theta^{[k]})$. ■

The procedure is given in Algorithm 2.1. In many applications the maximization of the Q-function is much easier than maximizing the objective in Eq. (2.76).

Example 2.6.1 *Consider an HMP specified as in Section 2.4.5. In the EM framework,* $\boldsymbol{y}^N = [y_0, y_1, ..., y_N]$ *represents the observed data, and* $\boldsymbol{s}^N = [s_0, s_1, .., s_N]$ *is the unobservable state sequence, and the distribution* $p(\boldsymbol{y}^N, \boldsymbol{s}^N;\theta)$ *is specified by the parameters* $\theta = \{\boldsymbol{\lambda}, \boldsymbol{P}, p(y_n|s_n)\}$. *The Q-function is given by*

$$Q(\theta;\theta^{[k]}) = \sum_{\boldsymbol{s}^N} p(\boldsymbol{s}^N|\boldsymbol{y}^N;\theta^{[k]}) \log(p(\boldsymbol{s}^N,\boldsymbol{y}^N;\theta)). \tag{2.81}$$

Algorithm 2.1: Expectation-maximization algorithm

Require: Observed data $\boldsymbol{y}$, missing data $\boldsymbol{x}$,

Initialization: guess an initial parameter $\theta^{[0]} \in \Theta$, set a threshold t_h, $k = 0$, $\delta_{ll} = \infty$

while $\delta_{ll} > t_h$ **do**

(E-step): Expectation

$$Q(\theta, \theta^{[k]}) = \sum_{\boldsymbol{x}} p(\boldsymbol{x}|\boldsymbol{y}; \theta^{[k]}) \log(p(\boldsymbol{y}, \boldsymbol{x}; \theta)).$$

(M-step): Maximization

$$\theta^{[k+1]} = \arg\max_{\theta \in \Theta} \{Q(\theta, \theta^{[k]})\}$$

compute

$$\delta_{ll} = ll_{\boldsymbol{y}}(\theta^{[k+1]}) - ll_{\boldsymbol{y}}(\theta^{[k]})$$

$k \leftarrow k + 1.$

end while

output $\theta^{[k]}$.

By Eq. (2.55d)*,and using the convention* $p(y_0, s_0|s_{-1}) = p(y_0, s_0)$*, we find:*

$$\log(p(\boldsymbol{s}^N, \boldsymbol{y}^N)) = \sum_{n=0}^{N} \log(p(y_n, s_n|s_{n-1})). \tag{2.82}$$

Hence,

$$Q(\theta; \theta^{[k]}) = \sum_{s_0,..,s_n,..,s_N} p(s_0, .., s_n, ..s_N|\boldsymbol{y}^N; \theta^{[k]}) \sum_{n=0}^{N} \log(p(y_n, s_n|s_{n-1})) \tag{2.83a}$$

$$= \sum_{n=0}^{N} p(s_n|\boldsymbol{y}^N; \theta^{[k]}) \log(p(y_n, s_n|s_{n-1})). \tag{2.83b}$$

The Q-function can then efficiently be evaluated using the forward-backward recursions (2.63) *for the a posteriori state density* $p(s_n|\boldsymbol{y}^N; \theta^{[k]})$ *and* (2.56b) *for* $\log(p(y_n, s_n|s_{n-1}; \theta))$*.*

The MLE can even be analytically derived by setting the derivative of the Q-function with respect to θ *to zero. This results in the famous Baum-Welch algorithm for estimating the parameters of an HMP. For more details the reader is referred to [EM02, DEKM98].* □

2.7 Summary

This chapter gave a brief reference to the mathematical methods and models used in this work. We started with an overview of information theory, showing how information, entropy, divergence, and mutual information of random variables are related. The notion of transmission channels was defined, and the channel coding theorem was introduced, stating that lossless transmission of information over noisy channels is theoretically possible as long as a coding scheme operating at rates below the channel capacity is used.

Coding theory, introduced next, aims at developing practical methods that achieve the channel capacity. Over the past decades, powerful channel coding schemes were designed by engineers to protect digital information in the presence of noise. We described convolutional codes as a practical example, that exhibit good error correcting properties at a reasonable encoding and decoding complexity.

Markov processes are widely used as mathematical models in statistics. We shortly reviewed discrete time Markov chains and then focused on continuous time chains that are used in biology as probabilistic models for DNA sequence analysis. An overview about Hidden Markov models, an extension of Markov models, concluded this section. Hidden Markov models can be viewed as a Markov chain observed through a discrete memoryless channel.

Maximum likelihood methods are used throughout this thesis, and we introduced the basic principle of maximum likelihood and its asymptotic properties. Finally, we described the expectation maximization (EM) algorithm, an iterative procedure that is guaranteed to approach the maximum likelihood, given certain conditions are satisfied. The EM algorithm is particularly useful in situations when data is only partially observed. As an example for the latter, the application of EM to hidden Markov models was briefly discussed as an example.

3

Communication principles of the living cells

This chapter is an introduction to molecular biology from the perspective of a communications engineer, non-extensively covering the storage and replication of information at the sequence level, the cellular repair mechanisms, the concept of genes and the genetic code, and their organization in gene regulatory transcription factor networks.

Section 3.1 shows that the genomic material storing the genetic information (DNA and RNA) can be regarded as a digital, quarternary signal. During the life cycle of a cell, the genetic information is accessed to signal and perform the production of proteins and trigger metabolic reactions.

Like digital signals in a communications scenario, genetic information must be faithfully replicated, decoded and repaired. This analogy is addressed in Section 3.2, where we describe how this is achieved by enzymes, executing their task with astonishing precision and reliability. We characterize transmission errors occurring during these processes and show models that explain their occurrence.

Moving from the purely bio-physical to a functional layer of genome organization, Section 3.3 introduces the term *gene* and shows how genes are processed in the cell. Protein coding genes are decoded by means of the *genetic code*, introduced in the same section, that defines the mapping from DNA sequence to proteins.

Finally, Section 3.4 describes the next higher level of cellular information processing: the organization of genes in regulatory interaction networks.

3.1 The physical layer - DNA and RNA

3.1.1 Genomic material as digital signals

The information needed to construct a living organism is mainly provided by its *genome*, a long sequence of *nitrogenous bases* that are chained together by a *sugar phosphate* backbone. A nucleotide refers to the compound of the three chemical entities base, sugar and phosphate, and we often speak of a *sequence of nucleotides* when referring to genetic signals. The genome does not itself perform any active role in constructing the organism but its main role is the *storage* of information which is used to regulate and perform the production of proteins and metabolic reactions triggered by a complex series of interactions [Lew04].

Since all nucleotides have the same type of sugar and phosphate, the information is entirely contained in the sequence composition of nitrogenous bases. A nucleotide sequence can contain four different types of bases: *adenine* (A), *cytosine* (C), *guanine* (G) and *thymine* (T). Hence, a sequence of symbols from the quarternary alphabet $\{A, C, G, T\}$ is the minimal sufficient representation for a nucleotide sequence representing genetic information. We shall denote the set of possible bases (or nucleotides) as

$$\mathcal{A} = \{A, C, G, T\} \equiv \{1, 2, 3, 4\}, \tag{3.1}$$

and allways assume an alphabetical ordering on the elements.
The bases can be divided into two chemical subgroups called *pyrimidine* and *purine*:

$$\begin{aligned} \text{purines } &= \{A, G\} \\ \text{pyrimidines } &= \{C, T\}. \end{aligned}$$

The *deoxyribonucleic acid* (DNA) is formed by two nucleotide sequences that are joined by *hydrogen bonding* between the nitrogenous bases. Hydrogen bonding can only occur between C and G, while A can bond specifically only with T and vice versa. These reactions are described as *base pairing*. The two strands are twisted in the shape of a double helix.

Example 3.1.1 *The following two sequences form a valid DNA sequence:*

$$\begin{array}{l} TAAAGCGTGGGTATTCTT \\ ATTTCGCACCCATAAGAA \end{array}$$

□

From an information theoretic perspective, it is sufficient to reduce the representation to a single nucleotide sequence over the alphabet $\mathcal{A}$ because any of the two strands of DNA is completely determined by the other. Most genomes are DNA (some viruses have genomes that consist of RNA, described below). The spatial dimension of the DNA sequence differs among organisms: the human genome has 3×10^9 bases, whereas the genome of the bacterium *Escherichia coli* (*E. coli*) comprises around 4.65×10^6 bases. The length of the human genome is approximately 1.8 meters but must be kept in the *nucleus* of the cell having a diameter of only 6μm. It is evident that the DNA must be packed into the compartment that contains it. The DNA is therefore coiled around a certain class of proteins that are again packed into a dense structure which is further coiled into a higher structure. Finally, the *chromosomes* are at the end of this structural organization. The structural organization of DNA is very dynamic and changes during the cell cycle, for example, the structure may have to be locally unwound to access the sequence information. In fact, the individual chromosomes become only visible within a certain time period of the cell cycle [Kni06].

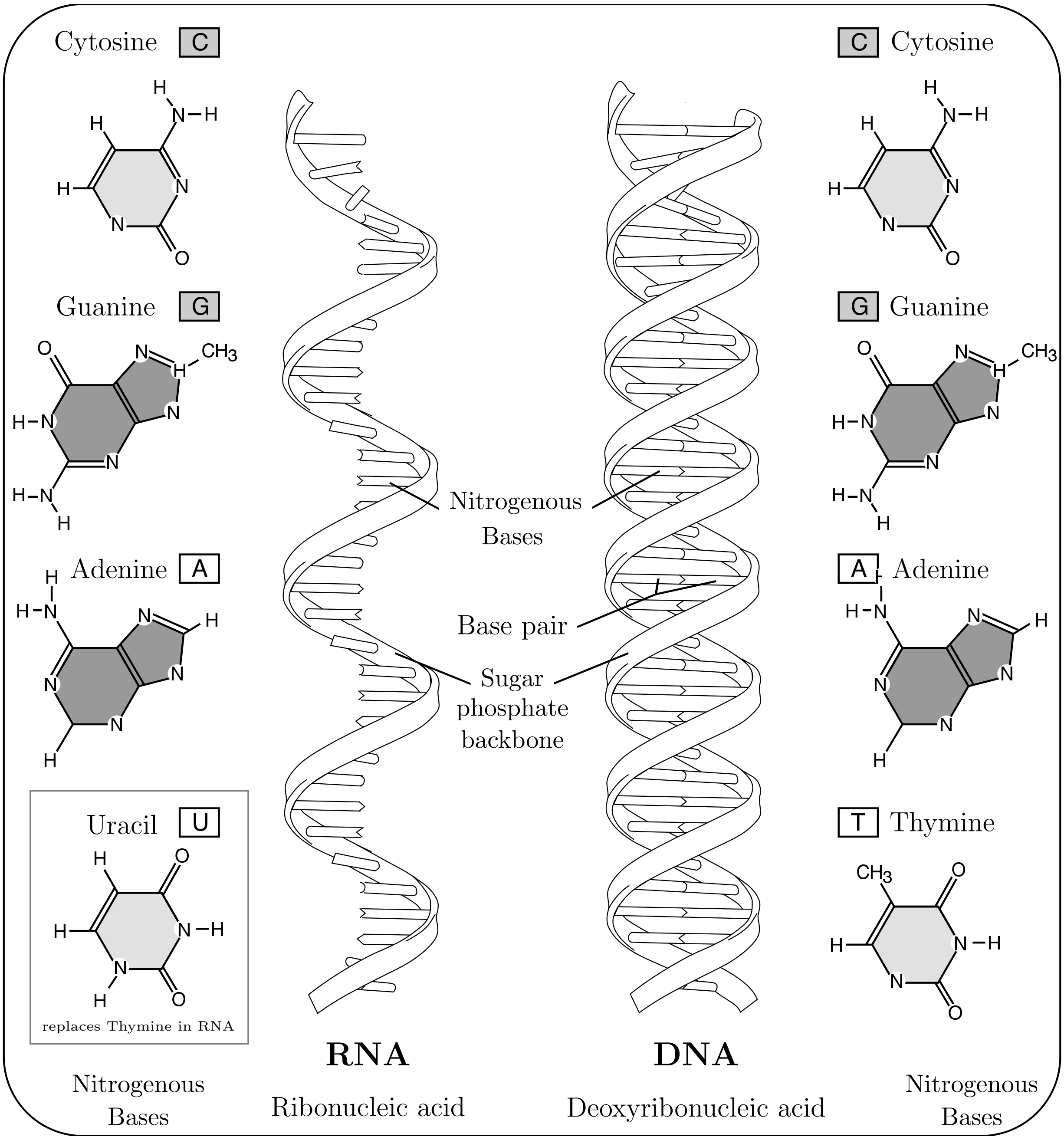


Figure 3.1: Structure of RNA (left) and DNA (right) molecules. A nucleotide sequence consists of *nitrogenous bases* that are chained together by a *sugar-phosphate* backbone. DNA is formed by two nucleotide sequences that are joined by *hydrogen bonding* between the nitrogenous bases. RNA is single stranded, formed by a different kind of sugar compared to DNA, and in RNA the base uracil (U) is found instead of thymine. (Figure modified from: National human genome research institute, http://genome.gov/10012096, September 2008).

3.1.2 From DNA to RNA

Another kind of nucleotide sequence is *ribonucleic acid* (RNA). The main difference is that it is formed by a different kind of sugar compared to DNA, and that in RNA the base uracil (U) is found instead of thymine. Cellular genomes are DNA, but viruses can have RNA genomes. The structure of DNA and RNA is shown in Figure 3.1. In cells, RNA is synthesized from DNA in a process called *transcription*. Transcription starts when the enzyme *RNA polymerase* binds to the DNA at a specific binding sequence called *promoter*. From this point, the polymerase moves along the DNA, synthesizing RNA, until it reaches a stop signal. The resulting RNA is identical to one of the DNA strands from which it has been synthesized (with T replaced by U) and complementary to the other strand which is called the *template strand*.

Unlike DNA, RNA is generally produced as a single stranded molecule with a strong affinity to *fold* via intramolecular base pairing. The reason for this behavior is that the RNA molecule seeks to attain a state of lower energy. The folded, two dimensional structure is called the *secondary structure* of the RNA. The mere symbolic sequence is therefore sometimes referred to as the *primary structure*. An example is shown in Figure 3.2. RNAs can form complex structures and are involved in many regulatory processes in the cell. Recent studies suggest that the majority of mammalian genomes is transcribed into RNAs [The07], most of whose functions are unknown [Mat07, MM06]. It is believed that the function of most RNAs is determined by their secondary structures. Evidence supporting this hypothesis is given by observations that secondary structure of RNAs is often highly conserved across different species while the primary structure may differ (different primary structures can give rise to the same secondary structure). On the other hand, conservation of the primary structure does not necessarily imply conservation of function, since single mutations can lead to different secondary structures [BMG$^+$04]. The secondary structure of a long functional RNA known to be involved in the transport of proteins, a so-called *signal recognition particle*, is shown in Figure 3.3.

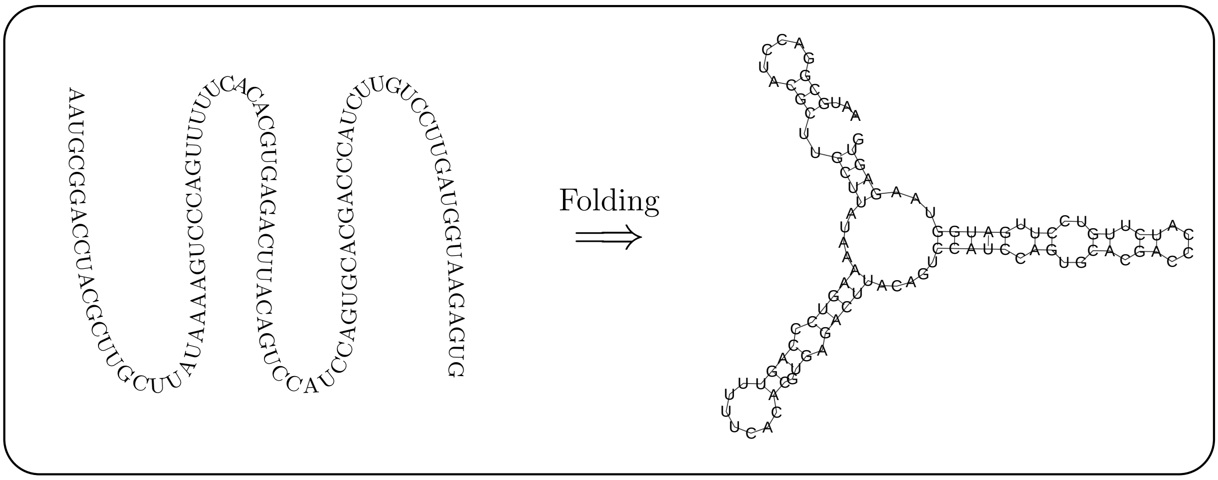


Figure 3.2: A single stranded RNA molecule (sequence shown on the left) folds to a secondary structure by intramolecule base pairing (right). Lines between letters indicate pairing of bases. Structure of a random RNA sequence was predicted using the Vienna RNA package [GLB$^+$08].

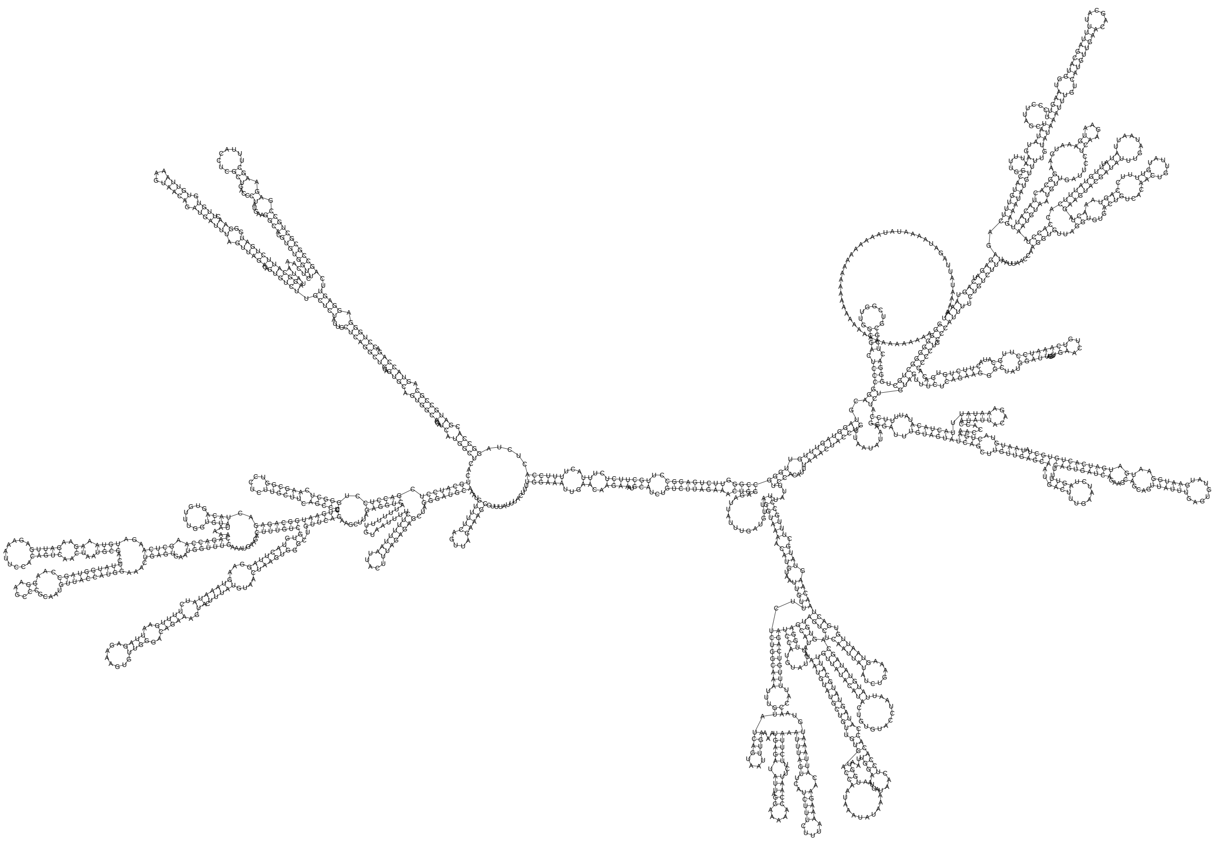


Figure 3.3: Secondary structure of the signal recognition particle (Symbol: SRP9, Entrez identifier: 6726). The secondary structure shown was predicted from the sequence obtained from GenBank [BKML+08] using the Vienna RNA package [GLB+08].

3.2 Transmission and maintenance of DNA

3.2.1 DNA replication

DNA is the storage medium for genetic information, and a complete copy of DNA is kept in each cell of an organism. When a cell divides, it first replicates its DNA and passes a copy on to its descendants. Replication starts by unzipping the two DNA strands. Each separated strand then serves as a template for the synthesis of a new complementary partner strand that is identical to its former partner. An enzyme called *DNA polymerase* is responsible for the synthesis of the new strand, moving along the template strand assembling nucleotides in the order that complements the template strand. When the replication process is complete, two identical DNA molecules have been produced. DNA replication is *semiconservative* because the new molecules consist of an original strand conserved from the "old" DNA and a complementary new strand synthesized by the polymerase. Figure 3.4 depicts the process of DNA replication.

Replication of DNA is a noisy process, therefore, the cell is equipped with several error correcting mechanisms to reduce the numbers of errors to a tolerable level. The fidelity of DNA synthesis relies on the specifity of base pairing: although A-T and C-G are the most stable base pairs, less stable pairings such as G-T or C-A can be formed. When the

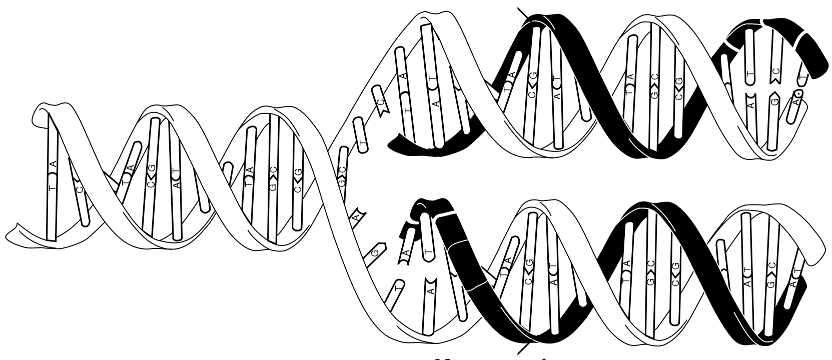


Figure 3.4: DNA replication is *semi-conservative*. (Figure modified from: National human genome research institute, http://genome.gov/100012096, September 2008)

polymerase synthesizes the new strand, it can make errors that result in a "mismatch" pairing of nucleotides. The error rate of the synthesis mechanism of DNA polymerase is approximately 10^{-3} per base pair replicated which is not tolerable for producing viable new cells. Therefore, a *proofreading* mechanism is implemented in the polymerase itself: before adding a new nucleotide to the growing DNA chain, the polymerase enzyme is able to check whether the previous one has been correctly base paired. If not, the polymerase removes the nucleotide and repeats the synthesis. Proofreading decreases the error rate in replication to $10^{-5} - 10^{-7}$ per base pair.

As we shall see, additional protein machinery is available in the cell that continuously monitors DNA for mistakes and initiates error control, if necessary. These processes then correct remaining errors made during DNA replication bringing the overall error rate to $< 10^{-9}$ per base pair replicated [Kni06, Lew04].

3.2.2 Error control mechanisms in the cell

Even though the DNA polymerase proceeds with astonishing accuracy, it may also introduce errors into the new DNAs with a low probability. In addition, the DNA is an unstable macromolecule which is constantly suffering from damage caused by intracellular (e.g., thermal collision with other molecules, chemically reactive byproducts) and environmental (e.g., sunlight, radiation) influences. We refer to these effects that cause DNA damage as the *genetic noise*. Such *spontaneous* changes of the DNA result in alterations of a base that do not properly pair with the partner base. For this reason, the cell is equipped with a set of efficient repair mechanisms, and evolutionary studies suggest that these mechanisms evolved at the early stages of evolution. This suggests that the genetic information needed for the construction of higher organisms cannot tolerate the level of genetic noise present in cells. Support of this hypothesis is given by the fact that it was found that the cause of cancer in human is often related to mutations in these repair systems [Lew04].

Repair systems can identify a range of different DNA distortions, and each cell controls a whole set of systems able to deal with DNA errors. Most damage creates deformation in the structural continuity of a DNA strand, which is then detected by *mismatch repair* mechanisms. *Excision repair* is triggered by a damaged base or a change in the spatial path of DNA: *base excision repair* replaces a single damaged base by the correct one; *nucleotide excision repair* removes a whole sequence of nucleotides and resynthesizes a new stretch of DNA to replace the excised material.

When DNA is damaged such that one base is degenerated to an "invalid" chemical structure (not A, C, G, T for DNA) recognition of the "correct" strand is easy. But how is the repair system able to identify the correct strand in case of a mismatch where the mispairing involves two valid A, C, G, T bases caused, for example, by the replication process? As with many processes in molecular biology, the mechanism underlying this phenomenom is not yet fully understood [Kni06]. One hypothesis is that chemical modifications of the old strand (*DNA methylation*) can be recognized (this can be thought of as a "chemical time-stamp").

In addition to small scale damage affecting only a single base or a stretch of nucleotides, the cell is able to deal with large scale distortions that affect whole chromosomes. An example are *double strand breaks* of the DNA molecule caused by environmental influence (such as radiation) potentially leading to rearrangement of chromosomes if unrepaired. These can result in a chromosome part being deleted, translocated to another position or chromosome, or duplicated. Further, combinations of these events are possible [Kni06, Lew04]. The majority of certain types of cancers show high rates of large-scale alterations in chromosomes [Kni06,Lew04]. As a last resort, if the repair mechanisms fail, the cell can therefore trigger *programmed cell death* (so-called *apoptosis*) causing the cell to commit suicide if it recognizes that it has been damaged beyond an acceptable level. Apoptosis is a key defense against cancer and crucial to the development of multicellular organisms providing control over the total cell number.

3.2.3 Characterization of genetic errors

There are three types of small scale errors (affecting only a single or a few bases) that can occur in transmission of genetic information: substitutions, insertions and deletions. While a mutation is damage to the DNA that can still be corrected by the cell, these errors are fixed in the DNA. We shall briefly describe the processes that explain genetic errors: when a mismatch is introduced in the DNA, the repair mechanisms must correct the error in a limited amount of time, before the cell divides. If they fail to do so, fixation of the mutation will occur upon replication of the DNA, as shown in Figure 3.5. We call the replacement of a base in the nucleotide sequence by another one a *substitution*.

Insertion errors introduce additional base pairs into the DNA molecule that were not present previously. Deletion errors have exactly the opposite effect, erasing one or multiple base pairs from the DNA. These types of errors are also often referred to as *InDels* (**In**sertions and **Del**etions). Such errors most likely occur because of the affinity of single stranded DNA to fold and form loops. A simple scheme explaining such errors was

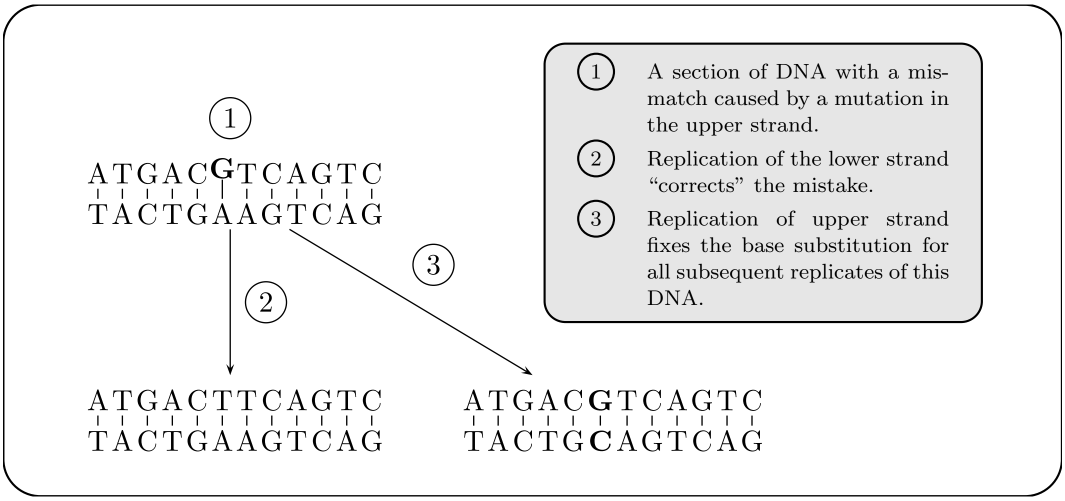


Figure 3.5: A substitution occurs if a DNA with mismatch is replicated. A mismatch can occur because of a mistake of the replication machinery or a false synthesis in excision repair.

developed by Streisinger in 1966 [Kni06] and is depicted in Figure 3.6.

So far, we characterized how information is represented and processed on the biochemical level. We introduced DNA and RNA as carriers of genetic information that is contained in sequences of nucleotides which form a finite alphabet $\mathcal{A}$. Yet, we have not said anything about the encoding of the information and how it gives rise to function. We shall now see how genetic information is organized and structured on a higher, functional level, which leads us to the description of a fundamental organizational genomic unit: the *gene*.

3.3 Information packets - the genes

On the biochemical layer, the genome consists of a number of different nucleic acid molecules that are copied, monitored, and mantained by the cell as described in the previous sections. The nucleotides do not proactively carry out any function but encode information that is needed to construct functional molecular entities. These entities form a *functional layer* that is organized into different genes. Genes are confined regions in the DNA, encoding information which is further processed by molecular complexes. The information stored in a gene is used to assemble the molecules that carry out diverse functions within the cell. This section gives a brief overview about this functional layer.

3.3.1 What is a gene?

While the term *gene* is frequently used in scientific and non-scientific literature, it is not clearly defined, and since it was first proposed, the concept of what a gene is has been frequently adapted according to the rapidly changing state of current knowledge [GBR$^+$07]. In this work, we define a gene as a *genomic sequence encoding a protein or RNA product* (a simplified version of the definition given in [GBR$^+$07]). We hence distinguish two classes

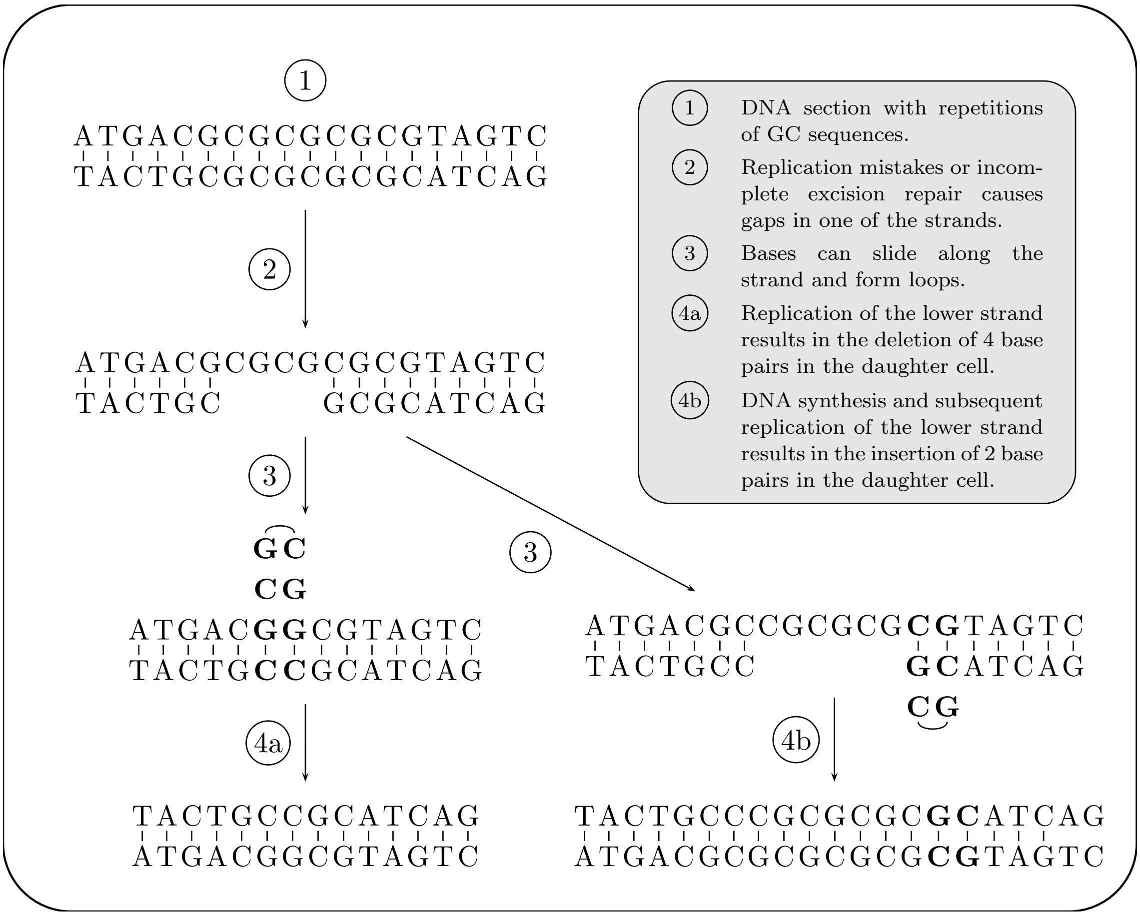


Figure 3.6: A model that explains insertion and deletion mutations. Such errors most likely occur because of the affinity of single stranded DNA to fold and form loops.

of genes: *protein-coding* and *non-(protein-)coding* genes, the latter producing RNAs that directly execute functions, and the former producing intermediate RNAs that are further processed to make proteins. The definition implies that all genes are *transcribed* into single stranded RNAs, a process that was described in Section 3.1, summarized again in Figure 3.7. The roles of non-coding genes are diverse, including gene regulation, RNA processing, and protein synthesis; most of the non-coding RNAs identified in genomic studies have yet to be ascribed any function. However, there is strong evidence that these RNAs are biologically important.

3.3.2 Protein coding genes and the genetic code

Protein coding genes produce a special RNA called *messenger RNA* (mRNA) that is further processed to make a protein. Proteins are macro molecules that are formed by chains of *amino acids*. The information needed to produce the amino acid sequence is contained within a single gene. Proteins are crucial to most cellular processes carrying out basic and diverse cell functions such as material transport, catalysis of chemical reactions, signal detection as well as maintaining the cell scaffold. A cell permanently produces proteins from DNA by "decoding" the genes through *gene expression* that occurs by a

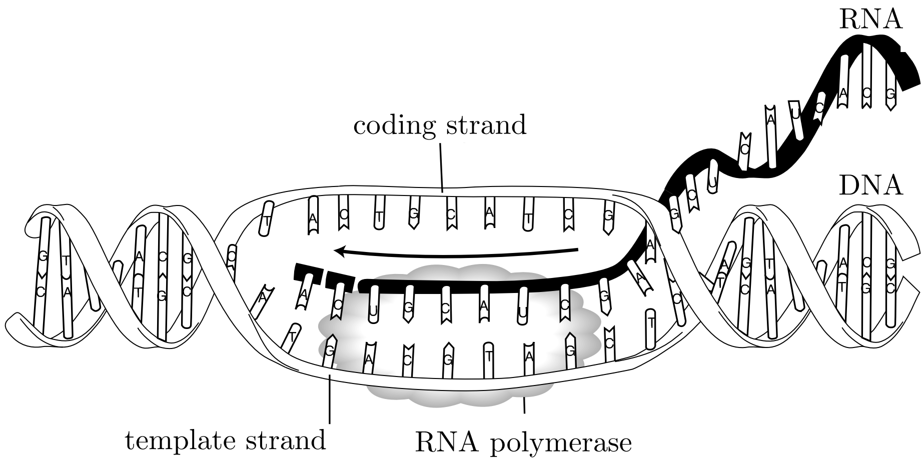


Figure 3.7: Transcription of DNA to RNA is executed by *RNA polymerase* moving along DNA and synthesizing RNA from the template strand. The RNA is identical to the coding strand (with T replaced by U) and complementary to the template strand (modified from: National human genome research institute, http://genome.gov/100012096, September 2008).

two-stage process: the transcription of the gene to mRNA and subsequent synthesis of amino acids, called *translation*.

In higher organisms (*eucaryotes*), genes are divided into *exons* and *introns*. The information for the production of a protein is only contained within the *exons* which are therefore often called *coding regions* whereas all other parts of the genome are referred to as *non-coding regions*. Introns are transcribed but removed from the mRNA before translation in a process called *splicing*.

Exons have a well specified structure being organized in triplets of nucleotides, so-called *codons*. During the translation step, a codon is mapped to an amino acid according to the *genetic code*. The resulting chains forming the proteins consist of 20 different amino acids but there are $4^3 = 64$ different codons. Therefore, the mapping is not unique, and some codons code for the same amino acid. Additionally, there are start and stop codons that signalize the beginning and the end of protein synthesis. This is because only a subsequence of the mRNA is translated into a protein, which is delimited by the start codon AUG and one of the stop codons UAA, UAG, or UGA. The genetic code is shown in Figure 3.9. The process of gene expression is summarized in Figure 3.8. Once a sequence of amino acids has been synthesized, it folds, similar to RNA (cf. Section 3.1.2), to a three dimensional structure called the *tertiary structure* of the protein. The structure of the protein determines its function. It is, however, an open problem to predict this structure computationally in an efficient way and the characterization of all existing protein structures in the human genomes is far from being complete.

Upon completion of gene expression, the gene product interacts with other molecules. Some proteins can interact with DNA and thus influence the production rate of other genes. This yields a higher level of information processing that forms a regulation network of interacting entities, as described in the next section.

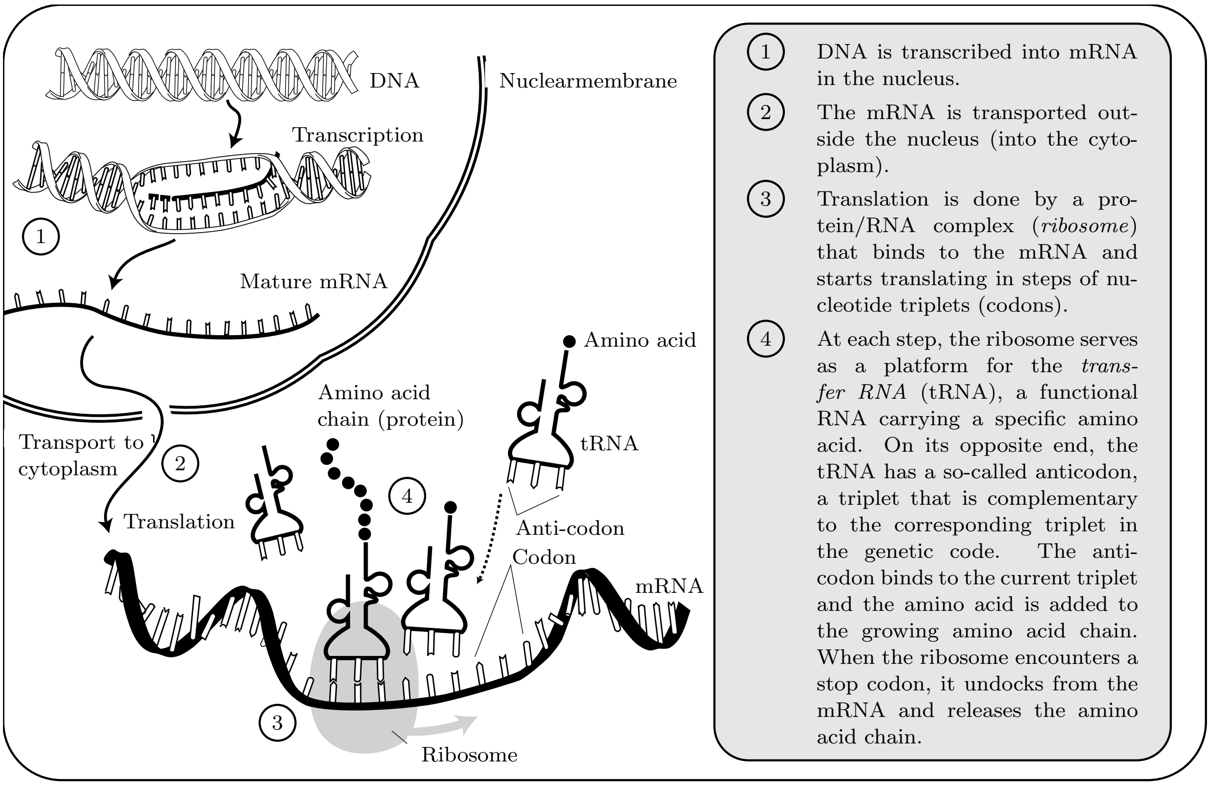


Figure 3.8: Gene expression is a two-stage process: transcription of a gene first produces mRNA. A protein/RNA complex subsequently translates mRNA into a protein following the genetic code (modified from: National human genome research institute, http://genome.gov/100012096, September 2008).

3.4 Network layer: gene interaction

We saw that genetic information processing is executed in a sequential manner: genes are transcribed into mRNAs which, in turn, are subsequently translated into proteins. Upon modification, proteins are used either as building blocks of the cellular physical structure or as functional units. A gene is said to be *expressed* if its mRNA or protein can be found in the cell cytoplasm, and the activity of a gene is measured through the amount of its corresponding product. Each cell holds a whole copy of the genome, and many genes are, in fact, expressed in all cell types. On the other hand, each different cell type also produces specialized proteins. Different expression patterns are to a great extend responsible for the cell's distinctive properties.

Genes, however, do not act independently from each other - they are organized in large control networks. Cells live in a complex environment and must be able to react to external signals (e.g., molecules from other cells, temperature, harmful chemicals), and information about their internal state (e.g., DNA damage). Cells respond to these signals by producing appropriate proteins that control the expression patterns of one or multiple other genes. The cell uses special proteins called *transcription factors* (TF) that are modulated by a specific environmental signal input, causing the particular TF to transit between an inactive and active molecular state. An active TF can bind to a specific

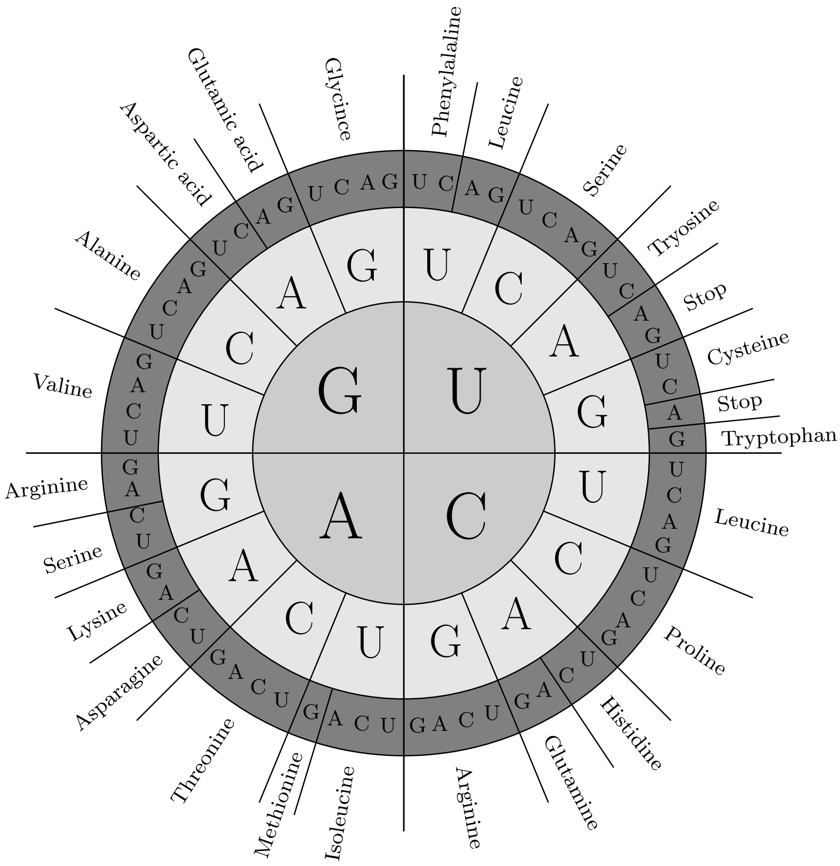


Figure 3.9: The genetic code defines the mapping between nucleotide triplets and amino acids. There are three stop codons (UGA,UAA,UAG) that serve as stop signals for translation and are not coding for an amino acid. The codon AUG represents the start codon, coding for methionine at the same time. The codons are given as RNA code, the DNA code as it appears on the coding strand of genes is obtained by exchanging U (Uracil) with T (Thymine).

regulatory sequence (*transcription factor binding site*) in proximity of its target gene(s) to regulate the transcription rate. Transcription factors can act as *activators* that increase the transcription rate of a gene, or as *repressors* that reduce or even completely inhibit transcription. A TF can regulate multiple genes, and a gene is usually regulated by one or multiple TFs. Since TFs are proteins themselves, they are also encoded by genes, which are regulated by other TFs, which in turn may be regulated by yet other TFs, and so on. These interactions allow the formation of complex regulatory networks (Figure 3.10). Nodes in the network represent genes, whereas edges denote regulatory connections among the nodes. The output of a node in the network is controlled by input received from connected nodes.

The network thus represents a dynamical system: an environmental signal changes the state of transcription factor activities, leading to changes in the production rate of proteins. Some of those proteins are TFs themselves that further control activity of other genes, and so on. This continues until gene expressions have reached a steady (or oscillating) state. Depending on the kind of signals the network receives, it will end up in different states. In addition to transcription networks, the cell contains several other

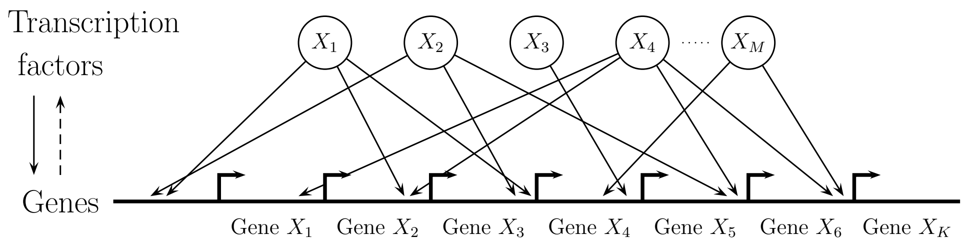


Figure 3.10: Graphical representation of a transcriptional regulation network. The control of gene activity is much more complex than this figure suggests. Figure reproduced from [Alo07, Chapter 2]

networks of interactions, such as those based on protein-protein interactions.

A remarkable feature of transcription networks is modularity [Alo03]. One can take a small gene network of one organism, e.g., the DNA coding for green fluorescent protein from jellyfish, and place it in a different organism, e.g., a bacterium [Alo07], causing the bacterium to turn green. Such organization makes transcription networks adaptable during evolution and able to rapidly incorporate new genes or reconnect existing modules [Alo07]. In fact, there is evidence that gene networks evolve much faster than coding regions of genes. Closely related animals exhibit conserved DNA coding regions yet their transcription regulation is often quite different, indicating that many of the differences between animal species lie in the different activation patterns of genes rather than in difference on the sequence level [Alo07].

Gene networks are *sparse*, and there is experimental evidence that gene networks are *scale free*, i.e., there are very few nodes with many connections and vice versa. A scale-free topology has, on average, more nodes with very many connections (called *hubs*) and more nodes with very few connections than appearing in random networks. It is a well known result from graph theory that the average minimum paths between nodes in scale-free networks are shorter than in random networks. Furthermore, the minimum path length is typically robust to disruptions of random nodes. However, if a hub is compromised, the distance tends to increase dramatically [SPB07]. Surprisingly, it has been observed that there appears to be at best only a weak relationship between connectivities in the yeast gene expression network and relative growth rates of yeast deletion mutants that had highly connected genes removed. In other words: other than expected the network seems to be robust with respect to the disruption of hubs [SPB07]. How this robustness is achieved is yet unknown (some simulation studies and theories are provided by [SPB07]).

Understanding gene networks is of focal importance in order to understand cellular information processing. Recently, a whole discipline called *systems biology* has emerged that approaches modeling and simulation of gene networks by system theoretic approaches, using methods from mathematics, computer science and engineering [dG05,Alo07,DSCW05]. Some of these models are briefly reviewed in Chapter 7, followed by a coding theoretic approach to reverse engineering of the dynamics of biological networks.

3.5 Summary

We gave a brief overview to molecular biology and cellular information processing. We shall briefly summarize the most important facts from this chapter that the reader should keep in mind for the remainder of this thesis:

The genomic information of each organism is stored in its DNA. A single DNA strand consists of four nucleotides A, C, G, T (also referred to as bases) and can thus be regarded as a digital signal. Chemically, the nucleotides can be subdived into the purine class (A,G) and the pyrimidine class (C,T). DNA is a double stranded molecule, with the bases of the two strands pairing according to the following determinism: A$\leftrightarrow$T and C$\leftrightarrow$G. RNA is similar to DNA with T replaced by U, and single stranded instead of double stranded. RNA has the affinity to fold into complex secondary structures. The structure of the RNA often determines its function.

Each cell contains its own copy of DNA. When the cell divides, it has to replicate its DNA. Replication is a *semi-conservative*, noisy process. DNA is constantly suffering from damage; the cell is, however, equipped with several error correcting mechanisms that are able to detect and correct degenerated DNA. As a last resort, the cell can trigger its own suicide (*Apoptosis*) when DNA damage has reached an unacceptable level. The overall error rate per nucleotide is estimated to be $10^{-9} - 10^{-10}$ per cell generation, for example, in a human cell. There are two types of small scale errors that can occur: substitution errors exchange a DNA base by another. Insertions/deletions insert/remove a single or multiple bases into/from the DNA strand. Insertions and deletions (InDels) occur far less frequently than substitutions.

There is no common definition of what a *gene* is. Here, we define a gene as a genomic sequence encoding a protein or RNA product. We distinguish between the class of *non-coding* and *protein-coding* genes. We say genes are *transcribed* into an RNA product. Protein-coding genes are divided into *exons* and *introns* and transcribed into mRNA. Exons are further *translated* into chains of *amino acids* according to the *genetic code* while introns are removed from the mRNA before translation. The amino acid chains form three dimensional structures, and the final product is called a protein.

The cell responds to internal and external signals by producing appropriate proteins called *transcription factors* (TFs). These bind to specific regulatory regions in proximity of target genes causing the gene to increase (activate) or decrease (repress) its transcription rate. Since transcription factors are encoded by genes themselves, this set of interactions forms a complex dynamical network. The cell thus responds to external and internal stimuli by changing its gene expression patterns.

A gene network can be drawn as a sparse graph, where nodes represent genes and edges represent interactions among genes. It was observed that the topology of transcription networks are not random but scale-free, i.e., they have more nodes with a very high number of connections and fewer nodes with less connections than a random network on average. Experiments suggest that gene networks are remarkably robust with respect to disruption of single or multiple genes.

4

Bioinformatics

We introduce important concepts from bioinformatics, focusing on probabilistic sequence analysis. This chapter is most relevant to Chapter 5 since it introduces the basic definitions and methods of computational molecular evolution, that provide the theoretical framework for Chapter 5.

Section 4.1 establishes the probabilistic framework for analyzing two temporally diverged DNA sequences. Scoring schemes are used to detect sequences that have evolved from a common ancestor. Models of sequence evolution, based on continuous time Markov chains, are presented and classified. The concept of sequence alignment is introduced.

Section 4.2 generalizes the statistical analysis to the case when multiple sequences share common ancestry. Phylogenetic trees are defined, and it is shown how to efficiently calculate the likelihood of sequences on a tree using the Felsenstein algorithm. Finally, multiple sequence alignments are briefly discussed.

4.1 Analysis of two DNA sequences

On the sequence level, many problems in bioinformatics center around analyzing the statistical properties of so-called *homologous* DNA sequences:

Definition 4.1.1 (Homology, Orthology, Paralogy) A pair of DNA sequences is said to be *homologous* if the sequences share common ancestry. There are two different types of homology: the sequences are *orthologous* if their homology is the result of a speciation from an immediate ancestral species, i.e. they evolved from a common ancestor sequence in genomes of different species. They are called *paralogous* if they originate from a sequence duplication event (e.g., gene duplication caused by mutation) and then evolved within the genome of same species. □

A comparative analysis of homologous sequences within the same genome or among genomes often allows to draw biologically meaningful conclusions about the sequences or the relationship and evolutionary history of species. For example, homology of genes may imply a similar function on the protein level. In this section, we shall briefly review the basic concepts of probabilistic sequence analysis.

We first describe how scoring schemes are used to detect homology between two sequences. Then, we present probabilistic models of sequence evolution that allow for a statistical analysis of DNA sequences.

4.1.1 Sequence homology and scoring schemes

Suppose two sequences are homologous due to a speciation or duplication event in the past. After the event, the sequences evolved and were independently subjected to mutations as described in Chapter 3, Section 3.2. As a result, the two sequences will both look different from the original, ancestral sequence, and different from each other. An important task is to reliably detect the homology of sequences. For example, we may be given a gene sequence, and we want to detect possible orthologous genes in another species. The question to answer is how similar the sequences have to be in order to decide for homology?

We shall first restrict our attention to point substitutions, meaning that we assume that no insertion or deletion has occurred in the evolution of the homologous sequences. The problem can be formalized as follows: given a short sequence $\boldsymbol{x} = [x_1, .., x_N] \in \mathcal{A}^N$ and a longer sequence (e.g., a genome) $\boldsymbol{y} = [y_1, .., y_M] \in \mathcal{A}^M$, $M > N$, detect homologous representatives of $\boldsymbol{x}$ in $\boldsymbol{y}$. For this purpose, a symmetric scoring function

$$s : \mathcal{A} \times \mathcal{A} \to \mathbb{R}, \quad (x, y) \mapsto s(x, y) = s(y, x) \tag{4.1}$$

is introduced that is used to score sequence matches, i.e., for each possible overlap of $\boldsymbol{x}$ with $\boldsymbol{y}$ a score at overlap n is computed as follows:

$$S(n) = \sum_{i=1}^{N} s(x_{n+i-1}, y_i), \quad n = 1, .., M - N + 1. \tag{4.2}$$

To detect sequences that are homologous with high probability, the distribution of $S(n)$, or at least for $S(n)$ of high scoring segments, is desired. Let us first consider the case of a single overlap of length N. The following scoring model is assumed: $\boldsymbol{x}$ and $\boldsymbol{y}$ are realizations of iid random processes $\{\mathsf{X}_i\}_{i=1}^N$ and $\{\mathsf{Y}_i\}_{i=1}^N$ with distributions $\mathsf{X}_i \sim q(x)$ and $\mathsf{Y}_i \sim q(y)$. A pair $(\mathsf{X}_i, \mathsf{Y}_i)$ of independent (non-homologous) nucleotides then has distribution $(\mathsf{X}_i, \mathsf{Y}_i)_{ind} \sim q(x)q(y)$, and a pair of homologous nucleotides is assumed to have distribution $(\mathsf{X}_i, \mathsf{Y}_i)_{hom} \sim p(x, y)$. Let $s(x, y)$ be a scoring function as described above with the additional requirements that $s(x, y)$ is positive for at least one pair x, y and that the expected score in two independent sequences is negative

$$E\{s(x, y)\} = \sum_{x,y \in \mathcal{A}} q(x)q(y)s(x, y) < 0.$$

In 1990, Karlin and Altschul showed that under these conditions, and as the size N of the sequences grows, an optimal target distribution has the form [KA90, Alt91]

$$p(x, y) = q(x)q(y)e^{\eta s(x,y)}, \tag{4.3}$$

with η implicitly determined by

$$\sum_{x,y} p(x, y) = \sum_{x,y} q(x)q(y)e^{\eta s(x,y)} \stackrel{!}{=} 1. \tag{4.4}$$

An *optimal scoring scheme* can hence be specified from any chosen target distribution $p(x, y)$

$$s(x,y) = \frac{1}{\eta} \log \left(\frac{p(x,y)}{q(x)q(y)} \right). \tag{4.5}$$

The scoring scheme has an information theoretic interpretation: if we choose $\eta = \log(2)$, then the expected score in two aligned homologous sequences is

$$E\{s(x,y)\} = \sum_{x,y} p(x,y) \log_2 \left(\frac{p(x,y)}{q(x)q(y)} \right) = \mathcal{D}(p(x,y)||q(x)q(y)), \tag{4.6}$$

which is the Kullback-Leibler (KL) divergence between $p(x, y)$ and $q(x)q(y)$, according to Eq. (2.9). If we assume that $q(x) = p(x)$ and $q(y) = p(y)$ are the marginals of $p(x, y)$, this becomes the mutual information (cf. Eq. (2.12a))

$$\mathcal{D}(p(x,y)||p(x)p(y)) = I(\mathsf{X};\mathsf{Y}).$$

Within the assumed iid scenario and two sequences of length N, it holds that: $I(\mathbf{X};\mathbf{Y}) = NI(\mathsf{X};\mathsf{Y})$.

Now we switch back to our original problem, where we wanted to find a homologous sequence of length N in a longer sequence of length M. Karlin and Altschul showed that the length N of a scoring segment to be statistically significant is inverse proportional to the mutual information and depends on the length of the sequences. They derived exact formulas of score distributions for large N and gave the following approximation for deciding whether a sequence match score is significant:

$$NI(\mathsf{X};\mathsf{Y}) \gtrapprox \log_2(NM) \text{ bits} \tag{4.8a}$$

$$\log_2(NM)/N \lessapprox I(\mathsf{X};\mathsf{Y}) \tag{4.8b}$$

bits are required to decide for a statistically significant match. Hence, the more "diverged" the sequences are in terms of their mutual information, the longer the length N of the query sequence has to be chosen. On the other hand, for fixed N,M, we can estimate how large the mutual information between the sequences has to be in order to make statistically sound statements.

It is interesting that Eq. (4.8a) gives a similar inequality condition as the fundamental channel capacity theorem by Shannon (cf. Section 2.2.2). The interpretation follows the one presented in Section 2.3: there are $2^{NI(\mathsf{X};\mathsf{Y})}$ sequences that are distinguishable, and we have NM different sequences in our database to compare $\boldsymbol{x}$ with. So, if we want to identify the unique sequence among these sequences that $\boldsymbol{x}$ was derived from (i.e., homologous) with high probability, $\log_2(NM)/N$ must not exceed $I(\mathsf{X};\mathsf{Y})$.

Example 4.1.1 *A typical protein consists of* 2^8 *amino acids. For comparing such a sequence with a database containing* 2^{22} *residues,*

$$I(\mathsf{X};\mathsf{Y}) \gtrapprox \log_2(2^{30})/N \approx 0.1172 \textit{ bits}$$

are required to detect the homologous sequence in the database. □

The arguments above define an optimal scoring scheme and they state that an optimal score is the log ratio of the target and the background distribution. However, it is not specified how to derive the distribution $p(x, y)$. Below, we show how to derive such distributions based on probabilistic sequence models. The mathematical description and the most commonly used models are presented next.

4.1.2 Models of sequence evolution

An information theoretic view of homology is depicted in Figure 4.1: the single ancestral sequence modeled as r. v. Z is transmitted over two *independent* channels, and the two observable outcomes Y, X are received. It follows that $\mathsf{X} \to \mathsf{Z} \to \mathsf{Y}$ form a Markov chain

$$p(x, y|z) = p(x|z)p(y|z). \tag{4.9}$$

We wish to model the conditional distributions $p(x|z)$ and $p(y|z)$ in order to specify the joint distribution for (X, Y)

$$p(x, y) = \sum_z p(x, y|z)p(z) = \sum_z p(x|z)p(y|z)p(z), \tag{4.10}$$

used in Equation (4.5) to derive optimal scores. In the framework of DNA sequence evolution, distances are assigned to sequences representing the time since divergence from the ancestor. Let X and Y have distances t_1 and $t_2 > t_1$, respectively, then we require that $I(\mathsf{X}; \mathsf{Z}) > I(\mathsf{Y}; \mathsf{Z})$. Furthermore, X, Y should be independent of any previous ancestor given Z, meaning $\mathsf{X} \to \mathsf{Z} \to \mathsf{Y}$ form a Markov chain. A continuous time Markov chain model as described in Section 2.4 satisfies the required properties. Recall that such a model is determined by a rate matrix $\boldsymbol{R}$, and time-dependent transition probabilities are calculated using the matrix exponential $\boldsymbol{P}(t) = e^{t\boldsymbol{R}}$. In a sequence evolution framework, it is usually assumed that processes are homogenous and stationary with equilibrium distribution $\boldsymbol{\pi}$.

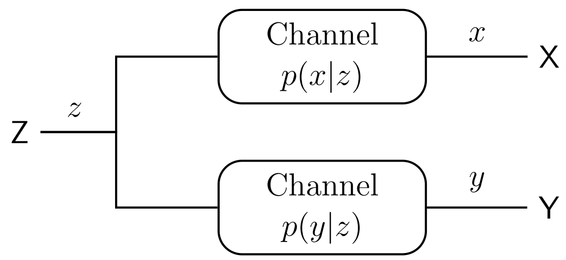


Figure 4.1: A transmission diagram of homology.

In the following, we consider the evolution of a *single* nucleotide: after divergence from the common ancestor, the nucleotides evolve as two independent, identical $\texttt{Markov}(\boldsymbol{\pi}, \boldsymbol{R})$ processes with state space $\mathcal{A}$. In the evolution framework, we refer to the $\texttt{Markov}(\boldsymbol{\pi}, \boldsymbol{R})$ process as *substitution process*.

Now, suppose we know from an independent source the (fixed) divergence times t_1, t_2 of the sequences, then we can calculate the transition probabilities via

$$\boldsymbol{P}(t_1) = e^{\boldsymbol{R}t_1}, \quad \boldsymbol{P}(t_2) = e^{\boldsymbol{R}t_2},$$

and thus the nucleotide pair distribution given in Eq. (4.10). Note that the stationarity assumption implies

$$p(z) = p(x) = p(y) = \boldsymbol{\pi} \tag{4.11}$$

because of Theorem 7 in Chapter 2.4.

For the remainder of this section, we shall consider the quarternary case $\mathcal{A} = \{A, C, G, T\}$ for DNA sequences. Of course, the same concepts can be generalized to any finite alphabet (e.g., amino acid alphabet). The substitution process is defined by the parameters $\boldsymbol{\pi}$ and $\boldsymbol{R}$, specified by 4 and 16 parameters respectively. Certain constraints reduce the numbers of parameters: to yield a valid rate matrix, the diagonal elements of $\boldsymbol{R}$ must be chosen such that $\boldsymbol{Re} = \boldsymbol{0}$, and a valid distribution satisfies $\boldsymbol{\pi e} = 1$ (where $\boldsymbol{e}$ is the all-one vector $\boldsymbol{e} = [1, 1, ..1]^T$). Further, as established in Theorem 6 in Chapter 2.4, the constraint $\boldsymbol{\pi R} = \boldsymbol{0}$ applies for stationary processes. Therefore, an overall number of 11 parameters is required to specify a substitution process. In practice, these parameters are unknown and have to be estimated from data, and it is therefore often necessary to further reduce the numbers of parameters in order to avoid overfitting effects [WLG01, Yan06, Nie05, DEKM98].

A common assumption is that the process is reversible, i.e., it is in detailed balance (Definition (2.4.8)), i.e., for all $s_i, s_j \in \{A, C, G, T\}$:

$$\pi_{s_i} r_{s_i s_j} = \pi_{s_j} r_{s_j s_i}. \tag{4.12}$$

We can then write $\boldsymbol{R}$ as the product of $\boldsymbol{\Pi}$ and a symmetric matrix $\boldsymbol{R}_s$, i.e., $\boldsymbol{R} = \boldsymbol{R}_s\boldsymbol{\Pi}$, which satisfies the balance equation

$$\boldsymbol{\Pi R} = \boldsymbol{\Pi R}_s\boldsymbol{\Pi} = (\boldsymbol{R}_s\boldsymbol{\Pi})^T\boldsymbol{\Pi} = \boldsymbol{R}^T\boldsymbol{\Pi}. \tag{4.13}$$

By construction, the process has stationary distribution $\boldsymbol{\pi}$. In phylogenetic theory such a model is called *general reversible* (GREV) model and can be parameterized by a rate matrix of the form[1]

$$\boldsymbol{R} = \begin{pmatrix} \star & \pi_C r_\alpha & \pi_G r_\beta & \pi_T r_\gamma \\ \pi_A r_\alpha & \star & \pi_G r_\delta & \pi_T r_\epsilon \\ \pi_A r_\beta & \pi_C r_\delta & \star & \pi_T r_\zeta \\ \pi_A r_\gamma & \pi_C r_\epsilon & \pi_G r_\zeta & \star \end{pmatrix}, \tag{4.14}$$

where the $\star$ symbols in the diagonals represent entries that are chosen such that the row sums equal zero. The GREV model is determined by 9 parameters (6 rates plus $4 - 1$ stationary frequencies because of the $\boldsymbol{\pi e} = 1$ constraint).

The HKY model, determined by 5 parameters, enforces an additional constraint that only allows differences in rates within the Purine/Pyrimidine class[2], denoted r_β, and

[1] Throughout it is assumed that the ordering A, C, G, T applies to all rate matrices defined over the DNA alphabet $\{A, C, G, T\}$.

[2] This is called a transition in biology. Whenever we refer to a transition, it should be clear from the context whether the biological or statistical meaning applies.

rates between the two classes, called *transversions*, denoted as r_α:

$$\boldsymbol{R} = \begin{pmatrix} \star & \pi_C r_\alpha & \pi_G r_\beta & \pi_T r_\alpha \\ \pi_A r_\alpha & \star & \pi_G r_\alpha & \pi_T r_\beta \\ \pi_A r_\beta & \pi_C r_\alpha & \star & \pi_T r_\alpha \\ \pi_A r_\alpha & \pi_C r_\beta & \pi_G r_\alpha & \star \end{pmatrix}. \tag{4.15}$$

The Kimura 2 parameter (K2P) model is derived from this model assuming a uniform stationary distribution $\boldsymbol{\pi} = [1/4, ..., 1/4]$.

The F84 model (proposed by Felsenstein in 1984 [Yan06]) has a single free rate parameter, while allowing for an arbitrary distribution $\boldsymbol{\pi}$:

$$\boldsymbol{R} = \begin{pmatrix} \star & \pi_C r_\alpha & \pi_G r_\alpha & \pi_T r_\alpha \\ \pi_A r_\alpha & \star & \pi_G r_\alpha & \pi_T r_\alpha \\ \pi_A r_\alpha & \pi_C r_\alpha & \star & \pi_T r_\alpha \\ \pi_A r_\alpha & \pi_C r_\alpha & \pi_G r_\alpha & \star \end{pmatrix}, \tag{4.16}$$

and the Jukes-Cantor (JC) model, determined via 1 parameter only, is the reduced F84 model enforcing the uniform stationary distribution. Note that the Jukes-Cantor model gives rise to the QSC channel as discussed in Example 2.2.2. The nested relationship among the models is depicted in Figure 4.2. The channel models resulting from the JC and K2P rate matrices are compared in Figure 4.3.

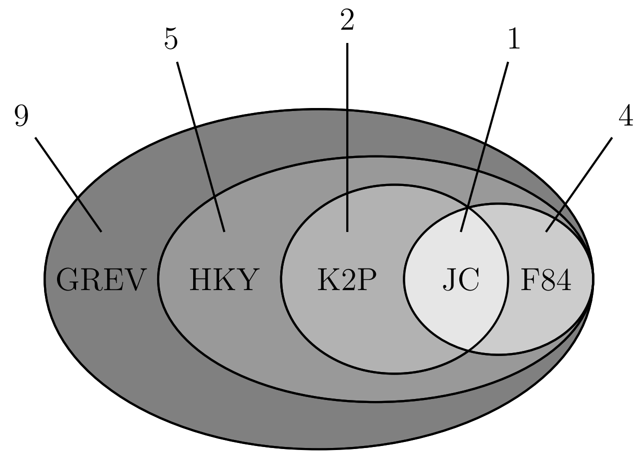


Figure 4.2: A Venn diagram showing the relationship among models of sequence evolution. The numbers at the top indicate the numbers of parameters that are required to specify the model.

Remark: The assumptions of identical, stationary and reversible substitution processes are mainly imposed to ease the computation and to reduce the number of parameters. There is no biological justification to make these assumptions. For example, as a consequence of stationarity, identical nucleotide distribution in the different sequences is assumed (Theorem (7)), which is often violated in real datasets. However, in this case $\boldsymbol{\pi}$ can be estimated directly from the observed sequences (e.g., by a simple frequency count), and the number of parameters for the GREV, HKY and F84 models is then further reduced.

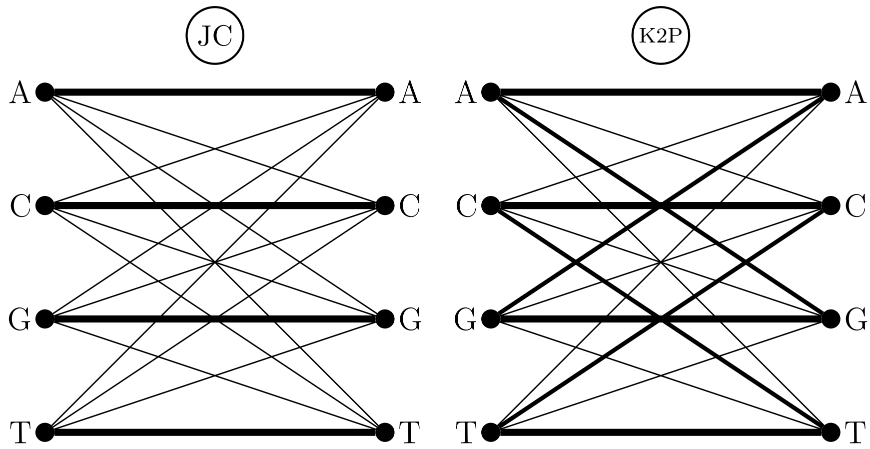


Figure 4.3: Transition diagrams of Jukes-Cantor (JC) and Kimura 2-parameter (K2P) model. Thickness of lines is proportional to probabilities: the JC model only allows for a uniform mutation probability while the K2P model can allow higher mutation rates within Pyrimidine/Purine classes than between classes.

The HKY model has two rate parameters determining rates of transitions (mutations within the Purine/Pyrimidine class) and transversions. Different genome studies suggest that this is a reasonable assumption [Nie05, HRY$^+$03]. For some cases it can be shown that the single parameter model (F84) is sufficient [YW95]. The JC and K2P assume a uniform stationary distribution which suggests an equal frequency of nucleotides in the observed sequences. While allowing the derivation of simple analytical expressions for transition probabilities, the assumption is often clearly violated in real datasets. Therefore, these models are more of theoretical than practical interest.

4.1.3 Gaps and pairwise sequence alignment

So far, we considered the evolution of sequences under point substitutions. We derived optimal scoring schemes for homologous sequences (Eq. (4.5)) and presented models of sequence evolution that determine nucleotide distributions. However, as discussed in Chapter 3, Section 3.2.3, DNA sequences additionally suffer from insertions and deletions that have to be dealt with. Since InDels most likely result in sequences of unequal lengths, the corresponding (homologous) nucleotides have to be identified first. Therefore, a pre-processing, that is called *alignment*, is necessary for most computational analysis of sequence data:

Definition 4.1.2 (Sequence alignment) Given two sequences $\boldsymbol{x} \in \mathcal{A}^N$, $\boldsymbol{y} \in \mathcal{A}^M$ and a "gap" symbol denoted by "$-$" not in $\mathcal{A}$, a *pairwise alignment* is a matrix $\boldsymbol{A} \in \{\mathcal{A}, -\}^{2\times L}$ with the following properties:

(i) $\max\{N, M\} \leq L \leq N + M$,
(ii) there is no column of $\boldsymbol{A}$ that only contains gap symbols
(iii) the first (second) row of $\boldsymbol{A}$ with gaps removed is identical to $\boldsymbol{x}$ ($\boldsymbol{y}$) □

Example 4.1.2 *Given the DNA sequences*

$$\begin{aligned} \boldsymbol{x} = &\ AAAAGCGTGGGTATTCTT \\ \boldsymbol{y} = &\ ATTAACGGGTCTTCTT \end{aligned}$$

a possible sequence alignment is given by

$$\boldsymbol{A} = \begin{array}{cccccccccccccccccc} A & A & A & A & G & C & G & T & G & G & G & T & A & T & T & C & T & T \\ | & : & : & | & : & | & & & | & | & | & | & : & | & | & | & | & | \\ A & T & T & A & A & C & - & - & G & G & G & T & C & T & T & C & T & T \end{array},$$

where we show additional symbols $|$ *and* $:$ *indicating match and substitution respectively. The alignment implies that at least* 4 *mutations have occurred and that an insertion or deletion has occurred in the sequences: two nucleotides GT were deleted from* $\boldsymbol{y}$ *or inserted from* $\boldsymbol{x}$. *This is accounted for by introducing two gap symbols in the row representing* $\boldsymbol{y}$. *Note that there are always many possible explanations of how sequences have evolved given a particular alignment.* □

An alignment $\boldsymbol{A}$ is optimal if it maximizes some predefined cost function. Cost functions for alignments are usually based on scoring schemes as presented in Section 4.1 extended by scores $(s(-,y),\ y \in \mathcal{A})$ for gaps, called *gap penalties*. It is, in general, not feasible to examine all possible alignments. For two DNA sequences of length 1000 nucleotides, the number of all possible alignments is well approximated by 10^{600} [EG05]. However, efficient algorithms based on dynamic programming were developed that find optimal alignments under certain cost functions. They are not discussed in this thesis, neither will be the choice of cost functions. The reader is referred to [DEKM98, EG05, Pev00] for an in-depth treatment of the subject.

4.2 Analysis of multiple sequences

4.2.1 Evolution model and phylogenetic trees

Many of the concepts developed in the previous section generalize to multiple sequences as well. Instead of two sequences, we now consider N homologous sequences all having evolved from the same ancestor sequence. This ancestor is assumed to have diverged into two independent sequences some time ago, and every of those two sequences has become the ancestor of another two sequences and so on. The generalized multiple sequence evolution model is depicted in Figure 4.4. The relationship among the sequences can be drawn as a tree-like graph which is referred to as *phylogenetic tree*.

Definition 4.2.1 (Phylogenetic tree) A *phylogenetic tree* is a graph $\mathcal{T} = \{V, E\}$ with nodes V representing DNA sequences and edges $E \subset \{(u,v) : u, v \in V, u \neq v\}$. An edge is drawn from node u to v if u is an ancestor of v. We refer to u as the *parent* of v and v is called the *child* of u. A phylogenetic tree is binary, i.e., every node has either one

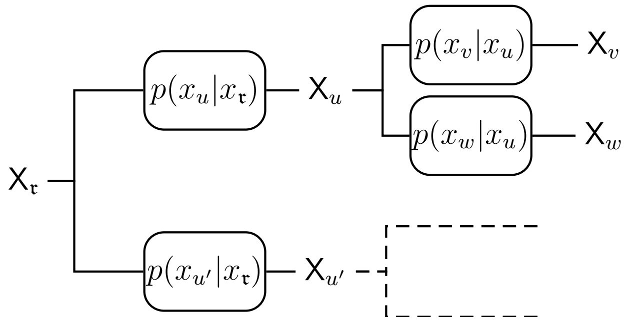


Figure 4.4: A transmission diagram for multiple homologous nucleotides.

parent or none, and every node has either two children ore none. The single node having no parent is called the *root* of the tree denoted by $\mathfrak{r}$, and the nodes having no children are called the *leaves*. We denote the set of leaf nodes as $\{\ell_1, .., \ell_N\}$. $\square$

A phylogenetic tree relating N homologous sequences has N leaves representing these sequences. The inner nodes represent ancestral sequences, and the root node represents the ancestral sequence common to all N leaf sequences. As for two sequences, the evolution of sequences between parent and child is modeled as a CTMP with rate matrix $\boldsymbol{R}$, parameterized by any of the sequence models described above. In order to calculate transition probabilities between nodes, we have to assign distances to the branches in the tree.

Definition 4.2.2 A *phylogenetic tree with branch lengths* $\{\mathcal{T}, \tau\}$ is a phylogenetic tree with positive real numbers $\tau = \{t_{uv} \in \mathbb{R}^+, \forall (u,v) \in E\}$ assigned to every branch in $\mathcal{T}$, where t_{uv} is called the *length* or *distance* of the branch connecting u and v. $\square$

The branch lengths can have different meanings such as years since divergence or expected frequency of mutations between sequences. The transition probabilities between two nodes u and v connected by branch $(u,v) \in E$ are then given by

$$\boldsymbol{P}_{u \to v} = e^{t_{uv}\boldsymbol{R}}. \tag{4.17}$$

We shall consider the evolution of a single nucleotide given a tree with branch lengths $\{\mathcal{T}, \tau\}$ and N leaves: denote X_u as the random variable modeling the nucleotide at node $u \in V$ with realization x_u. Starting from nucleotide $x_\mathfrak{r}$ in the common ancestor, at any node u the nucleotide x_u is transmitted to each child v with transition probabilities specified by Eq (4.17). This process continues until the nucleotides $x_{\ell_i},\ i = 1, .., N$ are observed at the leaves.

4.2.2 Felsenstein algorithm

In many applications we are interested in calculating the likelihood of observing nucleotides at the leaves of a tree given the parameters $\{\boldsymbol{R}, \boldsymbol{\pi}, \mathcal{T}, \tau\}$. Denote the observable nucleotides at the leaves as $\boldsymbol{x}_\ell = [x_{\ell_1}, ..x_{\ell_N}]$ and the unobservable ones at the inner nodes as $\boldsymbol{x}_{\ell-}$. The brute force solution to the likelihood problem yields

$$p(\boldsymbol{x}_\ell) = \sum_{\boldsymbol{x}_{\ell-}} p(\boldsymbol{x}_{\ell-}, \boldsymbol{x}_\ell), \tag{4.18}$$

which is feasible for small N only. An efficient iterative procedure for calculating the likelihood was presented by Felsenstein in 1981. Felsenstein's algorithm is an instance of the sum-product algorithm for tree-like graphical models and derived in the following: let $\boldsymbol{x}_{\ell(u)}$ denote the observations at the leaves of the tree rooted at u as depicted in Figure 4.5, and let v and w be children of u. Note that $x_{\ell(\mathfrak{r})} = x_\ell$ and $\boldsymbol{x}_{\ell(u)} = [\boldsymbol{x}_{\ell(v)}, \boldsymbol{x}_{\ell(w)}]$. Because X_v and X_w are conditionally independent given X_u, we have

$$\begin{aligned}
p(\boldsymbol{x}_{\ell(u)}|x_u) &= p(\boldsymbol{x}_{\ell(v)}, \boldsymbol{x}_{\ell(w)}|x_u) && \text{(4.19a)}\\
&= p(\boldsymbol{x}_{\ell(v)}|x_u)p(\boldsymbol{x}_{\ell(w)}|x_u) && \text{(4.19b)}\\
&= \left(\sum_{x_v\in\mathcal{A}} p(\boldsymbol{x}_{\ell(v)}, x_v|x_u)\right) \times \left(\sum_{x_w\in\mathcal{A}} p(\boldsymbol{x}_{\ell(w)}, x_w|x_u)\right) && \text{(4.19c)}\\
&= \left(\sum_{x_v\in\mathcal{A}} p(\boldsymbol{x}_{\ell(v)}|x_v)p(x_v|x_u)\right) \times \left(\sum_{x_w\in\mathcal{A}} p(\boldsymbol{x}_{\ell(w)}|x_w)p(x_w|x_u)\right), && \text{(4.19d)}
\end{aligned}$$

where the transition probabilities $p(x_v|x_u)$ and $p(x_w|x_u)$ are given by $\boldsymbol{P}_{u\to v}$ and $\boldsymbol{P}_{u\to w}$ from Eq. (4.17), and $p(\boldsymbol{x}_{\ell(v)}|x_v)$ was calculated in the previous iteration.

The algorithm starts at the leaves with initial probabilities

$$p(x|x_\ell) = \begin{cases} 1 & \text{if } x = x_\ell \\ 0 & \text{otherwise} \end{cases}, \tag{4.20}$$

for a leave node x_ℓ and proceeds up to the root in an iterative fashion according to Eq. (4.19d). At the root, the likelihood can finally be calculated by

$$p(\boldsymbol{x}_{\ell(\mathfrak{r})}) = \sum_{x_\mathfrak{r}\in\mathcal{A}} p(\boldsymbol{x}_{\ell(\mathfrak{r})}|x_\mathfrak{r})p(x_\mathfrak{r}) = \sum_{x_\mathfrak{r}\in\mathcal{A}} p(\boldsymbol{x}_{\ell(\mathfrak{r})}|x_\mathfrak{r})\pi_{x_\mathfrak{r}}. \tag{4.21}$$

where stationarity is assumed, therefore $p(x_\mathfrak{r})$ is given by $\boldsymbol{\pi}$, hence The algorithm can be given in a compact vectorized form: without loss of generality, assume that $\mathcal{A} = \{A, C, G, T\}$. Then, using the notation

$$\boldsymbol{p}^{(u)} = \big(p(\boldsymbol{x}_{\ell(u)}|x_u = A), p(\boldsymbol{x}_{\ell(u)}|x_u = C), p(\boldsymbol{x}_{\ell(u)}|x_u = G), p(\boldsymbol{x}_{\ell(u)}|x_u = T)\big), \tag{4.22}$$

the update rule of the FA in (4.19d) can be written as

$$\boldsymbol{p}^{(u)} = (\boldsymbol{p}^{(v)}\boldsymbol{P}^T_{u\to v}) \odot (\boldsymbol{p}^{(w)}\boldsymbol{P}^T_{u\to w}), \tag{4.23}$$

where $\odot$ denotes the entrywise (Hadamard-)product. Equation (4.21) then becomes

$$p(\boldsymbol{x}_{\ell(\mathfrak{r})}) = \boldsymbol{p}^{(\mathfrak{r})}\boldsymbol{\pi}^T. \tag{4.24}$$

A detailed listing of the FA is given in Algorithm 4.1, the message passing procedure is summarized in Figure 4.5.

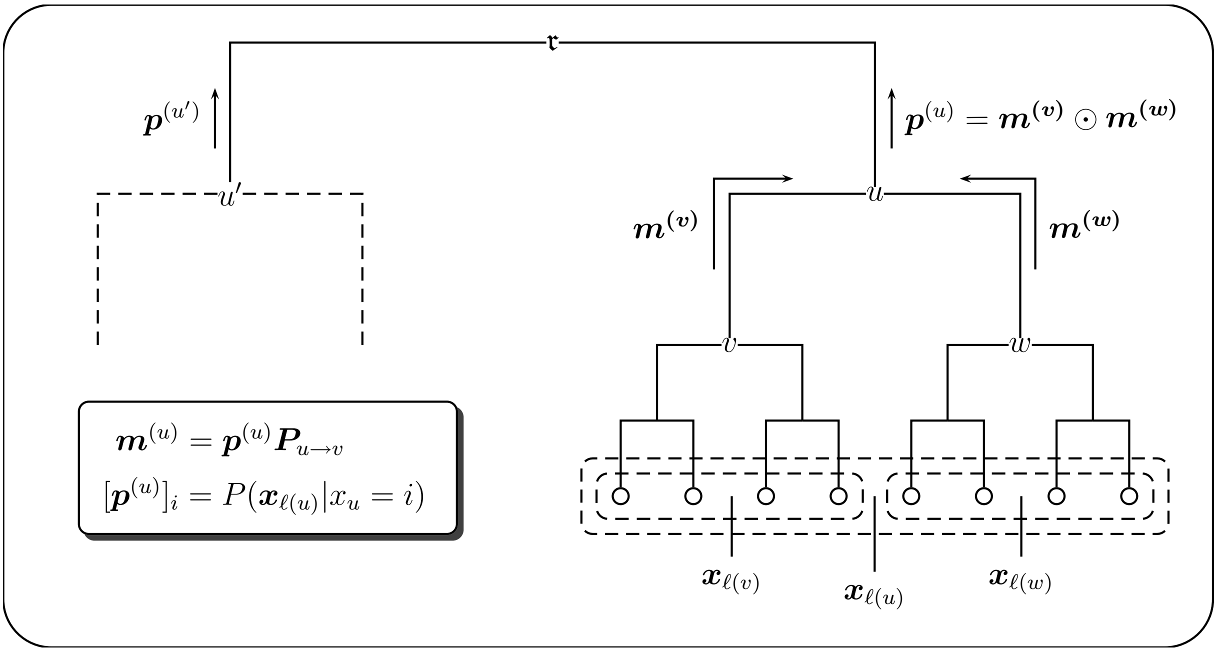


Figure 4.5: Felsenstein algorithm for tree-like graphical models, given in vectorized form: the algorithm starts at the leaves and proceeds by sending messages up to the root. Node u receives the messages $\boldsymbol{m}^{(v)}$ and $\boldsymbol{m}^{(w)}$ and calculates $\boldsymbol{m}^{(u)}$, which is sent to the parent of u. The vector $\boldsymbol{x}_{\ell(u)}$ denotes the observations at the leaves of the tree rooted at u. The ith entry of $\boldsymbol{p}^{(u)}$ is the probability of this observation given that the nucleotide at u was $x_u = i$.

Remark: An evolutionary model $\{\boldsymbol{R}, \boldsymbol{\pi}, \mathcal{T}, \tau\}$ can be interpreted as a probabilistic graphical model, and efficient calculation of the likelihood under such models has been solved independently in the bioinformatics and information theory/machine learning community. In information theory, the Felsenstein algorithm is referred to as *belief propagation* (first presented by Pearl) or *sum-product* algorithm. It is, for example, used in modern coding theory to decode some of the best performing error control codes defined on graphs [KFL01]. Pearl and Felsenstein developed the algorithm almost at the same time, Felsenstein published his algorithm in 1981 in the biological literature [Fel81], and Pearl presented his result (a more general approach than Felsentein's) 1982 at a conference on artificial intelligence [Pea82]. Among others, Siepel and Haussler [SH05] and Jordan et.al. [MPJ04] recently further extended the application of graphical models in phylogenetic analysis.

Algorithm 4.1: Felsenstein Algorithm

Require: evolutionary model $\{\boldsymbol{R}, \boldsymbol{\pi}, \mathcal{T}, \tau\}$,

$\boldsymbol{p}^{(\mathfrak{r})} = \text{get\_message}(\mathfrak{r})$
Output $p(\boldsymbol{x}_{\ell(\mathfrak{r})}) = \boldsymbol{p}^{(\mathfrak{r})}\boldsymbol{\pi}^T$
Return

Procedure get_message(node u)

if $u \in \{\ell_1, .., \ell_N\}$ **then**
    Output $\boldsymbol{p}^{(u)}$ according to Eq. (4.20)
else
    set c_1 first child of u.
    $\boldsymbol{p}^{(c_1)} = \text{get\_message}(c_1)$
    set c_2 second child of u.
    $\boldsymbol{p}^{(c_2)} = \text{get\_message}(c_2)$
    Output $\boldsymbol{p}^{(u)} = \left(\boldsymbol{p}^{(c_1)}\boldsymbol{P}^T_{u\to c_1}\right) \odot \left(\boldsymbol{p}^{(c_2)}\boldsymbol{P}^T_{u\to c_2}\right)$
end if
Return

4.2.3 Multiple sequence alignment

As for two sequences, we need to locally reconstruct the homology of nucleotides by *aligning* the sequences:

Definition 4.2.3 Given M_s sequences $\boldsymbol{x}_1, \boldsymbol{x}_2, .., \boldsymbol{x}_{M_s}$ over $\mathcal{A}$ of lengths $N_1, N_2, .., N_{M_s}$, and denoting "$-$" the "gap" symbol which is not in $\mathcal{A}$, a (global) *multiple sequence alignment* (MSA) is a matrix $\boldsymbol{A} \in \{\mathcal{A}, -\}^{M_s \times L}$ with the following properties:

(i) $\max\{N_1, .., N_{M_s}\} \leq L \leq \sum_{m=1}^{M_s} N_m$,
(ii) there is no column of $\boldsymbol{A}$ that only contains gap symbols,
(iii) the ith row of $\boldsymbol{A}$ with gaps removed is identical to $\boldsymbol{x}_i$. □

An optimal alignment is the one optimizing a predefined cost function, ideally derived from a complete probabilistic model of molecular sequence evolution. Such desired evolutionary model is too complex, and the number of possible evolutionary scenarios explaining an alignment is huge. Simplifying assumptions partly or entirely ignoring the phylogenetic tree must be made. We give two examples:

Example 4.2.1 (Minimum entropy scoring) *Let $[\boldsymbol{A}]_{ij}$ be the symbol in the ith sequence in the jth column. Let further*

$$f_j(x) = \frac{1}{M_s}|\{i : [\boldsymbol{A}]_{ij} = x \in \mathcal{A}\}|$$

be the observed frequencies for residue x in column j. The empirical entropy

$$cost_j(\boldsymbol{A}) = -\sum_x f_j(x)\log(f_j(x)),$$

is a measure of the column's nucleotide composition. An optimal alignment is then defined as [DEKM98]

$$\hat{\boldsymbol{A}} = \arg\min_{\boldsymbol{A}} \left\{ \sum_{j=1}^{L} cost_j(\boldsymbol{A}) \right\}.$$

□

Example 4.2.2 (Sum of pairs (SP) scoring) *A multiple alignment can be found by scoring all pairs of sequences:*

$$cost_j(\boldsymbol{A}) = \sum_{k,l:k<l} s([\boldsymbol{A}]_{kj}, [\boldsymbol{A}]_{lj}),$$

where the score function $s(\cdot,\cdot)$ is usually a scoring scheme as described in Section 4.1 with an additional gap penalty. The optimal alignment minimizes the sum of pairs

$$\hat{\boldsymbol{A}} = \arg\max_{\boldsymbol{A}} \left\{ \sum_{j=1}^{L} cost_j(\boldsymbol{A}) \right\}.$$

Note that there is no probabilistic justification of the SP score; each sequence is scored as if it descended from the $M_s - 1$ other species instead of a single ancestor. The SP scoring is a widely used scheme for multiple sequence alignment [DEKM98]. □

Even under simplified cost functions, the running time of the best known schemes for finding an optimal alignment, based on dynamic programming, increases exponentially with the number of input sequences. Modern MSA techniques rely on heuristic optimization strategies [BKR$^+$04, BP04, BDC$^+$03, CSK$^+$03]. In particular, alignments of whole genomes require a capacity of computational resources that exceeds those of most laboratories today. Upon generation, the data is centrally provided and maintained by large databases [KBD$^+$03] where it is freely accessible by researchers. The subject is not further discussed in this thesis, and we shall assume throughout that we are given "perfect" alignments whenever dealing with data produced by state-of-the-art MSA software tools obtained from online databases. The alignment shown in Figure 4.6 was created by a method described in [BKR$^+$04]. It was obtained through the UCSC database [KBD$^+$03] and aligns DNA sequences from 22 vertebrate species annotated to the left of the corresponding sequence. In each column of the alignment, the consensus nucleotide, i.e., the nucleotide that occurs with the highest frequency in that column, is indicated by a gray background. Deviations from consensus have white background.

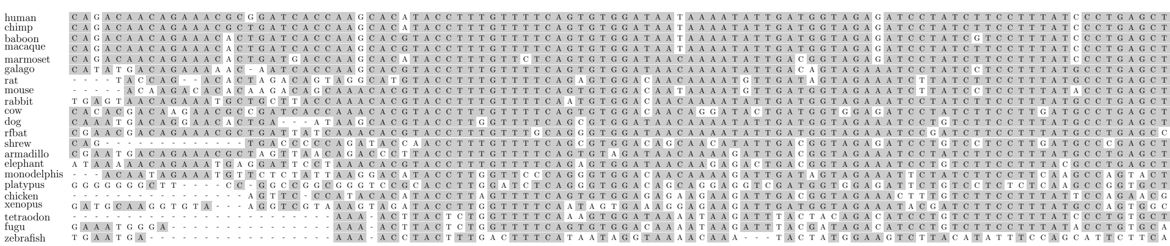


Figure 4.6: Multiple sequence alignment of 22 vertebrate species.

4.3 Summary

The material provided in this chapter outlined the fundamental concepts from bioinformatics necessary in order to understand the next Chapter 5.

Homology means that two sequences share ancestry. We showed how scoring schemes are used to detect two homologous DNA sequences. The substitution process of DNA sequences is modeled by a continuous time Markov process with rate matrix $\boldsymbol{R}$. The process is assumed to be stationary and reversible. Different constraints, based on biological considerations, are imposed on $\boldsymbol{R}$ and lead to different substitution models with reduced number of parameters. Some of these models do not conform with reality.

Due to evolutionary insertion and deletion (InDel) events, a sequence alignment is a necessary preprocessing step in order to reconstruct homology on a nucleotide level. InDels give rise to gaps in the alignment.

The evolution of multiple homologous sequences is described by a phylogenetic (binary) tree with branch lengths accounting for evolutionary distance among species. The likelihood of observed sequences given such a tree along with a model of evolution is efficiently calculated by the Felsenstein algorithm, a variant of the sum-product algorithm.

Multiple sequence alignments are computationally hard and state-of-the-art methods are based on suboptimal heuristics. Evaluating the quality of an alignment is also difficult. Whole genome sequence alignments are stored in centralized databases. We assume throughout this thesis that we are given perfect alignments and neglect all issues that arise from the alignment problem itself.

5

Identification of highly conserved DNA sequences

Based on concepts of mathematical evolutionary models introduced in the previous chapter, this chapter considers *phylogenetic systems* and the inference of biological insights from multiple DNA sequences analysis. The system under study has a single discrete input sequence (common ancestor) and multiple outputs (the genomes of today's species). Only the outputs can be observed since there is usually no information about the common ancestor available[1]. Sequences evolve over time according to a stochastic substitution process, and the system is therefore characterized by transition probabilities relating input and output symbols. The observed sequences are not independent but depend on each other through their *phylogeny*. *Comparative genomics* is the discipline studying these systems in order to draw biologically meaningful conclusions from homologous DNA sequences of multiple species [Nie05, Yan06].

In Section 5.2, we shall first briefly discuss the reverse engineering of phylogenetic models, i.e., the estimation of substitution process parameters and reconstruction of phylogenetic trees, thereby focusing on maximum likelihood methods. We try to give a coherent picture while it should be clear that, within the scope of this work, we can only outline basic principles of this extensively studied and vibrant research topic [Yan07, Yan06, Fel81, FC96, Nie05, WLG01, MFP05, EG05, DEKM98].

For the remainder of the chapter we then focus on a specific subproblem, that is, the estimation of rate variation, assuming that the previously mentioned parameters of the system were identified. We show that estimation of spatial rate variation along DNA sequences is directly related to the identification of *conserved regions* that recently gained high interest in the biology community since they are believed to be potential candidates for fundamental functional genomic units.

Space-time processes of evolution that allow for modeling rate variation are introduced in Section 5.3. In Section 5.4 we argue that mutual information is the *best* measure of conservation (incomputable for large numbers of species) and show by simulation that

[1] Here we focus on DNA sequences of vertebrate species. The situation may be different for fast evolving organisms such as viruses, where the ancestor sequence may actually be known.

rate variation *is* the crucial parameter related to conservation and that other parameters have minor influence. We present state-of-the-art methods for detecting conserved regions and derive an ML estimator for conservation under a realistic model of rate variation in Section 5.4.2. To the best of our knowledge ML estimation of rate variation under this model was not analyzed previously.

We present our results in Section 5.5: first, we evaluate our method by *in silico* simulations. Using a measure of correlation, we compare the performance of our approach to existing methods and find that the proposed algorithm performs best within the assumed simulation framework. Then, we apply our method to alignment data of 28 vertebrate species which was used in the recent ENCODE project and compare our conservation scores qualitatively to the three computational methods used in the ENCODE project.

For most parts of this Chapter we assume that evolution is solely driven by substitution errors, completely neglecting insertion and deletion events (InDels). Gaps in a multiple sequence alignment are then neglected, i.e., gaps in an alignment column are treated as if the corresponding species were not present in that alignment column. In fact, most existing conservation scores deal with gaps in this way. In contrast to that, we show how our conservation measure is extended to take InDels into account in a probabilistic fashion based on an algorithm developed for estimating InDel probabilities. Results on ENCODE data are presented and compared to the standard approaches.

5.1 Background and notation

We assume that we are given an alignment $\boldsymbol{A} \in \{\mathcal{A}, -\}^{M_s \times L}$ as defined in Section 4.2.3, and the lth column of an alignment is denoted by $\boldsymbol{a}_l$. The DNA alphabet $\mathcal{A} = \{A, C, G, T\}$ is assumed, yet the generalization to arbitrary alphabets is straightforward. Following our introduction in Section 4.1, a single nucleotide is modeled by a continuous time Markov process (CTMP) $\{\mathsf{X}_t\}_{t \geq 0}$ over state space $\mathcal{A}$. A DNA sequence of length N is a sequence of N such nucleotides denoted by $\{\mathsf{X}_t^{(l)}\}_{t \geq 0},\ l = 1, ..N$.

In this chapter we assume that such a sequence evolves along a phylogenetic tree $\mathcal{T}$ with branch lengths τ, subject to a substitution process specified by rate matrix $\boldsymbol{R}$, as described in Section 4.2. We refer to such a scenario as *phylogenetic system* with a single sequence as input and multiple homologous sequences as output. The observed alignment $\boldsymbol{A}$ represents the system output, i.e., the observed DNA sequences at the leaves of the phylogenetic tree.

As stated above, the substitution process between any two nodes u and v in that tree is characterized by a single rate matrix $\boldsymbol{R}$, and the transition probabilities between u and v are hence calculated as

$$\boldsymbol{P}_{u \to v}(t) = e^{t_{uv} \boldsymbol{R}}, \tag{5.1}$$

where t_{uv} denotes the branch length between u and v. The element in the ith row and jth column of $\boldsymbol{P}(t)$ is denoted by $p_{ij}(t)$. Substitution processes are assumed to be stationary with equilibrium distribution $\boldsymbol{\pi}$, reversible, and parameterized by one of the substitution models $\mathcal{M} \in$ {GREV, HKY, K2P, F84, JC} presented in Section 4.1. The following table summarizes the basic properties (derived in Section 2.4) that $\boldsymbol{R}$ and $\boldsymbol{P}$ satisfy:

#	Description	Property	Remark
(1)	Rate matrix constraint	$\boldsymbol{Re} = 0$	$\boldsymbol{e} = [1,1,1,1]^T$, Def. 2.4.5
(2)	Model constraint	$\boldsymbol{R} = \boldsymbol{R}_s\boldsymbol{\Pi}$	$\boldsymbol{R}_s = \boldsymbol{R}_s^T, \boldsymbol{\Pi} = \text{diag}(\boldsymbol{\pi})$, Eq. (4.13)
(3)	Transition probability	$\boldsymbol{P}(t) = e^{t\boldsymbol{R}}$	$t \in \mathbb{R}^+$, Thm. 5
		$\boldsymbol{Pe} = \boldsymbol{e}$	Def. 2.4.1
		$\boldsymbol{PR} = \boldsymbol{RP}$	Thm. 3
(4)	Stationarity	$\boldsymbol{\pi R} = \boldsymbol{0}$	Thm. 6
		$\boldsymbol{\pi P} = \boldsymbol{\pi}$	Def. 2.4.4
(5)	Reversibility	$\boldsymbol{\Pi R} = \boldsymbol{R}^T\boldsymbol{\Pi}$	Def. 2.4.8
		$\boldsymbol{\Pi P} = \boldsymbol{P}^T\boldsymbol{\Pi}$	Eq. (2.46)

In the following, a *phylogenetic system* or *evolutionary model* is represented by the set of parameters

$$\boldsymbol{\psi} = \{\mathcal{T}, \tau, \boldsymbol{R}, \boldsymbol{\pi}\}.$$

A single column in the alignment then follows the distribution $p(\boldsymbol{a}; \boldsymbol{\psi})$ depending on the evolutionary parameters. The likelihood

$$l_{\boldsymbol{A}}(\boldsymbol{\psi}) \sim P(\boldsymbol{A}; \boldsymbol{\psi}) \tag{5.2}$$

of the observed sequences can be calculated by Felsenstein's message passing algorithm (Section 4.2.2) when $\boldsymbol{\psi}$ is known. The following section introduces likelihood based inference of a phylogenetic system from observed DNA sequences.

5.2 Identification of phylogenetic systems

5.2.1 Substitution process estimation

The substitution process is characterized by the stationary distribution $\boldsymbol{\pi}$ and the rate matrix $\boldsymbol{R}$. The rate matrix is constrained by the chosen model $\mathcal{M}$ of substitutions discussed in Section 4.1, i.e., $\mathcal{M} \in$ {GREV,HKY,K2P,F84,JC}. Since the process is ergodic and assumed in equilibrium at the common ancestor, the distribution of bases is always $\boldsymbol{\pi}$, i.e., the substitution process is stationary. The stationary distribution can then be estimated from the observed sequences using a simple frequency count. For a given alignment $\boldsymbol{A} \in \{\mathcal{A}, -\}^{M_s \times L}$:

$$\hat{\pi}_i = \frac{1}{M_s L} \sum_{m=1}^{M_s} \sum_{l=1}^{L} (1 - d_H([\boldsymbol{A}]_{ml}, i)),\ i = A, C, G, T, \tag{5.3}$$

where $d_H(\cdot, \cdot)$ denotes the Hamming distance. While this is not an ML estimator for $\boldsymbol{\pi}$ [Yan06], it is the procedure which is most commonly applied in practice. In the following, we shall assume that $\boldsymbol{\pi}$ is given.

Estimation of $\boldsymbol{R}$ is more difficult: given an evolutionary process $\{\mathsf{X}_t\}_{t\geq 0}$, the expected number of mutations $N(t)$ in a sequence per site after time t is calculated as

$$E\{N(t)\} = \sum_{x_0 \neq x_t} p(x_0, x_t) = \sum_{x_0} \pi_{x_0} \sum_{x_t \neq x_0} p(x_t|x_0) \tag{5.4a}$$

$$= \sum_{x_0} \pi_{x_0} \left(1 - p(x_t = x_0|x_0)\right) \tag{5.4b}$$

$$= 1 - \sum_i \pi_i p_{ii}(t) \left[\frac{\#\text{Mutations}}{\text{site}}\right], \tag{5.4c}$$

where $p_{ii}(t)$ denotes the diagonal elements in the transition matrix given by $\boldsymbol{P}(t) = e^{t\boldsymbol{R}}$, and the unit is *number of mutations per site.* It is shown in Appendix B (Eq. (B.9a)) that $p_{ii}(t)$ and r_{ii} are related by

$$p_{ii}(t) = 1 + r_{ii}t + o(t), \tag{5.5}$$

where $o(t)/t \to 0$ as $t \to 0$. Plugging this into Eq. (5.4c) we find that the expected number of mutations per time unit reads as

$$\frac{E\{N(t)\}}{t} = 1/t - \sum_i \pi_i/t - \sum_i \pi_i r_{ii} - \sum_i \pi_i o(t)/t \tag{5.6a}$$

$$= -\sum_i \pi_i r_{ii} - o(t)/t \left[\frac{\text{Avg. \#Mutations}}{\text{site} \times \text{time}}\right]. \tag{5.6b}$$

And as t gets small:

$$\lim_{t\to 0} \frac{E\{N(t)\}}{t} = -\sum_i \pi_i r_{ii} \left[\frac{\text{Avg. \#Mutations}}{\text{site} \times \text{time}}\right]. \tag{5.7}$$

Substitution processes are often normalized such that

$$-\sum_i \pi_i r_{ii} = -\text{trace}\left\{\boldsymbol{\Pi}\boldsymbol{R}\right\} = 1/c, \tag{5.8}$$

where trace$\{\cdot\}$ returns the sum of diagonal elements of its argument. In this case, phylogenetic branch lengths can be interpreted as having unit *expected number of mutations per c sites* (c is usually chosen 1 or 100). The additional constraint reduces the number of parameters in the substitution models by 1.

Now, given two sequences $(\boldsymbol{x}_0, \boldsymbol{x}_t)$, where $\boldsymbol{x}_0 = [x_0^{(1)}, ..., x_0^{(L)}]$ is the realization of $\{\mathsf{X}_t^{(l)}\}_{t\geq 0}$, $l = 1, ..., L$ at $t = 0$ and $\boldsymbol{x}_t$ the realization at t, we need to estimate

$$\{\hat{r}_\alpha, \hat{r}_\beta, \hat{r}_\delta, \hat{r}_\gamma, \hat{r}_\varepsilon, \hat{r}_\zeta\} = \underset{r_\alpha,..,r_\zeta}{\arg\max}\{p(\boldsymbol{x}_0, \boldsymbol{x}_t)\} = \underset{r_\alpha,..,r_\zeta}{\arg\max}\{p(\boldsymbol{x}_t|\boldsymbol{x}_0)p(\boldsymbol{x}_0)\},$$
$$\text{s.t. } \boldsymbol{R}_s\boldsymbol{\Pi}\boldsymbol{e} = \boldsymbol{0}, \{r_\alpha, r_\beta, .., r_\zeta\} \subset \mathcal{M}, \tag{5.9a}$$

where $\boldsymbol{\Pi} = \text{diag}(\boldsymbol{\pi})$ and $\boldsymbol{R}_s$ is the symmetric part of the rate matrix with $r_\alpha, .., r_\zeta$ in the upper and lower triangular entries (cf. (Eq. (4.13))). Maximization is constrained by the properties of the rate matrix $\boldsymbol{R}_s\boldsymbol{\Pi}\boldsymbol{e} = \boldsymbol{0}$ and the constraints imposed by the chosen model of substitutions $\mathcal{M}$ (cf. Section 4.1). If we assume that sites evolve independently, we have

$$\begin{aligned}\arg\max_{r_\alpha,..,r_\zeta}\{p(\boldsymbol{x}_0,\boldsymbol{x}_t)\} &= \arg\max_{r_\alpha,..,r_\zeta}\left\{\prod_{l=1}^{L} p(x_t^{(l)}|x_0^{(l)})\pi_{x_0^{(l)}}\right\}\\ &\equiv \arg\max_{r_\alpha,..,r_\zeta}\left\{\sum_{l=1}^{L}\log\left(p(x_t^{(l)}|x_0^{(l)})\right)\right\}. \qquad (5.10a)\end{aligned}$$

The log likelihood function that is to be maximized depends on $r_\alpha,..,r_\zeta$ through $\boldsymbol{P} = e^{\boldsymbol{R}}$. Closed form expressions for $\boldsymbol{P}$ are only available for the most simple models (cf. Example in Section 2.4.4) and the rates have to be found via numerical optimization techniques. A rate matrix under the GREV model published by Siepel et al. is given by [SBP05a]:

$$\boldsymbol{R} = \begin{pmatrix} -0.9906 & 0.1788 & 0.4907 & 0.3211\\ 0.2573 & -1.0017 & 0.1866 & 0.5578\\ 0.7062 & 0.1866 & -1.1619 & 0.2691\\ 0.3211 & 0.3876 & 0.1870 & -0.8957 \end{pmatrix}. \qquad (5.11)$$

It is easily checked that the matrix is reversible and calibrated such that trace$\{\boldsymbol{\Pi R}\} = -1$. The corresponding stationary distribution is given by

$$\boldsymbol{\pi} = [0.295,\ 0.205,\ 0.205,\ 0.295]. \qquad (5.12)$$

We generated two sequences of 5000 basepairs with $t = 0.25$ using this matrix. The log likelihood function over r_α is shown in Figure 5.1. The true r_α is given by 0.8. Inspection of the function yields that the maximum likelihood estimate is around $\hat{r}_\alpha \approx 0.8$. We see that, in this case, the underlying univariate optimization problem is solved using hill climbing techniques such as Newton-type algorithms. Schadt and Lange showed that the derivative of $e^{\boldsymbol{R}}$ with respect to a rate parameter r for general, reversible $\boldsymbol{R}$ ($\boldsymbol{R}$ diagonalizable: $\boldsymbol{R} = \boldsymbol{U}\boldsymbol{\phi}\boldsymbol{U}^{-1}$, $\boldsymbol{\phi} = \mathrm{diag}([\phi_1,..,\phi_4])$) is given by [SL02]

$$\frac{\partial}{\partial r}e^{\boldsymbol{R}} = \boldsymbol{U}\left[\boldsymbol{D}\odot\left(\boldsymbol{U}^{-1}\frac{\partial}{\partial r}\boldsymbol{R}\boldsymbol{U}\right)\right]\boldsymbol{U}^{-1}, \qquad (5.13)$$

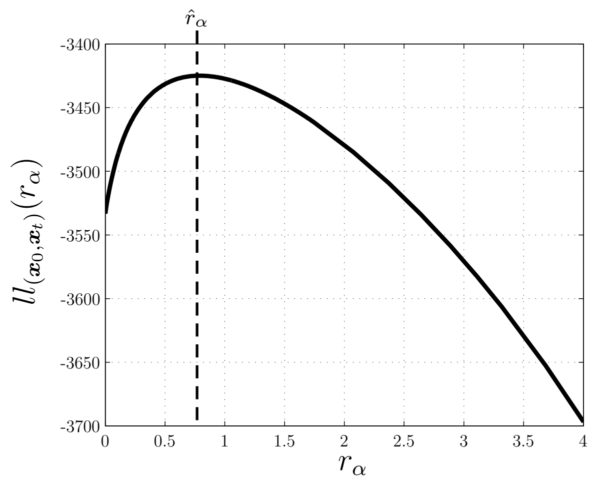


Figure 5.1: Log likelihood function over r_α for two *in silico* generated sequences $\boldsymbol{x}_0$, $\boldsymbol{x}_t$ of 5000 bases length with $t = 0.25$ using the matrix given in Eq. (5.11). The *true* r_α is given by 0.8.

where $\boldsymbol{D}$ is the matrix with

$$[\boldsymbol{D}]_{jk} = \begin{cases} e^{\phi_j} & \text{if } j = k \\ \frac{e^{\phi_j} - e^{\phi_k}}{\phi_j - \phi_k} & \text{if } j \neq k \end{cases} . \tag{5.14}$$

Since t is fixed but unknown, we first estimate $\hat{t}\hat{\boldsymbol{R}}$ and recover $\hat{t}$ and $\hat{\boldsymbol{R}}$ as follows: Let f_{kj} be the frequency count of nucleotide pairs k, j appearing jointly in $\boldsymbol{x}_0$ and $\boldsymbol{x}_t$, respectively. Then an estimate of $p(x_0^{(l)} = k, x_t^{(l)} = j)$ is given by

$$\hat{p}(x_0^{(l)} = k, x_t^{(l)} = j) = \hat{p}(x_t^{(l)} = j | x_0^{(l)} = k)\pi_k = \hat{p}_{kj}(t)\pi_k \overset{(a)}{=} \hat{p}_{jk}(t)\pi_j = \frac{f_{kj}}{L}, \tag{5.15a}$$

where (a) is due to reversibility. Let the matrix of frequency counts be denoted by $\boldsymbol{F}$, then $\hat{\boldsymbol{P}} = \boldsymbol{\Pi}^{-1}\boldsymbol{F}$. Now, if $\boldsymbol{R}$ is normalized such that Eq. (5.8) holds, then

$$-\text{trace}\,\{\boldsymbol{\Pi}\boldsymbol{R}\} = 1/c \tag{5.16a}$$

$$-\text{trace}\,\{\boldsymbol{\Pi}t\boldsymbol{R}\} = t/c, \tag{5.16b}$$

and we can recover both, $\hat{t}$ and $\hat{\boldsymbol{R}}$:

$$\hat{t}\hat{\boldsymbol{R}} = \log(\hat{\boldsymbol{P}}) \tag{5.17a}$$

$$\Rightarrow \hat{t}/c = -\text{trace}\left\{\boldsymbol{\Pi}\log(\hat{\boldsymbol{P}})\right\} \tag{5.17b}$$

$$\Rightarrow \hat{\boldsymbol{R}} = -\frac{1/c\log(\hat{\boldsymbol{P}})}{\text{trace}\left\{\boldsymbol{\Pi}\log(\hat{\boldsymbol{P}})\right\}}. \tag{5.17c}$$

5.2.2 Tree reconstruction and evolutionary distance

Given the multiple sequence alignment $\boldsymbol{A} \in \{\mathcal{A}, -\}^{M_s \times L}$ the tree reconstruction problem is to find the topology of a phylogenetic tree that "best" explains the data. A likelihood approach will select the topology $\mathcal{T}$ according to

$$\hat{\mathcal{T}} = \arg\max_{\mathcal{T}} \{l_{\boldsymbol{A}}(\mathcal{T})\} \tag{5.18}$$

from the space of all possible topologies. Given N labels, there are $\prod_{i=3}^{N}(2N-3)$ different trees with these labels at the leaves, and a brute force solution to Eq. (5.18) is therefore intractable even for small N. In fact, it was shown in [CT06, Roc06] that finding the ML tree is an NP-hard problem. The search for good heuristic methods for reconstructing phylogenetic trees from multiple sequence alignments is an ongoing area of research. Besides ML, distance methods, maximum parsimony (a model free approach explaining alignments with the minimum number of evolutionary events) and Bayesian methods (taking tree priors into account) have been proposed.

Even the comparison and evaluation of statistical properties of these methods is controversial [Yan06]. Today, the most commonly used method for assessing uncertainty in estimated phylogenies is bootstrapping (sampling multiple data sets from the original

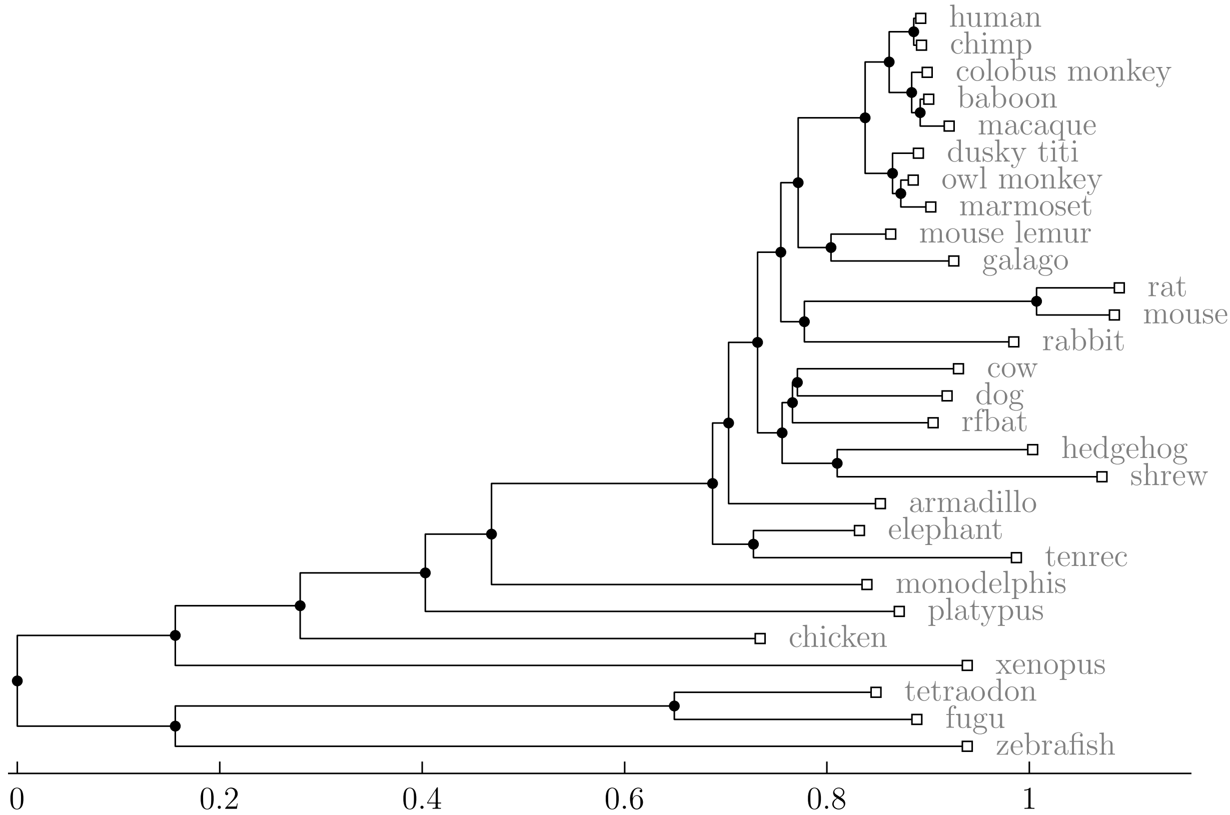


Figure 5.2: The phylogenetic tree used in the ENCODE project.

data with replacement, so generating a whole set of phylogenies that can be analyzed, e.g., by empirical calculation of the variance) [WLG01]. In practice, instead of selecting a single algorithm, biologists often use a whole set of computational methods and take the consensus of the inferred trees. In conlusion: there is no *best method* and if there is, the means to assess it are yet to be developed.

The tree estimated from 28 vertebrate genomes used in the ENCODE project is depicted in Figure 5.2. Given a fixed tree, the branch lengths can be optimized iteratively. The likelihood of a given MSA is calculated via message passing (Section 4.2.2). Branches are then optimized one at a time, keeping all other branches and parameters fixed. Figure 5.3 shows the log-likelihood function over varying t_{uv} for an *in silico* generated alignment $\boldsymbol{A}$ of 100 columns, where the tree in Figure 5.2 was used, and t_{uv} represents the branch separating the mammalian species (human,..., platypus) from the rest (chicken,..., zebrafish). The ML estimate is close to the true value which is 0.089. We see that, in this case, we have a convex optimization problem, and hill-climbing algorithms (such as Newton type algorithms) can be used to find the MLE.

Felsenstein's message passing procedure can be naturally extended to compute the derivative of the likelihood, which can then be used to speed up the optimization: In Appendix D it is derived that update messages at node u' with children v' and w' are calculated as follows:

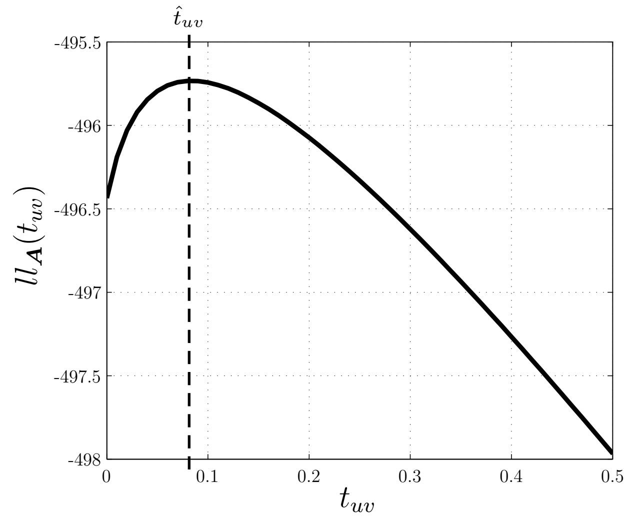


Figure 5.3: Log-likelihood function over varying t_{uv} for an *in silico* generated alignment spanning 100 columns.

$$\frac{\partial}{\partial t_{uv}}\boldsymbol{p}^{(u')} = \begin{cases} (\frac{\partial}{\partial t_{uv}}\boldsymbol{p}^{(v')}\boldsymbol{P}^T_{v'\to u'}) \odot (\boldsymbol{p}^{(w')}\boldsymbol{P}^T_{w'\to u'}) & \text{if } v' \text{ is ancestor of or equal } u \\ (\boldsymbol{p}^{(v')}\boldsymbol{R}^T\boldsymbol{P}^T_{v'\to u'}) \odot (\boldsymbol{p}^{(w')}\boldsymbol{P}^T_{w'\to u'}) & \text{if } u' = u, v' = v \\ (\boldsymbol{p}^{(v')}\boldsymbol{P}^T_{v'\to u'}) \odot (\boldsymbol{p}^{(v')}\boldsymbol{R}^T\boldsymbol{P}^T_{w'\to u'}) & \text{if } u' = u, w' = v \\ \boldsymbol{0} & \text{if } u' \text{is descendant of } u. \end{cases} \tag{5.19}$$

It was proved that under the F84 model, the log likelihood function is convex when other branch lengths are fixed and it was conjectured that this applies to more general substitution models as well. However, as discussed by Yang in [Yan06], this does not guarantee the multivariate problem to be convex. In fact, counter examples were constructed demonstrating the existence of likelihood surfaces with multiple peaks even on small trees with four species. In this case hill climbing methods get stuck in a local maximum, and (mostly heuristic) global optimization algorithms such as stochastic optimization or genetic algorithms have to be applied. Simulation studies have shown that multiple local maxima are much less common under simplistic models like the JC model than under more realistic parameter rich models [Yan06].

5.3 Extended models of evolution

5.3.1 A space-time process accounts for variable rates

Computational genome analysis and biological experiments show that different sites of DNA exhibit different rates of substitutions. Suppose we are given a phylogenetic model $\{\mathcal{T}, \tau, \boldsymbol{R}, \boldsymbol{\pi}\}$. The model describes the evolution of sequences along the phylogenetic tree which we refer to as the *time-process*. The model is extended to account for variable rates

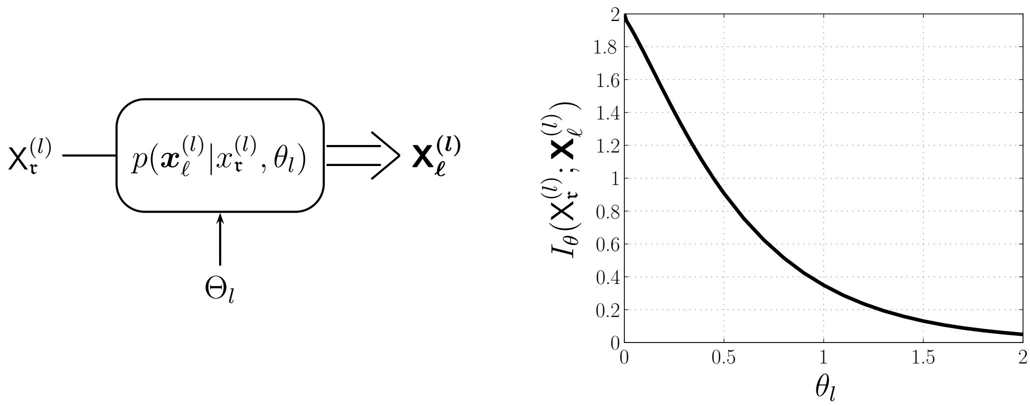


Figure 5.4: Left: transmission model of evolution with spatial rate variation. Right: the mutual information $I_\theta(\mathsf{X}_{\mathfrak{r}}^{(l)}; \mathbf{X}_\ell^{(l)})$ between $\mathsf{X}_{\mathfrak{r}}^{(l)}$ and $\mathbf{X}_\ell^{(l)}$ over θ_l for a five species subtree.

by introducing a discrete *space-process* $\{\Theta_l\}_{l\geq 0}$, $l \in \mathbb{N}^+$, where each index l corresponds to a site in the common ancestor DNA sequence. The spatial process "interacts" with the time-process by scaling the distances of the tree. Evolution is now modeled as follows: at any position l of the common ancestor sequence, the space-process is sampled yielding the realization θ_l. The nucleotide at site l then evolves along the phylogenetic tree with transition probabilities depending globally on θ_l through

$$\boldsymbol{P}_{u\to v} = e^{\theta_l t_{uv} \boldsymbol{R}}. \tag{5.20}$$

From an information theoretic perspective this is equivalent to sending information over a channel with varying state of quality. A transmission model for the space-time model of variable rates is depicted in Figure 5.4: the lth base at the common ancestor sequence $\mathsf{X}_{\mathfrak{r}}^{(l)}$ enters the channel and the observations at the leaves of the tree represent the outcome of the channel $\mathbf{X}_\ell^{(l)}$. The channel is characterized by the transition probabilities $p(\boldsymbol{x}_\ell^{(l)}|x_{\mathfrak{r}}^{(l)}, \theta_l)$, depending on the realization of the rate heterogeneity at site l. From Eq. (5.20) it is clear that higher values of θ_l lead to higher substitution probabilities. The mutual information $I_\theta(\mathsf{X}_{\mathfrak{r}}^{(l)}; \mathbf{X}_\ell^{(l)})$ between $\mathsf{X}_{\mathfrak{r}}^{(l)}$ and $\mathbf{X}_\ell^{(l)}$ depending on the values of θ_l for a five species subtree (including human, chimp, mouse, rat and fugu) derived from the tree shown in Figure 5.2 is also depicted in Figure 5.4. In the following, we shall often skip the site index l.

5.3.2 Models of rate heterogeneity

Different models for the space process $\{\Theta_l\}_{l\geq 0}$ were proposed. Yang [Yan93] was the first to propose an iid model with Θ_l distributed according to a gamma distribution

$$\Theta \sim \theta^{\alpha-1} \frac{e^{-\theta/\beta}}{\beta^\alpha\ \Gamma(\alpha)} \doteq G_\Theta(\theta; \alpha, \beta), \tag{5.21}$$

which was later extended to a *Markov gamma* process with [Yan95]

$$0 < \mathrm{Corr}(\Theta_{l+1}, \Theta_l) = \frac{1}{\sigma_{\Theta_{l+1}}\sigma_{\Theta_l}} E\{(\Theta_{l+1} - E\{\Theta_{l+1}\})(\Theta_l - E\{\Theta_l\})\}, \tag{5.22}$$

and

$$\sigma^2_{\Theta_l} = E\{(\Theta_l - E\{\Theta_l\})^2\}. \tag{5.23}$$

Models assuming $\{\Theta_l\}_{l\geq 0}$ to be Markovian base on the biological observation that neighboring sites of DNA sequences often have correlated rates [Nie97, YW95].

In [Yan94] the increased complexity of calculating the likelihood under such models was reduced by discretizing the continuous r. v. Θ_l into *rate categories*. Introducing transition probabilities among these categories leads to an HMM approach [FC96] where discretized rates represent states of the Markov chain and alignment columns correspond to observations. Recently, Mayrose et al. suggested a gamma mixture model [MFP05] and showed that it provides a better fit to alignment data.

It is clear that additional parameters are required to describe the rate process. For example, under the iid gamma model the parameters α and β in Eq. (5.21) have to be estimated from data. For HMM models, initial and transition probabilities must be learned, using, for example, the EM algorithm as outlined in Chapter 2, Section 2.6.

5.4 Detection of potentially functional elements

Since the completion of the Human Genome Project (HGP) in 2003 that provided a high quality sequence of the human genome, comprehensive identification of biologically functional elements in the DNA represents a central and ambitious goal in modern genomics. The reliable detection and analysis of functional elements is crucial to a deep understanding of how complex organisms work.

Early approaches were limited to the use of information from a single species and exploited certain properties of the functional regions to be detected. Short, recurrent molecular sequences, in genetics often referred to as *motifs*, are good candidates for functional regions such as *promoters* [BST00]. A common approach was to assemble a collection of sequences from a single genome believed to contain the region of interest and search in these sequences for patterns that occur in a statistically significant overabundance.

Rapid progress in whole genome sequencing efforts allowed to suggest an approach taking into account multiple sources of information: it was proposed to assemble a collection of sequences from multiple species looking for regions that are *well conserved* in most or all of the species [Ker99]. The underlying idea is the following: during evolution, the genomic information, represented by DNA, of an organism is passed on to its descendants. DNA is subject to mutations that cause genetic variations, and *natural selection* decides about the success of the transmitted, altered information. In some regions, variation negatively influences the fitness of an organism and diminishes its capability to reproduce, eventually preventing it from passing its DNA to the next generation. Mutations in regions which are not important for the fitness of the organism are passed on to further generations without restrictions. Thus, those elements within the genome carrying information for important basic functions are believed to remain conserved during evolution. We say those regions are under negative selection or constraint. Today, joint analysis of DNA orthologues from multiple species conveys important information about sequence properties. Such a *comparative approach* is a powerful concept in genome analysis today [LSM+03]. DNA sequences with unexpected conservation across species have recently gained particular interest [DRA05, SBP05a, BPM+04].

Throughout this Section the term *conserved* will refer to *primary sequence conservation* among multiple species. Certainly, there are many types of conservation acting at different constraint levels upon the genome. Secondary and tertiary structures as well as interactions of non-coding RNA may be preserved with little primary sequence information remaining conserved [WRT07] (cf. Section 3.1).

The problem of measuring the conservation of sequences across multiple species has been addressed in a number of publications: exact algorithms for the detection of small functional regions (*motifs*) of around 5 to 25 bases were developed in [Bla01, Bla03, SBT04]. Stojanovic et. al. compared 5 different methods for scoring the conservation of a multiple sequence alignments in gene regulatory regions [SFR+99]. Margulies et al. presented two alignment based methods that incorporate phylogenetic information, suitable for whole genome analysis [MBHG03]. Siepel and Haussler introduced a phylogenetic hidden Markov model (phylo-HMM) that allows high throughput measurement of evolutionary constraint (phastCons) [SH05]. Cooper et al. introduced GERP [CSA+05] and more recently Asthana et al. presented SCONE [ARSS07], which both produce per-base scores of conservation and constraint.

Existing genome scale methods require the *a priori* estimation of a neutral evolutionary rate and measure conservation as the "surprise" of observing the analyzed data, assuming the neutral model. Neutral substitution rates are usually estimated from fourfold degenerated sites (the bases in codons that can be changed to any other base without changing the amino acid) or ancestral repeats (replications of DNA segments that happened a long time ago, believed to evolve without evolutionary pressure) [CBS+04, HRY+03]. The ENCODE project[2], however, revealed that about half of the analyzed functional elements found in non-coding DNA had been classified as unconstrained [The07, MCA+07] by existing computational methods. Pheasant and Mattick [PM07], among others, have argued that this could partly be explained by questioning the neutral rate of evolution used by existing methods. Wrong assumptions about the neutral rate would lead to biased conservation measures and eventually to an over- or underestimate of the fraction of the genome under evolutionary constraint. For example, ancestral repeats are often assumed to evolve neutrally, but have been previously shown to include a nontrivial amount of constrained DNA [CSA+05, KXL06].

In the remainder of this chapter, we outline an algorithm for the detection of conserved regions which has several improvements over existing methods allowing for a detailed analysis of conserved DNA elements. Our method can be used to help gaining new insights into the function of still poorly understood conserved non-coding sequences [DBN+06, BPM+04, DRA05].

[2] The ENCODE (**Enc**yclopedia **o**f **D**NA **e**lements) research consortium, launched by the National Human Genome Research Institute in 2003, aims for the identification of all functional elements in the human genome sequence. The pilot phase tested and compared existing computational and experimental methods to rigorously analyze approximately 1% of the human genome sequence. The results of this pilot phase were published in June 2007 in [The07, MCA+07].

Our method - that we call KuLcons - avoids *a priori* assumptions about properties of conserved sequences. We suggest that the maximum likelihood (ML) estimate of rate heterogeneity is a more direct measure for sequence conservation. We obtain the ML estimate of the rate process using an optimized window function, accounting for autocorrelation among rates. While our approach does not require assumptions about neutral rates, prior distribution of rates, or transition probabilities between rate categories, we show *in silico* that reliable estimation in the mean squared error (MSE) sense is achieved in regions of conserved sequence. We present a qualitative comparison of the scores calculated by KuLcons and the established methods phastCons, GERP and SCONE that all assume a neutral model. ENCODE regions were used for comparison.

Furthermore, our method allows for richer or more complex parameter models like those taking insertion and deletion (InDel) rates into consideration. Results of scores accounting for gaps in the alignment as InDels are presented and compared to standard methods.

5.4.1 Identification problem and overview of existing methods

We consider the following problem: given an alignment $\boldsymbol{A} \in \{\mathcal{A}, -\}^{M_s \times L}$, we want to identify the columns $\boldsymbol{a}_l$ in $\boldsymbol{A}$ that are more *conserved* than others. A *conservation estimator* assigns a score $\mathfrak{s}_l$ to the lth column of an alignment reflecting the degree of conservation. An ad hoc approach would simply look at how many different bases occur in one column and assign a high conservation score when the base composition of a single column is very diverse. However, such an approach does not take into account the phylogenetic relationship among the species. Inconsistent bases in very distant species such as, for instance, human and zebrafish should decrease the conservation score less than a base change between human and, let's say, chimp. We call such methods *naive* since they approach the problem without incorporating additional knowledge about the phylogeny. Non-trivial approaches take a model of evolution into account. In the following, we briefly sketch algorithms that compute conservation scores for alignments. We start with naive methods and proceed with methods taking evolutionary models into account. Then, we derive the requirements for an optimal conservation estimator and present our approach that is shown to satisfy these requirements.

Naive approaches

When no model of evolution is assumed, all one can do is to look how diverse the base composition in an alignment column appears. Let $\boldsymbol{f}_l$ be the frequency count of bases in the lth column with the ith entry in $\boldsymbol{f}_l$ given by

$$f_l^{(i)} = \frac{1}{M_s} \sum_{j=1}^{M_s} (1 - d_H(\boldsymbol{a}_l^{(j)}, i)), \ i = A, C, G, T \tag{5.24}$$

where $\boldsymbol{a}_l^{(j)}$ denotes the jth element in the alignment column $\boldsymbol{a}_l$ at position l. From the vector $\boldsymbol{f}_l$, a real valued score can naturally be derived by calculating the empirical entropy of that column

$$\mathfrak{s}_l = 2 - \hat{H}_{\text{emp}}(\boldsymbol{f}_l) = 2 + \sum_{i \in \mathcal{A}} f_l^{(i)} \log_2(f_l^{(i)}). \tag{5.25}$$

The score is 0 when $\boldsymbol{f}_l$ is uniform and 2 if the column is fully conserved (consists only of one type of base). Conservation scores based on the empirical entropy as described above were suggested in [SDM+05].

In order to account for the base composition of the alignment, divergence based approaches estimate the background distribution $\hat{\boldsymbol{\pi}}$ of the alignment according to Eq. (5.3) and use a pdf divergence measure such as the Kullback-Leibler divergence (Def. 2.2.5):

$$\mathfrak{s}_l = \mathcal{D}(\boldsymbol{f}_l || \hat{\boldsymbol{\pi}}). \tag{5.26}$$

The score is greater or equal zero with equality iff $\boldsymbol{f}_l = \hat{\boldsymbol{\pi}}$. Note that this score is equivalent to Eq. (5.25) if $\boldsymbol{\pi}$ is the uniform distribution. In [CS07], the Jensen-Shannon (JS) divergence was suggested as a measure for the detection of functional amino acid residues (note that for amino acids the problem setting doesn't change except one uses a different alphabet). The JS divergence is a symmetrized version of the KL divergence [Lin91],

$$\mathcal{JS}(p||q) = \frac{1}{2}\mathcal{D}(p||p^*) + \frac{1}{2}\mathcal{D}(q||p^*), \; p^* = \frac{1}{2}(q+p). \tag{5.27}$$

The advantage of the JS divergence is that it is symmetric and bounded between zero and one. It is zero if and only if $\boldsymbol{f}_l = \hat{\boldsymbol{\pi}}$.

Methods taking phylogeny into account

For the remainder of this chapter, we consider extended models of evolution, i.e., we take rate heterogeneity among sites into account. Evolution at site l is then modeled by the set of parameters

$$\boldsymbol{\psi} = \{\mathcal{T}, \tau, \boldsymbol{R}, \boldsymbol{\pi}, \theta_l\}, \tag{5.28}$$

where θ_l is the realization of the rate variation process $\{\Theta_l\}_{l \geq 0}$ at position l. As before, we often skip the index l in the following.

phastCons

Siepel et al. developed phastCons, based on the concept of phylogenetic hidden Markov models (phylo-HMMs) [SH05]. Phylo-HMMs are hidden Markov models whose states are associated with different evolutionary models $\boldsymbol{\psi}$ [SH04]. A phylo-HMM probabilistically generates a multiple alignment, column by column, such that each column is defined by the phylogenetic model corresponding to the state in which the HMM currently resides. When moving to the next column, a new $\boldsymbol{\psi}$ is chosen randomly, conditional on the $\boldsymbol{\psi}$ in the previous step. A four state phylo-HMM is depicted in Figure 5.5. The symbols emitted by the HMM are alignment columns with emission probabilities $p(\boldsymbol{a}|s_j) \equiv p(\boldsymbol{a}; \boldsymbol{\psi}_j)$.

In the framework of conservation estimation, a phylo-HMM is reduced to two states denoted by $\boldsymbol{\psi}_c$ and $\boldsymbol{\psi}_n$. The phylogenetic model $\boldsymbol{\psi}_c$ is supposed to describe positions

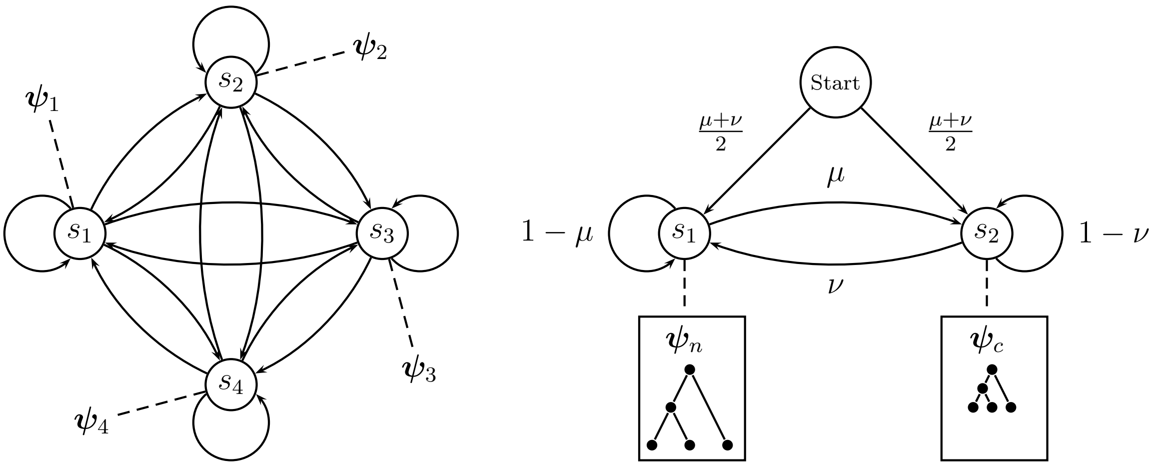


Figure 5.5: Left: a phylogenetic hidden Markov model (phylo-HMM). Upon initiation, a phylo-HMM randomly jumps into a state according to the initial state probabilities. At each step, the phylo-HMM emits an alignment column drawn from the distribution associated with the current state. Then, it randomly jumps into another state according to the state-transition probabilities. Right: A phylo-HMM setup for conservation estimation with two states representing conserved and non-conserved.

in the genome that are under purifying selection, i.e., positions that will be conserved during evolution. The model $\boldsymbol{\psi}_n$ represents genome regions that evolve neutrally. The models $\boldsymbol{\psi}_n$ and $\boldsymbol{\psi}_c$ are identical except for the rates θ. Note that the HMM is then equivalent to a single phylogenetic model with the rate variation process $\{\Theta_l\}_{l\geq 0}$ modeled as a two state discrete Markov process. Two parameters μ and ν define state-transition and initial state probabilities as shown in Figure 5.5. These are tuning parameters that are related to the expected coverage and expected length of conserved elements. The expected length parameter $\omega = \frac{1}{\mu}$ is given by the expected number of steps for which the Markov chain will remain in the conserved state. The expected coverage $\gamma = \frac{\nu}{\mu+\nu}$ is the expected percentage of conserved columns in an alignment at equilibrium. A detailed discussion about these parameters and how they relate to properties of conserved regions is given in [SBP$^+$05b]. It is important to note that γ and ω are *a priori* rather than a posteriori quantities. They define properties of the model which influence final (posterior) inferences. While full maximum likelihood estimation of μ and ν may seem to be an ideal way, it does not work well in practice (according to the phastCons HOWTO available at http://www.soe.ucsc.edu/~acs/PhastCons-HOWTO.html, June 2005). Thus, μ and ν may really be seen as tuning parameters.

Given the alignment $\boldsymbol{A}$, the conservation score $\mathfrak{s}_l$ assigned to column $\boldsymbol{a}_l$ is the posterior probability that the HMM was in the conserved state at site l

$$\mathfrak{s}_l = p(s_l = \boldsymbol{\psi}_c | \boldsymbol{A}), \tag{5.29}$$

and is computed using the forward-backward algorithm described in Section 2.4.5.

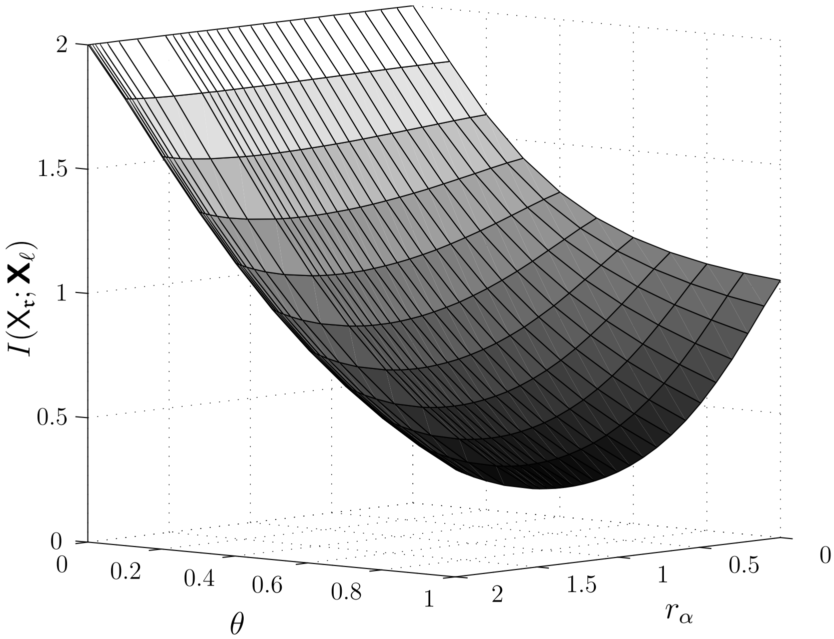


Figure 5.6: Mutual information over θ and rate parameter r_α of a substitution matrix under the K2P model. Because of the normalization constraint of the substitution matrix, r_α cannot exceed 2.

Likelihood and Bayesian methods

Given a fixed phylogenetic tree with branch lengths $\{\mathcal{T}, \tau\}$, the rate variation θ is the crucial parameter for detecting evolutionary conservation. An ideal measure of conservation is the mutual information between the common ancestor nucleotide $\mathsf{X}_\mathfrak{r}$ and the nucleotides $\mathbf{X}_\ell$ at the leaves of a tree. Figure 5.6 shows the MI over θ and the rate parameter r_α of a substitution matrix under the K2P model (cf. Section 4.1). A five species subtree (including human, chimp, mouse, rat and fugu) derived from the tree shown in Figure 5.2 was used. Note that the second parameter r_β in the K2P model is fixed due to the $\mathrm{trace}(\mathbf{\Pi}\boldsymbol{R}) = -1$ constraint. For the same reason, r_α cannot exceed 2. The MI is shown for $\theta \in [0, 1]$. We observe that over a large range of θ ($\approx$ 0..0.5), the MI is greatly influenced by θ but does not vary with r_α. Furthermore, for all reasonable values of r_α, the MI is mainly influenced by the rate variation. Significant changes are only observed when θ is high and at very low values of r_α, i.e., an extremely low probability of transversions (change from Purine to Pyrimidine and vice versa) which is usually not observed in reality[3].

The finding is intuitive: the substitution process describes the affinity of a base to mutate to another base. However, for the mutual information, and hence for measuring conservation, the quality of mutations (e.g., whether A is more likely to mutate to C than to T) is irrelevant. The rate variation describes the expected quantity of mutations and is therefore the important parameter for mutual information. Conservation scores should therefore have a high correlation with this parameter because it is directly related to the

[3]When $r_\alpha = 0$, then the transversion probability is 0 and the channel model basically reduces to a binary alphabet. As θ gets large the mutual information therefore approaches 1 for $r_\alpha = 0$.

mutual information between common ancestor base and observed nucleotides.

In the following, we discuss estimators for θ under different models for rate variation. Figure 5.7 shows the simulated log likelihood function $ll_{\boldsymbol{A}}(\theta)$ for an alignment $\boldsymbol{A}$ of length 50. Alignments were sampled based on the tree in Figure 5.2 using the rate matrix in Eq. (5.11) with different rates $\theta_s \in [0, 0.3]$ constant over the alignment. The log likelihood is shown in the interval $\theta \in [0, 1.5]$. For each simulated θ_s, the likelihood function was averaged over 1000 sample alignments. Figure 5.7 also shows the one dimensional likelihood functions for a single alignment of length 50 generated under $\theta_s = 0.2$ and $\theta_s = 0.8$.

The likelihood function is calculated using the Felsenstein algorithm (Section 4.2.2) and has a single maximum with respect to θ. Message passing can be extended to calculate the gradient of $ll_{\boldsymbol{A}}(\theta)$. For a single column $\boldsymbol{a}$ in the alignment:

$$\frac{\partial}{\partial\theta} l_{\boldsymbol{a}}(\theta) = \frac{\partial}{\partial\theta} \boldsymbol{p}^{(\mathfrak{r})} \boldsymbol{\pi}^T. \tag{5.30}$$

For each node u with children v, w, we calculate

$$\frac{\partial}{\partial\theta} \boldsymbol{p}^{(u)} = (\frac{\partial}{\partial\theta} \boldsymbol{m}^{(v)} \odot \boldsymbol{m}^{(w)}) + (\boldsymbol{m}^{(v)} \odot \frac{\partial}{\partial\theta} \boldsymbol{m}^{(w)}) \tag{5.31a}$$

$$\frac{\partial}{\partial\theta} \boldsymbol{m}^{(v)} = t_{uv} \boldsymbol{p}^{(v)} \boldsymbol{R}^T \boldsymbol{P}^T_{u\to v} + \frac{\partial}{\partial\theta} \boldsymbol{p}^{(v)} \boldsymbol{P}^T_{u\to v}. \tag{5.31b}$$

The MLE of the rate variation θ is given by

$$\hat{\theta}_{\text{MLE}} = \arg\max_{\theta\in\mathbb{R}^+} \{ll_{\boldsymbol{A}}(\theta)\}, \tag{5.32}$$

where θ is treated as a parameter. When sites are assumed to evolve independently:

$$\hat{\theta}_{\text{MLE}} = \arg\max_{\theta\in\mathbb{R}^+} \left\{ \sum_i ll_{\boldsymbol{a}_i}(\theta) \right\}. \tag{5.33}$$

Depending on the model of rate heterogeneity, different estimators for θ were proposed (Table 5.1): likelihood methods do not assume a prior distribution and treat θ as a deterministic parameter. Under the discrete rates model, the likelihood for each rate is evaluated and the one returning the highest value is chosen. Estimators for continuous θ use the message passing approach for finding the MLE using numerical optimization techniques (hill-climbing methods) [Nie97]. Bayesian approaches treat θ as realization of a random variable with prior distribution. The gamma distribution is most commonly used in this case [Yan96]. It is well known that the conditional mean estimator (CME) is optimal in the mean squared error (MSE) sense under this assumption. Consider the observation $\boldsymbol{x}_\ell$, the CME is given by

$$\hat{\theta}_{\text{CME}} = E_\theta\{\theta|\boldsymbol{x}_\ell\} = \frac{\int_0^\infty \theta\, p(\boldsymbol{x}_\ell|\theta)p(\theta)\mathrm{d}\theta}{\int_0^\infty p(\boldsymbol{x}_\ell, \theta)\mathrm{d}\theta}. \tag{5.34}$$

Under the discrete rates model, the rate θ takes values with probabilities $p(\theta = \theta_k) = p_k$. The integrals in (5.34) turn into sums, and one may calculate the posterior mean or use

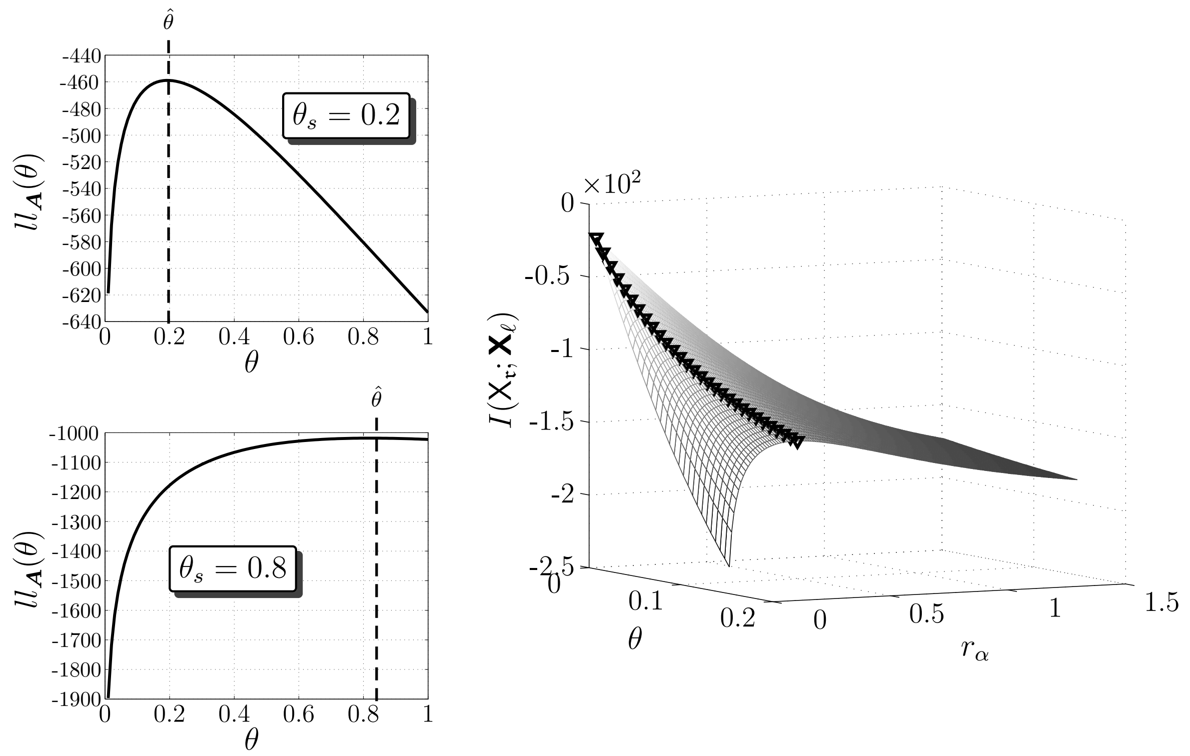


Figure 5.7: Left: Likelihood functions for an alignment of length 50 generated with $\theta_s = 0.2$ and $\theta_s = 0.8$. Right: Simulated log-likelihood function varying with θ for different simulated θ_s. The maximum value for each θ_s is indicated by triangles on the likelihood surface.

	discrete iid	continuous iid	discrete w auto-correlation
Likelihood	$\hat{\theta}_{\text{MLE}} = \underset{k\in\{0,..,K\}}{\arg\max}\{p(\boldsymbol{x}_\ell;\theta_k)\}$	$\hat{\theta}_{\text{MLE}} = \underset{\theta\in\mathbb{R}^+}{\arg\max}\{p(\boldsymbol{x}_\ell;\theta)\}$	/
Bayesian	discrete CME/MAP	$\hat{\theta}_{\text{CME}} = E_\theta\{\theta\vert\boldsymbol{x}_\ell\}$	HMP: MAP using forward backward

Table 5.1: Estimators for different models of rate heterogeneity.

the rate with highest posterior probability (maximum a posteriori probability (MAP)). The CME for substitution rates under these models was derived by Yang in [YW95]. When discrete rates with correlation is assumed, we obtain an HMP [FC96] with states corresponding to rates and alignment columns corresponding to observations. The posterior state probabilities are estimated using the forward-backward algorithm (phastCons, as described above, belongs to this class of approaches).

5.4.2 KuLcons

Having reviewed existing methods for scoring conservation, the remainder of this section presents our method focusing on the algorithmic details. Results involving simulations and application to biological data are presented in Section 5.5 followed by a detailed discussion in Section 5.6.

Our goal was to address the following issues:

- ⋆ The method should be probabilistic, taking the phylogeny among species into account using the established models of evolution.
- ⋆ Since we identified the rate variation process $\{\Theta_l\}_{l\geq 0}$ as the crucial parameter in $\boldsymbol{\psi}$ responsible for conservation, we aim for a high correlation with this process. The topology of the tree and the branch lengths τ greatly influence conservation. However, we can assume these parameters not to change considerably over alignment columns. Furthermore, due to their richness of parameters, they have to be estimated from a large amount of data and are usually fixed by their estimates for further analysis.
- ⋆ As few *a priori* assumptions as possible should be made about the rate variation process (such as degree of autocorrelation, distribution, mean length of conserved element etc.). We therefore aim for a prior free likelihood approach. To our knowledge, the estimation of the rate heterogeneity process under autocorrelation in the continuous case was not analyzed before.
- ⋆ In particular, no assumptions about neutral rates shall be made, which is motivated by the conjecture of Mattick et. al. that these assumptions may lead to biased estimates overlooking many conserved regions [PM07].
- ⋆ The method should be extendable to richer models of evolution such as those taking insertion and deletion events into account.

In the framework of conservation estimation, a subset of parameters in $\boldsymbol{\psi}$ will be fixed over the alignment. For example, it is reasonable to assume that the topology $\mathcal{T}$ and the branch lengths τ do not change considerably over alignment columns. These parameters as well as the stationary distribution $\boldsymbol{\pi}$ are generally replaced by their ML estimates from a large data set. We use $\hat{\boldsymbol{\psi}}$ to refer to the estimate of the remaining free parameters in $\boldsymbol{\psi}$.

We use a sliding window approach to increase reliability of estimates and to take autocorrelation into account, meaning that we consider a section of the alignment

$$\boldsymbol{A}_{l-\delta}^{l+\delta} = [\boldsymbol{a}_{l-\delta}, \ldots, \boldsymbol{a}_{l+\delta}],\ l = \delta+1, \ldots, l-\delta \tag{5.35}$$

of length $2\delta + 1$ around the column of interest to estimate the parameters[4]. A window function $w[n]$ assigns weights to the likelihoods of neighboring columns. For example, a uniform weighting scheme could be chosen with $w[n] = 1,\ n = 0, ..., 2\delta$. However, it is well known that the choice of window functions with good spectral properties can significantly improve estimation of autocorrelated processes. Several types of window

[4]For convenience, we restrict ourself to odd window sizes. The generalization to even window sizes is trivial.

functions optimized with respect to different properties exist, e.g., the Hamming-, Kaiser- or Gauss-window. The Gauss-window, for example, is given as

$$w[n] = \begin{cases} e^{-\frac{1}{2}\left(\frac{n-\delta}{\sigma_W(n-\delta)}\right)^2} & n = 0, ..., 2\delta \\ 0 & \text{else} \end{cases} . \tag{5.36}$$

The resulting estimator is then given by

$$\hat{\boldsymbol{\psi}}_l = \arg\max_{\boldsymbol{\psi}} \left\{ \sum_{n=l-\delta}^{l+\delta} w[n-l+\delta] \, \log\left(p(\boldsymbol{a}_n; \boldsymbol{\psi})\right) \right\}, \tag{5.37}$$

yielding a set of parameters describing the local evolutionary process for the data in the window.

In order to obtain a scalar conservation score from the estimated parameters $\hat{\boldsymbol{\psi}}_l$, we consider the probability mass function (pmf) $p(\boldsymbol{a}; \hat{\boldsymbol{\psi}}_l)$ of an alignment column $\boldsymbol{a}$ under the model $\hat{\boldsymbol{\psi}}_l$. Avoiding assumptions about the neutral evolutionary rate, we compare the estimated distribution to the distribution of the well defined absolute conservation, parameterized by the imaginary set of parameters $\boldsymbol{\psi}^0$ that does not allow for any substitution to occur, i.e.,

$$p(\boldsymbol{a}; \boldsymbol{\psi}^0) = \begin{cases} \pi_b & \text{if } \forall (i,j) : [\boldsymbol{a}]_i = [\boldsymbol{a}]_j = b \in \mathcal{A}, \\ 0 & \text{else} \end{cases} . \tag{5.38}$$

A measure for the divergence between two probability mass functions is the Kullback-Leibler (KL) divergence which was introduced in Section 2.2.1. Our conservation score function is given by

$$\mathfrak{s}_l = \mathcal{D}(p(\boldsymbol{a}; \boldsymbol{\psi}^0) || p(\boldsymbol{a}; \hat{\boldsymbol{\psi}}_l)). \tag{5.39}$$

As we measure the divergence to the maximum conservation, low score values indicate high conservation. Note that $p(\boldsymbol{a}; \boldsymbol{\psi}^0)$ is equal to zero whenever $\exists (i,j) : [\boldsymbol{a}]_i \neq [\boldsymbol{a}]_j$, i.e., it is only nonzero for fully conserved columns. That is, in order to evaluate Eq. (5.39) we only have to consider the four columns having only As, Cs, Gs or Ts which are the only possible realizations of maximum conservation. Let $\boldsymbol{a}_{[b]}$ denote a fully conserved column of nucleotide b, i.e., $\boldsymbol{a}_{[b]} = [b, b, ..., b]^T, \; b \in \mathcal{A}$. Then, $p(\boldsymbol{a}_{[b]}; \boldsymbol{\psi}^0) = \pi_b$ under the maximum conserving model and we can rewrite Eq. (5.39) as

$$\sum_{b \in \mathcal{A}} \pi_b \log \frac{\pi_b}{p(\boldsymbol{a}_{[b]}; \hat{\boldsymbol{\psi}}_l)} = H(\boldsymbol{\pi}) - E\{\log(p(\boldsymbol{a}_{[b]}; \hat{\boldsymbol{\psi}}_l))\}. \tag{5.40}$$

Our algorithm works as follows: given an alignment $\boldsymbol{A}$, we choose a suitable window type and fix the size of our sliding window by choosing a suitable δ. Then we obtain the local ML estimate $\hat{\boldsymbol{\psi}}_l$ over $\boldsymbol{A}_{l-\delta}^{l+\delta}$ according to (5.37) by message passing (FA) and numerical maximization using a Newton method. The estimate is projected to a score via Kullback-Leibler divergence according to (5.39) and assigned to the column $\boldsymbol{a}_l$. The sliding window is shifted forward, increasing l by 1 and the procedure is repeated until l reaches $L - \delta$. A score $\mathfrak{s}_l$ is now assigned to every alignment column $\boldsymbol{a}_l$ (scores at the borders of the alignment can be obtained by setting $p(\boldsymbol{a}; \boldsymbol{\psi}_l) = 1$ for $l < 1$ and $l > L$).

5.4.3 Run time of the algorithm

Even though the run time of our algorithm is significantly higher than the computation times achieved by algorithms designed for high throughput analysis such as phastCons, our method is still feasible for assaying whole genome alignments. Using a single standard Linux PC (2Gb RAM, 2.4GHz) it was possible to calculate the scores for the human (hg18) reference 28-species alignment from UCSC Genome Browser [KBD+03] in less than one month. The complexity of the algorithm scales linear with the length of the alignment and linear with the number of inner nodes in the inspected phylogenetic tree. The latter is explained by the complexity of the Felsenstein algorithm that has to visit every node in the tree where the same update function is evaluated.

5.5 Results

5.5.1 Small sample sizes and error distribution

Since the proposed ML estimate is based on a relatively small sample size, we study whether the ML approach is optimal in this case by comparing the density of the estimated rate variation $\hat{\theta}$ to the theoretically achievable density. We assume all parameters in $\boldsymbol{\psi}$ to be fixed except for θ_l. We shall check whether the MLE attains the Cramér-Rao lower bound

$$E\left\{(\hat{\theta}-\theta)^2\right\} \geq -E\left\{\frac{\partial^2 \log p(\boldsymbol{x};\boldsymbol{\psi})}{\partial\theta^2}\right\}^{-1} = \frac{1}{I(\theta)}, \tag{5.41}$$

in the considered small sample size setting. An unbiased estimator that attains the CRLB is a *minimum variance unbiased estimator* and no other estimator has a lower variance. As discussed in Section 2.5, the MLE asymptotically achieves this bound for large sample sizes, i.e., $\hat{\theta} \overset{a}{\sim} \mathcal{N}(\theta, I(\theta)^{-1})$. We performed a computer simulation using $100,000$ realizations of alignments of length $(2\delta+1)$, generated according to a fixed evolutionary model $\boldsymbol{\psi}$. We estimated $\hat{\theta}$ and computed $I(\theta)$ for each sample.

Figure 5.8 shows the theoretical achievable pdfs $\mathcal{N}(\theta, I(\theta)^{-1})$ versus the observed pdfs of $\hat{\theta}$ for different simulated θ_s. Even for small window sizes, e.g., $\delta = 7$, the MLE closely approaches its asymptotic distribution. At low values of θ, the variances are relatively small, i.e., different values of θ can be distinguished with high probability. It can also be observed that the variance of the estimation increases with increasing θ. Hence, our estimator is best discriminating between different degrees of conservation in relatively conserved regions even at small window sizes, whereas in non-conserved regions, the information revealed by the window is not enough to allow for precise differentiation.

We propose an estimation model for θ with a multiplicative error as follows

$$\hat{\theta} = (1+\eta)\theta, \tag{5.42}$$

where $\eta \sim \mathcal{N}(0, \sigma_\eta^2)$ is a normally distributed random variable. The variance of the estimate $\hat{\theta}$ then depends on its mean and higher values will have a higher variance such

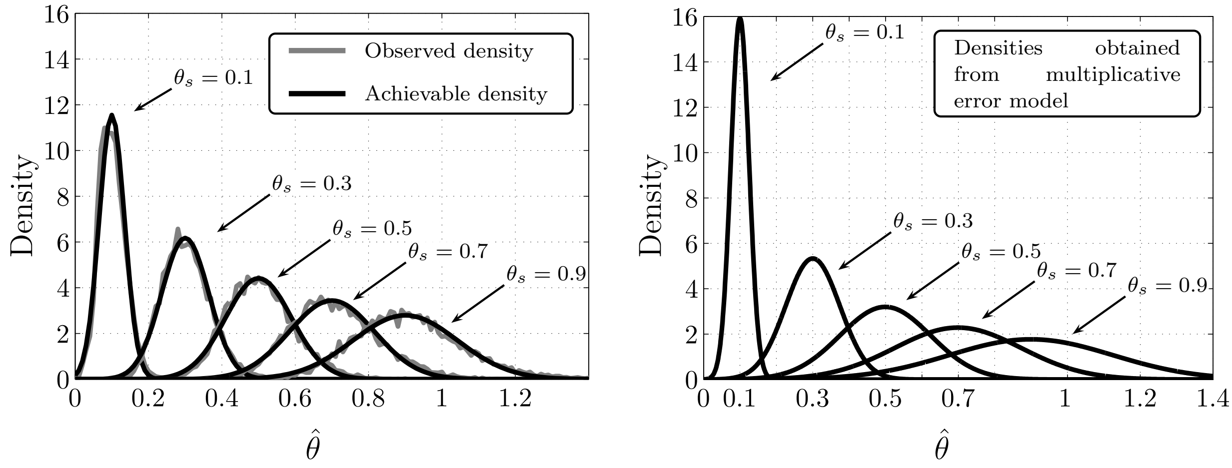


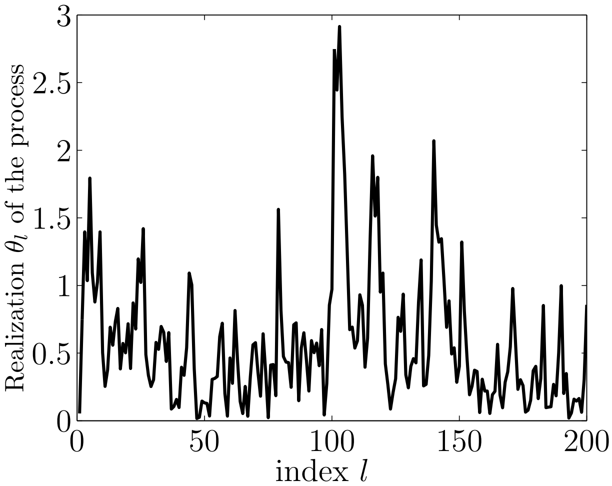
Figure 5.8: Left: observed and theoretical pdfs of estimates $\hat{\theta}$ for $\delta = 7$ simulated at different θ_s. Simulation are based on 100000 realizations of alignments of length $(2\delta + 1)$, sampled according to a fixed evolutionary model $\boldsymbol{\psi}$ based on the phylogenetic tree in Fig. 5.2. Right: distribution of rate heterogeneity estimates under a multiplicative error model.

as observed in Figure 5.8. The best fitting variance σ_η^2 can be determined via simulations on synthetic data. A simulation of the multiplicative model is also shown in Figure 5.8, demonstrating that it provides a good fit to the distribution of estimates obtained from the simulated genomic data.

5.5.2 Sliding window estimation of a Markov gamma process

In this Section, we show via simulations of synthetic data generated *in silico* that our approach is well suited for the estimation of the rate heterogeneity process under a Markov gamma model. As discussed in Section 5.3.2, iid and Markov, continuous and discrete space models were proposed for the rate process $\{\Theta_l\}$ among sites [Yan06, FC96]. In the continuous case, the stationary distribution of $\{\Theta_l\}$ is commonly assumed to follow a gamma distribution (cf. Eq. (5.21)) [Yan95]. Correlation among sites is introduced to account for the fact that neighboring sites are likely to experience similar substitution rates [YW95, Nie97].

Simulation model: In the context of conservation measurement, the estimator is not required to give reliable results on the whole spectrum of possible rates, but to provide a good estimate for the degree of conservation of a region. The situation that we simulate mimics a moderately conserved region with "islands" of more or less conservation due to variance and autocorrelation of the rate. A good conservation estimator takes into account autocorrelation among sites while retaining the sensitivity of reporting variability within regions. Using a Markov gamma rate model, we generated alignment columns and estimated the rates using site-by-site ML estimation and the sliding window ML procedure described in Section 5.4.2. Simulation of Markov gamma processes was performed as described by Moran [Mor69] and Phatarfod [Pha87]. The process $\{\Theta_l\}$ has distribution



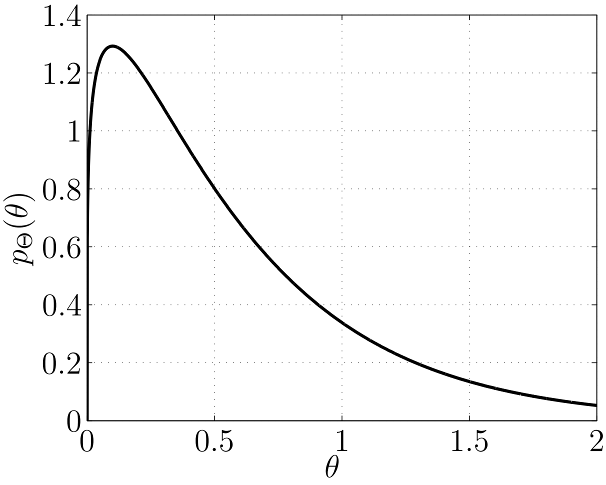


Figure 5.9: Left: A typical sample path of the process $\{\Theta_l\}_{l=0}^{200}$ with marginal $G_\Theta(\theta, 1.2, 0.5)$ and $\rho_\theta = 0.7$. Right: Probability density $G_\Theta(\theta, 1.2, 0.5)$ of θ that was used in the simulation.

$G_\Theta(\theta_l; 1.2, 0.5)$ and autocorrelation

$$\rho_\theta \doteq \frac{1}{\sigma_{\Theta_{l+1}}\sigma_{\Theta_l}} E\{(\Theta_{l+1} - E\{\Theta_{l+1}\})(\Theta_l - E\{\Theta_l\})\},\ \rho_\theta \in [0,1], \tag{5.43}$$

among sites. Analysis of substitution rates in biological data sets revealed that θ is mostly in the range $[0, 1]$. For the chosen parameters in this simulation, 80% of the θ_l are expected to fall in this interval. We simulate an overall moderately conserved region ($E\{\Theta\} = 0.6$) with varying conservation inside, which is modeled by the rate variance ($\mathrm{VAR}\{\Theta\} = 0.3$), and different degrees of autocorrelation. In Figure 5.9 a typical realization of the rate process $\{\Theta_l\}_{l=1}^{L}$ is shown for $L = 200$ with the parameters described above and $\rho_\theta = 0.7$, revealing several regions with different degrees of substitution rates. Alignment columns were simulated under the described model on a subtree of the 28 species ENCODE tree, comprising 18 species.

Simulation results of rate process estimation using sliding window ML: The true simulated θ is compared to its estimate $\hat{\theta}$ obtained by the different methods. In Figure 5.10 two performance measures are shown, the MSE and $\mathrm{Corr}(\theta, \hat{\theta})$, for different window types over the range of among site rate autocorrelation ρ_θ. The sample correlation (Pearson correlation coefficient) between estimated and true values was calculated as

$$\mathrm{Corr}(\hat{\theta}, \theta_l) = \frac{L\sum_{l=1}^{L}\theta_l\hat{\theta}_l - \sum_{l=1}^{L}\theta_l\sum_{l=1}^{L}\hat{\theta}_l}{\sqrt{L\sum_{l=1}^{L}\theta_l^2 - (\sum_{l=1}^{L}\theta_l)^2}\sqrt{L\sum_{l=1}^{L}\hat{\theta}_l^2 - (\sum_{l=1}^{L}\hat{\theta}_l)^2}}. \tag{5.44}$$

For site-by-site ML estimates we restricted the maximum value of $\hat{\theta}$ to 3 because it was reported by Nielsen that estimates of highly variable columns tend to go to infinity [Nie97]. Around 99% of θ will have values lower 3 under the assumed gamma distribution. Choosing different maximum values had minor effects on the results.

The best MSE is achieved with the Gauss window of variance 0.2 (Eq. (5.36) with $\sigma_w = 0.2$) in the complete range of ρ_θ. As expected, for very slowly changing rates

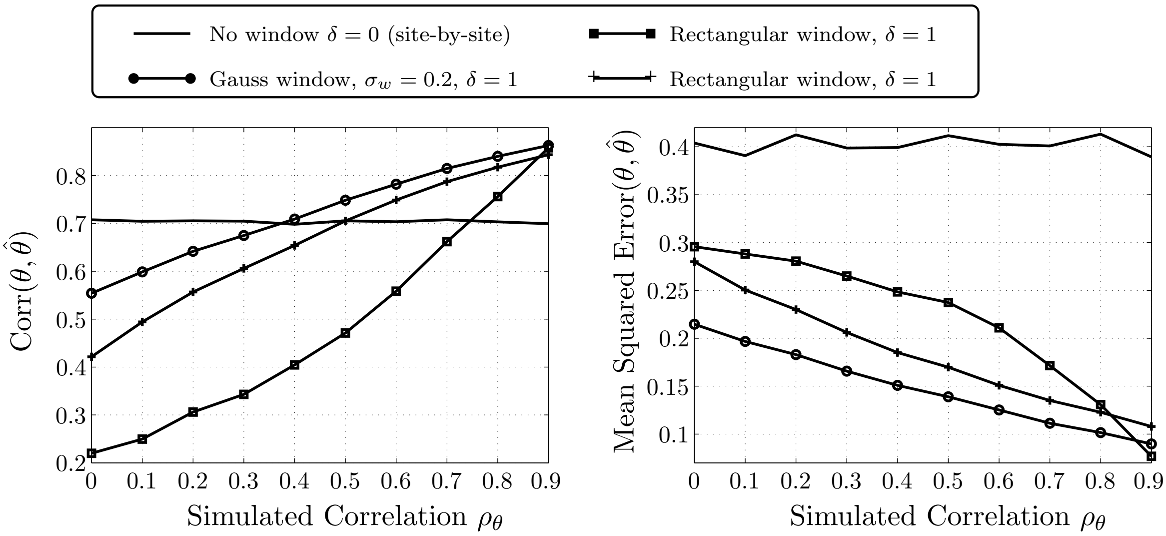


Figure 5.10: Performance of ML estimation of a Markov gamma process using different window functions. Left: correlation between true (θ) and estimated ($\hat{\theta}$) rate. Right: mean squared error.

($\rho_\theta = 0.9$) the performance coincides with the large rectangular window. Interestingly, for uncorrelated sites, the large Gauss window clearly gives the best results, outperforming the small rectangular window and site-by-site estimation. Apparently, even though the window introduces a bias, the error variance is reduced leading to an overall performance improvement. The minimum MSE is achieved for all simulated ρ_θ. The highest correlation is achieved for $\rho_\theta > 0.4$. The results suggest that the method is very well suited for estimating θ with unknown prior distribution and with arbitrary autocorrelation among adjacent sites. We found that other types of window functions did not significantly change the performance, and therefore use the Gauss window for subsequent analysis.

5.5.3 *In silico* comparison of methods

Our analysis based on mutual information suggested that optimal scores should have a high correlation with the rate variation θ. Figure 5.11 shows the result of an *in silico* analysis comparing KuLcons with scores produced by phastCons, and by two naive methods not taking phylogeny into account. The empirical correlation between column-by-column scores and true values of θ_l that were used to generate alignment columns is chosen as a performance measure.

Our simulation framework is based on the assumption that biological data is highly inhomogeneous. We therefore used a mixed gamma process to generate alignments, switching among different distributions and degrees of autocorrelation for different alignment sections: alignments $\boldsymbol{A}_i$ of different lengths L_i were generated according to one of the distributions shown in Figure 5.12 with autocorrelation ρ_θ randomly chosen from [0..0.9]. The distributions account for highly conserved, moderately conserved, and unconserved regions, respectively. Each alignment $\boldsymbol{A}_i$ was generated using a randomly selected distribution and autocorrelation. Lengths L_i of alignment sections were varied uniformly and

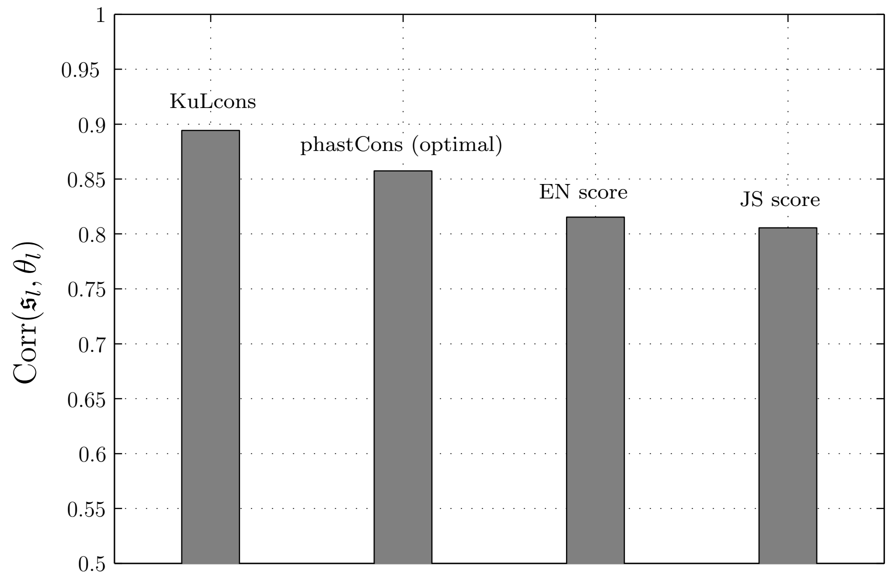


Figure 5.11: Result of an *in silico* analysis comparing KuLcons with scores produced by phastCons and by two naive methods not taking phylogeny into account. A mixed gamma process was used to generate alignments, switching among different distributions and degrees of autocorrelation for different sections in the alignment.

at random between 10 and 100. Alignments $\boldsymbol{A}_i$ were appended until the overall alignment length exceeded 1000 columns.

Conservation scores were calculated for this alignment as follows: KuLcons used a Gauss window with $\sigma_w = 0.2$, phastCons scores were obtained using different models $\boldsymbol{\psi}_c$ and the scores yielding the *overall highest correlation* were selected. The performance of phastCons shown in the Figure therefore represents an upper bound on the phastCons performance, when the optimal tuning is known. The entropy scoring (EN score, Eq. (5.25)) and the score based on the Jensen-Shannon divergence (JS score, Eq. (5.27)) suggested in [CS07] are used as a comparison to scores that neglect phylogenetic information among sequences.

Our approach outperforms all methods in terms of the correlation measure. Even though the entropy and Jensen-Shannon approaches do not take an evolutionary model into account, their performance is not much worse than phastCons's. The entropy approach slightly outperforms the Jensen-Shannon based measure.

Note, however, that for phastCons and KuLcons it was assumed that perfect information about the evolutionary model $\boldsymbol{\psi}$ is available. A situation which is of course unrealistic since estimates about the phylogeny of species from biological data are prone to uncertainties.

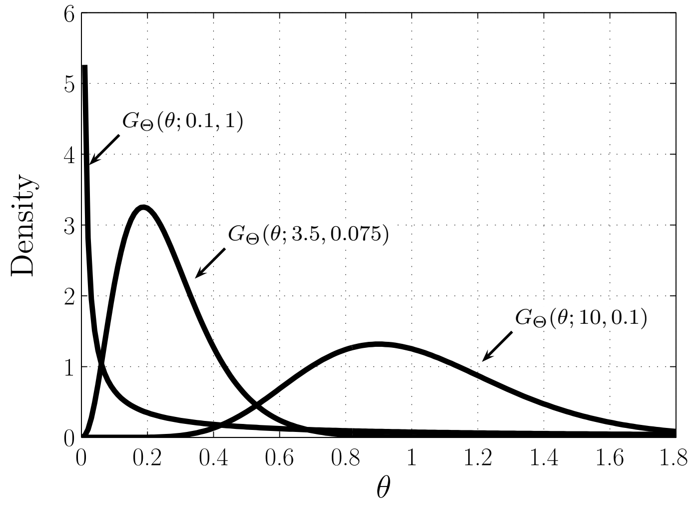


Figure 5.12: The distributions for θ used in the simulation based performance comparison of conservation scoring schemes. The distributions are chosen such as to account for highly conserved, moderately conserved, and unconserved regions, respectively.

5.5.4 Application to ENCODE data

In order to evaluate our conservation score qualitatively, we applied our algorithm to the ENCODE data available from UCSC genome browser [KBD+03]. Three different conservation measures are available at the browser's *conservation track* for ENCODE data: phastCons, GERP and SCONE. GERP and SCONE are likelihood based methods that are briefly explained in Section 5.6. All three algorithms depend on the calibration of neutral substitution rates. Figure 5.13 compares KuLcons scores to the scores produced by phastCons over a 200 bp nucleotide sequence alignment in an ENCODE region (ENm005). Comparison with the scores produced by GERP and SCONE are provided in Appendix C. In order to facilitate the comparison, we show a transformed version of our score, that is,

$$1 - \frac{\mathfrak{s}_l}{\max_l\{\mathfrak{s}_l\}}, \tag{5.45}$$

where $\mathfrak{s}_l$ denotes the conservation score as derived in Section 5.4.2. This has the effect that 1 represents the highest and zero the lowest possible conservation, which is already the case for phastCons scores. The transformation serves solely visualization purposes to ease the qualitative comparison. We would like to emphasize that, while scores are normalized to be in the interval $[0, 1]$, only qualitative comparison is possible since different scores are based on different models (c.f. Section 5.6). For the calculation of our score, all parameters in $\boldsymbol{\psi}$ were replaced by estimates except the rate heterogeneity parameter θ. We used the global average rate matrix $\boldsymbol{R}$ (non-conserved) published by Siepel et. al. [SBP05a] as given in Eq. (5.11). However, using different realistic matrices had minor impact on the scores which is in accordance with previously published observations [CSA+05, YW95] and our own analysis discussed earlier. Single base resolution results in highly varying scores among columns. One can suggest that functional units, such as binding sites, are constrained at least over several neighboring base pairs. For comparison, we therefore applied the same window function as for KuLcons to smooth SCONE and GERP scores.

It can be observed in Figure 5.13 that our score signal is in good agreement with the conservation estimate obtained by visual inspection of the multiple sequence alignment. The phastCons signal shows a binary characteristic and does not allow for discrimination among different conservation degrees. Consequently, phastCons shows a relatively rough-scale pattern of conservation which is different from the pattern revealed by KuLcons. This is explained by phastCon's underlying two-state phylo-HMM model (cf. Section 5.4.1). Interestingly, the smoothed GERP and SCONE scores (cf. Appendix C) show a very similar characteristic to KuLcons with still some notable exceptions: in the region around $31-37$ KuLcons and GERP indicate a relatively weak conservation while SCONE indicates higher conservation. On the other hand, KuLcons and SCONE both indicate higher conservation around $87-93$ while GERP deviates significantly indicating weaker constraint. A different pattern can be observed in region $161-167$ with KuLcons being intermediate.

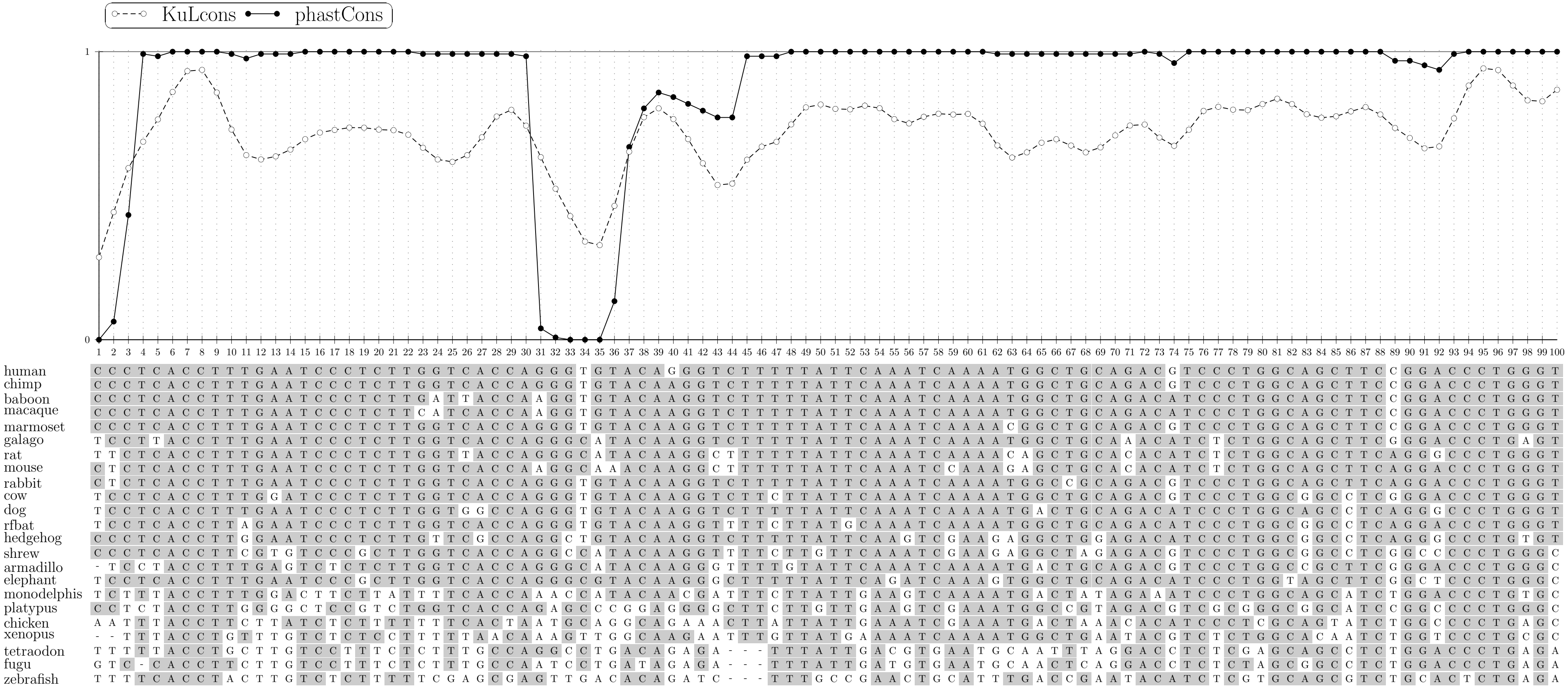


Figure 5.13: Comparison of KuLcons score signal to the phastCons score over an ENCODE region (hg17, ENm005, chr21:32677595-32677794). A Gauss window with $\sigma_w = 0.2$, size 15 ($\delta = 7$) was used for KuLcons scores. In the alignment, bases with gray background represent bases identical to consensus.

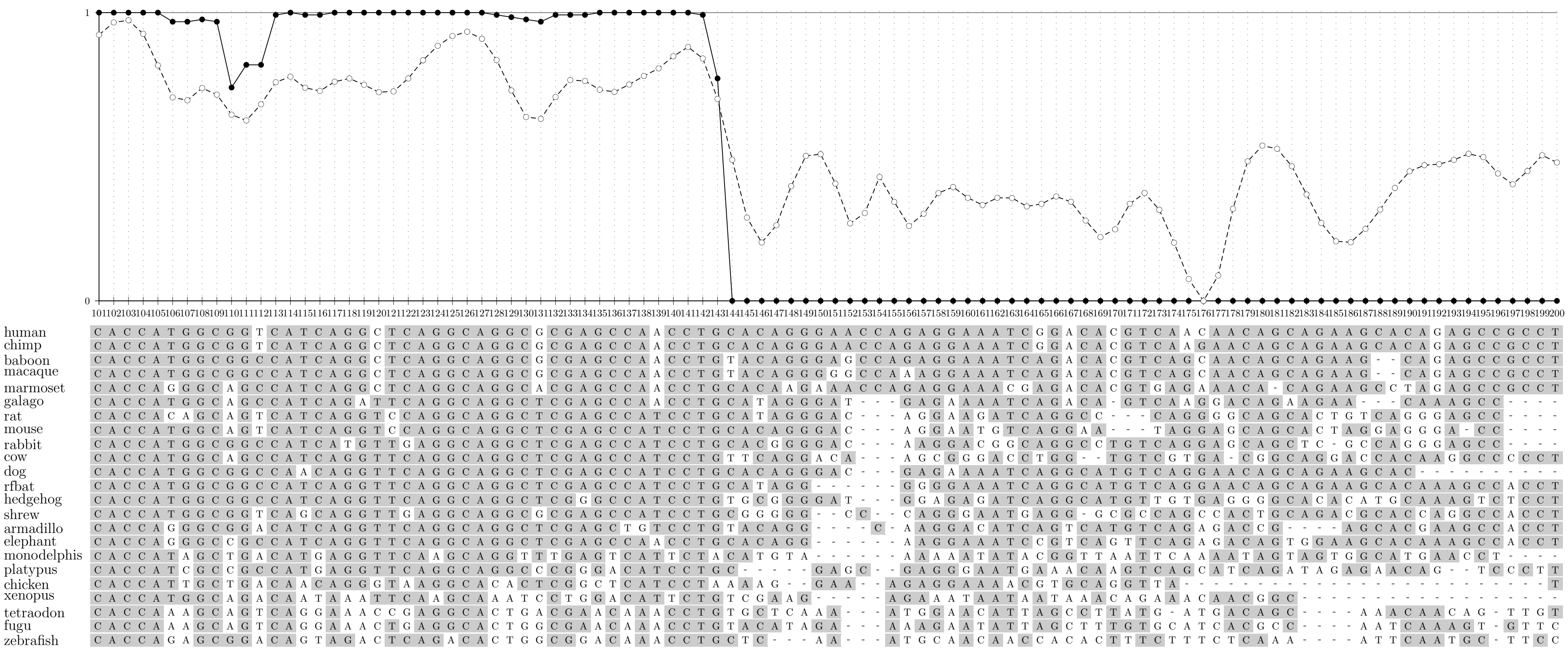


Figure 5.13 continued

5.5.5 Extending $\boldsymbol{\psi}$ to model insertion and deletion events

The projection of the model $\boldsymbol{\psi}$ via KL divergence allows more than one parameter in $\boldsymbol{\psi}$ to contribute to the conservation score. In particular, taking into account insertions and deletions in a proper way is a challenging task in most phylogenetic analysis. We extended the standard evolutionary model to include rates of insertions and deletions. These InDels (cf. Section 3.2) give rise to gaps in the alignment (cf. Section 4.2.3) which are usually neglected by most approaches for measuring the conservation. The probabilistic inference of insertion and deletion events along a phylogenetic tree is a difficult problem. Several methods have been proposed in the literature: Rivas showed in [Riv05] how to extend the matrix $\boldsymbol{R}$ in order to model gaps as a fifth character. A maximum likelihood approach for inferring InDel scenarios was proposed by Blanchette et. al. in [DMB06]. Recently, Kim and Sinha presented an algorithm, *InDelign*, for the annotation of InDels in a probabilistic framework [KS07]. Here we used InDelign to estimate the probability of deletions and insertions in the sliding window. The estimated probabilities $\hat{p}_I^{(e,k)}$ and $\hat{p}_D^{(e,k)}$ of an InDel of length $k = 1, 2, .., 2\delta + 1$ on branch e were calculated. In Indelign, InDel probabilities p_I, p_D are assumed to be proportional to the branch length t_e

$$p_I = c_I t_e, \quad p_D = c_D t_e, \tag{5.46}$$

where c_I and c_D are constants, estimated as follows: let N_I,N_D be the numbers of InDels from the parent of the two closest related species to either species. The constants are estimated as

$$\hat{c}_I = \frac{N_I}{(t_{e_1} + t_{e_2})L}, \quad \hat{c}_D = \frac{N_D}{(t_{e_1} + t_{e_2})L}, \tag{5.47}$$

where $(t_{e_1} + t_{e_2})$ is the sum of the distances of these species to their common ancestor, and L is the length of the sequence which is in our case the size of the sliding window $2\delta+1$ [KS07]. The score was then calculated based on the 3 free parameters θ, c_I, c_D. Since we are only interested in the probability of a fully conserved column, we have to calculate the probability that such a column is observed for the estimated values. The substitution and InDel processes are assumed to act independently, i.e., let again $p(\boldsymbol{a}_{[b]}; \hat{c}_I, \hat{c}_D, \hat{\theta})$ denote the probability of a fully conserved column under the estimated parameters, then

$$p(\boldsymbol{a}_{[b]}; \hat{c}_I, \hat{c}_D, \hat{\theta}) = p(\boldsymbol{a}_{[b]}; \hat{c}_I, \hat{c}_I) p(\boldsymbol{a}_{[b]}; \hat{\theta}). \tag{5.48}$$

Simplifying assumptions about the InDel process imposed by the InDelign algorithm allow to express the first term as the probability that no InDel occurred on any of the branches e, at the actual and the k preceeding positions:

$$p(\boldsymbol{a}_{[b]}; \hat{c}_I, \hat{c}_D, \hat{\theta}) = p(\boldsymbol{a}_{[b]}; \hat{\theta}) \prod_{\forall e} \prod_{k=1}^{\delta+1} (1 - p_D^{(e,k)})^k (1 - p_I^{(e,k)})^k. \tag{5.49}$$

The scores (Eq. (5.39)) are then calculated using this probability.

In Figure 5.14 we show the application of our extended algorithm to ENCODE data. Two different KuLcons scores for a 200bp fragment of an ENCODE region are shown. One score (*KuLcons no InDels*) represents conservation estimation based only on local substitution rate estimates, treating gaps as missing data. For the other score (*KuLcons* *with InDels*), 3 parameters were estimated: the substitution rate θ, and InDel parameters $\hat{c}_I$ and $\hat{c}_D$. All parameters were estimated in a rectangular sliding window of length $21(\delta = 10)$ over the alignment. Note that in this framework $\boldsymbol{\psi}$ comprises 2 additional parameters c_I and c_D.

In [SPH06], Siepel et al. present an extension of phastCons accounting for lineage-specific "gained" or "lost" elements. Similar to our approach, the authors use a separately reconstructed InDel history and compute emission probabilities of InDels for a phylo-HMM. However, to our knowledge phastCons has not yet been further developed in this direction, and the signal of phastCons shown in Figure 5.14 treats gaps as missing data.

As expected, the KuLcons score including the InDel estimation is upper bounded by the version neglecting the InDels. The scores coincide where no gaps are observed in the sliding window (positions 41-42) and differ when one or more gaps are observed (e.g., 74-98). A significant difference in the scores is observed in regions with many gaps. While the score based solely on the substitution rate indicates high conservation, the score respecting the gaps indicates low conservation in these regions.

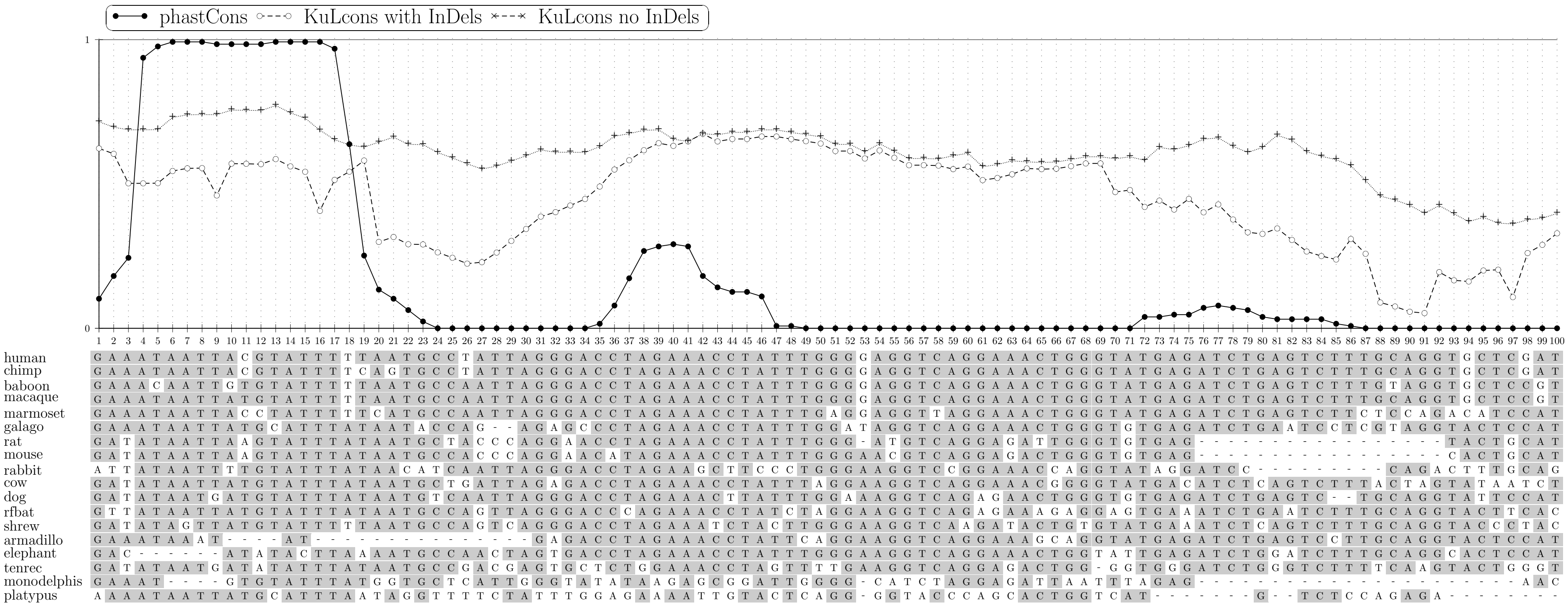


Figure 5.14: Comparison of KuLcons score taking gaps as InDels into account and KuLcons score treating them as missing data in an ENCODE region (hg17, ENr212,chr5:142147118- 142147317). Scores are based on estimating the parameters in a rectangular window ($\delta = 10$). PhastCons scores are shown for comparison.

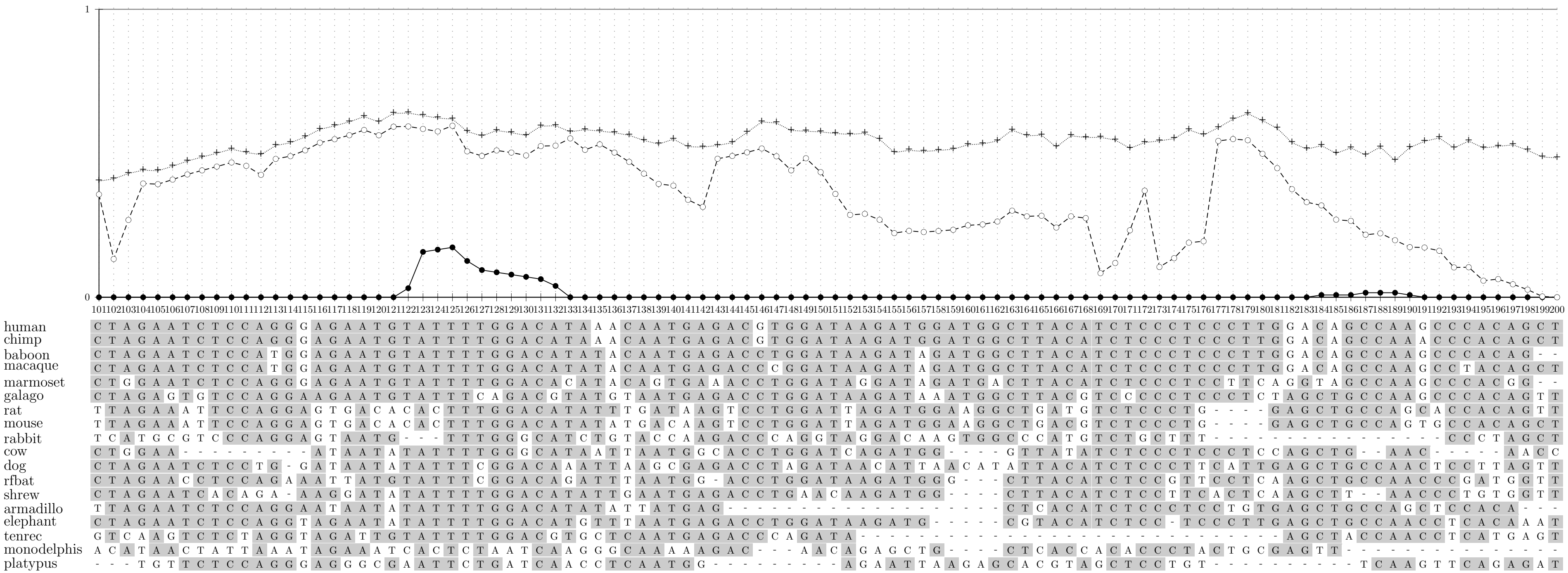


Figure 5.14 continued.

5.6 Discussion

In Figure 5.13 and Appendix C we showed a comparisons of KuLcons to phastCons, GERP and SCONE. These methods aim to detect sequence conservation and/or constraint based on different models:

phastCons scores reflect the a posteriori state probabilities of the underlying HMM and thus express the probability of constraint, based on the assumed degree of conservation and the assumptions about neutral evolution imposed on the hidden Markov model. While this is very well suited for high throughput processing, a simplistic binary model on genome evolution is imposed. The two state HMM implies that evolution is either conserving or neutral. The model has to be tuned with a priori information such as transition rates among the conserved and the neutral state, which implicitly imposes assumptions about the expected length and coverage of conserved regions. Phylogenetic HMM were analyzed in [FZSL07] with respect to their statistical power for detection of conserved elements. The authors found that these parameters, in fact, have major influence on the detection power of phylo-HMMs.

The assumed binary model is clearly reflected in the scores. The phastCons score in Figure 5.13 provides clear indication for strong or weak conservation but lacks sensitivity for different degrees of conservation.

GERP compares observed and expected substitution rates on a phylogenetic tree with fixed topology. The branch lengths of the observed tree are estimated for each column separately, and branch lengths of the expected tree are compared to the average of estimates from neutral sites. The final score is the difference of the observed to the expected substitution rate induced by the corresponding estimated trees [CSA+05]. GERP predicts constraint elements using a null model of shuffled alignments.

SCONE scores express the p-value that a position evolved neutrally given a model that accounts for context-dependency, InDel events, and neutral evolution. Hence, the score can as well be interpreted as a probability of constraint [ARSS07].

Another method used in the ENCODE analysis, BinCons developed by Margulies et al. [MBHG03], was not included in the comparison because it was noted by Siepel [SH05] that scores of BinCons and phastCons give qualitatively similar results.

In contrast to the approaches mentioned above, **KuLcons** considers the direct estimation of the rate heterogeneity $\theta_l \in \mathbb{R}_+$ or more parameters from an evolutionary model $\boldsymbol{\psi}$ via maximum likelihood using an optimized sliding window and accounting for autocorrelation among sites. The Kullback-Leibler divergence is used to project the estimated parameters to a conservation score. As shown in Section 5.4.1, the rate parameter θ is the crucial parameter for detecting evolutionary conservation, and we showed that the presented ML sliding window approach achieves high estimation accuracy *in silico*, assuming a model of gamma distributed rates with autocorrelation.

Our method allows other parameters than θ to contribute to the conservation score. As expected, scores obtained from multiple parameter estimates, with additional free parameters in $\boldsymbol{R}$, did not significantly deviate from estimates solely based on θ. For example, estimating the score for the ENCODE data shown above using the HKY model for $\boldsymbol{R}$, i.e., allowing for two substitution parameters (different transition/transversion rates) to be estimated, leads to a correlation of 0.99 to scores obtained with $\boldsymbol{R}$ fixed (results not shown). This is in accordance with earlier results on the negligible influence of different parameterizations of $\boldsymbol{R}$ on rate estimation [YW95, CSA+05] and our own findings based on the mutual information analysis.

However, conservation estimation under more complex models of evolution, such as those accounting for insertions and deletions, can be incorporated in the procedure in a probabilistic fashion. As implied by our results shown in Figure 5.14, scores neglecting insertion and deletion events and scores taking these complex evolutionary events into account can differ significantly.

We believe that KuLcons has the following advantages:

1. The presented algorithm is free of assumptions about neutral evolutionary rates that are notoriously hard to determine [PM07, CBS+04, HRY+03]. Furthermore, it uses few *a priori* parameters that require biological considerations. We have shown that our ML estimation of substitution rates in an optimized Gauss window without assumptions on the rate prior leads to good performance in the MSE sense.
2. Our score reflects well the different degrees of conservations. This soft score may disclose new possibilities in comparative genome analysis allowing the comparison of different finescale conservation patterns within conserved regions of interest. A first analysis in this direction was carried out in [Heg08] in which micro RNAs were compared based on their conservation profile rather than their sequence similarity.
3. The KL divergence based measure is intuitive and makes scores obtained from different data sets using different models $\boldsymbol{\psi}$ comparable.
4. Our method can deal with extended phylogenetic models taking into account more evolutionary factors such as insertions and deletions. A whole set of different process parameters can then be mapped to a conservation score via the Kullback-Leibler divergence. A score was shown in Figure 5.14 that uses co-estimated InDel rate parameters. Another possibility would be to assign different θ to different subtrees thus allowing for lineage-specific rate heterogeneities.

Our results show that the KuLcons score qualitatively exhibits similar conservation patterns in different regions as GERP and SCONE. This observation has two important consequences: first, it is possible to score the conservation of DNA sequences without having assumptions or estimates on neutral rates. The estimation and potential bias of these rates have been controversially discussed in the past [CBS+04, HRY+03, KXL06, PM07].

Secondly, our results suggest that conserved elements inferred from this method would probably not be very different from those discovered by GERP and SCONE, opposed to the conjecture raised in [PM07]. This would mean that the discrepancies of experimentally verified functional elements and computationally predicted conserved regions [MVH+07, MCA+07, Che07] cannot be explained in majority by biased assumptions on neutral rates. One explanation might be that low scoring sequences experience constraints at a different information level (e.g., secondary structure) that is not detectable by simple sequence alignments but rather structural alignments. An alternative explanation is that species specific functional elements that are not conserved across a given set of species are more important in functional evolution than currently discussed. In our opinion, it would be highly desirable to develop test sets for objective comparison of the different methods based on *in silico* and biological data. Other disciplines from computational biology are fostering similar concerted efforts by computational and experimental biologists [SMC07, MSY03]. However, for various reasons this may be more difficult in the discipline of comparative genomics.

5.7 Summary

This chapter dealt with the identification of phylogenetic systems from multiple species sequence data under a probabilistic model of DNA evolution. It was assumed that evolution of species is modeled by a phylogenetic tree, evolutionary distances, and a substitution process as introduced in Chapter 4. It was further assumed that a perfect multiple sequence alignment was available for the considered sequences.

We first briefly introduced methods for reconstructing the tree topology as well as for estimating branch lengths and substitution process parameters. After extending the model of evolution to account for variable rates among sites, introducing a rate variation parameter, we focused on the problem of detecting DNA sequences that are conserved among species. Conserved regions recently are candidates for potentially functional elements. This is motivated by the assumption that conservation is a strong indicator for natural selection.

Previously applied detection methods were reviewed and classified, distinguishing between naive (model free) methods and those taking phylogenetic models into account. We then presented our method KuLcons, which is based on a continuous, autocorrelated model of rate variation. KuLcons proceeds in two steps: it first locally estimates phylogenetic parameters in a sliding window and then uses the Kullback-Leibler divergence to calculate a score that is independent of assumptions about neutral substitution rates.

We carried out simulation studies to analyze KuLcons: results suggest that the MLE is nearly optimal even if the size of the sliding window is relatively small, and that it can reliably estimate the rate variation parameter under the continuous, autocorrelated model. An *in silico* comparison with previous methods was performed under a very general simulation model of rate variation. KuLcons achieved the best performance compared to phastCons and model free methods. The gain in performance is expected to increase when alignments with more species are considered.

We applied KuLcons to recent alignment data from the ENCODE project and qualitatively compared our score to those of phastCons, GERP and SCONE, that were applied in the ENCODE project. We found that KuLcons better reflects the different degrees of conservation than phastCons that only gives a rough scale pattern of conservation. Our comparison with GERP and SCONE further suggested that biased estimates of neutral evolutionary rates is not the major cause of computational methods overlooking functional DNA regions. This had been proposed as an explanation for the discrepancy between computationally predicted and experimentally verified functional regions in the ENCODE project [PM07].

A major advantage of our score is that it is easily extendable to take multiple factors contributing to evolutionary conservation into account. We showed how scores can be calculated under insertion and deletion models, applying our modified scoring scheme to an ENCODE region. Comparison suggests that these scores can diverge significantly in regions where gaps are observed.

A discussion of the obtained results concluded this chapter.

5.8 Future research

While the presented conservation score KuLcons was shown to better identify conserved DNA sequences, its computational complexity makes high-throughput application difficult, at least for the user without access to a high performance computer. Reduction of running time without sacrificing accuracy would be highly desirable. One way to achieve this could be the use of simplified evolutionary models, e.g., the use of substitution rate matrices that allow for analytical solutions of the transition probabilities between nodes in the tree graph. In this case, singular value decompositions - which are costly - can be avoided, and the running time is expected to be significantly decreased. In the framework of communication theory, Hagenauer et al. developed approximate message passing algorithms using metrics instead of probabilities known as log-max approximation [HOP96]. These were shown to provide sufficient accuracy while significantly reducing the algorithmic complexity in data transmission applications. An application to the Felsenstein algorithm for large-scale phylogenetic analysis could lead to interesting results.

KuLcons provides a soft score that may prove more helpful in deciphering and classifying functional DNA than scores that behave binary such as phastCons. In fact, first results [Heg08] suggest that KuLcons scores of *micro RNAs* (miRNAs) exhibit statistically significant similarities using Pearson correlation as well as empirical mutual information as divergence measures. Does conservation of score patterns imply similar function? It would be interesting to investigate whether miRNAs or other conserved regions could be clustered on a functional level using soft scores. However, this requires a large, carefuly curated and annotated database that allows for rapid whole genome analysis.

Our algorithm has the advantage to be easily extendable to more complex models of evolution. Such models could allow for different substitution rates in different lineages or take context-dependency into account. Here, we presented an extension to the case when gaps in multiple sequence alignments are modeled as insertions and deletions instead of neglecting them (as done by most algorithms). Different evolutionary models exist, and our algorithm enables comparison of conservation scores under different models - to be investigated in extending work.

6

On the DNA code reverse engineering problem

This chapter is a first attempt to systematically address the question whether there is an error correcting code in the DNA that protects genetic information from mutations.

In Section 6.1 we analyze known error correcting mechanisms in the cell from a coding theoretic perspective and review previous investigations on the hypothesis of nature using a channel coding scheme in the DNA. We present the theoretical and experimental evidences previously found and also give a critical discussion on these evidences.

Most of the remainder of this chapter then focuses on the formal code reverse engineering problem. Maximum likelihood methods for reconstructing a convolutional encoder from observed noisy coded bits are presented. The methods are based on a probabilistic encoder tap model transforming the problem into the log-likelihood ratio domain. Section 6.2.2 presents an approach based on the expectation maximization algorithm. Section 6.2.6 introduces a method based on global stochastic maximization, and Section 6.2.7 provides the extension to non-binary alphabets. Simulation results are shown for both methods (Sections 6.2.5 and 6.2.8).

Finally, in Section 6.3, an application to DNA sequence data is presented. Our method is used to analyze an *ultraconserved* sequence, 100% identical in the human, mouse, and rat genome. However, our results based on comparison to a a random sequence show no evidence that the ultraconserved region belongs to the class of convolutional codes considered in the analysis.

6.1 Is there an error correcting code in the DNA?

It is an astonishing fact that DNA is a digital signal, i.e., a discrete sequence of symbols from a finite - in case of DNA quarternary - alphabet (cf. Chapter 3, Section 3.1). As discussed in Chapter 3, all cellular information processing problems on the DNA level are essentially related to reliable storage and transmission of digital information. Over the past decades, communication engineers have developed many powerful tools and algorithms in order to design digital data transmission systems. It should not surprise us if nature and engineers partly developed similar concepts in dealing with transmission of digital information.

A main result of information theory is the *channel coding theorem* stating that faithful transmission of information is possible even if signals are distorted by noise during transmission (cf. Chapter 2, Section 2.2.2). One of the main advantages of digital communication systems is the possibility of using *forward error correction* or *channel coding schemes* allowing to reconstruct messages at the receiver from noisy signals. Very powerful error correcting mechanisms were developed over the last 60 years by communication engineers, that are now an integral part of any modern communication system [CHIW98, HOP96, CFR+01, RU07].

6.1.1 DNA error correction from a coding theoretic perspective

The DNA replication machinery reproduces DNA with an astonishing accuracy. It introduces an error only every $10^{-3} - 10^{-5}$ base pairs. The known error correcting mechanisms (base excision and mismatch repair, cf. Chapter 3, Section 3.2) further reduce the rate of nucleotide substitution error to $10^{-9} - 10^{-10}$ per cell generation. Do these repair mechanisms make use of error correcting codes? As we shall see in this section, the answer is clearly yes; but from a coding theoretic perspective, most of these are trivial.

Recall that, formally, a code $\mathcal{C}$ is a collection of elements which form a subset of all possible elements. A crucial parameter of a code is its minimum distance, i.e., the minimum distance between any two elements from $\mathcal{C}$ (cf. Chapter 2, Section 2.3 for a brief introduction). Nature uses A,C,G,T as an alphabet. However, there are many more chemical structures that could be used. Donail, for example, lists 16 possible nucleotides or nucleotide analogues. Some of those were experimentally shown to be replicated *in vivo* by cellular replication machinery implying that larger alphabets were in fact possible [Dón06]. So the alphabet of DNA forms a *subset* of all possible elements, which could be interpreted as a code. Consider a nucleotide $x \in \{A, C, G, T\}$ that experiences a mutation and mutates to x'. Most mutations that occur do not result in a change of the nucleotide but alter it in a way that it is not a "valid" nucleotide (not A,C,G,T) anymore. In words of a coding theorist: the mutation results in an element that is not an element of the code A,C,G,T. As a result, the error can be detected. Now, in a coding theoretic approach, one would look for the element A,C,G,T which is closest to the mutated x'. However, this is not what happens in the cell; the cell rather uses a repetition code to correct mutated bases: recall that DNA is a double stranded molecule consisting of base pairs that pair up according to deterministic rules (A with T, C with G). A coding theorist calls this a repetition code (cf. Chapter 2, Example 2.3.1). Once an error is detected (first code), the mutated nucleotide x' can be corrected by simply looking at its complementary base (second code).

The genetic code can be interpreted as a block code: recall that the genetic code maps triplets of nucleotides to amino acids. There are 64 distinct triplets but only 20 different amino acids. The genetic code is hence a rate $\log_4(20)/3 \approx 0.72$ block code. There exist pairs of triplets in the code that code for different amino acids and only differ in one position (cf. Figure 3.9). The minimum distance of the code is therefore 1. To summarize, the genetic code is a $(3, \log_4(20), 1)$ block code. A minimum distance 1 code is of course not a good code since there exist triplets for which the code does not have error correcting nor error detecting capability. Still, the code provides some error tolerance afterall (e.g., some third positions of triplets are completely interchangeable without altering the resulting amino acid). The code can even be shown to be close to optimal when taking into account that properties of proteins are almost invariant to substitutions of amino acids with similar properties [FH98].

The latter observation is actually a crucial point: when considering cellular information processing systems, it is not enough to count the overall number of errors. Rather, one has to take a distortion measure into account: different kind of errors have different effects. For example, the change of an amino acid to another might be tolerable as long as the property of the resulting proteins are conserved. However, such effects are not taken into account by classical coding theory that concentrates on minimizing the total number of errors. Furthermore, systems in nature are mostly designed such as to optimize multiple objectives. For example, it was shown that the genetic code, in addition to its error distortion minimizing property, is nearly optimal with respect to frameshift errors [BVK07] and in allowing for additional signals to be embedded as secondary information [IA07, SP07].

Error correction mechanisms are spread among the different layers of cellular information processing. For instance, it was observed that genetic networks are robust with respect to the failure of a single gene. Knockout experiments have shown that other genes are able to take over the tasks of silenced genes. A very simple explanation is that most genes occur in multiple copies in the genome. In fact, in the cells of eukaryotic organisms each gene occurs twice, since each cell contains two sets of chromosomes (repetition code). However, gene copies cannot explain all experimental observations regarding network robustness. For instance, experiments that completely removed some ultraconserved regions from the mouse genome generated viable mice [NZPF+04], not showing any phenotype. How the robustness of genetic networks is achieved is not completely understood; it shows, however, that there is an error correcting mechanism beyond the DNA sequence level.

To summarize, known cellular repair mechanisms seem to function according to the following principles:

- ⋆ The mechanisms are very simple from a coding theoretic perspective (repetition codes, very small codes).
- ⋆ Not the overall number of errors but rather some resulting distortion is minimized. Classical coding theory is therefore probably not sufficient for an analysis of genomic codes. Such an analysis, however, is difficult since the underlying distortion measures are often impossible to model or quantify.
- ⋆ Cellular systems often have multiple objectives. As an example, the genetic code jointly provides robustness against substitution errors and frameshifts [BVK07], and allows for inclusion of additional secondary information [IA07].
- ⋆ Error correction happens at different levels of information processing (sequence level, network level, etc.). Mechanisms are probably linked across the different layers.

Coming back to our original question, one may conclude that it might therefore seem too naive to look for a classical error correcting mechanism on the DNA sequence level. On the other hand, there are observations of sequence error correction that still cannot be explained with known mechanisms: how does the mismatch repair system, facing two mispaired nucleotides, know on which strand the error occurred in the first place (cf. Chapter 3, Section 3.2.2)? How can the non-mendelian inheritance of sequence information in *Arabidopsis* be explained [LVYP05]? How is the surprising robustness of gene networks achieved?

6.1.2 Theoretical and experimental evidence

Recently, Gérard Battail published a series of papers that consider the problem of conserving genomic information from an information theoretic perspective [Bat08, Bat06]. Battail postulates that

> "Any genome belongs to an efficient error-correcting code [...]. Its necessity stems from the need of a faithful conservation of the genetic information."

He then mathematically shows that, under certain assumptions, genome conservation can only be explained by the presence of such codes in the genome. He further discusses biological consequences that his hypothesis would imply and compares them to the situation observed in reality. He finally concludes that a genomic error correcting code must exist [Bat08].

Battail regards DNA as a memory in a noisy environment. He assumes that the information contained within this memory is to be preserved *ultimatively*. Information theory enables measuring the ability of a channel or memory element to convey information via the channel capacity (cf. Chapter 2, Section 2.2.2). Battail calculates the capacity of a DNA strand assuming the following model:

Consider a symbol from the alphabet $\mathcal{A} = \{A, C, G, T\}$ and assume that this symbol incurs a substitution during the infinitesimal time interval dt with probability αdt, with constant α. Battail finds that the probability of substitution $p_{\mathrm{su}}(t)$ can be expressed as

$$p_{\mathrm{su}}(t) = 3/4(1 - e^{-3/4\alpha t}). \tag{6.1}$$

Note that this is equivalent to the Jukes-Cantor (JC) substitution process model introduced in Chapter 4, Section 4.1.2. In Section 2.4.4, the substitution probability expressed in Eq. (6.1) was derived for the JC model. We also know from Chapter 4 that the JC model is equivalent to the quarternary symmetric channel (QSC) as defined in Chapter 2, Section 2.2.2. The capacity of the QSC channel was derived in Example 2.2.2, and substituting Eq. (6.1) we find that

$$C_{\mathrm{DNA}}(t) = 2 - \left[(1 - p_{\mathrm{su}}(t)) \log_2(1/(1 - p_{\mathrm{su}}(t))) + p_{\mathrm{su}}(t) \log_2(1/(3p_{\mathrm{su}}(t)))\right]. \tag{6.2}$$

The capacity depends on evolutionary time through the product αt. We show a plot of the capacity over αt in Figure 6.1. For $t = 0$, the DNA reveals 2 bit of information as

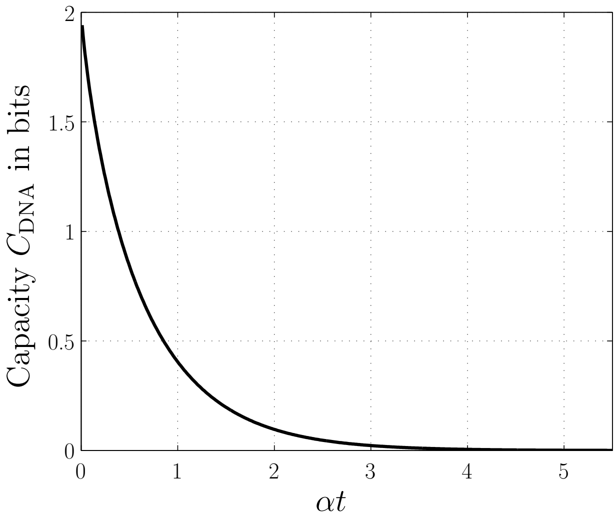


Figure 6.1: Time-variant capacity of a single stranded DNA under Battail's model.

expected. For $t > 0$, however, the capacity decreases exponentially fast and approaches 0 for $t \rightarrow \infty$. Battail also extends these calculations to the case of double stranded DNA and erasure errors and shows that the capacity also vanishes exponentially in these cases.

Battail concludes that, under any reasonable assumptions about mutation rates, DNA is an *ephemeral* memory, hence unable to conserve hereditary information during geological timescales. He argues that an intrinsic error-correcting code must frequently act upon the genome to efficiently regenerate the genome before its capacity has decreased beyond a level that precludes regeneration.

Recently, Battail's hypothesis was strengthened by experimental results from plant biology: Lolle et. al. experimentally introduced a substitution in a single position of a gene from *Arabidopsis thaliana* [LVYP05]. According to the mendelian rules, the offspring of these modified plants must inherit the same mutation from their parents. Surprisingly, a non-negligible (10%) fraction of plants was observed that inherited sequence information that was not present in the chromosomal genome of their parents but had been present in previous generations. The authors conjectured that an ancestral RNA sequence cache is inherited, that can serve as a template to restore the original sequence information; yet there is no experimental evidence for this hypothesis, and the exact error correcting mechanism is still not understood.

6.1.3 Limitations of Battail's model

From a purely information theoretic point of view, there is no other explanation for genome conservation than the existence of a genomic error correcting code under Battail's model. However, the model is based on certain assumptions that do not conform with reality: Battail assumes that the primary goal in nature is genome conservation. However, evolution is a constant process of competition for resources and adaptation to a changing environment. A certain amount of mutations must be possible in order to allow species to develop new traits and adapt to the environment. For example, a main reason of why finding a cure to HIV is so difficult is the high mutation rate of the virus; it allows HIV to adapt quickly to new treatments, developing new resistances. Even the human genome is quite different from individual to individual. *Single nucleotide polymorphisms* (SNPs) denote nucleotide positions in the genome that vary in at least 1% of a given population. SNPs occur every 100 to 300 bases along the human genome. Recently, a comparative study of 270 individuals from four different populations revealed that more than 1140 regions in the genome differ in the number of copies in which they occur in a given genome (copy number variation). These regions cover no less than 12% of the genome [RIF$^+$06].

Still, it can be argued that there are parts in (at least) eukaryotic genomes that are so fundamental to the organism that they must be conserved. Comparative genomic studies support this assumption: it was found that there are many (ultra-)conserved elements located in, for example, the human coding and non-coding regions. Some of those are several hundreds of basepairs in length and 100% conserved among human, rat, and mouse. Applying Battail's model to these sequences, such high degree of conservation is only explicable by error correcting codes. However, this is only true when death as an error correcting mechanism is excluded from the model as done in Battail's considerations. A more realistic model would account for the fact that a genome is sometimes not passed on to the next generation at all, i.e., would account for *natural selection.* We saw in Chapter 3 that death is in fact an important concept in cell development: a cell has the ability to monitor its DNA for errors and can trigger programmed cell death (*apoptosis*) when the DNA is damaged beyond an acceptable level. This mechanism is crucial: mutations in cells that damage the pathway of apoptosis are known to be related to certain types of cancers [Kni06]. Furthermore, in Battail's model a genome is replicated once per generation which is actually not the case since higher organisms produce a whole set of germ cells that (simplified spoken) already compete for passing on their genome.

A more realistic model of DNA evolution must consider these factors. This is, however, admittedly very difficult.

6.1.4 Previous work

Several researchers have conjectured that there might exist a channel coding scheme also in DNA, protecting genetic information from mutations: in 1981, Forsdyke proposed a model of *introns* serving as in-series error detecting sequences [For81]. A protein coding gene is not a continuous coding sequence but is subdivided into *exons* (actually coding for the amino acids following the genetic code) and *introns*, spliced out before translation. At that time, the function of introns was not well understood yet it *was* known that they differ in sequence from the exons, not showing any similarity nor following the genetic code. Forsdyke therefore proposed a parity check code where parities are stored in the introns and information symbols in the exons (coding part of a gene). Such a parity model accounts for the unsimilarity between exons and their corresponding introns as well as for their difference in length. Access to DNA sequence data was difficult in 1981, and Forsdyke did not test his hypothesis on biological data. Also, Forsdyke's contribution was not constructive in the sense that it provided algorithms how to detect such a code.

Liebovitch et. al. [LTTL96], in 1996, were the first to propose a simple method how to find a linear parity-check block code in the DNA base sequence. They presented two approaches based on brute force search and on Gaussian elimination of variables, respectively. It was not considered that sequences could be distorted by noise. The method was applied to two genes and compared to results obtained from random sequences. However, the authors could not find evidence for the existence of a linear block code in the considered genes.

In 2003, Mac Dónaill presented a parity-check nucleotide model, showing that, under some assumptions, the nucleotides A,C,G,T exhibit optimal error correcting properties [Dón03]. The author then suggested how a parity code structure in the alphabet could offer a replication fidelity advantage. His conclusion suggests that A,C,G,T/U were not selected by accident as nature's alphabet but because of their optimal parity check properties.

The remainder of this chapter is a first attempt to systematically approach the problem of identifying a code in a DNA sequence. We first concentrate on the formal code reverse engineering problem. Since most of the previous work in DNA code identification considered block codes without finding indication for such a code in DNA, we shall focus on convolutional codes.

6.2 The code reverse engineering problem

We address a problem strongly related to cryptanalysis and data security: in a reverse engineering context, an observer wants to extract the transmitted information from a received data stream without knowing all the parameters of the transmission. It is further assumed that the observed signal was corrupted by noise during transmission. Even without employing advanced cryptologic protocols, modern communication systems are hard to decipher if the parameters of the different elements of the transmission chain are not, or only partially, known. Reverse engineering of channel codes (cf. Figure 6.2) was considered for communication systems [Clu06, Fil97, Ric95, Val01] and also for DNA sequences [LTTL96, For81], looking for possible error correcting codes in the genetic code. Most of these works concentrate on linear block codes and algebraic approaches.

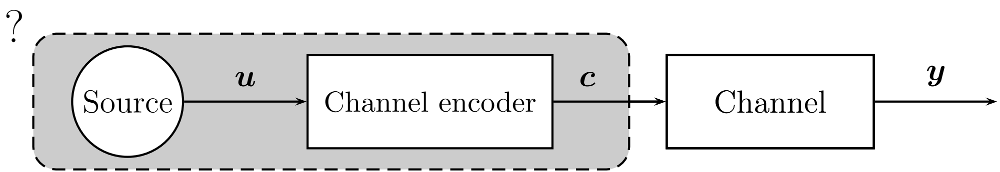


Figure 6.2: Code reverse engineering problem. A source emits a data stream $\boldsymbol{u}$. An observer has only access to the noise corrupted version of the channel encoded stream $\boldsymbol{y}$ and wants to estimate the channel coding scheme.

In the following, we derive a new probabilistic algorithm for the estimation of encoder parameters of a convolutional code from a noisy data stream. Reverse engineering of convolutional encoders was considered in an algebraic framework by Rice [Ric95] and later by Filiol [Fil97, Fil00]: parameters are recovered from a subsequence of bits that is unaffected by noise and then tested for significance on the whole observed sequence. However, if no noise-free subsequence exists for which the parameters can be algebraically recovered, the method will fail.

Here, we introduce iterative, probabilistic approaches based on the Expectation Maximization (EM) algorithm and stochastic global optimization. The EM algorithm, as presented in Chapter 2, Section 2.6, is a standard statistical method for local likelihood maximization, applied in many communications and signal processing problems today, such as blind channel estimation [KV94] and system identification [Moo96]. These problems are closely related to the one considered here. In fact the problem is essentially the same as fitting a hidden Markov model, typically done with the Baum-Welch algorithm, a variant of the EM [EM02]. Unlike in these cases, we investigate systems where computations are carried out in a finite field, and methods developed for traditional blind estimation scenarios do not apply. This implies that an approach has to combine concepts from both, coding theory and blind signal processing.

Moon [Moo02] applied the EM algorithm to the synthesis of linear feedback shift-registers in a similar framework. However, here we show how to transform the problem into the log likelihood ratio domain (LLR) and use LLR algebra [HOP96] to greatly simplify the derivation of our algorithm. We find that local optimization can only recover systematic encoders, and we show how global optimization techniques can be used to overcome this limitation. Furthermore, we extend the methods to sequences defined over arbitrary alphabets, and show simulation results for the binary and quarternary (DNA) case. The next sections introduce the problem more formally and describe our methods.

6.2.1 Problem statement

We assume that an information stream $\boldsymbol{u} = [u_0, u_1, \ldots, u_t, \ldots]$, $u_t \in \{+1, -1\}$ passes through a convolutional encoder with memory M as introduced in Chapter 2, Section 2.3.2. We shall first consider codes defined over the binary field $\mathbb{F}_2$ with elements $\{+1, -1\}$, where $+1$ represents the zero element under the $\oplus$ addition. Our approach is presented for codes of rate $\frac{1}{N}$, i.e., the encoder has a single input and N outputs. We exclude encoders with feedback from our considerations. The algorithm is easily generalized to any rate $\frac{K}{N}$.

In Figure 6.3, the situation for the nth output of such a convolutional encoder is shown: the encoder taps $\boldsymbol{g}^{(n)} = [g_0^{(n)}, \ldots, g_M^{(n)}]$ determine how the information symbol u_t and the content of the memory elements, i.e., $[u_{t-1}, \ldots, u_{t-M}]$, at time t, are mapped to the nth coded output symbol $c_t^{(n)}$. We assume that the received symbol $y_t^{(n)}$ is observed through a binary symmetric channel (BSC) modeled by a r. v. ε with $P(\varepsilon = -1) = p_\varepsilon$, i.e.,

$$y_t^{(n)} = c_t^{(n)} \oplus \varepsilon. \tag{6.3}$$

We define the encoder state $\boldsymbol{s}_t$ as the concatenation of the current input symbol with the content of the memory elements, i.e.,

$$\boldsymbol{s}_t = [s_{t_0}, \ldots, s_{t_M}] = [u_t, u_{t-1}, \ldots, u_{t-M}], \tag{6.4}$$

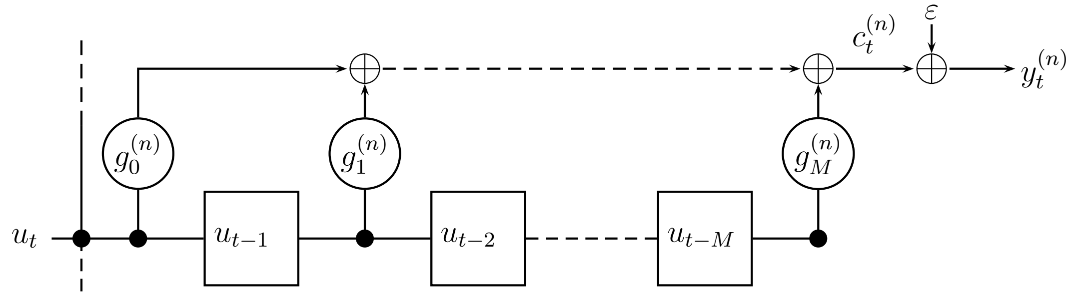


Figure 6.3: Convolutional encoder with memory M and encoder parameters $\{g_m^{(n)}\}$. Mapping of u_t to the nth output $c_t^{(n)}$ is shown. Additive, memoryless noise leads to observation $y_t^{(n)}$.

and we can therefore write

$$y_t^{(n)} = \sum_{m=0}^{M} \oplus s_{t_k} g_m^{(n)} \oplus \varepsilon, \tag{6.5}$$

where multiplication and addition are in $\mathbb{F}_2$. Note that the nth output only depends on the nth parameter subset $\boldsymbol{g}^{(n)}$, and that the N outputs are not independent since each output $\boldsymbol{y}_t = [y_t^{(1)}, \ldots, y_t^{(N)}]$ depends on the same encoder state $\boldsymbol{s}_t$.

An encoder is fully specified by the vector

$$\boldsymbol{g} = [\boldsymbol{g}^{(1)}, \ldots, \boldsymbol{g}^{(n)}] \in \{+1, -1\}^{N(M+1)}, \tag{6.6}$$

which are the parameters that we are about to estimate given the observation

$$\boldsymbol{y} = [\boldsymbol{y}^{(1)}, \ldots, \boldsymbol{y}^{(N)}], \quad \text{where} \quad \boldsymbol{y}^{(n)} = [y_1^{(n)}, \ldots, y_T^{(n)}]. \tag{6.7}$$

The length of $\boldsymbol{y}$ is NT.

In the next section we show how to obtain the maximum likelihood estimate on $\boldsymbol{g}$ in an iterative manner, using the expectation maximization algorithm.

6.2.2 EM encoder tap estimation

The Expectation Maximization (EM) Algorithm iteratively approaches the maximum likelihood solution in what are called incomplete-data problems [Moo96]. The EM algorithm was derived earlier in Chapter 2, Section 2.6. In brief: let $\boldsymbol{y}$ be the observed data and $\boldsymbol{x}$ the unobservable data, that depend on a deterministic parameter $\boldsymbol{\theta}$ that we want to estimate. With a *fixed* current estimate on the parameter, say $\boldsymbol{\theta}^{[k]}$, the expected value of $\log(p(\boldsymbol{x}, \boldsymbol{y}; \boldsymbol{\theta}))$ (so-called Q-function $Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{[k]})$) is evaluated with respect to $\boldsymbol{x}$ and conditional on $\boldsymbol{y}$ in the E-Step. In the M-step of the algorithm, $Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{[k]})$ is maximized with respect to $\boldsymbol{\theta}$ to yield a new estimate $\boldsymbol{\theta}^{[k+1]}$ in iteration $k+1$:

$$Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{[k]}) = \sum_{\boldsymbol{x}} p(\boldsymbol{x}|\boldsymbol{y}; \boldsymbol{\theta}^{[k]}) \log(p(\boldsymbol{x}, \boldsymbol{y}; \boldsymbol{\theta})) \quad \text{(E-Step)}, \tag{6.8a}$$

$$\boldsymbol{\theta}^{[k+1]} = \arg\max_{\boldsymbol{\theta}} \{Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{[k]})\} \qquad \text{(M-Step)}. \tag{6.8b}$$

The EM algorithm is guaranteed (cf. Theorem 11) to converge to a stationary point which is a global maximum, yielding the unique ML estimate of $\boldsymbol{\theta}$.

In our setting, we identify $\boldsymbol{y}$ as the observed and $\boldsymbol{s}$ as the missing data. The unobserved state at time t, $\boldsymbol{s}_t$, is mapped to the observation $\boldsymbol{y}_t = [y_t^{(1)}, \dots, y_t^{(N)}]$, and the joint distribution of $(\boldsymbol{s}, \boldsymbol{y})$ depends on the parameter $\boldsymbol{g}$ of the convolutional encoder (cf. Chapter2, Section 2.4.5 and [EM02, Moo02]):

$$p(\boldsymbol{s}, \boldsymbol{y}; \boldsymbol{g}) = p(\boldsymbol{s}_1) \prod_{t=1}^{T} p(\boldsymbol{s}_{t+1}|\boldsymbol{s}_t) p(\boldsymbol{y}_t|\boldsymbol{s}_t; \boldsymbol{g}). \tag{6.9}$$

Applying Eq. (6.8a) yields the Q-function

$$Q(\boldsymbol{g}; \boldsymbol{g}^{[k]}) = \sum_{\boldsymbol{s}} p(\boldsymbol{s}|\boldsymbol{y}; \boldsymbol{g}^{[k]}) \left(\log(p(\boldsymbol{s}_1)) + \sum_{t=1}^{T} (\log(p(\boldsymbol{s}_{t+1}|\boldsymbol{s}_t)) + \log(p(\boldsymbol{y}_t|\boldsymbol{s}_t; \boldsymbol{g})) \right) \tag{6.10a}$$

$$= \underbrace{\sum_{\boldsymbol{s}_1} p(\boldsymbol{s}_1|\boldsymbol{y}; \boldsymbol{g}^{[k]}) \log(p(\boldsymbol{s}_1))}_{\text{independent of } \boldsymbol{g}} + \underbrace{\sum_{t=1}^{T} \sum_{\boldsymbol{s}_{t+1}, \boldsymbol{s}_t} p(\boldsymbol{s}_{t+1}, \boldsymbol{s}_t|\boldsymbol{y}; \boldsymbol{g}^{[k]}) \log(p(\boldsymbol{s}_{t+1}|\boldsymbol{s}_t))}_{\text{independent of } \boldsymbol{g}}$$

$$+ \sum_{t=1}^{T} \sum_{\boldsymbol{s}_t} p(\boldsymbol{s}_t|\boldsymbol{y}; \boldsymbol{g}^{[k]}) \log(p(\boldsymbol{y}_t|\boldsymbol{s}_t; \boldsymbol{g})) \tag{6.10b}$$

which, according to (6.8b), has to be maximized with respect to $\boldsymbol{g}$. The state transition probabilities do not depend on $\boldsymbol{g}$, nor does the initial state distribution $p(\boldsymbol{s}_1)$. Applied to our problem, the maximization in (6.8b) can therefore be written as

$$\boldsymbol{g}^{[k+1]} = \arg\max_{\boldsymbol{g}} \left\{ \sum_{t=1}^{T} \sum_{\boldsymbol{s}_t} p(\boldsymbol{s}_t|\boldsymbol{y}; \boldsymbol{g}^{[k]}) \log(p(\boldsymbol{y}_t|\boldsymbol{s}_t; \boldsymbol{g})) \right\} \tag{6.11a}$$

$$= \arg\max_{\boldsymbol{g}} \left\{ \sum_{t=1}^{T} \sum_{\boldsymbol{s}_t} p(\boldsymbol{s}_t|\boldsymbol{y}; \boldsymbol{g}^{[k]}) \sum_{n=1}^{N} \log(p(y_t^{(n)}|\boldsymbol{s}_t; \boldsymbol{g}^{(n)})) \right\}. \tag{6.11b}$$

Note that, given the state $\boldsymbol{s}_t$, the N outputs $y_t^{(n)}$ at time t are independent of each other and depend only on the subset of parameters $\boldsymbol{g}^{(n)}$, therefore:

$$\log\left(p(\boldsymbol{y}_t|\boldsymbol{s}_t; \boldsymbol{g})\right) = \sum_{n=1}^{N} \log(p(y_t^{(n)}|\boldsymbol{s}_t; \boldsymbol{g}^{(n)})). \tag{6.12}$$

So far, we regarded $\boldsymbol{g}$ as an element from a discrete parameter space. However, the EM based approach requires the parameter space to be continuous. We achieve this by transforming the parameters into the log-likelihood ratio (LLR) space, which shall allow us to apply the concept of LLR algebra [HOP96]. The next sections describe this approach and derive the computations necessary to perform an EM iteration in the LLR domain.

6.2.3 Transformation into the log-likelihood domain

Instead of regarding the encoder parameters as discrete values, we shall work with probabilities. In iterative decoding, calculations are carried out using log-likelihood ratios and log-likelihood ratio (LLR) algebra is applied [HOP96]. We shall see that it is also a natural choice for our problem.

Definition 6.2.1 (LLR and soft bit) A log-likelihood ratio $L(x)$ of the binary random variable X with pdf $p_{\mathsf{X}}(x) = [P_{\mathsf{X}}(x=+1),\ P_{\mathsf{X}}(x=-1)]$ is defined as

$$L(x) = \log\left(\frac{P_{\mathsf{X}}(x=+1)}{P_{\mathsf{X}}(x=-1)}\right). \tag{6.13}$$

□

This quantity has the following properties [HOP96]:

$$P_{\mathsf{X}}(x=\pm 1) = \frac{\pm e^{L(x)/2}}{e^{L(x)/2}+e^{-L(x)/2}}, \tag{6.14a}$$

$$\bar{x} \doteq E\{x\} = \tanh\left(\frac{L(x)}{2}\right), \tag{6.14b}$$

$$L(x_1 \oplus x_2) \doteq L(x_1) \boxplus L(x_2) = 2\tanh^{-1}\left(\bar{x}_1\bar{x}_2\right), \tag{6.14c}$$

$$\operatorname{sign}\left(L(x)\right) = \operatorname{sign}\left(x\right). \tag{6.14d}$$

The expected value of x, the so-called *soft bit*, is denoted by $\bar{x}$. A special operator $\boxplus$, the *boxplus*, is introduced to calculate the LLR of a sum of binary variables. The magnitude of the absolute value of $L(x)$ can be interpreted as the reliability about x, i.e., high positive values indicate that $x=+1$, and high negative values indicate $x=-1$.

We denote the log-likelihood ratio and the soft bit of the encoder parameter $g_m^{(n)}$ as

$$L_m^{(n)} = \log\left(\frac{P(g_m^{(n)}=+1)}{P(g_m^{(n)}=-1)}\right) \text{ and } \bar{g}_m^{(n)} = E\{g_m^{(n)}\} = \tanh\left(\frac{L_m^{(n)}}{2}\right). \tag{6.15}$$

Furthermore, we denote

$$\boldsymbol{L} = [\boldsymbol{L}^{(1)}, .., \boldsymbol{L}^{(N)}] = [L_0^{(1)}, L_1^{(1)}, .., L_{M-1}^{(N)}, L_M^{(N)}] \tag{6.16}$$

as our parameter vector.

For the EM approach we now consider $Q(\boldsymbol{L}, \boldsymbol{L}^{[k]})$, and the objective function in (6.11a) transforms into

$$\sum_{t=1}^{T}\sum_{\boldsymbol{s}_t} p(\boldsymbol{s}_t|\boldsymbol{y}; \boldsymbol{L}^{[k]}) \log(p(\boldsymbol{y}_t|\boldsymbol{s}_t; \boldsymbol{L})), \tag{6.17}$$

now maximized with respect to $\boldsymbol{L} \in \mathbb{R}^{N(M+1)}$, i.e., in a continuous parameter space. In order to find an analytical solution, we relax the strict maximization of (6.17) and require only $Q(\boldsymbol{L}^{[k+1]}, \boldsymbol{L}^{[k]}) > Q(\boldsymbol{L}^{[k]}, \boldsymbol{L}^{[k]})$, which, according to Theorem 11, suffices to increase the likelihood in iteration step $k+1$. This approach is commonly referred to as the *generalized* EM algorithm [EM02]. An increase of $Q(\cdot,\cdot)$ is achieved by a steepest ascent approach, i.e., the maximization is replaced by

$$L_m^{(n),[k+1]} = L_m^{(n),[k]} + \mu_{\text{step}} \frac{\partial Q(\boldsymbol{L}, \boldsymbol{L}^{[k]})}{\partial L_m^{(n)}}, \tag{6.18}$$

where μ_{step} is a suitable chosen step size. The gradient is evaluated at $L_m^{(n),[k]}$.

As mentioned earlier, most of the terms in $Q(\boldsymbol{L}, \boldsymbol{L}^{[k]})$ do not depend on $\boldsymbol{L}$ and only equation (6.17) has to be considered for the derivative of the Q-function. We observe that in (6.17) only the logarithmal factor depends on the variable and that the other factor is evaluated at a fixed $\boldsymbol{L}^{[k]}$. With the independence property from (6.12), this yields

$$\frac{\partial Q(\boldsymbol{L}, \boldsymbol{L}^{[k]})}{\partial L_m^{(n)}} = \sum_{t,\ \boldsymbol{s}_t} p(\boldsymbol{s}_t|\boldsymbol{y}; \boldsymbol{L}^{[k]}) \frac{\partial}{\partial L_m^{(n)}} \log\left(p(y_t^{(n)}|\boldsymbol{s}_t; \boldsymbol{L}^{(n)})\right). \tag{6.19}$$

We identify $p(\boldsymbol{s}_t|\boldsymbol{y}; \boldsymbol{L}^{[k]})$ as the a posteriori probability distribution for the state of the encoder at time t, given all observations and the current guess of the LLRs. It was shown in Chapter 2, Section 2.4.5 how to efficiently compute this quantity. A number of computationally efficient and numerically stable forward-backward recursions were developed to calculate this distribution since it is of great interest in coding and signal processing (cf. [EM02] for an overview).

For the evaluation of (6.19), we need an analytical expression for the derivative of the conditional log-likelihood of $y_t^{(n)}$. At this point, the choice of log-likelihood ratios for the parameters turns out to be an elegant solution: remember that, assuming a BSC channel model, $y_t^{(n)}$ is a modulo 2 sum (Eq. (6.5)):

$$y_t^{(n)} = \sum_{m=0}^{M}\!\oplus s_{t_m} g_m^{(n)} + \varepsilon. \tag{6.20}$$

Using the boxplus notation, the LLR of $y_t^{(n)}$ is given as

$$L(y_t^{(n)}|\boldsymbol{s}_t) = \sum_{\substack{m=0\\ s_{t_m}=-1}}^{M}\!\boxplus\, L_m^{(n)} \boxplus L(\varepsilon) = 2\tanh^{-1}\left(\prod_{\substack{m=0\\ s_{t_m}=-1}}^{M} \tanh(\frac{L_m^{(n)}}{2})\tanh(\frac{L(\varepsilon)}{2})\right).$$

Note that $L(y_t^{(n)}|\boldsymbol{s}_t)$ inherently depends on the parameter $\boldsymbol{L}^{(n)}$ which shall not explicitly appear in the notation for the sake of simplicity. For the BSC channel model with transition probability p_ε we set

$$L(\varepsilon) = \log\left(\frac{1-p_\varepsilon}{p_\varepsilon}\right). \tag{6.22}$$

Introducing LLRs allows for a convenient way to process soft channel values, and different expressions of $L(y_t^{(n)}|\boldsymbol{s}_t)$ for the Gaussian or the multiplicative fading channel can be derived [HOP96]. This is an obvious advantage of our probabilistic approach compared to the algebraic solution presented in [Fil97] and is expected to lead to better results alike decoding methods that process soft-values outperform their algebraic counterparts. In the following, we concentrate on the BSC channel model.

6.2.4 Derivation of the gradient

In order to yield an analytical solution for (6.19), we have to evaluate an expression of the form

$$\frac{\partial}{\partial\theta_m}\log\left(p(y;\boldsymbol{\theta})\right), \tag{6.23}$$

i.e., the derivative of a log-likelihood depending on a parameter vector $\boldsymbol{\theta}$ with respect to a single parameter θ_m.

Substituting

$$p(y=\pm 1;\boldsymbol{\theta}) = \frac{e^{\pm L(y;\boldsymbol{\theta})/2}}{e^{L(y;\boldsymbol{\theta})/2}+e^{-L(y;\boldsymbol{\theta})/2}} \tag{6.24}$$

in (6.23) and carrying out some simple calculations, we find that

$$\frac{\partial\log\left(p(y=\pm 1;\boldsymbol{\theta})\right)}{\partial\theta_m} = \pm p(y=\mp 1;\boldsymbol{\theta})\frac{\partial}{\partial\theta_m}L(y;\boldsymbol{\theta}). \tag{6.25}$$

In the log-domain, Eq. (6.23) now reads as

$$\frac{\partial}{\partial\theta_m}\log\left(p(y=\pm 1;\boldsymbol{\theta})\right) = \frac{\pm e^{\mp L(y;\boldsymbol{\theta})/2}}{e^{L(y;\boldsymbol{\theta})/2}+e^{-L(y;\boldsymbol{\theta})/2}}\frac{\partial L(y;\boldsymbol{\theta})}{\partial\theta_m}. \tag{6.26}$$

It remains to find the derivative of $L(y;\boldsymbol{\theta})$ which we identify as $L(y_t^{(n)}|\boldsymbol{s}_t)$. As shown in Eq. (6.21), the LLR of the encoder output is a boxplus-sum:

$$\frac{\partial L(y_t^{(n)}|\boldsymbol{s}_t)}{\partial L_m^{(n)}} = \frac{\partial}{\partial L_m^{(n)}}\sum_{j,s_{t_j}=-1}\!\!\boxplus\, L_j^{(n)} \boxplus L(\varepsilon) \tag{6.27a}$$

$$= \frac{\prod\limits_{j:(j\neq m\wedge s_{t_j}=-1)}\tanh\left(L_j^{(n)}/2\right)\tanh\left(\frac{L(\varepsilon)}{2}\right)}{1-\left(\prod\limits_{j:(s_{t_j}=-1)}\tanh\left(\frac{L_j^{(n)}}{2}\right)\tanh\left(\frac{L(\varepsilon)}{2}\right)\right)^2}\left(1-\tanh\left(\frac{L_m^{(n)}}{2}\right)^2\right) \tag{6.27b}$$

$$= \frac{\bar{\varepsilon}}{\bar{g}_m^{(n)}}\frac{E\{(g_m^{(n)}-\bar{g}_m^{(n)})^2\}}{1-(\bar{\varepsilon}\prod\limits_{j:(s_{t_j}=-1)}\bar{g}_j^{(n)})^2}\prod_{j:(s_{t_j}=-1)}\bar{g}_j^{(n)}. \tag{6.27c}$$

The gradient is zero if $s_{t_m}=+1$ as (6.21) does not depend on $L_m^{(n)}$ in this case. Equation (6.27c) depends only on soft bits and the variance of $g_m^{(n)}$, where we used the relation

$$E\{(g_m^{(n)}-\bar{g}_m^{(n)})^2\} = 1-(\bar{g}_m^{(n)})^2 = 2\frac{\partial\bar{g}_m^{(n)}}{\partial L_m^{(n)}}. \tag{6.28}$$

Our algorithm starts with an initialization of $\boldsymbol{L}^{[0]}$. Setting the LLRs of the parameters close to zero indicates maximum uncertainty. One EM iteration is performed by multiplying the terms obtained from Eqs. (6.26) and (6.27c) with the state probabilities $p(\boldsymbol{s}_t|\boldsymbol{y};\boldsymbol{L})$, summing up over all possible states and timesteps. The gradient of the Q-function (Eq. (6.19)) is then used to update the parameter estimates according to Eq. (6.18), where a suitable step size μ_{step} has to be chosen. Iterations are performed until a certain stop condition is met such as the exceeding of a maximum number of iterations or undershooting of minimum increase in likelihood. The procedure is summarized in Algorithm 6.1.

Algorithm 6.1: EM algorithm for estimating convolutional encoder parameters

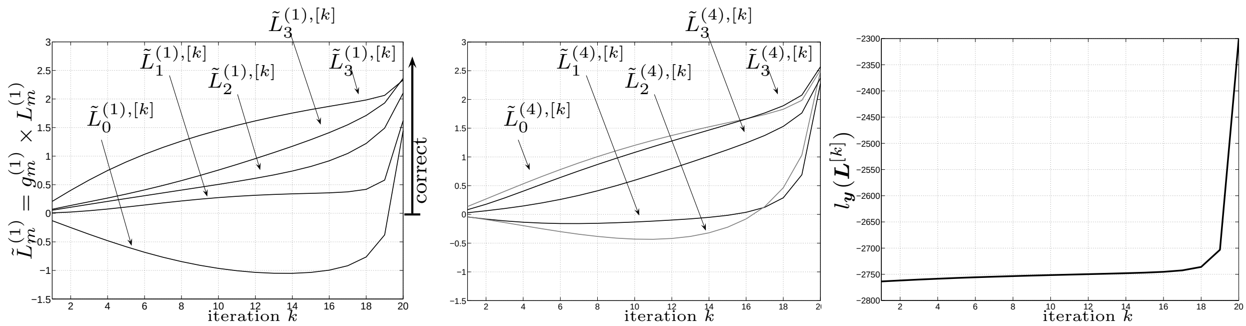
Require: Observation $\boldsymbol{y}$, rate $\frac{1}{N}$, memory M convolutional code, noise distribution p_ε.

Set $k = 0$
Choose μ_{step}
Initialize $\boldsymbol{L}^{[0]}$
while stopcondition = **false** **do**
    $\boldsymbol{L}^{[k+1]} = \boldsymbol{L}^{[k]}$
    for all t **and** $\boldsymbol{s}_t$ **do**
        $\beta_{\boldsymbol{s}_t}^{[k]} = p(\boldsymbol{s}_t|\boldsymbol{y};\boldsymbol{L}^{[k]})$ (forward-backward)
        for all $n = 1,\ldots,N$ **and** $m = 0,\ldots,M$ **do**
            $\gamma_{t,m}^{(n)} = \frac{\partial}{\partial L_m^{(n)}} \log\left(p(y_t^{(n)}|\boldsymbol{s}_t;\boldsymbol{L}^{(n),[k]})\right)$ with (6.26), (6.27c)
            $L_m^{(n),[k+1]} \leftarrow L_m^{(n),[k+1]} + \mu_{\text{step}}\, \beta_{\boldsymbol{s}_t}^{[k]} \gamma_{t,m}^{(n)}$
        end for
    end for
    $k \leftarrow k+1$
end while
return $\hat{\boldsymbol{g}} = \text{sign}\left(\boldsymbol{L}^{[k-1]}\right)$

6.2.5 Simulation results

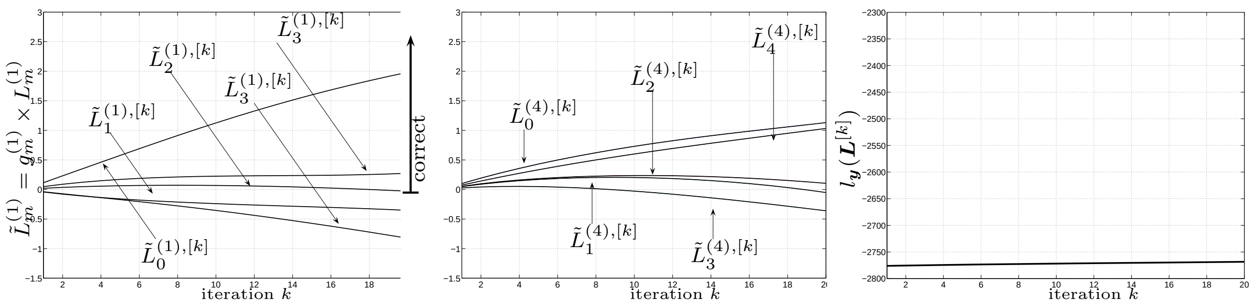
Information bits were encoded using a rate $\frac{1}{4}$ non-systematic encoder with $M = 4$ ($[53, 51, 63, 73]$ in octal form) and sent over a BSC with transition probability $p_\varepsilon = 0.1$. For each output, 1000 noisy bits were observed, i.e., we expect 100 wrong bits per output. A brute force approach would evaluate and compare 2^{20} likelihood values. In Figure 6.4(a) and 6.4(b) we show the progression of the LLRs of the parameters $\boldsymbol{g}^{(1)}$ and $\boldsymbol{g}^{(4)}$ over 20 EM iterations. On the y-axis, the LLRs are multiplied with the true values, i.e., $\tilde{L}_m^{(n)} = L_m^{(n)} g_m^{(n)}$, for visualization purposes. As a result, positive values indicate the correct estimation of a parameter after hard decision, whereas negative values indicate a false estimate. Upon initialization, several parameters start to move in the wrong direction, reaching even high reliability. After a certain number of iterations, however, the algorithm is able to correct those values, and at iteration 19 the LLRs predict the correct encoder. Once all LLRs are correct, their absolute values are highly increased towards the correct direction. Since absolute values of the LLRs reflect the reliability about an estimate, we stop iterating when no unreliable values are left in $\boldsymbol{L}^{[k]}$. Figure 6.4(c) shows the corresponding log-likelihood $\log\left(p(\boldsymbol{y};\boldsymbol{L}^{[k+1]})\right)$. A constant increase is observed and a steep increase for the last iterations where the algorithm found the correct configuration.

In Figure 6.5, we show the results for the same simulation parameters for the case that the algorithm converges to a false solution. The $\tilde{L}_m^{(n)}$ are shown for outputs $n = 2$ and $n = 4$. At iteration 20, some of the LLRs $\tilde{L}_m^{(n)}$ still show negative values indicating an incorrect estimate after hard decision. The corresponding log-likelihood does not significantly increase, and it can be assumed that the algorithm got stuck in a local minimum. Nevertheless, the small absolute values of the LLRs indicate that the estimate is not reliable. Observing an unreliable convergence, the algorithm could be restarted from a new random initial point until a sufficient level of certainty is reached. Consequently, by monitoring the reliability of convergence, it is possible to keep the rate of false positive estimates very low.



(a) LLR of $\boldsymbol{g}^{(1)}$ over iterations. (b) LLR of $\boldsymbol{g}^{(4)}$ over iterations. (c) Log-likelihood over iterations.

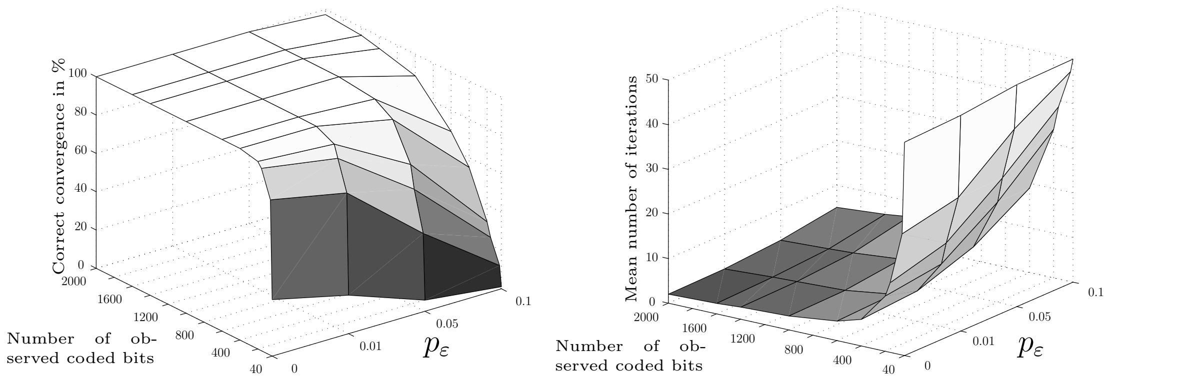
Figure 6.4: Sample of a successful estimation after 20 iterations of a rate $\frac{1}{4}$, $M = 4$, non-systematic encoder from 4000 observed coded bits and $\varepsilon = 0.1$.



(a) LLR of $\boldsymbol{g}^{(2)}$ over iterations. (b) LLR of $\boldsymbol{g}^{(4)}$ over iterations. (c) Log-likelihood over iterations.

Figure 6.5: Non-successful estimation after 20 iterations of a rate $\frac{1}{4}$, $M = 4$, non-systematic encoder from 4000 observed bits and noise level $\varepsilon = 0.1$.

Figure 6.6(a) shows the rate of correct convergence for different numbers of observed coded bits and varying noise levels in case of *systematic* convolutional encoding. Information streams were encoded with a rate $\frac{1}{4}$, memory $M = 4$, optimal distance profile code with generator polynomial $[\boldsymbol{g}^{(2)}, \ldots, \boldsymbol{g}^{(4)}] = [56, 62, 72]$ (the first output being systematic) [JZ99]. Correct convergence refers to $\boldsymbol{L}$ indicating high reliability (here defined as $|L_m^{(n)}| \geq 2 \ \forall m, n$) *and* correct estimate of $\boldsymbol{g}$ after hard decision. Figure 6.6(b) shows the corresponding mean number of iterations required until reliability is achieved. An error was declared when convergence in the defined sense was not achieved within 50



(a) Percentage of correct convergence. (b) Required iterations until convergence.

Figure 6.6: Percentage of correct convergence and mean number of required iterations for different numbers of observed coded bits and varying noise level in case of systematic convolutional encoding.

iterations. We observe high rates of convergence even at high noise levels for a relatively small number of observed coded bits. For example with 400 coded bits at a noise level $p_\varepsilon \leq 0.05$ we are always able to recover the correct encoder.

Considering non-systematic encoders, we observed poor convergence rates using the EM based approach. The success seems to heavily depend on the starting value, and the gradient ascent often causes the algorithm to get stuck in a local maximum immediately. We show how to use global optimization techniques to overcome this problem in the next section.

6.2.6 Inference via global optimization

In this Section, we use the global optimization approach based on stochastic differential equations presented by Schäffler [Sch97] to reconstruct encoder taps. The method has been previously applied to error correcting codes by Schäffler and Hagenauer in the framework of maximum likelihood decoding of BCH and Turbo codes [SH97].

Briefly, the global approach works as follows: define $l_{\boldsymbol{y}}(\boldsymbol{L}) \doteq -\log(p(\boldsymbol{y};\boldsymbol{L}))$ as the likelihood function depending on the tap LLRs, and denote

$$\boldsymbol{L}^* : l_{\boldsymbol{y}}(\boldsymbol{L}^*) \leq l_{\boldsymbol{y}}(\boldsymbol{L}),\ \forall \boldsymbol{L} \tag{6.29}$$

the global minimizer of this function. We consider a random variable L whose density is given by

$$p_{\mathsf{L}}(\boldsymbol{L}) = \frac{\exp(-2\frac{l_{\boldsymbol{y}}(\boldsymbol{L})-l_{\boldsymbol{y}}(\boldsymbol{L}^*)}{\epsilon^2})}{\int_{\mathbb{R}^{N(M+1)}} \exp(-2\frac{l_{\boldsymbol{y}}(\boldsymbol{L})-l_{\boldsymbol{y}}(\boldsymbol{L}^*)}{\epsilon^2})d\boldsymbol{L}}. \tag{6.30}$$

This density has the interesting property that it has its *global maximum* where $l_{\boldsymbol{y}}(\boldsymbol{L})$ has its *global minimum*. If we were able to draw samples from L having this density, we could expect that a non-negligible fraction of samples will be in proximity of the global minimizer of $l_{\boldsymbol{y}}(\boldsymbol{L})$. Consider the following stochastic differential equation:

$$\boldsymbol{L}^{\epsilon}(t) = \boldsymbol{L}_0^{\epsilon} - \int_0^t \frac{\partial}{\partial \boldsymbol{L}} l_{\boldsymbol{y}}(\boldsymbol{L}^{\epsilon}(\tau))d\tau + \epsilon \mathbf{B}(t), \tag{6.31}$$

where $\boldsymbol{L}_0^{\epsilon}$ is an arbitrary starting point and $\mathbf{B}(t)$ denotes the $N(M+1)$ dimensional Brownian motion. The integral can be interpreted as the deterministic part that moves the solution in the direction of the gradient. The stochastic part introduces randomness (noise) in order to avoid the method getting stuck in local minima. Unlike for local minimization, successive decrease of the function is not desired. The key to global optimization is that the solution to integral Equation (6.31) can be shown to have the following properties [Sch97]:

1. The density of the random variable $\boldsymbol{L}^{\epsilon}(t)$ converges for $t \to \infty$ to the density given in Eq. (6.30) (Note that the density is independent of the starting value $\boldsymbol{L}_0^{\epsilon}$).
2. For any $\gamma > 0$: $P(\inf\{t : ||\boldsymbol{L}^{\epsilon}(t) - \boldsymbol{L}^*|| < \gamma\} < \infty) = 1$, i.e., any function $\boldsymbol{L}^{\epsilon}(t)$ approaches the global minimum arbitrarily close within finite time, independent of $\boldsymbol{L}_0^{\epsilon}$.

Based on these properties, it is sufficient to numerically find a solution to Eq. (6.31) for any starting value $\boldsymbol{L}_0^{\epsilon}$. Here, we used the semi-implicit Euler method as described in [Sch97].

In order to apply this method in our framework, we must be able to calculate the likelihood function $l_{\boldsymbol{L}}(\boldsymbol{y})$ and its derivative. Straightforward calculation of $p(\boldsymbol{y}; \boldsymbol{L})$ is numerically unstable for long sequences $\boldsymbol{y}$. However, stable and efficient recursions for calculating $p(\boldsymbol{y}; \boldsymbol{L})$ were presented in [CBM98]. These involve calculation of the likelihood $p(\boldsymbol{y}_t|\boldsymbol{s}_t)$ and its derivatives. These quantities are easily calculated using log-likelihood algebra as presented previously in Section 6.2.4.

Algorithm 6.2 summarizes how the procedure is applied to the data stream. We start with a randomly sampled initial parameter vector. After each iteration of the numerical integration, the current likelihood is compared with the initial one. If the likelihood has increased beyond a certain threshold, it is tested whether the corresponding hard decision of the current estimate is consistent with the observed data. If this is not the case, the algorithm is restarted from a new random initialization.

6.2.7 Extension to quarternary alphabet

In order to apply the methods described above to symbolic datastreams, such as DNA sequences, we have to extend the log-likelihood formalism to the non-binary case. In general, the nth output of the encoder is a symbolic sequence $\boldsymbol{y}^{(n)}$ of length T, where $y_t^{(n)}$ are elements from a finite set $\mathcal{Z}$ of cardinality $|\mathcal{Z}| = q$. In the considered framework, addition and multiplication $(+, \cdot)$ for the elements in $\mathcal{Z}$ must be defined and $\mathcal{Z}$ must be closed under these operations. We assume that there is a 0 element for which $a \cdot 0 = 0\ \forall a \in \mathcal{Z}$. We denote the elements of $\mathcal{Z}$ as $\{\alpha_0, \alpha_1, .., \alpha_{q-1}\}$, where α_0 denotes the 0 element.

Algorithm 6.2: Encoder estimation using global optimization

Require: Noisy coded bits $\boldsymbol{y}$ and known noise distribution p_ε.

```
while 1 do
    sample a random starting point L_0^ε
    S = ∅, t_0 = 0, k = 0
    Choose k_stop
    while k < k_stop do
        sample from L^ε(t) using the Euler-integration method described in [Sch97] and
        add samples to S
                S ← S ∪ L^ε(t_k)
        t_k ← t_k + △t_k
        k ← k + 1
    end while
    find current maximum likelihood and corresponding estimate
        l_max = max_{L∈S} {l_y(L)},   L_max = arg max_{L∈S} {l_y(L)}
    if hard decision of L_max consistent with y then
        return L_max
    end if
end while
```

$$S \leftarrow S \cup \boldsymbol{L}^\epsilon(t_k)$$

$$l_{\max} = \max_{\boldsymbol{L}\in S}\{l_{\boldsymbol{y}}(\boldsymbol{L})\}, \quad \boldsymbol{L}_{\max} = \arg\max_{\boldsymbol{L}\in S}\{l_{\boldsymbol{y}}(\boldsymbol{L})\}$$

As we move to the non-binary case, log-likelihood ratios (LLR) are extended to $q$ary log-likelihood ratio *vectors*. In [WSM04, Ber01], these vectors are defined by normalizing the probabilities to the probability of the 0 element (any other element could be used):

Definition 6.2.2 (LLR vector) For a random variable X with realizations $x \in \mathcal{Z}$, the LLR vector $\boldsymbol{L}(x) \in \mathbb{R}^{q-1}$ is defined as

$$\boldsymbol{L}(x) = \left[\log\left(\frac{P(x=\alpha_1)}{P(x=\alpha_0)}\right), ..., \log\left(\frac{P(x=\alpha_{q-1})}{P(x=\alpha_0)}\right)\right], \tag{6.32}$$

where the following notation applies:

$$\boldsymbol{L}(\alpha_i) = \boldsymbol{L}(x=\alpha_i) = \log\left(\frac{P(x=\alpha_i)}{P(x=\alpha_0)}\right), \; i = 1, .., q-1 \tag{6.33a}$$

$$\boldsymbol{L}(\alpha_0) = \boldsymbol{L}(x=\alpha_0) = 0 \tag{6.33b}$$

□

It is easily verified that the probabilities can be recovered from

$$P(x=\alpha_i) = \frac{e^{\boldsymbol{L}(x=\alpha_i)}}{\sum_{k=0}^{q-1} e^{\boldsymbol{L}(x=\alpha_k)}}. \tag{6.34}$$

The generalized non-binary boxplus and its derivative are derived in Appendix E. In the convolutional encoder, each tap is now modeled as a $(q-1)$ary LLR vector

$$\boldsymbol{L}_m^{(n)} = \left[\log\left(\frac{P(g_m^{(n)}=\alpha_1)}{P(g_m^{(n)}=\alpha_0)}\right), ..., \log\left(\frac{P(g_m^{(n)}=\alpha_{q-1})}{P(g_m^{(n)}=\alpha_0)}\right)\right]. \tag{6.35}$$

The conditional state probability for the nth output of the encoder at time t is given by

$$p(y_t^{(n)} = \alpha_j | \boldsymbol{s}_t) = p(\sum_{m=0}^{M} s_{t_m} g_m^{(n)} = \alpha_j), \tag{6.36}$$

with the corresponding LLR vector as derived in Eq. (E.3c). Finally, the derivative of the log-likelihood with respect to $\boldsymbol{L}(x_m = \alpha_j)$ is calculated as

$$\frac{\partial\ p(y_t^{(n)} = \alpha_j | \boldsymbol{s}_t)}{\partial L_m^{(n)}(\alpha_i)} = \frac{e^{L(y_t^{(n)}=\alpha_j)}}{(\sum_k e^{L(y_t^{(n)}=\alpha_k)})^2} \left(\frac{\partial L(y_t^{(n)} = \alpha_j)}{\partial L_m^{(n)}(\alpha_i)} \sum_k e^{L(y_t^{(n)}=\alpha_k)} \right. \\ \left. - \sum_k e^{L(y_t^{(n)}=\alpha_k)} \frac{\partial L(y_t^{(n)} = \alpha_k)}{\partial L_m^{(n)}(\alpha_i)} \right). \tag{6.37}$$

Algorithm 6.2 is then easily extended to symbolic sequences over $\mathcal{Z}$.

6.2.8 Simulation results

Figure 6.7 shows simulation results for the binary and quarternary ($q = 4$) case for non-systematic encoders with $N = 3$ and varying memory M. Optimum distance profile encoders, taken from [JZ99, Chapter 8], were used in the binary case. Quarternary encoders were randomly selected. The global optimization procedure described above was applied to the observed data streams. By tuning the stopping criteria, it is possible to achieve a very low false positive rate. In fact, we never obtained a wrong encoder estimate using the procedure outlined in Algorithm 6.2.

The Figure shows the mean number of iterations required until a consistent solution was found, over the number of observed coded symbols. Each point in the Figure represents the average of 1000 simulated data streams. A common characteristic observed is that the number of iterations first rapidly decreases with increasing number of symbols. After that, the curves show a saturation effect.

The noise levels in all simulations was $p_\varepsilon = 0.01$. For comparison we also show one example with a highly increased noise level ($p_\varepsilon = 0.1$ for the quarternary case, $M = 1$). The comparison shows that reconstruction is also possible in the high noise regime, and that more iterations or longer streams are required in this case.

6.3 Application to DNA sequence data

The previous sections introduced likelihood based methods for reconstructing convolutional encoders from observed noisy codesymbols. Various other methods focusing on linear block- and convolutional codes were proposed previously [Clu06,LTTL96,Val01,Fil97]. However, we are facing the following difficult issues when applying these methods to DNA sequence data in order to find error correcting codes in the genome:

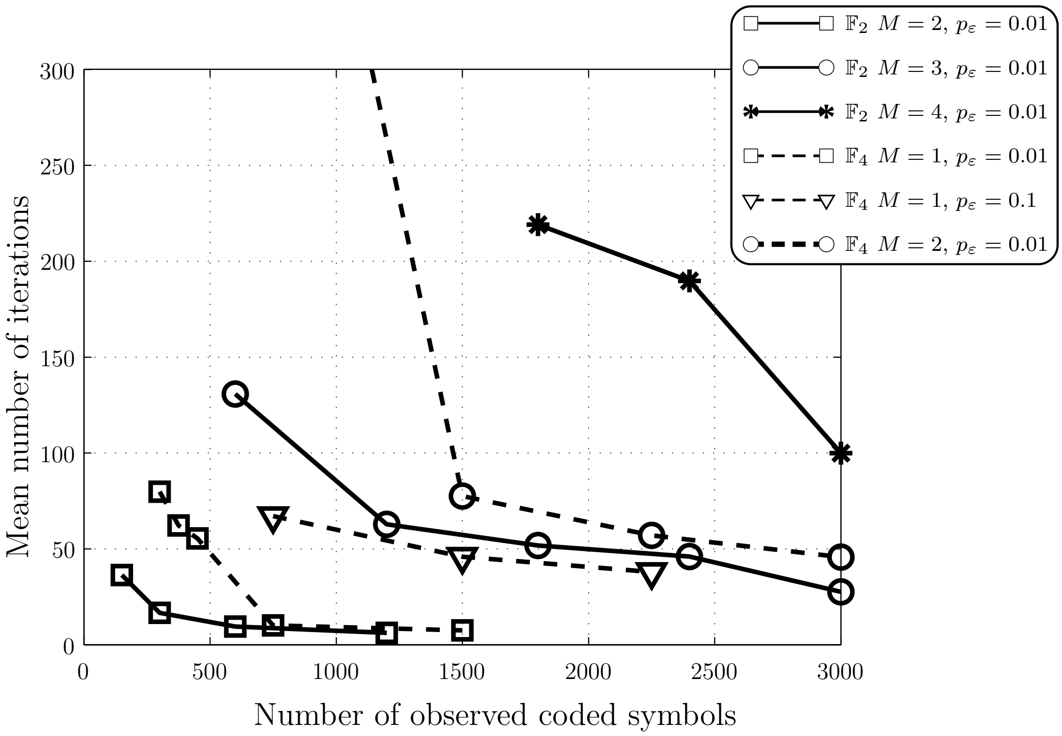


Figure 6.7: Simulation results of encoder reconstruction based on global optimization. For the quarternary case, computations were carried out in the Galois field $\mathbb{F}_4$.

1. Where to apply?
   The length of the human genome is 3×10^9 bases. Assuming that only parts of the genome are encoded by a channel coding scheme, one has to identify candidate sequences to apply reverse engineering methods to. Previous approaches concentrated on coding regions in genes [LTTL96, Ros06] without giving any justification why they chose a particular region or gene.
2. What is the algebra of DNA?
   DNA is a sequence of symbols from a quarternary alphabet. There is, however, no indication that these four symbols form an algebraic structure that allows to carry out mathematical operations such as addition and multiplication within this alphabet. In other words, what is the result of A + C or G× T? Which nucleotide is the zero element among A, C, G, T? It is also not clear how to interprete operations like addition or multiplications biochemically.
3. What are the parameters of the code?
   Most approaches to the code reverse engineering problem assume that at least a few of the parameters, such as rate, block length etc., are known. This is of course not the case for DNA sequences. In particular for convolutional codes, the memory and rate of the encoder, and the synchronization of the code bits in the data stream have to be determined.

We believe that we can provide a reasonable answer at least to the first question: recently identified ultraconserved [BPM+04] regions were found to be 100% identical over geological timescales, i.e., among the human, rat, and mouse genome. It seems that sequence conservation in these regions is of primary interest to the cell. If Battail's hypothesis is true, an error correcting code should be present in these regions.

Sequence data was obtained from a recently developed database (UCbase) of ultraconserved regions [TFV+08]. We analyzed the longest of those ultraconserved DNA region (No. 462 in UCbase, located on human X chromosome), 779 nucleotides in length, using our reconstruction algorithm. DNA symbols A, C, G, T were assumed to be elements from the finite field $\mathbb{F}_4$. Algorithm 6.2 was applied to the ultraconserved sequence and a random sequence, derived from the ultraconserved one by random shuffling. The algorithm was applied to all possible mappings of the finite field elements to A, C, G, T (24 possible combinations), memory $M = 1, 2$, number of encoder outputs $N = 2..4$, and possible number of frameshifts $N_S = 0..N-1$ yielding an overall number of $(24 \times 2 \times (2+3+4) = 432)$ parameter combinations. Since the sequence is conserved in human, rat, and mouse, it can be assumed that the noise level is very close to zero. For each possible finite field mapping, and each possible combination of parameters (M, N, N_S), 500 samples were taken using the implicit Euler integration method and $k_{stop} = 15$. The maximum likelihood observed was stored for both, the ultraconserved and the randomly shuffled sequence. Figure 6.8 shows the log-likelihood ratios[1] over all possible parameter configurations. If

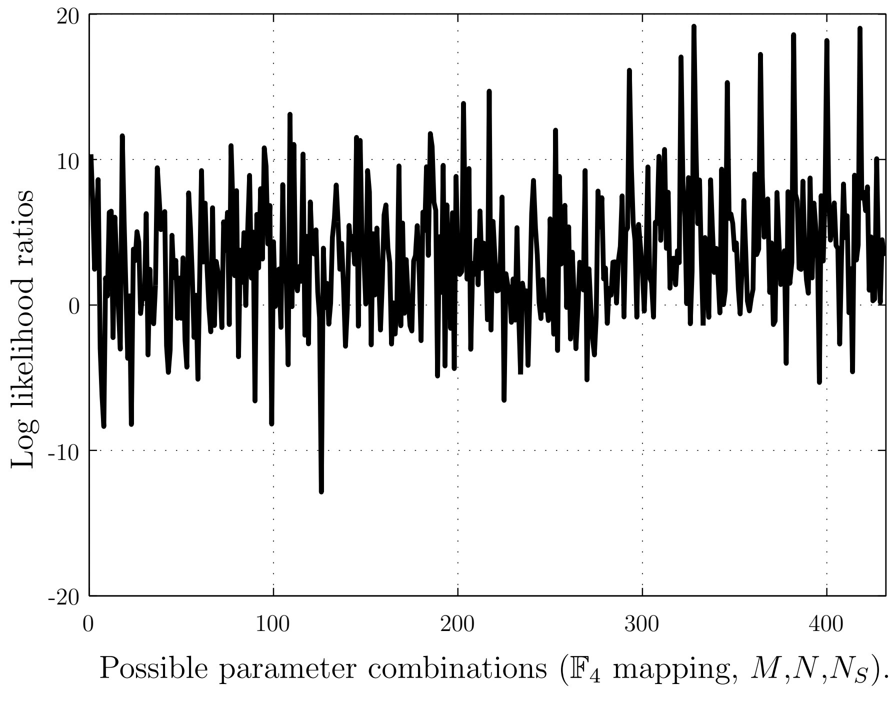


Figure 6.8: Log ratios of the observed maximum likelihood obtained after 500 samples for an ultraconserved and a random sequence. Within the considered ranges of parameters, one ratio is shown for each possible combination of encoder parameters.

a convolutional code was present in the ultraconserved sequence, its likelihood should be much higher than that of the random sequence, resulting in a very high peak. The Figure, however, does not indicate evidence for the ultraconserved element belonging to

[1] We refer to the ratios of likelihood for the observed sequences. This is not to be confused with the LLRs of encoder taps considered previously in this chapter.

a convolutional code having the considered parameters (assuming DNA uses finite field $\mathbb{F}_4$ computations).

Here, we only analyzed a single candidate sequence, assuming a particular coding scheme (convolutional codes). A complete approach needs to consider many conserved regions, and has to apply additional methods [Clu06, LTTL96, Val01] to detect all kinds of codes. More coding scheme parameters have to be considered and mathematical structures of the A, C, G, T alphabet tested. This seems to be, however, not very practical. A more promising approach must take biochemical considerations into account. One should first ask: what kind of computations can be realized with the nucleotides A, C, G, T on the biochemical level? What kind of code constructions are possible using such computations? These investigations may lead to interesting new results for both, coding theory and molecular biology.

6.4 Summary

We considered the problem of estimating the parameters of a convolutional encoder given only the noisy coded symbols. We were motivated by the question whether there is an error correcting code in the DNA base sequence. Most of the previous work on error correcting codes in DNA rely on theoretical considerations and models. Few constructive results how to detect such codes were presented, all of those concentrating on linear block codes. In an attempt to approach the DNA error code hypothesis more systematically, we considered the formal code reverse engineering problem in which an observer wants to detect the coding scheme from intercepted noisy codewords. We thereby focused on a particular class of error correcting codes, convolutional codes. Algebraic methods for the reverse engineering problem in this framework were considered before [Ric95, Fil97]. Here, we presented an iterative, probabilistic approach based on the EM algorithm, combining soft-taps, modeled as log-likelihood ratios (LLRs), and LLR algebra with blind signal processing techniques.

Simulation results on distorted data streams suggest that high rates of correct reconstruction can be achieved even at high noise levels in case of systematic encoder structures. The algorithm, however, was found to exhibit poor convergence rates for non-systematic encoders, presumably caused by local maxima of the likelihood function. We therefore presented an extended approach based on stochastic global optimization to overcome this limitation. Furthermore, we provided the generalization to the non-binary case, where symbols from a q-ary alphabet are encoded. Simulation results suggest that reverse engineering encoder structures is possible within few iterations (compared to brute force maximum likelihood), at a very low false positive rate, for both, the binary and quarternary case.

We believe that our method has several advantages compared to the algebraic approach: 1) Our method is applicable at high noise levels. 2) The LLR based approach is easily extended to AWGN or fading channels and thus able to process soft values. 3) A priori information about parameters is easily incorporated by adapting initial LLRs. In this way, knowledge on the design rules of convolutional encoders can help to choose good starting values. In a similar fashion, constraints on the state sequence imposed by tail-biting codes could be incorporated in our algorithm. 4) Opposed to the algebraic method, our probabilistic approach allows for the co-estimation of the noise level p_ε by simply treating it as an additional parameter.

An outline of previous work addressing the genomic code hypothesis was given. In particular, the hypothesis of Battail and experimental evidence for a previously unknown error correcting mechanism in *Arabidopsis* were discussed. Known DNA repair mechanisms were analyzed from a coding theoretic perspective. Finally, we presented an application of our method to DNA sequence data. An ultraconserved region, 100% identical in the human, rat, and mouse genome was chosen as a test candidate. Our algorithm was applied using a variety of different parameters, and the maximum likelihood of the ultraconserved sequence was compared to the maximum likelihood obtained with a random sequence. Our results showed that, under the chosen parameters and assumptions, there is no evidence that the ultraconserved region belongs to an error correcting code. We also discussed the difficulties of applying such reverse engineering methods to DNA sequences.

6.5 Future research

The question whether an error correcting code in the genome exists remains unresolved. Many known error correcting mechanisms exist across different layers of molecular interactions that are clearly linked to enable a concerted DNA-damage response. The information theoretic modeling and analysis of cellular DNA-damage repair mechanisms represents a great challenge. A first step in this direction could extend Battail's model of evolution to a more realistic one taking into account cell death and natural selection. Another approach could try to synthetically (*in silico*) create a model which is then further refined to conform to biological reality.

While algorithms for a broad class of codes exist, reverse engineering of such codes on the DNA sequence level remains difficult because of the reasons discussed in Section 6.3. In our oppinion, these issues need to be addressed before going further in this direction. A reasonable starting point may be to ask "what is the algebra of DNA - and what kind of codes can be built using the pertinent possible computations?". Such an approach requires a deep understanding of the biochemical properties of DNA and, certainly, proper experimental validation.

There is also room for extensions as far as the technical reverse engineering problem is concerned. Here, we modeled noise as an additive discrete random variable. However, the probabilistic nature of our approach allows for a much broader class of channel models to be considered, such as additive Gaussian or fading models. We throughout assumed that key parameters of the encoding scheme such as memory and rate were known at the receiver and application to biological data was done using a brute force approach. A priori or joint estimation of these parameters could help to significantly reduce the running time of the reverse engineering algorithms.

7

List-decoding methods for algebraic gene network models

Gene regulatory networks[1] (GRNs), and in particular, transcriptional networks, are a topic of outstanding importance in the field of systems biology [de 02]. This interest mainly comes from the fact that complex interaction patterns of transcription factors that regulate the expressions of genes (cf. Chapter 3, Section 3.4) provide important cues for understanding disease development and progression on the level of a single cell.

In this chapter we present a coding theoretic approach to the reverse engineering of the dynamics of gene networks, cast within a recently proposed algebraic modeling framework [JLSS07]. We show how list-decoding can be applied to expression data in order to account for stochasticity of networks and noise. We present application to both, simulated networks and the global *E. coli* network based on available expression data.

Sections 7.1 and 7.2 give a short introduction to the problem, provide the used notation and necessary definitions and present common modeling frameworks. In Section 7.2.6, the algebraic model considered in the remainder of this chapter is introduced. Section 7.3 formulates the reverse engineering problem and presents the algorithm developed in [JLSS07] based on tools from computational algebra. In 7.4 we establish the connection to coding theory and show how decoding algorithms, especially list-decoding, can be applied to fit expression data in a stochastic framework. Section 7.6 then describes our algorithm in detail and provides theoretical and simulation analysis. Application to biological data is presented in Section 7.7, where we test our method on an *E. coli* sub-network as well as on the whole transcription factor network of *E. coli*. Section 7.8 discusses the obtained results and gives directions for possible extensions.

[1] In this chapter, we focus on transcriptional control networks, simply referred to as gene regulatory networks.

7.1 Background

Genes interact through their corresponding RNA and protein products, involved in sophisticated promoting, enhancing, and suppression processes. The bio-chemical processes supporting the expression of a gene occur with different rates and at different times and are inherently random. As a consequence, gene expressions and interaction profiles can be inferred only when averaged over large number of cells and over sufficiently long time intervals.

Modeling the averaged dynamics of GRNs is largely facilitated by high-throughput data obtained from DNA *microarrays* (cf. Section 7.6.1). Microarrays are biological sensors capable of identifying cDNA (complementary DNA) targets, corresponding to RNA transcripts of genes, through a process known as selective hybridization. Hybridization represents stacked bonding between Watson-Crick complementary bases on two DNA strands that leads to stable DNA duplexes.

The current costs of DNA microarray measurements are too large to allow for collecting a sufficiently large number of sample points needed for accurate time-series analysis of gene expressions. Consequently, one is facing the very difficult task of reverse engineering networks based on small sample sets that may contain a fairly large amount of noise.

The coupled dynamics of gene expression patterns are most accurately modeled via systems of coupled differential equations, derived by analyzing involved biochemical reactions of the cell cycle [de 02]. Besides their accuracy, these modeling approaches also have the advantage that they lend themselves to stochastic extensions that capture uncertainties in biological systems [Wil06]. However, systems of differential equations have very high parametric complexity and can often only be built *bottom-up*, i.e., by looking at a detailed list of reactions involved in gene transcription and by determining parameters of the model through extensive experimental studies.

An alternative approach to modeling gene networks relies on very coarse approximations of the system's dynamics; this coarse approximation allows only for capturing qualitative features of gene networks, rather than their exact dynamics. This approach was pioneered by Kauffman, who proposed using Boolean networks (BN) as models of gene regulatory networks (GRN) [Kau69]. Probabilistic Boolean networks (PBN) represent stochastic extensions of Boolean models [SDKZ02]. In a PBN, a *list* of Boolean functions is associated with each node in the network, and each time the state of a gene is updated, only one of these functions is randomly chosen to compute the new state of the gene [SDKZ02]. Many generalizations of these models exist, including hybrid systems [TT98], finite state linear models [BS03], and deterministic and probabilistic finite dynamical systems [AGM04, JLSS07].

Significant research effort has been devoted to fitting discrete models of GRNs to high throughput measurements in a manner that allows for good predictive descriptions of experimental data. Such a reverse engineering approach is fundamentally different from the bottom-up methods as it tries to select a model based purely on the observed data, without making use of detailed biochemical information. Early approaches to reverse engineering of GRNs under both, Boolean and PBN models have assumed that time series data from a network can be perfectly observed [AMK00, LFS98]. More recent methods account for uncertainty in the data by invoking information theoretic techniques. In [DTA08] it was shown that using the minimum description length principle allows for developing reverse engineering methods that outperform purely deterministic approaches. In addition, algebraic analytical frameworks allow for fitting a finite dynamical system model to time course microarray expressions, assuming purely deterministic observations [LS04, DJLS07]. However, noise in the expression measurements and the inherently stochastic nature of biological processes make reverse engineering within any of the above described deterministic frameworks only of limited practical value.

In this chapter, we present a new approach for reverse engineering gene expression dynamics that is casted within the algebraic framework developed in [LS04]. First, we develop a theoretical framework for the study of the reverse engineering problem and show that it is closely related to problems arising in coding theory. We then focus on the statistical predictive inference problem of network dynamics given the topology of the network. We address randomness, measurement errors, and small sample size issues jointly by applying powerful list-decoding algorithms that can be shown to optimally deal with missing observations and noise from a coding theoretic perspective [PW04]. This method overcomes the drawbacks of models that assume noiseless observations and that rely on large sample set sizes.

Our method is first validated and compared to existing methods using synthetic simulations. It is subsequently applied to *E. coli* DNA microarray data, and successfully used for decoding the responses of genes in the SOS repair network. We further introduce a framework for performance evaluation based on the notion of the influence of a gene, and show how the inferred network dynamics can be used to compute this variable. Our analysis of the complete transcription factor network of *E. coli*, available from the database RegulonDB [GCJJPG+08], reveals that the predictions made by the list-decoding algorithm can significantly improve the discrimination of basic features of network dynamics when compared to standard algebraic methods. To the best of our knowledge, this is the first analysis performed on real expression data and networks under the discrete algebraic model.

7.2 System and methods

This section introduces a formal description of gene networks and different models for their dynamic behavior. The choice of a model class involves a classical tradeoff: quantitative models are *fine scale* approaches that capture detailed *low-level* phenomena and consider factors associated with regulation of gene expression at various levels such as transcriptional regulation, protein concentrations, and reaction kinetics. Ordinary differential equation models are probably most widely used within this framework [dG05]. Reverse engineering quantitative models will therefore require large amounts of highly accurate data in order to avoid overfitting.

In contrast, qualitative models define gene networks on a higher level of abstraction than quantitative models. For example, Boolean networks are a very *coarse-scale* approximation, allowing genes to be either "ON" or "OFF". Qualitative models with lower complexity emphasize generic principles rather than quantitative biochemical details. In general, quantitative models can be applied to relatively small and isolated genetic subsystems, whereas qualitative models are more suited to analyze global measurements (such as those produced by microarrays). Stochastic extensions, available for both classes, account for the fact that chemical reactions are basically stochastic processes, and that randomness is therefore inherent in biological systems. Many approaches for modeling gene regulatory networks exist, and we can only give a brief summary of the most important models here. For a more thorough overview on the subject, the reader is referred to [de 02,SB07,DLS00] and references therein.

7.2.1 Basic definitions

Virtually all models start by defining a gene network as a graph. Here, we shall model GRNs as a *directed graph* as follows:

Definition 7.2.1 (Gene regulatory network (GRN)) A GRN is a directed graph $G = \{V, E\}$ with vertices $V = \{v_1, ..., v_{N_G}\}$ representing genes and edges $E \subseteq \{1, .., N_G\} \times \{1, .., N_G\}$ describing relationships among the genes. An edge (j, i) is drawn from gene v_j to gene v_i if gene v_j regulates the expression of gene v_i. In this case, we say that v_j is a *regulator* of v_i, and that v_i is a *regulatee* of v_j. □

Note that this very general definition allows for cycles as well as self-loops in the graph which are in fact prominent features of biological networks. Let $\check{\boldsymbol{v}}_i$ denote the vector of regulators of gene v_i, i.e.,

$$\check{\boldsymbol{v}}_i \doteq (v_{i_1}, .., v_{i_{m(i)}}),\ \forall i_k : (i_k, i) \in E. \tag{7.1}$$

We refer to the number of regulators $m(i)$ of a gene as its *in-degree.* The term *dynamics of the network* refers to the trajectories of the gene expression levels over time. We introduce a temporal dimension t and let $v_i(t)$ denote the expression of gene v_i at time t. The state of a network is the collection of current gene expression values and represented by the vector

$$\boldsymbol{v}(t) \doteq [v_1(t), .., v_{N_G}(t)]. \tag{7.2}$$

For continuous models, the expression at time $t + h$, $t, h \in \mathbb{R}_+$, is a function of the expression values of its regulators at time t

$$f_i : \mathbb{R}_+^{m(i)} \to \mathbb{R}_+,\ v_i(t + h) = f_i(\check{\boldsymbol{v}}_i(t), h). \tag{7.3}$$

The term *steady state* refers to the long-run behaviour of the network state vector $\boldsymbol{v}(t)$.

In this work, we focus on discrete time models and can therefore assume, without loss of generality, that $t \in \mathbb{N}_0$. Dynamics refer to the progression of network states at times

$t = 0, 1, 2, 3, \ldots$. The expression at time $t+1$ is a function of the expression values of its regulators at time t

$$v_i(t+1) = f_i(\check{\boldsymbol{v}}_i(t)). \tag{7.4}$$

Different models of GRNs differ in the choice of the type of functions f_i, and expression time and amplitude quantization schemes. The following sections introduce some of these models.

7.2.2 Ordinary differential equations:

Ordinary differential equation (ODE) models represent GRNs by continuous time systems of rate equations describing the dynamics of gene expression according to

$$\frac{\mathrm{d}v_i(t)}{\mathrm{d}t} = f_i(\check{v}_1(t), \ldots, \check{v}_{m(i)}(t)), \tag{7.5}$$

where $f_i : \mathbb{R}_+^{m(i)} \to \mathbb{R}_+$ is some non-linear function that depends on several parameters. A commonly used model obtains f_i by distinguishing between activating and inhibiting genes and multiplying together their sigmoidal contributions:

$$\frac{\mathrm{d}v_i(t)}{\mathrm{d}t} = f_i(\check{\boldsymbol{v}}_i(t)) - b_i v_i(t), \tag{7.6a}$$

$$f_i = V_i \prod_{j \in J_i} \left(1 + \frac{v_j^{h_{ij}}}{v_j^{h_{ij}} + \alpha_{ij}^{h_{ij}}} \right) \times \prod_{k \in K_i} \left(\frac{v_k^{h_{ik}}}{v_k^{h_{ik}} + \beta_{ik}^{h_{ik}}} \right), \tag{7.6b}$$

where J_i (K_i) represents the indices associated with the activators (inhibitors) of gene v_i. The model expressed by Eq. (7.6b) depends on many parameters (described in Table 7.1). Such a model can be quantitatively very accurate since it is based on detailed modeling of biochemical reactions involved. It is, however, apparent that a reverse engineering approach would require a lot of data, and that such models need to be built *bottom-up*.

Parameter	Description
V_i	Basal rate of expression
h_{ij}	Hill coefficient
α_j	Activator half-saturation constant
β_i	Inhibitor half-saturation constant
b_i	Degradation rate

Table 7.1: Parameters of the ODE model expressed in Eq. (7.6b).

Identification of nonlinear behavior is difficult, and a simpler approach considers only small perturbations of the network from the *steady state*, for which the gene expression behaviors can be approximated by a system of linear equations of the form

$$\frac{\mathrm{d}v_i(t)}{\mathrm{d}t} = \sum_{j=1}^{m(i)} c_{ji}\breve{v}_j(t). \tag{7.7}$$

Connectivity factors $c_{ji} \in \mathbb{R}$ model the influence of gene j on gene i.

A further simplification of the ODE model leads to a setting in which one can simply assume that the expression level of gene v_i is a linear combination of the expressions of its regulators:

$$v_i(t) = \sum_{j=1}^{m(i)} c_{ji}\breve{v}_j(t). \tag{7.8}$$

Neural network models [TB03] introduce an additional feature into the linear model by applying a nonlinear transform $g : \mathbb{R} \to \mathbb{R}_+$ to the linear combination of (7.8), i.e.,

$$v_i(t) = g\left(\sum_{j=1}^{m(i)} a_{ji}\breve{v}_j(t)\right), \tag{7.9}$$

where g is usually chosen to be the sigmoidal squashing function

$$g(x) = \frac{1}{1+e^{-x}}. \tag{7.10}$$

The function g can be seen as capturing the expression saturation effects that arise for highly expressed genes. Recently, this model was shown to achieve high predictive power for a particular bacterial gene regulatory system under a sparseness constraint for the network topology [BRS$^+$06]. Clearly, steady state models can only be used to infer topology but not dynamics of a gene network.

7.2.3 Stochastic master equations

Recognizing that biochemical reactions are stochastic processes, stochastic master equations (SME) provide extensions of deterministic ODE models. In contrast to the deterministic case, SMEs model the temporal evolution of probability distributions of the number of molecules of the different molecular species involved in the biochemical reactions under consideration. Discrete amounts of molecules are taken as state variables, and the joint pdf $p(\boldsymbol{v}, t)$ expresses the probability that at time t the cell contains $v_1 \in \mathbb{N}_0$ products of gene 1, $v_2 \in \mathbb{N}_0$ products of gene 2 and so on. The evolution of the system's state pdf can now be specified as follows [de Jong, 2000; Cai et al., 2007]:

$$p(\boldsymbol{v}, t+\Delta t) = p(\boldsymbol{v}, t)\left(1 - \sum_{r=1}^{R} \alpha_r \Delta t\right) + \sum_{r=1}^{R} \beta_r \Delta t, \tag{7.11}$$

where parameters are described in Table 7.2, and the stochastic master equation is obtained by taking the limit $\Delta t \to 0$:

$$\frac{\partial}{\partial t} p(\boldsymbol{v}, t) = \sum_{r=1}^{R} (\beta_r - \alpha_r p(\boldsymbol{v}, t)). \tag{7.12}$$

Parameter	Description
$p(\boldsymbol{v}, t)$	Joint pdf of number of molecules at time t
R	Number of reactions that can occur in the system
$\alpha_r \Delta t$	Probability that reaction r occurs in $[t, t+\Delta t]$ given system state $\boldsymbol{v}$
$\beta_r \Delta t$	Probability that reaction r will bring system into state $\boldsymbol{v}$

Table 7.2: Description of parameters in the stochastic model (Eq. (7.11)).

Behavior of small metabolic systems can be very accurately predicted using models of this kind. However, the master equation is difficult to solve analytically, and also numerical simulations are complicated [de 02]. Simulation approaches are often used to approximate molecular reality of gene regulation [CW07]. However, even the simulation of such models becomes intractable for large, genome-wide systems.

The models presented above assume a continuous time scale and (quasi)-continuous expression values. The complexity of arbitrary non-linear models is usually to high to be of practical value, while linear models are too simple to capture the most important features of GRNs. *Discrete models*, which rely on quantizing both, time and expression values, represent an alternative to continuous schemes. Discrete models are capable of reducing the complexity of continuous models while at the same time incorporating non-linear features into the GRN structure. In what follows, we henceforth focus on probabilistic and deterministic discrete models of gene interaction.

7.2.4 Boolean network models

Boolean network (BN) models [Kau69] are discrete deterministic models which allow genes to be only in two different states - “ON” or “OFF”. In other words, expression levels can be seen as elements from a finite field with two elements, denoted by $\mathbb{F}_2$. A BN model is defined via a map of the binary state vector from time t to $t+1$, i.e.,

$$\mathcal{F} : \mathbb{F}_2^{N_G} \to \mathbb{F}_2^{N_G},\ \boldsymbol{v}(t) \mapsto \mathcal{F}(\boldsymbol{v}(t)), \tag{7.13}$$

that can be decomposed into N_G Boolean functions

$$f_i : \mathbb{F}_2^{m(i)} \to \mathbb{F}_2, i = 1, .., N_G, \tag{7.14}$$

each associated with one gene v_i in the network. The binary expression of gene v_i at time $t+1$ is a Boolean rule on the state of the regulators at time t

$$v_i(t+1) = f_i(\breve{\boldsymbol{v}}_i(t)). \tag{7.15}$$

To construct a BN, continuous expression values $v_i(t) \in \mathbb{R}_+$, $t \in \mathbb{R}_+$ are first quantized at discrete time steps,

$$v_i(t) \in \mathbb{F}_2, t \in \mathbb{N}_0. \tag{7.16}$$

The dynamics of the network are simulated by starting from an initial state vector $\boldsymbol{v}(0)$, and by iteratively and synchronously updating the Boolean functions f_i. This gives rise to a sequence of states:

$$\boldsymbol{v}(0) \overset{\mathcal{F}}{\rightarrow} \boldsymbol{v}(1) \overset{\mathcal{F}}{\rightarrow} \boldsymbol{v}(2) \overset{\mathcal{F}}{\rightarrow} \ldots \tag{7.17}$$

It is clear that the dynamics of the network are completely deterministic. From the characterization of the dynamical behavior of the state sequence such as analyzing attractor states and cycles, biological conclusions can be drawn [SD05, Sch06].

Probabilistic Boolean networks (PBN) are stochastic extensions of Boolean networks. In a PBN, a whole list of functions

$$\mathcal{L}_i = \{f_i^{(1)}, f_i^{(2)}, ..., f_i^{(|\mathcal{L}_i|)}\} \tag{7.18}$$

is assigned to each gene v_i. At a given instant of time, a function is randomly selected from $\mathcal{L}_i$ to predict the output $v_i(t+1)$. The map $\mathcal{F}$ is now a random variable, and a realization of the PBN at a given time is determined by the vector function

$$\mathcal{F} : \mathbb{F}_2^{N_G} \rightarrow \mathbb{F}_2^{N_G},\ \check{\boldsymbol{v}}_i(t) \mapsto f_i^{(k)}(\check{\boldsymbol{v}}(t)),\ i = 1, ..., N_G, \tag{7.19}$$

where k is randomly chosen from $1, ..., |\mathcal{L}_i|$ according to some pdf.

When defined as above, a PBN forms a finite-state Markov chain (cf. Chapter 2 Section 2.4), and it was shown in [SDKZ02] how to derive explicit formulas for the state transition probabilities, depending on the gene functions $f_i^{(k)}$. Several extensions were presented, accounting for memory in the function selection process and noise caused by external factors [SD05]. The reverse engineering of PBNs from a single time course expression data sequence was discussed in [MXD07]. In [SDZ02], a strategy was presented how to *control* the behavior of such networks by carefully applied perturbations. PBNs are closely related to probabilistic graphical models, that are also widely used for modeling GRNs, which we shall describe next.

7.2.5 Probabilistic models

The inherent stochastic nature of biological processes makes deterministic models only of limited practical interest. Friedman et al. were among the first to apply Bayesian networks, well studied probabilistic graphical models, to the analysis of gene expression data [FLNP00]. In this framework, genes are assigned a finite number of states $v_i \in \mathcal{S} = \{1, 2, ..., q\}$, where $\mathcal{S}$ denotes the sample space of a random variable V_i with appropriate distribution function. Each node in the network is represented by a conditional probability distribution $p_{\mathsf{V}|\check{\mathsf{V}}}(v_i|\check{\boldsymbol{v}}_i)$, and the factorization of the joint probability distribution $p_{\mathsf{V}}(\boldsymbol{v})$ is inferred from the expression data, usually by Monte-Carlo type approaches. A severe drawback of Bayesian networks is the fact that only undirected acyclic graphs can be learned. This is clearly not the case observed in reality, since cycles are an important feature of GRNs [SPB07, Alo07].

Factor graphs [KFL01] are generalizations of Bayesian networks and Markov random fields. The main difference between factor graphs and Bayesian networks is that factor graphs are not required to be acyclic. Milenkovic et.al. [MV04] proposed a factor graph (FG) model for GRNs that allows for accommodating cycles in biological networks. In Figure 7.1, a gene network and its corresponding FG model are shown. As described

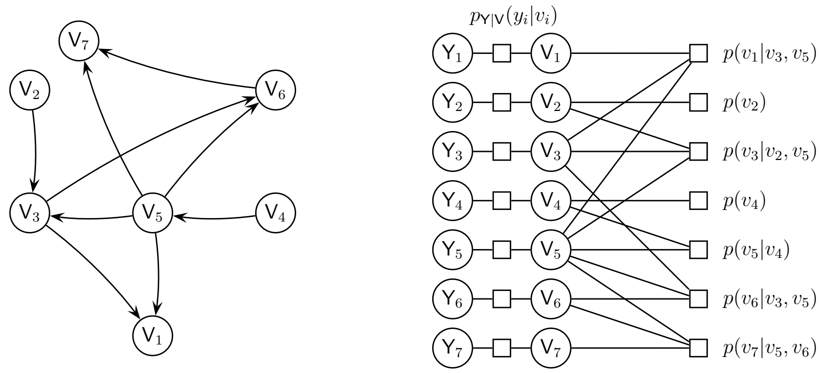


Figure 7.1: Gene network and corresponding factor graph model.

above, gene expressions are assumed to be discretized and dependencies among genes are modeled as conditional probability distributions $p_{\mathsf{V}|\breve{\mathsf{V}}}(v_i|\breve{\boldsymbol{v}}_i)$. The random variables Y_i represent the measured expression levels of the genes, and one can think of $p_{\mathsf{Y}|\mathsf{V}}(y_i|v_i)$ as describing the unknown and noisy measurement channel. The joint probability of the variables (V, Y) factors as

$$p_{\mathsf{V},\mathsf{Y}} \sim \prod_{i=1}^{N_g} p_{\mathsf{Y}|\mathsf{V}}(y_i|v_i) p_{\mathsf{V}|\breve{\mathsf{V}}}(v_i|\breve{\boldsymbol{v}}_i).$$

Recently, Gat-Viks et al. [GVTRS06] applied this model to two yeast networks consisting of 50 and 140 genes, respectively. Their approach is based on assuming at least partial or complete knowledge about the topology of the GRN. Loopy belief propagation [KFL01] is employed to learn the conditional probabilities $p_{\mathsf{V}|\breve{\mathsf{V}}}(v_i|\breve{\boldsymbol{v}}_i)$. Different methods for inference of these distributions and the "channel" $p_{\mathsf{YV}}(y_i|v_i)$ were proposed and analyzed in [GVTRS06].

We next describe the model considered in this chapter. Wee shall see that it is a generalization of PBNs to multivalued discretization of gene expression, cast in an algebraic framework.

7.2.6 Polynomial dynamical systems model

The algebraic framework for gene expression profiles considered in the remainder of this chapter is a generalization of Boolean networks and was first introduced in [LS04]. It allows for a finer representation of gene expression states while maintaining the analytical tractability of BN models by imposing a special form on the update functions f_i. More specifically, a gene is assumed to be in a finite number of states q that represent different expression levels. The number of states is restricted to be a power of a prime p,

$$q = p^s, \ \ s \in \mathbb{N}, \tag{7.20}$$

so that the states of the genes correspond to elements of a finite field, denoted by $\mathbb{F}_q$, i.e.,

$$v_i(t) \in \mathbb{F}_q.$$

This is not an overly restrictive assumption since many small integers such as 2, 3, 4, 5, 7, 8, 9, 11 can be expressed in the form of Eq. (7.20), and since very fine quantization schemes are not of practical interest. Analogous to Boolean networks, the dynamics of the system are specified through the equivalent mapping of the state vector $\boldsymbol{v}(t) \in \mathbb{F}_q^n, t \in \mathbb{N}_0$:

$$\mathcal{F}_q : \ \mathbb{F}_q^n \to \mathbb{F}_q^n, \ \boldsymbol{v}(t) \mapsto \boldsymbol{v}(t+1) = \mathcal{F}(\boldsymbol{v}(t)). \tag{7.21}$$

A crucial point for the methods used in this paper is the well known fact that any possible function $f : \mathbb{F}_q^m \to \mathbb{F}_q$ defined over an arbitrary finite field $\mathbb{F}_q$ is represented by a multivariate polynomial, i.e., every function f can be expressed as

$$f = \sum_{i_1,\ldots,i_m} a_{i_1 i_2 \ldots i_m} x_1^{i_1} \ldots x_m^{i_m}, \quad a_{i_1 i_2 \ldots i_m} \in \mathbb{F}_q. \tag{7.22}$$

We use $\mathbb{F}_q[x_1, x_2, ..., x_m]$ to represent the set of all possible multivariate polynomials in m variables given by Eq. (7.22). Hence, in our model, a function f_i associated with a node v_i in the GRN is a multivariate polynomial in the variables $\breve{\boldsymbol{v}}_i$, the regulators of v_i. In what follows, we refer to this model as a *polynomial dynamical system* (PDS). Note that a PDS can be generalized to capture probabilistic behavior in the same way as PBNs represent stochastic extensions of Boolean networks. We call a PDS that describes each gene by a list of randomly selected polynomials a *stochastic polynomial dynamical system* (SPDS).

7.3 Reverse engineering frameworks

We are interested in modeling the dynamics of gene expression via an SPDS model. We consider the following reverse engineering problem: one is given the topology of the network and a limited set of time course expression points for the genes in the network. In this setting, a reverse engineering algorithm has to infer an SPDS that *explains* the observed time course data.

7.3.1 Deterministic case and the Laubenbacher-Stigler algorithm

Let us first assume that the expression data is error-free and that a PDS model perfectly characterizes the real dynamics of the GRN. Then, we have to solve the following problem, which we henceforth refer to as the *noiseless reconstruction problem*:

Definition 7.3.1 (Noiseless reconstruction problem) Given a time series of $T + 1$ transitions, i.e., for each gene v_i in a network G we observe inputs $\breve{\boldsymbol{v}}_i(t)$ and corresponding outputs $v_i(t+1)$:

$$\begin{aligned}\mathcal{V}_i &= \{(\breve{\boldsymbol{v}}_i(0), v_i(1)), ..., (\breve{\boldsymbol{v}}_i(T), v_i(T+1))\}\\ &\doteq \{(\breve{\boldsymbol{v}}_i(t), v_i(t+1))\}_{t=0}^{T},\ i = 1, .., N_G,\end{aligned}$$

find the functions f_i that reproduce the time series *exactly*. □

This is the framework originally considered in [LS04] and follow-up work [DJLS07, AGM04, Jus06, DE09]. Due to small sample sizes, we are facing an underdetermined system, and it is clear that model constraints need to be imposed. In [LS04] the selection process in the noiseless reconstruction problem consists in identifying a *minimal* solution. Such a solution is obtained by first interpolating a polynomial $f_{i,(0)}$ through the $T+1$ points. Here, $f_{i,(0)}$ satisfies

$$f_{i,(0)}(\breve{\boldsymbol{v}}_i(t)) = v_i(t+1),\ \forall t = 0, ..., T. \tag{7.23}$$

Next, the set

$$\mathcal{I}(\mathcal{V}_i) = \{g_i \in \mathbb{F}_q[x_1, .., x_m] : g_i(\breve{\boldsymbol{v}}_i(t)) = 0,\ t = 0, ..., T\}, \tag{7.24}$$

of polynomials vanishing on the input points in $\mathcal{V}_i$ is constructed. Such a set, called a *vanishing ideal*, has the obvious property that any interpolator $f_{i,(0)}$ lies in the set

$$f_i + \mathcal{I}(\mathcal{V}_i) = \{f_i + g_i : g_i \in \mathcal{I}(\mathcal{V}_i)\}, \tag{7.25}$$

where f_i denotes an interpolator that cannot be further decomposed, i.e, a *reduced interpolator*. A reduced interpolator is defined by the following property:

$$\nexists(f_i^{'} \in \mathbb{F}_q[x_1, .., x_m],\ g_i^{'} \in \mathcal{I}(\mathcal{V}_i)) : f_i = f_i^{'} + g_i^{'}. \tag{7.26}$$

In other words, g_i represents the part of $f_{i,(0)}$ that lies in $\mathcal{I}(\mathcal{V}_i)$ and f_i represents the reduction of $f_{i,(0)}$ with respect to $\mathcal{I}(\mathcal{V}_i)$. The reduced solution f_i subsequently serves as an update model for the node v_i. It is a well known result from computer algebra that the ideal of vanishing polynomials $\mathcal{I}(\mathcal{V}_i)$ has a finite polynomial basis, a so-called Gröbner basis [CLO92]. This basis can be computed, for example, by the Buchberger-Möller algorithm, and then be used for identifying minimal solutions [LS04, DJLS07]. In the following, we shall refer to this procedure as the Laubenbacher-Stigler (LS) algorithm.

The LS algorithm has the following drawback: as noted by the authors in [LS04], the interpolation method is extremely sensitive to measurement errors and it does not consider missing expression observations. These problems arise due to the fact that in the first step of the modeling process, an interpolation polynomial is found that passes through *all* data points, which may cause over-fitting in the presence of noise.

An alternative approach to the method above is to use an SPDS model and to approximate the time series by using lists of update functions, thereby accounting for missing samples and read-out errors caused by noise. To implement this approach, we describe next how this reverse engineering problem is connected to coding theory, and introduce a reconstruction method based on list-decoding. List-decoders can identify the exact update functions in the presence of both, missing values and noise, provided that the total number of missing values and errors is properly bounded. Robustness with respect to measurement errors is achieved by bounding the degree of the update polynomial, thus allowing the interpolated functions in the list to agree only with a fraction of the observed transitions.

7.3.2 Accounting for stochasticity and noise

We now turn our attention to a more rigorous formulation of the reverse engineering problem. In this context, we address two important issues: availability of only a limited amount of time course data, even for the best studied organisms; and errors in the measured data. Noise in the data arises both due to measurement errors and the fact that biological systems are inherently stochastic and influenced by factors that cannot be measured. In this case, the purely deterministic view within the framework of the noiseless reconstruction problem represents an inadequate approach. We therefore propose to recast the reverse engineering problem into a probabilistic framework and consider the following *noisy reconstruction problem:*

Definition 7.3.2 (Noisy reconstruction problem) Given a time series of $T+1$ noisy transitions for each gene v_i, i.e., a sequence of pairs of inputs and corresponding outputs

$$\mathcal{V}_i^{\varepsilon} = \{(\breve{\boldsymbol{v}}_i(t), v_i(t+1))\}_{t=0}^{T}, \tag{7.27}$$

find a *set of functions* $\mathcal{L}_i = \{f_i^{(1)}, .., f_i^{(L)}\}$ that jointly provide the best *approximation* for the observed time series. □

Here, an additive noise model is assumed

$$v_i(t+1) = f_i^{(l)}(\breve{\boldsymbol{v}}_i(t)) + \varepsilon_i(t+1), \tag{7.28}$$

where $\varepsilon_i(t) \in \mathbb{F}_q$, and $P(\varepsilon_i(t) \neq 0) = p_\varepsilon$ denotes the noise samples. Note that in this setting, small sample sets and measurement errors are accounted for by forming a *list* of possible node update functions, similarly as for the case of probabilistic Boolean network models [SDKZ02], but with some major differences summarized in the exposition to follow.

Measurement noise affects *both* the inputs and the outputs of the time series. However, one can assume that only an *output noise* component exists, since the effect of input noise can be transformed into an equivalent output noise component. To clarify this issue, assume that the noise free input is $\breve{\boldsymbol{v}}_i'(t)$, with corresponding noise free output $v_i'(t+1)$, and that a noise sample $\breve{\boldsymbol{\varepsilon}}_i'(t)$ is added to the input, and a noise sample $\varepsilon_i'(t+1)$ is added to the output. Let

$$\breve{\boldsymbol{v}}_i(t) \doteq \breve{\boldsymbol{v}}_i'(t) + \breve{\boldsymbol{\varepsilon}}_i'(t) \tag{7.29}$$

and let

$$\varepsilon_i(t+1) \doteq f_i(\breve{\boldsymbol{v}}_i(t)) - v_i'(t+1) + \varepsilon_i'(t+1) \tag{7.30}$$

denote the *transformed noise* affecting only the output. Then,

$$f_i(\breve{\boldsymbol{v}}_i'(t) + \breve{\boldsymbol{\varepsilon}}_i'(t)) + \varepsilon_i'(t+1) = f_i(\breve{\boldsymbol{v}}_i'(t)) + \varepsilon_i(t+1) = v_i'(t+1) + \varepsilon_i(t+1), \tag{7.31}$$

as claimed.

7.4 Tools from coding theory

As shown above, the noiseless and noisy reverse engineering problems are essentially interpolation and approximation problems for polynomials over finite fields. This is a well studied subject in coding theory, due to its application in various decoding algorithms for polynomial codes. In this section, we briefly present the concept of codes defined over polynomials and their decoding; then we establish the connection to the noisy reconstruction problem.

7.4.1 Polynomial codes

Reed-Solomon (RS) codes are one of the most widely used class of codes today. RS codes are algebraic codes defined over univariate finite field polynomials of bounded degree. More precisely, let $\mathbb{F}_q[x]$ denote the ring of univariate finite polynomials, a generalized Reed-Solomon code $\mathcal{RS}_q(n,u)$ over a finite field $\mathbb{F}_q$ is defined as

$$\mathcal{RS}_q(n,u) = \{(f(\alpha_0), ..., f(\alpha_{n-1})) : f \in \mathbb{F}_q[x] \wedge \deg(f) < u\}, \tag{7.32}$$

where the points $(\alpha_0, ..., \alpha_{n-1})$, $\alpha_i \in \mathbb{F}_q$ are called the *evaluation set* of the code and $\deg(f)$ denotes the degree of the polynomial. The polynomials f that form the RS code are called *message polynomials*. It is clear that the $\mathcal{RS}_q(n,u)$ code is linear, i.e., $\boldsymbol{c} \in \mathcal{RS}_q(n,u)$ and $\boldsymbol{c}' \in \mathcal{RS}_q(n,u)$ implies $(\boldsymbol{c}+\boldsymbol{c}') \in \mathcal{RS}_q(n,u)$, since the set of polynomials of bounded degree is closed under addition. By definition, the all zero word (polynomial with all coefficients zero has degree $u = 0$) is an element of the code. The code has length n and rate u/n. Reed-Solomon codes belong to the class of *maximum distance separable* codes which says that they meet the largest minimum Hamming distance a linear code can have: an important result in coding theory, the Singleton bound, states that a linear block code of length n and rate u/n cannot have larger minimum distance than $n-u+1$ [MS77]. It is easy to show that an RS code, in fact, has the highest possible minimum distance: recall that a message polynomial of degree $u-1$ has at most $u-1$ zeros. Therefore, the minimum distance of an $\mathcal{RS}_q(n,u)$ code is given by

$$\begin{aligned} d_{\min} &= \min_{\boldsymbol{c},\boldsymbol{c}' \in \mathcal{RS}} \{d_{\mathrm{H}}(\boldsymbol{c},\boldsymbol{c}')\} && (7.33\mathrm{a}) \\ &= \min_{\boldsymbol{c},\boldsymbol{c}' \in \mathcal{RS}} \{w_{\mathrm{H}}(\boldsymbol{c}-\boldsymbol{c}')\} && (7.33\mathrm{b}) \\ &= \min_{\boldsymbol{c}'' \in \mathcal{RS}} \{w_{\mathrm{H}}(\boldsymbol{c}'')\} && (7.33\mathrm{c}) \\ &\geq n-u+1, && (7.33\mathrm{d}) \end{aligned}$$

where Eq. (7.33b) and (7.33c) follow from the linearity of the code, and the lower bound Eq. (7.33d) is due to the fact that a polynomial in the $\mathcal{RS}_q(u,n)$ has at most $u-1$ zeros. Since the Singleton bound provides an upper bound on the minimum distance, Eq. (7.33d) holds with equality and

$$d_{\min} = n-u+1. \tag{7.34}$$

An $\mathcal{RS}_q(n, u)$ code is therefore a $(n, u, n - u + 1)$ linear block code, and a message polynomial f can be uniquely recovered via bounded distance decoding from noisy samples

$$(f(\alpha_0) + \varepsilon_0, .., f(\alpha_{q^m-1}) + \varepsilon_{q^m-1}),$$

as long as the number of errors e (i.e., number of indices i for which $\varepsilon_i \neq 0$) satisfies (cf. Chapter 2, Section 2.3)

$$e \leq \lfloor \frac{d_{\min} - 1}{2} \rfloor. \tag{7.35}$$

The reverse engineering problem described in this chapter deals with multivariate polynomials in $\mathbb{F}_q$. This can be seen as the problem of decoding q-ary Reed-Muller codes. The coding-theoretic objects of interest are generalized q-ary Reed-Muller codes, a multivariate generalization of RS codes, defined as follows: let a multivariate polynomial f be of the form (7.22), then the total degree of f is defined as

$$\text{totdeg} = \max\{i_1 + i_2 + ... + i_m : a_{i_1 i_2 \dots i_m} \neq 0\}. \tag{7.36}$$

A q-ary Reed-Muller (RM) code $\mathcal{RM}_q(u, m)$ is the set of all m-variate polynomials $f \in \mathbb{F}_q[x_1, .., x_m]$ with $\text{totdeg}(f) \leq u$, evaluated at the q^m distinct elements of the finite field $\alpha_j \in \mathbb{F}_{q^m}$, i.e.,

$$\mathcal{RM}_q(u, m) = \{(f(\alpha_0), ..., f(\alpha_{q^m-1})) : f \in \mathbb{F}_q[x_1, .., x_m] \wedge \text{totdeg}(f) < u\}. \tag{7.37}$$

As already pointed out, in RM codes the encoded messages - codewords - are multivariate polynomials of bounded degree. Given the noisy sample vector

$$\boldsymbol{c}_\varepsilon = (f(\alpha_0) + \varepsilon_0, .., f(\alpha_{q^m-1}) + \varepsilon_{q^m-1}), \tag{7.38}$$

an optimal *maximum-likelihood* RM decoder has to find the polynomial f of bounded total degree u that most likely led to the observation $\boldsymbol{c}_\varepsilon$. However, this problem is computationally intractable for long codeword lengths, and suboptimal decoders must be used instead.

Recently, a class of algorithms was described in coding literature that can closely approach the optimal performance with polynomial computational times. Among these decoding algorithms, list-decoders are of special interest since they can operate in extremely noisy regimes. From the discussion above, it is apparent that the decoding problem is equivalent to the noisy reconstruction problem.

7.4.2 List-decoding

List-decoding algorithms can handle very large noise levels and missing sample points while allowing the decoder to generate a list of possible solutions, rather than a unique output. Bounded minimum distance (BMD) decoding was discussed in Chapter 2, Section 2.3. In contrast to BMD decoding, list-decoding does not stop at the $d_{\min}/2$ bound

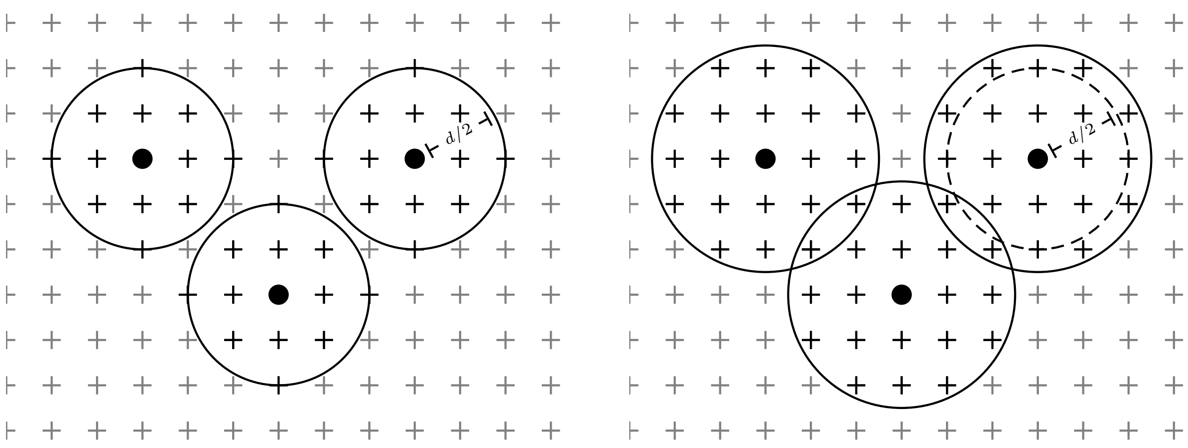


Figure 7.2: List-decoding allows codewords that lie beyond half the minimum Hamming distance. This can result in non-unique decisions. Black circles mark codewords, crosses mark elements from $\mathbb{F}_2^N$. Half the minimum distance is represented by $d/2$. Left: decision regions for BMD decoding, Right: decision regions for List-decoding.

but also looks for codewords that lie beyond that bound (cf. Figure 7.2). As a consequence, the decoder may find several codewords explaining the observed data instead of a single one. However, depending on how much the bound is increased, the probability of such a situation can be extremely small (cf. [McE03, Example 1]). With small sample sizes encountered in gene network reconstruction problems, one inevitably has to employ list-decoding and allow for non-unique solutions.

In 1997, Madhu Sudan showed that a polynomial time algorithm exists that is capable of decoding certain Reed-Solomon codes far beyond the classical $d_{\min}/2$ bound [Sud97]. In 1999 Guruswami and Sudan significantly improved this result, presenting a method capable of decoding virtually every RS code beyond the bounded distance decoding limit [GS99]. More precisely, given an $\mathcal{RS}_q(n,u)$ code with evaluation set $(\alpha_0, ..., \alpha_{n-1})$, the Guruswami-Sudan (GS) algorithm takes the noisy sample vector $\boldsymbol{c}_\varepsilon$ and an adjustable parameter ϑ and produces a list $\{g_1, ..., g_L\}$ that contains all polynomials $g \in \mathbb{F}_q[x]$ of total degree less or equal u, such that the Hamming distance of g and $\boldsymbol{c}_\varepsilon$ is smaller than or equal to a predefined constant e_ϑ, called the *decoding radius*. Alternatively, the list includes polynomials g such that

$$|\{j : g(\alpha_j) \neq f(\alpha_j) + \varepsilon_j\}| \leq e_\vartheta, \tag{7.39}$$

where e_ϑ can be close to the bound guaranteed by Shannon. Note that the parameters L and e_ϑ are intimately connected to the error rate, and that each list-decoding algorithm provides different operational regions for these parameters. Detailed overviews of list-decoding algorithms can be found in [GS99, GR06, GKS07, San07].

For the reverse engineering of SPDS models, we require list-decoding algorithms for Reed-Muller codes. We focus our attention on a list-decoding method recently described by Pellikaan and Wu [PW04] that exploits the fact that $\mathcal{RM}_q(u,m)$ codes defined over $\mathbb{F}_q$ are subfield-subcodes of generalized Reed-Solomon (RS) codes, defined over $\mathbb{F}_{q^m}$. The Pelilikaan-Wu (PW) algorithm is just one of many possible list-decoders for RM codes: some recent results that offer excellent performance for small field sizes include [GKZ08].

7.5 The Pellikaan-Wu list-decoder

We briefly describe the used list-decoder by Pellikaan and Wu lying at the heart of our gene reverse engineering algorithm [PW04]. A detailed listing of the algorithm is given in Algorithm 7.1.

For the purpose of RM list-decoding, the field $\mathbb{F}_{q^m}$ with primitive element ξ is viewed as an m-dimensional vector space over $\mathbb{F}_q$, with basis elements $\{1, \xi, .., \xi^{m-1}\}$, so that

$$\xi^j = \sum_{i=0}^{m-1} a_{ij}\xi^i, \quad 0 \leq j \leq q^m - 2,\ a_{ij} \in \mathbb{F}_q. \tag{7.40}$$

The vector space (field $\mathbb{F}_{q^m}$) has $n = q^m$ points

$$\begin{aligned} \alpha_0 &= (0, .., 0) \\ \alpha_j &= (a_{0,j-1}, .., a_{m-1,j-1}), \quad j = 1, .., n-1, \end{aligned} \tag{7.41a}$$

denoted by $\alpha_0, \alpha_1, .., \alpha_{n-1}$. It is well known that

$$\mathcal{RM}_q(u^\perp, m), \quad u^\perp = m(q-1) - u - 1 \tag{7.42}$$

is the dual code of the $\mathcal{RM}_q(u, m)$ code. Let σ, ρ be the factor and remainder after division of $u^\perp + 1$ by $q - 1$, i.e.,

$$u^\perp + 1 = \sigma(q-1) + \rho, \quad \rho < q - 1. \tag{7.43}$$

The minimum distance of the code $\mathcal{RM}_q(u, m)$, $d_{\min}$, between any two different codewords, equals [PW04, KLP68]

$$d_{\min} = (\rho + 1)q^\sigma. \tag{7.44}$$

As shown in the previous section, RS codes are specializations of RM codes to the case of *univariate* polynomials of bounded degree. Therefore, it can also be shown via enumeration of points in (7.41a) that the code $\mathcal{RM}_q(u, m)$ is a subfield-subcode of the code $\mathcal{RS}_{q^m}(n, n - d_{\min} + 1)$ with evaluation set $(\alpha_0, .., \alpha_n) = (0, 1, \xi, \xi^2, .., \xi^{q^m-2})$:

$$\mathcal{RM}_q(u, m) \subseteq (\mathcal{RS}_{q^m}(n, n - d_{\min} + 1) \cap \mathbb{F}_q^n), \tag{7.45}$$

i.e., a subset of the vectors from $\mathcal{RS}_{q^m}(n, n - d_{\min} + 1)$ with elements from $\mathbb{F}_q$.

List-decoding of RM codes proceeds in three steps: first, the received word is interpreted as a noisy version of an RS codeword; second, an RS list-decoder is used to decode the received word; third, a check is performed to see if the decoded word belongs to an RM subfield-subcode of the RS code. Algorithm 7.1 summarizes the list-decoding procedure for RM-codes introduced by Pellikaan and Wu.

Note that the decoding radii of the PW and Guruswami-Sudan-RS decoding algorithms are the same, and equal to

$$e_\vartheta = n(1 - \sqrt{((n - d_{\min})/n)}). \tag{7.46}$$

The list size of the latter algorithm L_ϑ is conservatively bounded from above by [McE03]:

$$L_\vartheta \leq (\vartheta + 1/2)\sqrt{\frac{n}{u-1}}, \tag{7.47}$$

where ϑ denotes the so-called multiplicity parameter of the GS algorithm. Since RM codes are subfield subcodes of RS codes, their corresponding L_ϑ parameter may be smaller. Note that the above given decoding radius is only obtained for $\vartheta \to \infty$, while in practice we have to use smaller values of ϑ in order to limit the decoding complexity.

Algorithm 7.1: List-Decoding of RM codes according to the PW algorithm. [PW04]

Require: $n = q^m$, $\boldsymbol{y} \in \mathbb{F}_q^n$.

1: Compute the minimum distance $d_{\min}$ of $\mathcal{RM}_q(u, m)$ and a parameter

$$E = \lceil n - \sqrt{n(n - d_{\min})} - 1 \rceil.$$

2: Construct the extension field $\mathbb{F}_{q^m}$ with primitive element ξ.

3: Generate the code $\mathcal{RS}_{q^m}(n, n - d_{\min} + 1)$ using the evaluation set

$$(\alpha_0, .., \alpha_{q^m-1}) = (0, 1, \xi, \xi^2, .., \xi^{q^m-2}).$$

4: Compute a parity check matrix $\boldsymbol{H}$ over $\mathbb{F}_q$ for $\mathcal{RM}_q(u, m)$.

5: Using the Guruswami-Sudan list-decoding algorithm find the set of codewords $\mathcal{C}_1$ of all codewords $\boldsymbol{c} \in \mathcal{RS}_{q^m}(n, n - d_{\min} + 1)$ that satisfy $d_H(\boldsymbol{c}, \boldsymbol{y}) \leq E$, where d_H denotes the Hamming distance.

6: Find the set of codewords $\mathcal{C}_2$ with elements in $\mathbb{F}_q$

$$\mathcal{C}_2 = \mathcal{C}_1 \cap \mathbb{F}_q^n.$$

7: Find the codewords in $\mathcal{C}_2$ that belong to $\mathcal{RM}_q(u, m)$ using the parity check matrix

$$\mathcal{C} = \{\boldsymbol{c} \in \mathcal{C}_2 : \boldsymbol{H}\boldsymbol{c} = 0\}$$

8: Output $\mathcal{C}$.

Exactly the same procedure can be applied for reverse engineering SPDS models for GRNs. In this case, the inputs $\check{\boldsymbol{v}}_i(t)$ of the $m(i)$ regulators are interpreted as points from the vector field $\mathbb{F}_q^{m(i)}$ corresponding to the evaluation points α_i of the Reed-Muller code with $m = m(i)$ in Eq. (7.37). The node function $f_i \in \mathbb{F}_q[x_1, ..., x_{m(i)}]$ corresponds to the encoded message and the outputs $v_i(t+1)$ represent (noisy) codeword symbols of the RM code. Note that the assumption of bounded degree of RM codewords does not impose a severe restriction on the biological properties of gene networks, since this parameter can be freely chosen and fairly large values for the degree bound can be tested in a sequential manner. A detailed description of our reverse engineering method under the SPDS model is described in the following.

7.6 The reverse engineering algorithm

This section presents our algorithm and provides theoretical and simulation analysis. We shall first briefly discuss microarray technology that is used to generate the data sets virtually used in all reverse engineering approaches. We describe how we obtained and preprocessed the data, then we explain our reverse engineering procedure in detail. We show that coding theoretic results lead to bounds on the required number of data samples, and we present performance comparisons with the Laubenbacher-Stigler (LS) algorithm (cf. Section 7.3) under *in silico* generated expression data.

7.6.1 Microarrays

The DNA microarray chip technology allows for surveying patterns of gene activity by assaying the expressions of thousands of genes simultaneously in a single experiment. A microarray chip, typically a glass slide or a quartz wafer, consist of a large number of spots arranged in a grid, each containing immobilized multiple samples of single stranded DNA of the gene to be monitored. Recall that a DNA sequence is a strand of nucleotides that can be viewed as a sequence over the alphabet $\{A, C, G, T\}$ (cf. Chapter 3). The nucleotides tend to bind in a complementary fashion, i.e., A always binds to T and C binds to G, and vice versa. The single stranded DNA sequences spotted on the microarray will bind - *hybridize* - to their complementary sequence. When a gene is expressed, it produces mRNA and as already pointed out, the amount of this mRNA is a direct measure of activity of a gene. The DNA sequences on the microarray are designed so as to uniquely bind to the mRNA of the respective gene. In order to measure the expression pattern of a cell under specific conditions, the mRNA has to be extracted from a cell population. The purified mRNAs are labeled with a fluorescent dye and flushed over the DNA microarray, so that they hybridize with their complementary spotted DNA sequences. Laser light is used after hybridization to scan the chip, producing an image from which an intensity measurement is derived that should be correlated to the gene expression value in the sample.

Stochastic variations in the experiment, e.g., differences between array spotting techniques and changing environmental conditions lead to different measurements even for the same sample. *Systematic effects* afflict a large number of measurements (for example the measurements on one array), whereas *stochastic effects* or *noise* are random with no well-understood pattern. In order to draw meaningful conclusions from microarray experiments, the raw data has to be pre-processed to account for these stochastic fluctuations in the measurement. The processing of the data that makes measurements from different arrays comparable is known as *normalization* [Spe03]. There are two classes of microarray data sets: static and time course data. Static data measures the expression pattern of a given sample at one particular time, usually after the gene network has arrived into a steady state. Time-course data is obtained by applying a certain treatment to a sample and taking snapshots of gene expressions at consecutive time points until a steady state is reached. It is obvious that for the analysis of dynamic behavior, time course data is required.

7.6.2 Microarray data preprocessing

We quantized gene expressions using q levels, in terms of k-means clustering. Quantization of microarray data is a critical step in the reverse engineering process, and the influence of quantization on network dynamics inference is still not completely understood. Still, quantization is an integral step of all finite state gene network modeling approaches [BS03, de 02, SB07] including Bayesian network techniques. Quantization is most often performed using simple thresholding and clustering techniques or fitting Gaussian mixture models [BE05, GVTRS06, PREF01, SZ02]. To the best of our knowledge, there still does not exist a quantization technique that takes into account error models for DNA microarray measurements [ITSH00]. As a consequence, different quantizers may lead to quite different outputs of the reverse engineering algorithm. By using a sufficiently large number of quantization levels, this problem can be avoided to a certain degree, and this motivates the use of non-binary network dynamics models.

Here, we used microarray data from a recently developed database called $\mathrm{M}^{3\mathrm{D}}$ and filtered time course experiments providing at least one transition of the network. The data set used is further described in Section 7.7.1.

7.6.3 Constructing lists of approximating polynomials

As described in the previous section, we are given a set of noisy transitions for each node in the network. The first step in the reconstruction algorithm is to reduce the observations to a set of *unique transitions*. The observed inputs $\breve{\boldsymbol{v}}_i(t)$ in the set $\mathcal{V}_i^{\varepsilon}$ may not be unique, i.e., there may exist pairs of indices $(t, t') \in \{0, .., T\}^2$, $t \neq t'$, for which $\breve{\boldsymbol{v}}_i(t) = \breve{\boldsymbol{v}}_i(t')$. Note that due to noise, the existence of such pairs does not imply that the corresponding outputs $v_i(t+1)$ and $v_i(t'+1)$ are equal. Hence, we can encounter *multiple transitions* for the same input.

Multiple transitions are coped with according to the procedure described next. First, the observed set $\mathcal{V}_i^{\varepsilon}$ is reduced to a set of unique transitions $\mathcal{U}_i$ as follows: assume we observe the same input to gene v_i exactly r times, $\breve{\boldsymbol{v}}_i(t_1) = \breve{\boldsymbol{v}}_i(t_2) =, .., = \breve{\boldsymbol{v}}_i(t_r)$, with corresponding outputs $v_i(t_1+1), .., v_i(t_r+1)$. We retain only one transition pair in $\mathcal{U}_i$, with input $\breve{\boldsymbol{v}}_i(t_1)$ and corresponding output equal to the majority vote of the observed outputs. Ties are broken arbitrarily. The inputs in $\mathcal{U}_i$ are subsequently regarded as the sample points of an RM code of unknown dimension, i.e., unknown bounded degree u of the encoder function f_i; then, the corresponding outputs represent noisy sample points of this polynomial. Upon reduction of the set $\mathcal{V}_i^{\varepsilon}$, we have $q^m - |\mathcal{U}_i|$ missing transitions (due to small sample sizes) at node v_i that we model as *erasures* during decoding.

List-decoding is used to explore the space of low-degree polynomials that approximate the above described time series: one starts with the smallest possible degree $u = 1$, and then tries to find polynomials that approximate the transitions in $\mathcal{U}_i$. In other words, the goal is to try to find interpolating functions that can disagree with the pairs in $\mathcal{U}_i$ up to a fraction of points. In the next iteration, we increase u by one and repeat the decoding process. For each value of u, the list of decoded codewords is stored in $\mathcal{C}_i^{(u)}$. The output of the algorithm for node v_i is the list $\mathcal{C}_i = \bigcup_u \mathcal{C}_i^{(u)}$. Optionally, the solution found by the Gröbner basis method as described in Section 7.3 can be added to the list as well.

Algorithm 7.2: List-Decoding of node v_i in the network

Require: $T+1$ transitions for node v_i: $\{(\breve{\boldsymbol{v}}_i(t), v_i(t+1)\}_{t=0}^{T}$, $\breve{\boldsymbol{v}}_i(t) \in \mathbb{F}_q^{m(i)}$, $v_i(t+1) \in \mathbb{F}_q$.

1: Reduce the set of transitions to the reduced set $\mathcal{U}_i$ and regard

$$\mathrm{eval}(f_i) = \{\breve{\boldsymbol{v}}_i(t)\}_{t=0}^{T}$$

as the evaluation points and

$$\boldsymbol{c}_\varepsilon = \{v_i(t+1)\}_{t=0}^{T}$$

as the corresponding noisy codeword from a Reed Muller code.

2: With primitive element ξ of $\mathbb{F}_{q^m}$:

$$\xi^j = \sum_{i=0}^{m-1} a_{ij}\xi^i, \quad 0 \le j \le q^m - 2,\ a_{ij} \in \mathbb{F}_q, \tag{7.48}$$

enumerate the points in $\mathbb{F}_{q^m}$:

$$\begin{aligned} \alpha_0 &= (0, .., 0) \\ \alpha_j &= (a_{0,j-1}, .., a_{m-1,j-1}), \quad j = 1, .., n-1. \end{aligned}$$

3: Sort $\mathrm{eval}(f_i)$ and the elements in $\boldsymbol{c}_\varepsilon$ according to the enumeration.

4: Compute the number of missing values (erasures) for node v_i:

$$s = q^{m(i)} - |eval(f_i)|.$$

5: Set $u = 1$. Choose threshold $t_h(u)$.

6: **while** $s < t_h(u)$ **do**

7: Try to decode $\boldsymbol{c}_\varepsilon$ using Algorithm 7.1 and store the output in $\mathcal{C}^{(u)}$.

8: Increase u

9: Compute the new erasure decoding radius λ_u.

10: **end while**

11: Output

$$\mathcal{L}_i = \bigcup_{u=1} \mathcal{C}^{(u)}.$$

When the algorithm produces a non-empty list for a given node, its output is replaced by one of the decoded words and the measurements of its regulatees are updated. In this way parent nodes can be used to aid the decoding of their children by correcting their input measurements. Our algorithm accounts for this feature through an iterative refinement technique that is used until no changes in the regulatee's lists are observed. The complexity of the algorithm is determined by the complexity of the list-decoding algorithm at hand, which for the PW method is roughly $O(|\mathcal{U}_i|^2\vartheta^4)$ operations per gene. Here, $\vartheta \ge 1$ denotes an adjustable integer parameter that determines the designed decoding radius e_ϑ given in Eq. (7.39). As each node in the network is decoded separately, the complexity

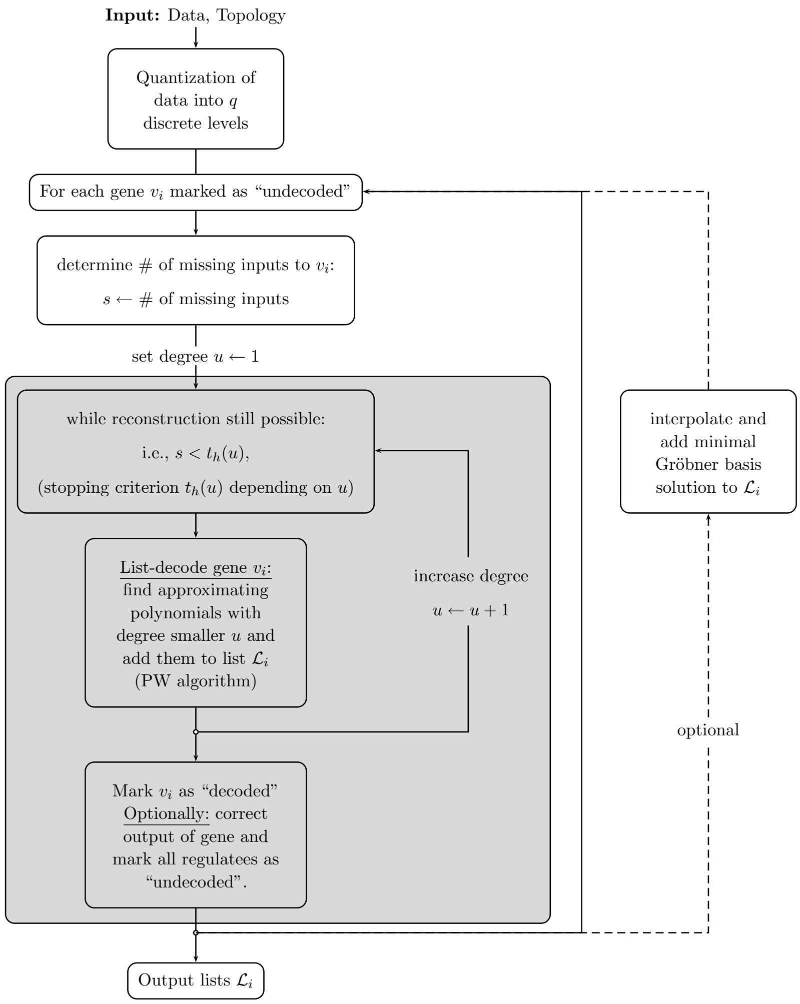


Figure 7.3: Flowchart of our method.

grows approximately linear with the number of nodes in the network (approximately because of the iterative update scheme). The steps of this algorithm are summarized in the Listing 7.2. A flowchart of the overall procedure is shown in Figure 7.3.

7.6.4 Reconstruction bounds

Consider the collection of noisy samples in Eq. (7.38). There are two types of errors: an *erasure* occurs when a sample is erased from the observations. This is equivalent to an unobserved transition at a given network node that arises due to small sample set sizes. An error due to noise occurs at positions j corresponding to $\varepsilon_j \neq 0$. Note the important difference between erasures and errors: in the former case, the positions of inaccurate symbols are known a priori to the algorithm, whereas in the latter case those positions are unknown. Based on the list-decoding framework, it is possible to describe *exact analytical bounds* for the minimum required number of measured transitions for reconstructing update function lists of a given size. It is easy to show that the polynomial f can be *uniquely* recovered from the noisy samples as long as the combined number of errors e and erasures s satisfies $s + 2e < d_{\min}$, where $d_{\min}$ denotes the minimum Hamming distance between any two codewords. Note that this bound applies to both the *noiseless* problem (for which $p_\varepsilon = 0$), as well as the *noisy* reconstruction problem. It is known that for an RM code, $d_{\min}$ depends on the total degree of the polynomials, and $d_{\min}$ decreases when u increases [PW04]. This is a very intuitive result because it basically states that the more non-linearity is present in our network the more samples must be taken in order to infer the functions and less noise can be tolerated in this case.

in-degree 2			in-degree 3		
u	e_ϑ	L_ϑ	u	e_ϑ	L_ϑ
1	13	9	1	66	11
2	8	3	2	44	11
3	5	5	3	27	11
4	2	1	4	12	1
5	1	1	5	9	1

Table 7.3: Decoding radius e_ϑ for different total degrees u and list size upper bound L_ϑ.

When list-decoding is employed, the number of errors and erasures that can be accounted for is significantly larger - linear in the size of the complete dataset. For illustration, several examples of performance characteristics for list-decoders of RM codes with small values of q are shown in Table 7.3. These numerical values are based on the bounds described in the previous section.

7.6.5 Synthetic networks

Performance of our algorithm was tested on random synthetic networks. We simulated 1000 PDS over the alphabet $\mathbb{F}_5$, with 20 nodes and random topologies. Nodes had either in-degree 0 (30%), 2 (50%) or 3 (20%). The degrees of the node update functions were randomly chosen from $\{1, 2, 3, 4, 5\}$. We restricted our attention to small degrees only, since biological networks have similar properties. The network was initialized with a random state $\boldsymbol{v}(0) \in \mathbb{F}_5^{20}$, and five transitions of the network were recorded. This was repeated 50 times, producing a set of 250 synthetic expression samples. Additive white noise samples were added to these "measurements". Figure 7.4 shows the fraction of correctly inferred node functions of our reverse engineering approach for the simulated network, given different noise levels. For comparison, the performance of the LS method described in Section 7.3 is shown as well. Our algorithm significantly outperforms the latter. Even in the noiseless case, we achieve a significantly higher reconstruction rate and the performance of the Gröbner approach drops quite fast when noise is introduced to the measurements. The weak performance of the LS algorithm in the noiseless case is caused by erasures that are independent of measurement noise but arise from the small sample sizes. Note that for the simulations, noise was also added to the inputs $\breve{\boldsymbol{v}}_i(t)$ and the probability that any symbol in the data was changed is termed "the noise level".

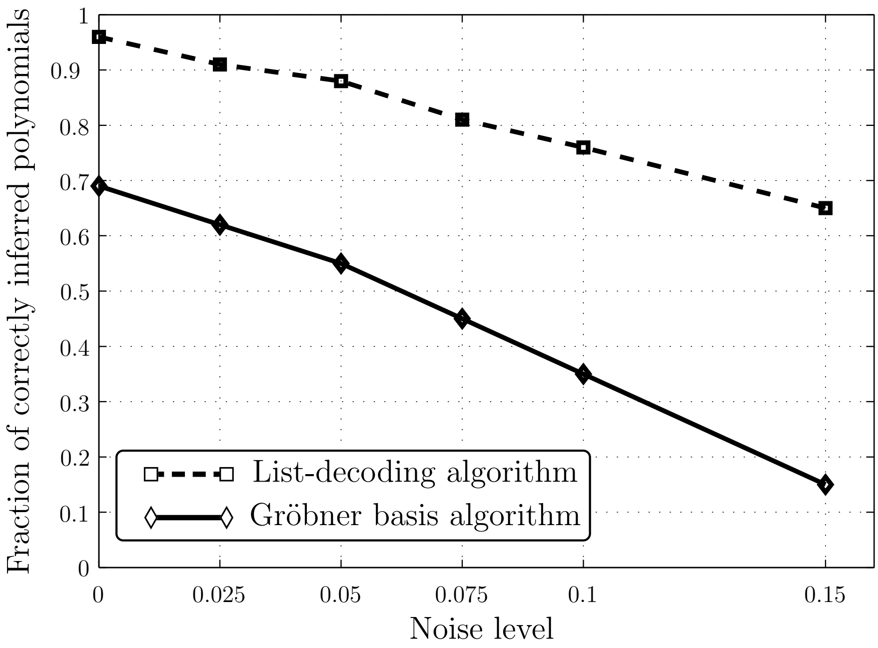


Figure 7.4: *In silico* performance of the proposed algorithm compared to the LS method.

7.7 Application to the *E. coli* network

In this work, our ultimate goal was to apply our algorithms to real data. As a first step, we had to identify datasets (i.e., topologies and microarray data), suitable to carry out meaningful experiments. In this Section, we apply our algorithm to two well studied networks (described below), using data from a recently developed database [FDFe08] (cf. Section 7.7.1). In fact, to the best of our knowledge we are the first to present such an analysis based on biological data under the multi-valued discrete model. In the first analysis (Section 7.7.2), we show that polynomials inferred with our algorithm are statistically significant, using a bootstrap approach. In the second analysis (Section 7.7.3), we consider a whole genome network and develop a new, innovative approach for evaluating our algorithm based on a gene influence measure. We first describe the microarray data used, followed by detailed description of the experiments performed.

7.7.1 Data

Microarray experiments, especially those generating time course data, are still very expensive, and often only extremely small datasets are available for a time course analysis of expression profiles. The number of time points measured in such experiments is typically in the range of $5-15$. That is why many studies based on time course experiments concentrate on synthetically generated data.

Yet, the many small microarray datasets generated by different research groups represent a large resource for network inference. Several online databases provide a central repository of expression data. However, as pointed out more thoroughly in [FDFe08], several problems prevent the efficient exploration or analysis of this data: platform-specific biases in expression data, lack of uniformity in the format and incompleteness and inconsistency of metadata describing the details of each experimental condition that make data fusion very difficult. The data deposited does often not employ a uniform preprocessing approach, nor is the raw intensity data or all the necessary information for normalization provided.

In order to facilitate the analysis of expression data compiled from multiple laboratories, Faith et al. recently developed M^{3D}, a unified microarray database for several microbial organisms [FDFe08]. M^{3D} contains only single-channel arrays using the same platform, and the raw expression data is uniformly normalized to enable the analysis across different experiments without any additional user-dependent processing. Among the species investigated in M^{3D} is the bacterium *E. coli.*, which is one of the best studied organisms today. In fact, the complete topology of the regulatory net of *E. coli.* is believed to be known. We used the M^{3D} microarray data for *E. coli.* and filtered time course experiments providing at least one transition of the network. We found a total of 90 time-points from 21 different experiments, resulting in a total of 69 transitions.

7.7.2 *E. coli* SOS response network

We extracted a small sub-network consisting of 9 genes whose topology was inferred in [GdLC03]. DNA is constantly suffering from environmental stresses. The bacterium *E. coli* responds to these attacks by expressing a certain number of genes allowing the DNA to be repaired or replicated despite damage, or to trigger other mechanisms such as apoptosis. In *E. coli* and other bacteria, the regulatory system coordinating this response is called the *SOS regulon* [dG05]. The topology of the network is depicted in Figure 7.5. Our nine gene test network includes the transcription factors *lexA* and *recA*, genes that catalyze DNA strand exchange and renaturation. Both genes are known to interact with each other and to regulate many genes directly and tens or possibly hundreds indirectly. LexA is a transcription factor repressing the genes of the SOS regulon during normal growth. RecA functions as a detector of DNA damage while LexA regulates the response to this stress. The four genes (*ssb, recF, dinI and umuDC*) are regulatory genes involved in the SOS response system, while the three genes (*rpoD, rpoH* and *rpoS*) code for sigma factors, important proteins involved in the process of gene transcription.

We applied our algorithm using different numbers of quantization levels q. At quantization level $q = 5$ we found statistically significant low-degree approximating polynomials for three of the nine genes in the network. The inferred input/output responses are shown in Figure 7.6. Statistical significance was evaluated by randomly choosing nine genes from the available set of 4292 genes, and by applying the reverse engineering algorithm assuming the same topology as in Figure 7.5. Functions found in this way are counted as false positives. This experiment was repeated 1000 times and the number of non-empty lists was counted. Only in six out of the 1000 iterations, three nodes were simultaneously inferred in the same network. The overall observed false positive rate per node was less than 3.8%. The sigma factors *rpoS* and *rpoH* exhibit identical responses; both are

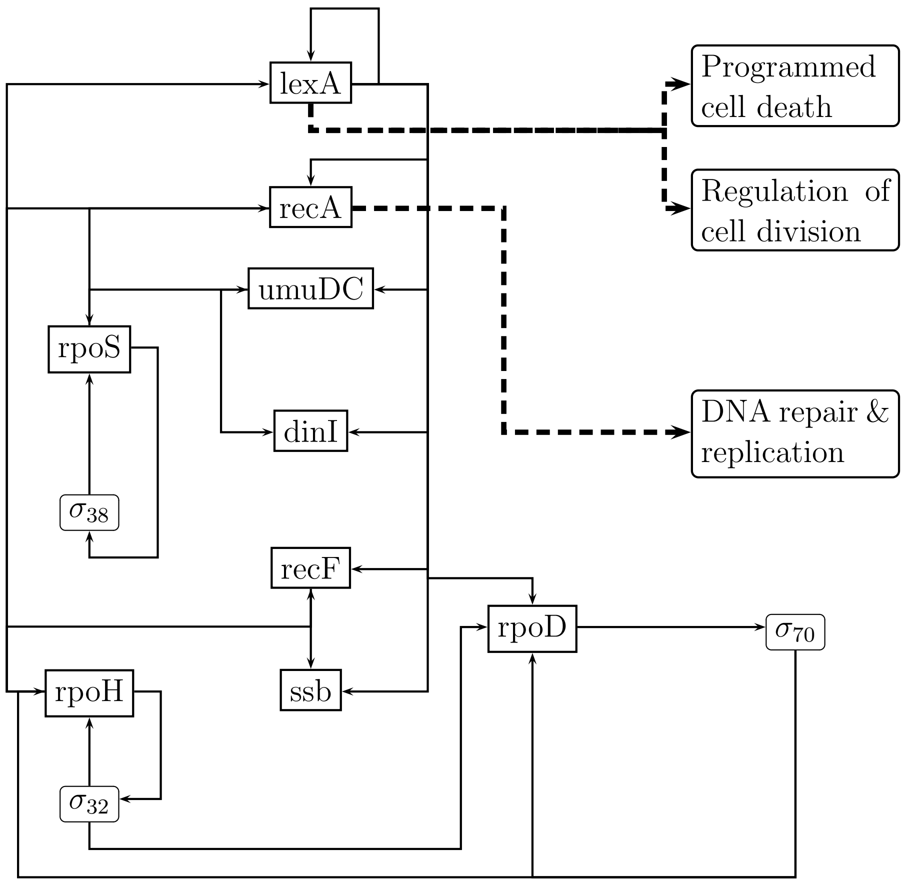


Figure 7.5: Diagram of interactions in the SOS network as described in [GdLC03]. Angled boxes denote genes, small rounded boxes proteins.

regulated by *rpoD* and include a self-loop, which may explain this finding. All inferred polynomials are linear ($u = 1$), indicating that non-linear functions could not be found with the provided number of samples and quality of data.

7.7.3 Global *E. coli* transcription factor network

We also applied our method to the complete transcription factor gene network available at the RegulonDB database [GCJJPG⁺08]. RegulonDB provides curated information on gene organization and regulation in *E. coli.*. The information includes the transcription units and the mechanical details of regulation at various levels. Among other information, the whole transcriptional regulation network is available for download. We removed all genes from the RegulonDB transcriptional regulation network for which there were no matching entries in the M^{3D} database, so that the reduced network consisted of 1384 genes. Our algorithm was applied to all nodes in this network with in-degree 2 or 3 (a total of 526 genes), by setting $q = 3$ for all nodes. In-degree distribution of the network is shown in Figure 7.7. Additionally, polynomials inferred by the LS method were added to the lists. In order to verify that the new constructive approach can improve the prediction of

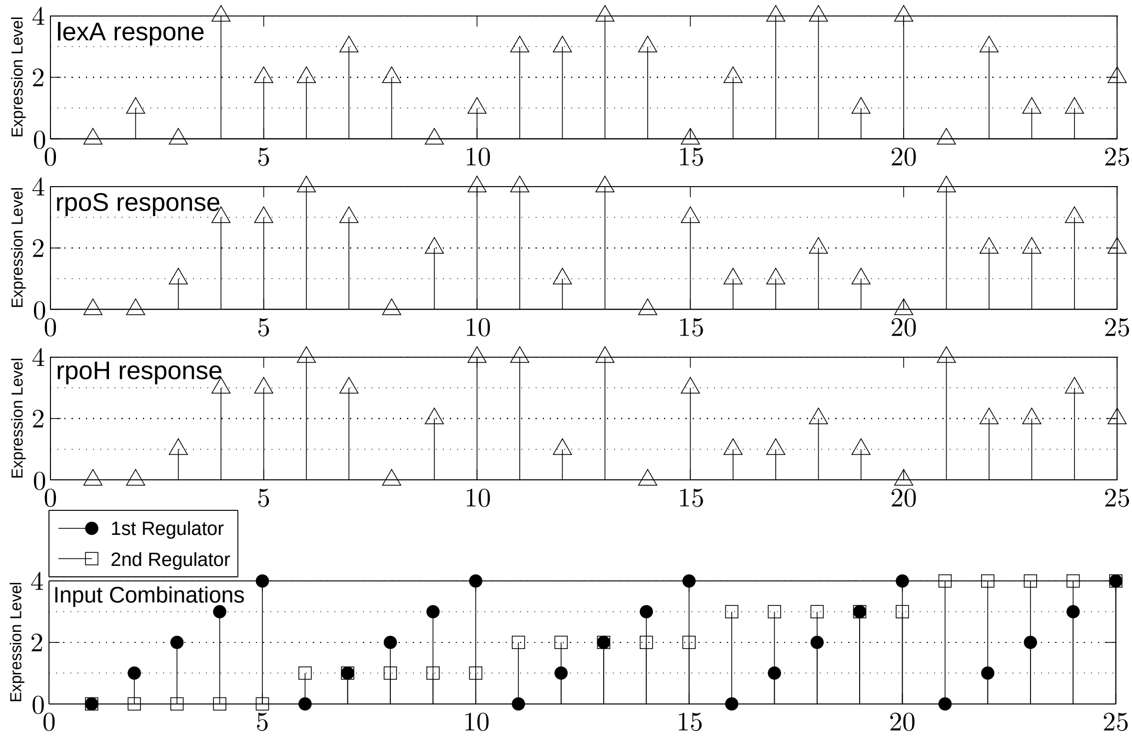


Figure 7.6: Inferred responses of the *E. coli.* genes *lexA, rpoS, rpoH.* Each of these genes has 2 regulators that can take 5 different values resulting in $5^2 = 25$ possible input combinations $\breve{\boldsymbol{v}}_i(t)$ that are shown in the last row. Rows 1 to 3 show the quinternary response $v_i(t+1)$ of the genes to the corresponding input at time t depicted in row 4.

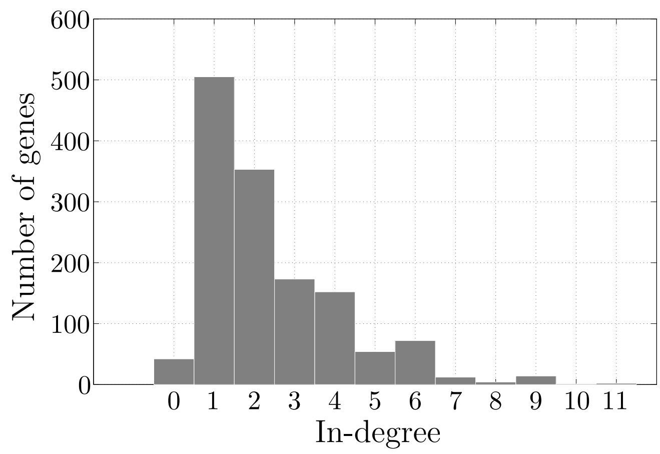


Figure 7.7: In-degree distribution in the considered RegulonDB gene network.

the system dynamics, the following experiment was performed: first, an influence measure was developed as follows. With the notation $\boldsymbol{x}^{(j,k)} = (x_1, .., x_j = k, .., x_m)$, the *influence* of gene v_j on v_i is defined as

$$I_{ji} = \mathbb{E}_{\boldsymbol{x}} \left\{ \frac{1}{2} \sum_{\substack{(k,l) \\ k \neq l}} \frac{f_i(\boldsymbol{x}^{(j,k)}) - f_i(\boldsymbol{x}^{(j,l)})}{k - l} \right\}. \tag{7.49}$$

Equation (7) is a mathematical formalism for measuring the influence of a gene on its regulatees. For all input combinations, we calculate the change in the output of gene v_i when changing the input of v_j from l to k, while fixing all regulators other than v_j. Expression levels are regarded as integers, and the expectation is taken with respect to all regulators except v_j. The resulting variable I_{ji} should have the following properties: the sign of I_{ji} must give the type of regulation (activating or repressing). The magnitude $|I_{ji}|$ should correlate with the strength of this interaction. Note that several other influence measures could be used.

In the second step, the values of I_{ji} were used to indicate an activating influence ($I_{ji} > 0$) of gene v_j on gene v_i, or repressing influence ($I_{ji} < 0$). The magnitude I_{ji} reflects the strength of the influence.

We compared our predictions obtained from Eq. (7.49) with the influences of regulators annotated in RegulonDB. Influences I_{ji} for all edges that connect to a gene with in-degree 2 or 3 (1225 edges) were calculated before and after applying list-decoding. Calculation of the influence before decoding was based on taking the consensus of a gene's output for multiple transitions and simply neglecting unobserved input combinations in Eq. (7.49). We then ranked interactions according to their influence magnitude $|I_{ji}|$ and compared the influence predictions based on $\mathrm{sign}(I_{ji})$ to the RegulonDB annotation for different set sizes of high ranking interactions. Figure 7.8 shows the fraction of the top $600, 500, ..., 200$ ranked genes in terms of $|I_{ji}|$ matching the annotation in RegulonDB. While there is no indication for correlation of matching predictions with high $|I_{ji}|$ before "decoding" we observe that after "decoding" the predictions were significantly improved up to approx. 70% matching for the 200 highest ranked interactions. The distributions of normalized influence values for the interactions in the *E.coli* network before and after decoding are shown in Figure 7.9. It is observed that the decoding leads to a notable change in the distributions which improves the predictions of interactions as indicated by the results shown in Figure 7.8.

7.8 Discussion

The presented approach to the reverse engineering of gene network dynamics builds upon two important principles: that biological systems are inherently stochastic and that only small and possibly erroneous data sets are available for analysis. Our approach accounts for these facts by setting the problem into a probabilistic and noisy framework, and by allowing for non-unique solutions explaining each node dynamics with a whole list

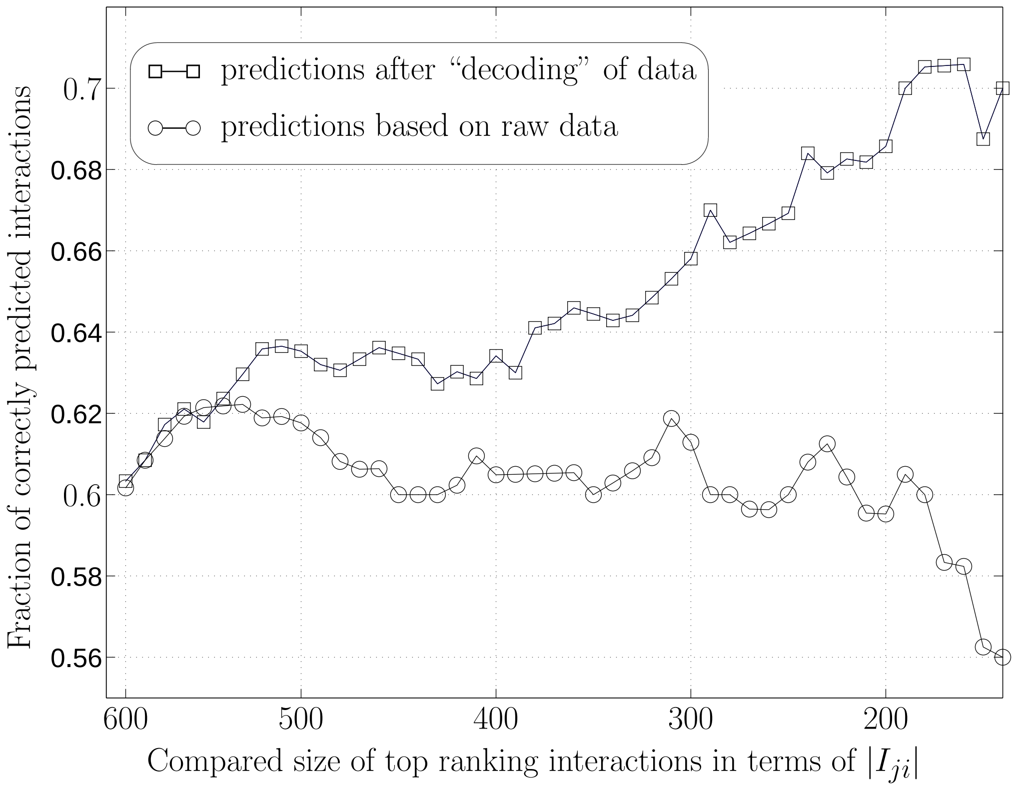


Figure 7.8: Predictive power of influence values as obtained from Eq. (7.49) for the *E. coli* transcription factor network before and after applying our algorithm.

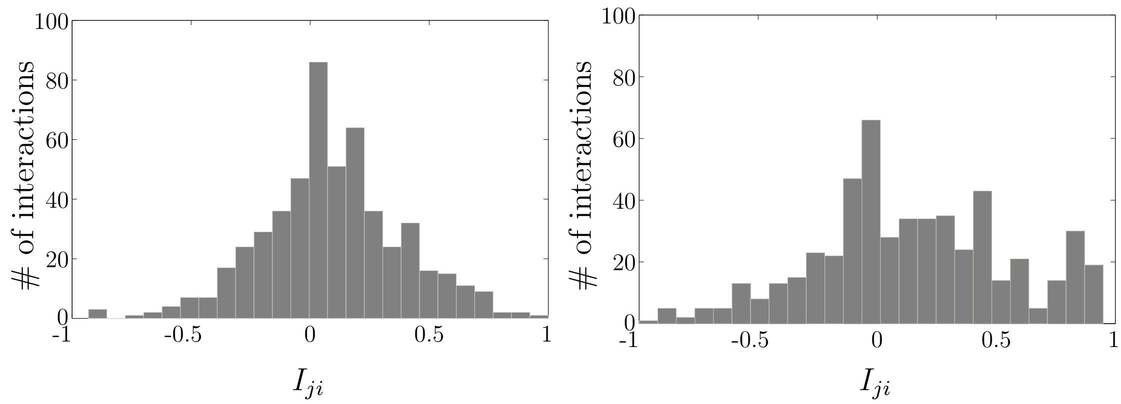


Figure 7.9: Left: distribution of network influence values calculated from quantized microarray data. Right: distribution of network influence values calculated after applying our algorithm.

of functions. Our algorithm is *guaranteed* to find all approximating polynomials that generate time series within a predefined Hamming distance from the measured data, as long as the combined number of missing values (erasures) and errors does not exceed the decoding radius of the list-decoder. The search space of all possible polynomials, however, can still be considerably large. The list-decoding algorithms applied here allow to find these polynomials with very reasonable complexity. Ideally, we would take the measurement error quantitatively into account and select solutions that are optimal under model selection criteria such as the *minimum description length principle*, the *Bayesian information* or *Akaike information criterion* [Gru04]. However, since the noise cannot be quantified, we successively raise the degree bound and add all solutions to the list that can be found via list-decoding for a certain degree restriction (i.e., we add all the polynomials that lie within the list-decoding radius). The final list then contains a mix of high degree and low degree polynomials.

The decoding radius of the list-decoder is influenced by an adjustable parameter and can be increased at the expense of increased reconstruction complexity. In our simulations, we used the parameter setting that ensures lowest complexity. Thus, the performance of our algorithm represents a lower bound on the achievable performance. As expected, our algorithm outperforms the approach presented by LS significantly both in the noisy and noiseless scenario. We used the LS method as a comparative reference; however, we would like to point out that this method was originally developed for joint inference of both, the topology *and* the dynamics of the network. Many methods concentrating solely on the inference of the network topology are known [DJLS07], and it may be reasonable to separate these two problems. While there exist many approaches focussing on the reverse engineering of network topology, few studies analyze the problem of inferring the dynamics assuming that information about the network topology is given *a priori* under the discrete model. In fact, we are not aware of any prior approach taking into account multivalued discrete models, noise, and missing data samples jointly. To the best of our knowledge, this work is the first to establish a connection between the reconstruction problem and coding theory, allowing for a rigorous theoretical analysis of the problem such as the derivation of minimal number of timepoints required. We therefore believe that our analysis can have great impact on both, theory and practice of the reverse engineering gene regulatory network problem, all within this modeling framework.

The application of our algorithm to time course data from *E. coli* validates the usefulness of our approach even with the present small amount of data available. The analysis on the SOS network suggests that at least some gene functions can be approximated by low degree polynomials while more data seems to be necessary for robust regression of other functions. Furthermore, our approach is constructive in the sense that it predicts well defined functions that specify the response to *all* possible combinations of inputs, even those that are not observed in the data. The E.coli SOS subnetwork involves transcription factors, genes coding for sigma factors, repair genes, and other genes involved in the SOS response. In its present form it is not possible to model multiple levels of gene expression in different ways with the algebraic model. However, this applies to discrete gene models in general. One could argue whether this is a drawback or a feature of these models. Discrete models define gene networks on a higher level of abstraction than, for example, quantitative models based on differential equations.

The results regarding the transcription factor network of *E. coli* also show that our constructive approach can significantly improve the analysis of certain features of system dynamics, such as gene influence. Furthermore, the presented framework based on influence measures could serve as an evaluation standard for other methods developed in the future.

7.9 Summary

In this chapter, we applied list-decoding of Reed-Muller codes to infer polynomials in algebraic gene network models. Our analysis is cast within a new modeling framework recently presented by Laubenbacher et al. [JLSS07], where each gene in the network is described by a multivariate polynomial over a finite field.

Our reverse engineering approach was based on two observations: that gene networks are inherently stochastic and that expression data obtained from microarray experiments are noisy. To account for noise, we proposed to search for solutions that *approximate* time series of gene expression, similar to a regression analysis. Stochasticity and small sample sizes were taken into consideration by describing each gene with a whole probabilistic list of functions, instead of a single deterministic solution. Interpolation and approximation of polynomials is a well studied problem in coding theory due to its application in the decoding of certain codes defined over polynomials. Here, we showed that the reverse engineering problem is connected to the decoding of generalized Reed-Muller codes.

We subsequently presented a list-decoding based algorithm for the construction of lists of polynomials for each gene from time course expression data. Using simulated networks, our algorithm was shown to outperform the algorithm presented by Laubenbacher and Stigler [JLSS07]. Our ultimate goal, however, was to apply our algorithm to biological expression data. Making use of a recently developed database, we were able to find polynomials for genes in a sub-network of *E. coli*, and statistical significance was confirmed using a bootstrap approach. Applied to the global *E. coli* transcription factor network, our decoding algorithm could significantly improve predictions about network features compared to an *ad-hoc* application of quantized expression data. Evaluation was based on a measure of gene influence developed for this purpose. To our knowledge we are the first to apply the multivalued discrete model to biological expression data - previous work was purely based on simulations.

7.10 Future research

A key step in the application of discrete gene network models to expression data is quantization. Although this is a well known fact, the development of quantization schemes especially designed for expression values has drawn only little attention. It is, in general, unknown how different reverse engineering methods perform under different quantization schemes. For example, a rigorous analysis of the influence of the number of quantization levels on the accuracy of the reconstruction method was beyond the scope of the present work and could be adressed in future contributions.

Since gene functions are not arbitrary, a major improvement is expected from incorporating biological prior knowledge into the reverse engineering procedure. In our case, this amounts to using a subset of codewords of RM codes, rather than the whole code. Subsets must be chosen based on biological considerations. On the algorithmic side, our method could be improved using information obtained from *multiple state transitions*. Such information is currently neglected but may be used in *soft-decision* list-decoding algorithms [KV03] that provide more accurate reconstructions than those obtainable from quantized data.

We used a heuristic approach to fit expression data under the multivalued discrete model that increased the total degree of polynomials until a stopping condition was reached. Ideally, we would trade-off goodness of fit versus parametric complexity of the polynomial class using model selection principles. However, such an approach would require to quantify the noise in expression data which might be difficult. We therefore recommend to investigate model selection as an extension as soon as measurement noise of microarray data is better understood.

8

Conclusion

The results of the Human Genome and the ENCODE project have shown that cellular information processing, in particular in higher organisms is far more complex than previously thought [LLB+01, The07, MCA+07, Che07]. Information and communication theory constitute the frameworks for an analysis and optimization of digital systems. Due to the digital nature of biological systems on the molecular level, these can be meaningfully applied in this context as well. Based on the idea that methods used in human-made systems could prove useful to explain phenomena observed in the cell, communication engineers recently started to cooperate with biologists [DGH+06, GDHM05, DHHM05, KH08, Mil06, MV04, AKL+07, DTA08, RGH+07]. This thesis aimed at further fostering this cooperation and achieving appreciation for this interdisciplinary approach in both scientific communities. We considered three problems in computational biology that have analogies to problems considered in communication theory. For each of the three cases, we developed algorithms based on probabilistic models and evaluated their performance using simulated data. Then, we applied our algorithms to carefully chosen biological data sets. This was the ultimate goal of this thesis since we believe that application to biological data must be included in any meaningful analysis in computational biology. We shall briefly summarize this work by pointing out the main achievements:

In Chapter 5, we considered the problem of identifying DNA sequences that are conserved among multiple species. Such sequences are potential candidates for basic functional units in the genome. We introduced probabilistic phylogenetic models of evolution from a single common ancestor DNA sequence to sequences observed in today's species (single input multiple output system). We showed how such systems are reconstructed under a continuous time Markov substitution model using maximum likelihood methods. We then focused on the detection of conserved sequences that evolve under such a model and presented current state-of-the-art approaches. We introduced our new method, KuLcons, based on relative entropy that allows for a fine-scale conservation score without assumptions about neutral substitution rates - an important advantage compared to most earlier approaches. After an analysis of our algorithm, simulations aiming at the comparison with other state-of-the-art methods were performed. Results based on a very general simulation model suggest that our approach outperforms other methods. We applied KuLcons to an ENCODE region and compared our scores qualitatively with those used by the ENCODE project consortium. It was observed that our scores were in agreement with two of the three ENCODE scores. Since our score is independent of neutral substitution rates, our result suggests that the discrepancy between computationally predicted and experimentally verified functional regions in the ENCODE project is most likely not a result of biased neutral substitution rate estimates - opposed to the conjecture raised by several researchers before [MVH+07, MCA+07, Che07, PM07].

In Chapter 6, we presented a first systematic approach to the claim of Battail that highly conserved regions, such as those detected by KuLcons, could only be explained by a genomic error correcting code [Bat08]. We discussed Battail's hypothesis and presented a coding theoretic view on known cellular error correcting mechanisms. We then focused on the technical code reverse engineering problem of convolutional codes given an observed noisy sequence of coded symbols. Our approach was based on maximum likelihood estimation under a log-likelihood ratio encoder tap model. Simulation results on synthetic data showed that reconstruction of encoders is achieved at very low false positive rates from a reasonable number of observed symbols for binary and quarternary alphabets. We applied the algorithm to an ultraconserved DNA region but found no indication for a coding structure compared to a random sequence. We also discussed difficulties of such an analysis and listed issues that remain to be resolved before further going in this direction.

A code reverse engineering problem also arises in the analysis of the dynamics of algebraic gene network models, generalizations of probabilistic Boolean models [SDKZ02, JLSS07, AGM04, Jus06, DE09]. In Chapter 7, we showed how the reverse engineering problem of inferring the network dynamics is related to coding theory and in particular the decoding of Reed-Muller codes. We then presented an algorithm based on list-decoding to account for noise and small sample size of expression data, and the inherent stochasticity of biological networks. We applied our algorithm to biological expression data - to our knowledge for the first time under the discrete multivalued model - adressing issues such as reconstruction bounds, data quantization, and multiple transitions. Using a bootstrap approach we obtained statistically significant results on a small *E. coli* network. Predictions of gene influence in the whole genome *E. coli* gene network were shown to be significantly improved after processing the data with our decoding algorithm. The established relation to coding theory is expected to lead to major improvements in the theoretical analysis of gene networks under the discrete model.

⋆ ⋆ ⋆

Our work showed that important insights to biological problems can be gained approaching them from a communication theoretic perspective. Analytical methods are becoming increasingly important in molecular biology, and new interdisciplinary sciences such as *systems biology* and *synthetic biology* just started to emerge. We believe that the formal, system-theoretic view provided by communication and information theory can play a major role in these research areas.

A

List of publications

Parts of this work were published in the following articles:

Journal papers

[1] P. Hanus, J. Dingel, G. Chalkidis and J. Hagenauer: Compression of multiple sequence alignments. *IEEE Transactions on Information Theory, Special Issue on Information Theory in Molecular Biology and Neuroscience (accepted for publication)*, 2009.

[2] J. Dingel and O. Milenkovic: List-Decoding methods for inferring the polynomials of finite discrete gene networks. *Bioinformatics*; doi: 10.1093/bioinformatics/btp281, Oxford University Press, 2009.

[3] J. Dingel, P. Hanus, N. Leonardi, J. Zech, J. Hagenauer, and J.C. Mueller: Local conservation scores without a priori assumptions on neutral substitution rates. - In: *BMC Bioinformatics*, (9):90, 2008.

[4] P. Hanus, B. Goebel, J. Dingel, J. Weindl, J. Zech, Z. Dawy, J. Hagenauer and J. C. Mueller: Information and communication theory in molecular biology. - In: *Electrical Engineering (Archiv für Elektrotechnik)*, Vol. 90, No. 2, Springer, December 2007.

Peer reviewed conference papers

[5] P. Hanus, J. Dingel, G. Chalkidis, and J. Hagenauer: Source coding scheme for multiple sequence alignments. - In: *Data Compression Conference*, Snowbird, Utah, March 2009, pp. 183-192. **Best paper award**.

[6] J. Dingel, N. Singh, and O. Milenkovic: Inferring algebraic gene networks using local decoding. - In: *Proc. IEEE International Conference on Bioinformatics and Biomedicine - Workshop on Systems Biology and Medicine*, Philadelphia, USA, September 2008, pp. 125-126.

[7] J. Dingel and O. Milenkovic: A list-decoding approach for inferring the dynamics of gene regulatory networks. - In: *Proc. IEEE International Symposium on Information Theory* (ISIT08), Toronto, Canada, July 2008, pp. 2282-6.

[8] J. Dingel and O. Milenkovic: Decoding the dynamics of gene regulatory networks. - In: *Proc. of the 3rd International Workshop on Computational Systems Biology* (WCSB08), Leipzig, Germany, June 2008, pp. 33-36. **Best paper award**.

[9] J. Dingel and J. Hagenauer: Parameter estimation of a convolutional encoder from noisy observations. - In: *Proc. of the International Symposium of Information Theory (ISIT07)*, Nice, France, June 2007, pp. 1776-80.

Invited papers & posters

[10] N. Santhanam, J. Dingel and O. Milenkovic: Information theoretic modeling of gene interactions (invited), - In: *Proc. of the IEEE Information Theory Workshop* (ITW09), Volos, Greece, June 2009.

[11] J. Dingel and O. Milenkovic: Coding theoretic methods for reverse engineering of gene regulatory networks (invited). - In: *Proc. of the IEEE Information Theory Workshop* (ITW08), Porto, Portugal, May 2008, pp. 114-118.

[12] Z. Dawy, P. Hanus, J. Weindl, J. Dingel, and F. Morcos: On genomic coding theory (invited). - In: *European Transactions on Telecommunications (ETT)*, December 2007, Wiley & Sons Ltd., InterScience, Vol. 18, No. 8, ISSN 1124-318X, pp. 873-879.

[13] P. Hanus, J. Dingel, J. Zech, J. Hagenauer, and J.C. Mueller: Information theoretic distance measures in phylogenomics (invited). - In: *Proc. of the International Workshop on Information Theory and Applications* (ITA07), San Diego, USA, January 2007, pp. 421-425.

[14] P. Hanus, J. Dingel, J. Hagenauer, and J.C. Mueller: An alternative method for detecting conserved elements in multiple sequence alignments. - In: *European Conference on Computational Biology - Student Council* (ECCB 2005), Madrid, Spain, September 2005, pp. 18-20.

[15] P. Hanus, J. Dingel and J. Hagenauer and J.C. Mueller: An alternative method for detecting conserved regions in multiple species. - In: *Proc. of the German Conference on Bioinformatics* (GCB06), Hamburg, Germany, October 2005, pp. 64.

B

Proofs

B.1 Proof of Theorem 2

Proof Due to the Perron-Frobenius theorem [Mey00, Chapter 8], a stochastic matrix with strictly positive entries has a simple eigenvalue $\omega_1 = 1$, and all other eigenvalues lie inside the unit disk, i.e., $\omega_1 > |\omega_i|, \ \forall i = 2, .., |\mathcal{S}|$. From Definition 2.4.4 of the stationary distribution, we see that $\boldsymbol{\pi}$ is the left eigenvector of $\boldsymbol{P}$ corresponding to $\omega_1 = 1$. Suppose $\boldsymbol{P}$ can be decomposed as $\boldsymbol{P} = \boldsymbol{U}^{-1}\boldsymbol{\Omega}\boldsymbol{U}$, where $\boldsymbol{\Omega} = \mathrm{diag}(\omega_1, ..., \omega_{|\mathcal{S}|})$ denotes the diagonal matrix with the eigenvalues in the diagonal and zeros in the off-diagonals, and $\boldsymbol{U} = [\boldsymbol{\pi}^T, \boldsymbol{u}_2^T ..., \boldsymbol{u}_{\mathcal{S}}^T]^T$ denotes the corresponding left eigenvectors. Define $\boldsymbol{U}^{-1} = \boldsymbol{V} = [\boldsymbol{v}_1^T, \boldsymbol{v}_2^T, ..]$, then for any distribution $\boldsymbol{\lambda}$ as n goes to infinity:

$$
\begin{aligned}
\boldsymbol{p}_\infty &= \lim_{n\to\infty} \boldsymbol{\lambda}\boldsymbol{P}^n = \boldsymbol{\lambda}\boldsymbol{V} \lim_{n\to\infty} \boldsymbol{\Omega}^n\boldsymbol{U} && \text{(B.1a)}\\
&= \boldsymbol{\lambda}\boldsymbol{V} \begin{pmatrix} 1 & 0 & ... & 0 \\ 0 & 0 & ... & 0 \\ \vdots & & & \vdots \\ 0 & & ... & 0 \end{pmatrix} \boldsymbol{U} = \boldsymbol{\lambda}\boldsymbol{V} \begin{pmatrix} \boldsymbol{\pi} \\ \boldsymbol{0} \\ \vdots \\ \boldsymbol{0} \end{pmatrix} = \boldsymbol{\lambda}(\boldsymbol{\pi} \otimes \boldsymbol{v}_1^T) && \text{(B.1b)}\\
&= \boldsymbol{\pi}\boldsymbol{\lambda}\boldsymbol{v}_1^T = \boldsymbol{\pi} c, && \text{(B.1c)}
\end{aligned}
$$

where $\otimes$ denotes the Kronecker symbol $(\boldsymbol{\pi} \otimes \boldsymbol{v}_1^T) = (\pi_1\boldsymbol{v}_1^T, \pi_2\boldsymbol{v}_1^T, ..., \pi_{\mathcal{S}}\boldsymbol{v}_1^T)$. Since $\boldsymbol{p}_\infty$ has to be a distribution, the constant $\boldsymbol{\lambda}\boldsymbol{v}_1^T$ must be 1 and therefore $\boldsymbol{p}_\infty = \boldsymbol{\pi}$. ■

B.2 Proof of Theorem 3

Proof Suppose $\boldsymbol{R}$ can be decomposed into $\boldsymbol{R} = \boldsymbol{U}\boldsymbol{\Phi}\boldsymbol{U}^{-1}$,

(i) For any $h, t \in \mathbb{R}^+$,

$$
\begin{aligned}
\boldsymbol{P}(h+t) &= \boldsymbol{U}e^{(h+t)\boldsymbol{\Phi}}\boldsymbol{U}^{-1} = \boldsymbol{U}e^{h\boldsymbol{\Phi}}e^{t\boldsymbol{\Phi}}\boldsymbol{U}^{-1} && \text{(B.2a)}\\
&= \boldsymbol{U}e^{h\boldsymbol{\Phi}}\boldsymbol{U}^{-1}\boldsymbol{U}e^{t\boldsymbol{\Phi}}\boldsymbol{U}^{-1} = \boldsymbol{P}(h)\boldsymbol{P}(t). && \text{(B.2b)}
\end{aligned}
$$

(ii) For any $t \in \mathbb{R}$,

$$\boldsymbol{P}(t)\boldsymbol{R} = \boldsymbol{U}e^{t\boldsymbol{\Phi}}\boldsymbol{U}^{-1}\boldsymbol{U}\boldsymbol{\Phi}\boldsymbol{U}^{-1} = \boldsymbol{U}e^{t\boldsymbol{\Phi}}\boldsymbol{\Phi}\boldsymbol{U}^{-1} \tag{B.3a}$$
$$= \boldsymbol{U}\boldsymbol{\Phi}e^{t\boldsymbol{\Phi}}\boldsymbol{U}^{-1} = \boldsymbol{U}\boldsymbol{\Phi}\boldsymbol{U}^{-1}\boldsymbol{U}e^{t\boldsymbol{\Phi}}\boldsymbol{U}^{-1} = \boldsymbol{R}\boldsymbol{P}(t). \tag{B.3b}$$

(iii) For $t \geq 0$,

$$\frac{d}{dt}\boldsymbol{P}(t) = \boldsymbol{U}\frac{d}{dt}e^{t\boldsymbol{\Phi}}\boldsymbol{U}^{-1} = \boldsymbol{U}e^{t\boldsymbol{\Phi}}\boldsymbol{\Phi}\boldsymbol{U}^{-1} = \boldsymbol{U}e^{t\boldsymbol{\Phi}}\boldsymbol{U}^{-1}\boldsymbol{U}\boldsymbol{\Phi}\boldsymbol{U}^{-1} = \boldsymbol{P}(\boldsymbol{t})\boldsymbol{R}. \tag{B.4}$$

It remains to show that $\boldsymbol{P}(t) = e^{t\boldsymbol{R}}$ is the unique solution: assume that $\boldsymbol{M}(t)$ satisfies the equation, then

$$\frac{d}{dt}(\boldsymbol{M}(t)e^{-t\boldsymbol{R}}) = \frac{d}{dt}\boldsymbol{M}(t)e^{-t\boldsymbol{R}} + \boldsymbol{M}(t)\frac{d}{dt}e^{-t\boldsymbol{R}} \tag{B.5a}$$
$$= \boldsymbol{M}(t)\boldsymbol{R}e^{-t\boldsymbol{R}} + \boldsymbol{M}(t)(-\boldsymbol{R})e^{-t\boldsymbol{R}} = 0. \tag{B.5b}$$

Hence $\boldsymbol{M}(t)e^{-t\boldsymbol{R}}$ is constant and $\boldsymbol{M}(t) = \boldsymbol{P}(t)$.

(iv) Follows from (iii) and (ii). ■

B.3 Proof of Theorem 5

Proof Let $T_{s_is_j}$ be the time that the process spends in state s_i before entering s_j, and let $T_{s_is_i}$ be the time that the process spends in s_i. Assume that $T_{s_is_j}$ and $T_{s_is_i}$ are exponentially distributed with means $E\{T_{s_is_j}\} = \frac{1}{r_{s_is_j}}$ and $E\{T_{s_is_i}\} = -\frac{1}{r_{s_is_i}}$ respectively:

$$T_{s_is_j} \sim r_{s_is_j}e^{-r_{s_is_j}T_{s_is_j}},$$
$$T_{s_is_i} \sim -r_{s_is_i}e^{r_{s_is_i}T_{s_is_i}}.$$

Suppose $T_{s_is_j}$ and $T_{s_is_i}$ have these distributions, let $o(t) = f(t)$ mean $f(t)/t \to 0$ as $t \to 0$. Then for a small h

$$P(x_{t+h} = s_i|x_t = s_i) = P_{ii}(h) = P(T_{s_is_i} \geq h) = e^{r_{s_is_i}h} = 1 + r_{s_is_i}h + o(h), \tag{B.6}$$

$$P(x_{t+h} = s_j|x_t = s_i) = P_{ij}(h) \geq P(T_{s_is_j} \leq h, T_{s_js_j} \geq h) \tag{B.7a}$$
$$= (1 - e^{-r_{s_is_j}h})(e^{-r_{s_js_j}h}) = r_{s_is_j}h + o(h). \tag{B.7b}$$

By taking the sum of transition probabilities over j we see that there must be equality in (B.7a)

$$1 = \sum_j P_{ij}(h) = 1 + h\sum_j r_{s_is_j} + o(h) = 1 + o(h). \tag{B.8}$$

So we find that the transition rates and transition probabilities are related by

$$P_{ii}(h) = 1 + r_{s_is_i}h + o(h), \tag{B.9a}$$
$$P_{ij}(h) = r_{s_is_j}h + o(h). \tag{B.9b}$$

Denote $p_{ij} = P(x_t = s_j | x_0 = s_i)$, for any $t, h \geq 0$

$$
\begin{aligned}
P(x_{t+h} = s_j | x_0 = s_i) &= \sum_k P(x_t = s_k | x_0 = s_i) P(x_{t+h} = s_j | x_t = s_k) \\
p_{ij}(t+h) &= \sum_k p_{ik}(t) P_{kj}(h) && \text{(B.10a)} \\
&\overset{\substack{\text{(B.9a)} \\ \text{(B.9b)}}}{=} \sum_{\substack{k \\ k \neq j}} p_{ik}(t)(r_{s_k s_j} h + o(h)) + p_{ij}(1 + r_{s_j s_j} h + o(h)), && \text{(B.10b)}
\end{aligned}
$$

and as h goes to zero,

$$
\begin{aligned}
\lim_{h \to 0} \frac{p_{ij}(t+h) - p_{ij}(t)}{h} &= \sum_k p_{ik}(t) r_{s_k s_j}, && \text{(B.11a)} \\
\frac{d}{dt} p_{ij}(t) &= \sum_k p_{ik}(t) r_{s_k s_j}. && \text{(B.11b)}
\end{aligned}
$$

Hence, $\boldsymbol{P}(t)$ must satisfy the forward equation

$$
\frac{d}{dt} \boldsymbol{P}(t) = \boldsymbol{P}\boldsymbol{R}, \quad \boldsymbol{P}(0) = \boldsymbol{I}.
$$

By Theorem 3, $\boldsymbol{P} = e^{t\boldsymbol{R}}$ is the unique solution to this equation.
We have assumed that $T_{s_i s_j}, T_{s_i s_i}$ are exponentially distributed and we have shown that if a process has such $T_{s_i s_j}, T_{s_i s_i}$, then its transition probabilities are given by $\boldsymbol{P} = e^{t\boldsymbol{R}}$. But $\boldsymbol{P}$ determines the distribution of $\{\mathsf{X}_t\}_{t \geq 0}$ and hence the distribution of $T_{s_i s_j}$ and $T_{s_i s_i}$. So every process satisfying $\boldsymbol{P} = e^{t\boldsymbol{R}}$ must satisfy the assumption. The definitions of the process in terms of $\boldsymbol{P} = e^{t\boldsymbol{R}}$ and $T_{s_i s_j}, T_{s_i s_i}$ are equivalent. ■

It is possible to prove a stronger version of this Theorem in a more rigorous way. However, we have presented a simplified version, as additional Definitions and Theorems are necessary in this case that are of limited interest for the rest of this work.

C

Comparison of KuLcons on ENCODE data

Plots of KuLcons scores versus smoothed versions of SCONE (Figure C.1) and GERP (Figure C.2) scores obtained from the UCSC database over an ENCODE region. In order to facilitate the comparison, we show a transformed version of our score, that is,

$$1 - \frac{\mathfrak{s}_l}{\max_l\{\mathfrak{s}_l\}}, \tag{C.1}$$

where $\mathfrak{s}_l$ denotes the conservation score as derived in Section 5.4.2. A similar transformation was applied to the GERP scores. This has the effect that 1 represents the highest and zero the lowest possible conservation, which is already the case for SCONE scores. Note that only qualitative comparison is possible since different scores are based on different models as briefly discussed in Section 5.6.

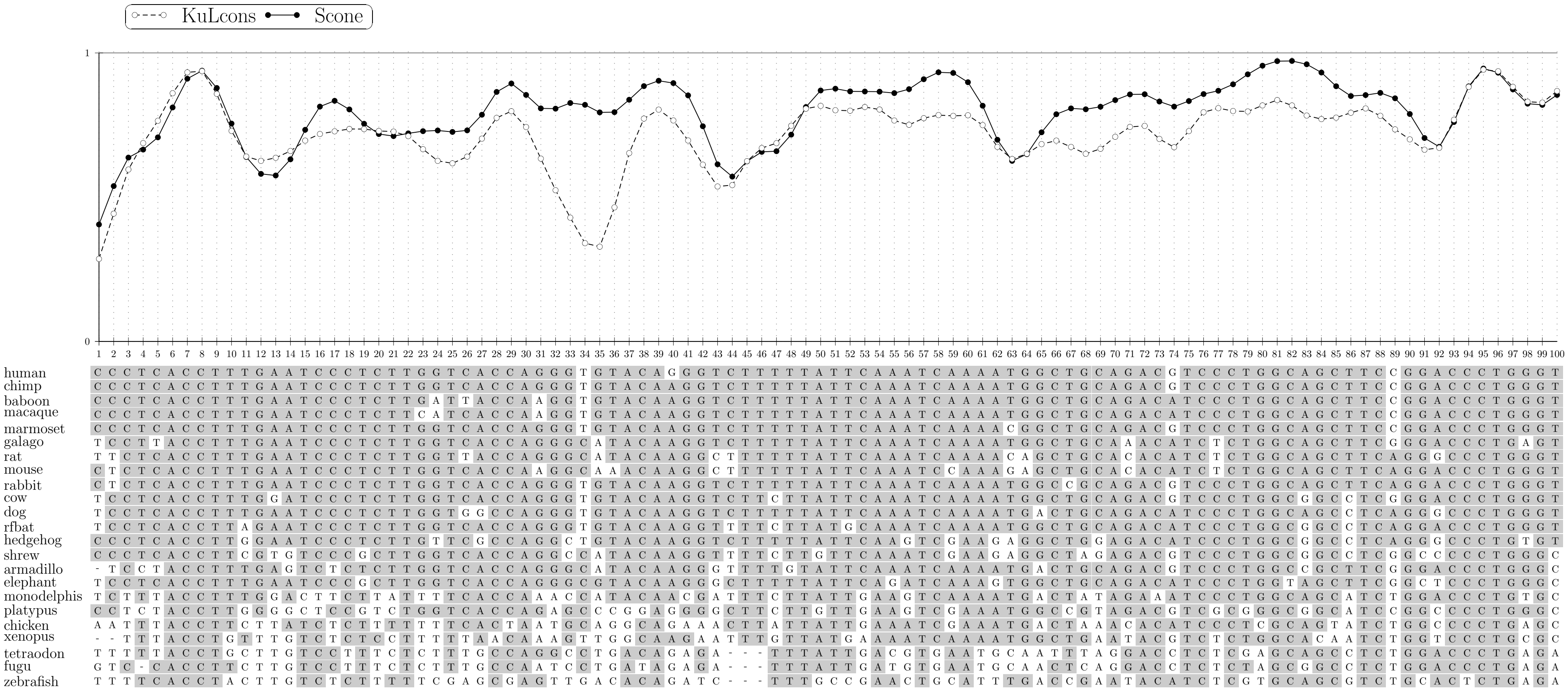
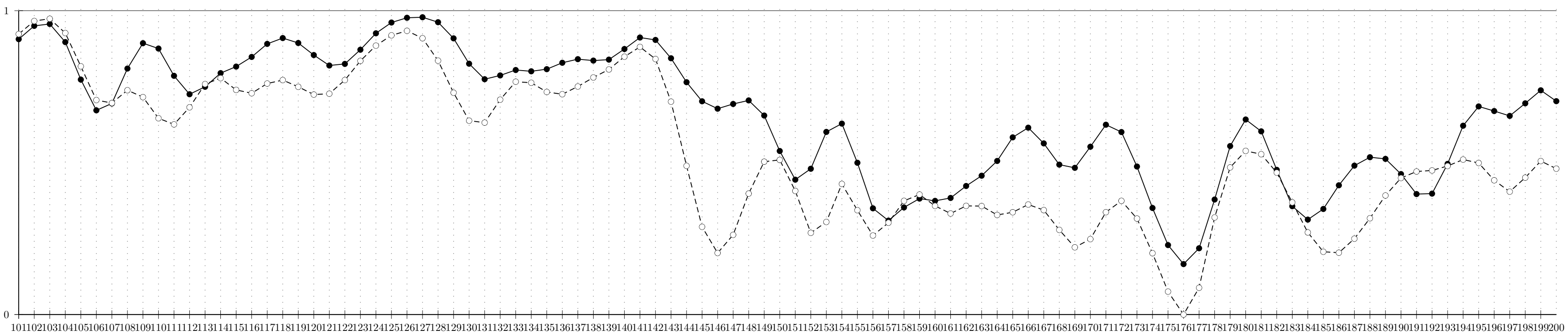


Figure C.1: Comparison of KuLcons score signal to the SCONE scores over an ENCODE region (hg17, ENm005, chr21:32677595-32677794). Scores were smoothed using a Gauss window with $\sigma_w = 0.2$, size 15 ($\delta = 7$).



human	CACCATGGCGGTCATCAGGCTCAGGCAGGCGCGAGCCAACCTGCACAGGGAACCAGAGGAAATCGGACACGTCAACAACAGCAGAAGCACAGAGCCGCCT
chimp	CACCATGGCGGTCATCAGGCTCAGGCAGGCGCGAGCCAACCTGCACAGGGAACCAGAGGAAATCGGACACGTCAAGAACAGCAGAAGCACAGAGCCGCCT
baboon	CACCATGGCGGCCATCAGGCTCAGGCAGGCGCGAGCCAACCTGTACAGGGAGCCAGAGGAAATCAGACACGTCAGCAACAGCAGAAG--CAGAGCCGCCT
macaque	CACCATGGCGGCCATCAGGCTCAGGCAGGCGCGAGCCAACCTGTACAGGGGGCCAAAGGAAATCAGACACGTCAGCAACAGCAGAAG--CAGAGCCGCCT
marmoset	CACCAGGGCAGCCATCAGGCTCAGGCAGGCACGAGCCAACCTGCACAAGAAACCAGAGGAAACGAGACACGTGAGAAACA-CAGAAGCCTAGAGCCGCCT
galago	CACCATGGCAGCCATCAGATTCAGGCAGGCTCGAGCCAACCTGCATAGGGAT---GAGAAAATCAGACA-GTCAAGGACAGAAGAA---CAAAGCC----
rat	CACCACAGCAGTCATCAGGTCCAGGCAGGCTCGAGCCATCCTGCATAGGGAC---AGGAAGATCAGGCC---CAGGGGCAGCACTGTCAGGGAGCC----
mouse	CACCATGGCAGTCATCAGGTCCAGGCAGGCTCGAGCCATCCTGCACAGGGAC---AGGAATGTCAGGAA---TAGGAGCAGCACTAGGAGGGA-CC----
rabbit	CACCATGGCGGCCATCATGTTGAGGCAGGCTCGAGCCATCCTGCACGGGGAC---AAGGACGGCAGGCCTGTCAGGAGCAGCTC-GCCAGGGAGCC----
cow	CACCATGGCAGCCATCAGGTTCAGGCAGGCTCGAGCCATCCTGTTCAGGACA---AGCGGGACCTGG--TGTCGTGA-CGGCAGGACCACAAGGCCCCCT
dog	CACCATGGCGGCCAACAGGTTCAGGCAGGCTCGAGCCATCCTGCACAGGGAC---GAGAAAATCAGGCATGTCAGGAACAGCAGAAGCAC----------
rfbat	CACCATGGCGGCCATCAGGTTCAGGCAGGCTCGAGCCATCCTGCATAGG------GGGGAAATCAGGCATGTCAGGAACAGCAGAAGCACAAAGCCACCT
hedgehog	CACCATGGCGGCCATCAGGTTCAGGCAGGCTCGGGCCATCCTGTGCGGGGAT---GGAGAGATCAGGCATGTTGTGAGGGGCACACATGCAAAGTCTCCT
shrew	CACCATGGCGGTCAGCAGGTTGAGGCAGGCGCGAGCCATCCTGCGGGGG--CC--CAGGGAATGAGG-GCGCCAGCCACTGCAGACGCACCAGGCCACCT
armadillo	CACCAGGGCGGACATCAGGTTCAGGCAGGCTCGAGCTGTCCTGTACAGG----C-AAGGACATCAGTCATGTCAGAGACCG----AGCACGAAGCCACCT
elephant	CACCAGGGCCGCCATCAGGTTCAGGCAGGCTCGAGCCAACCTGCACAGG------AAGGAAATCCGTCAGTTCAGAGACAGTGGAAGCACAAAGCCACCT
monodelphis	CACCATAGCTGACATGAGGTTCAAGCAGGTTTGAGTCATTCTACATGTA------AAAAATATACGGTTAATTCAAAATAGTAGTGGCATGAACCT----
platypus	CACCATCGCCGCCATGAGGTTCAGGCAGGCCCGGGACATCCTGC-----GAGC--GAGGGAATGAAACAAGTCAGCATCAGATAGAGAACAG--TCCCTT
chicken	CACCATTGCTGACAACAGGGTAAGGCACACTCGGCTCATCCTAAAAG--GAA--AGAGGAAAACGTGCAGGTTA-------------------------T
xenopus	CACCATGGCAGACAATAAATTCAAGCAAATCCTGGACATTCTGTCGAAG------AGAAATAATAATAAACAGAAACAACGGC-----------------
tetraodon	CACCAAAGCAGTCAGGAAACCGAGGCACTGACGAACAAACCTGTGCTCAAA---ATGGAACATTAGCCTTATG-ATGACAGC----AAACAACAG-TTGT
fugu	CACCAAAGCAGTCAGGAAACTGAGGCACTGGCGAACAAACCTGTACATAGA---AAAGAATATTAGCTTTGTGCATCACGCC----AATCAAAGT-GTTC
zebrafish	CACCAGAGCGGACAGTAGACTCAGACACTGGCGGACAAACCTGCTC---AA---ATGCAACAACCACACTTTCTTTCTCAAA----ATTCAATGC-TTCC

Figure C.1 continued

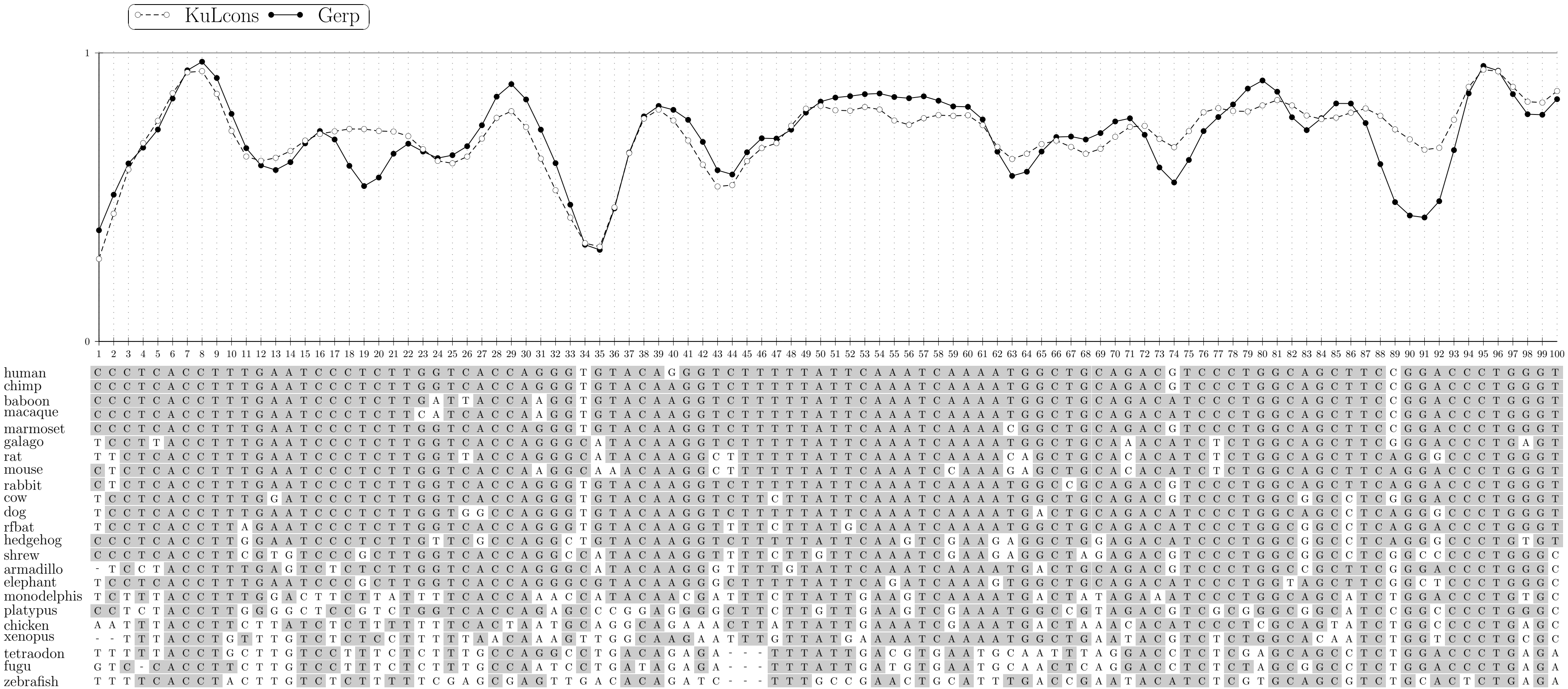


Figure C.2: Comparison of KuLcons score signal to the GERP scores over an ENCODE region (hg17, ENm005, chr21:32677595-32677794). Scores were smoothed using a Gauss window with $\sigma_w = 0.2$, size 15 ($\delta = 7$).

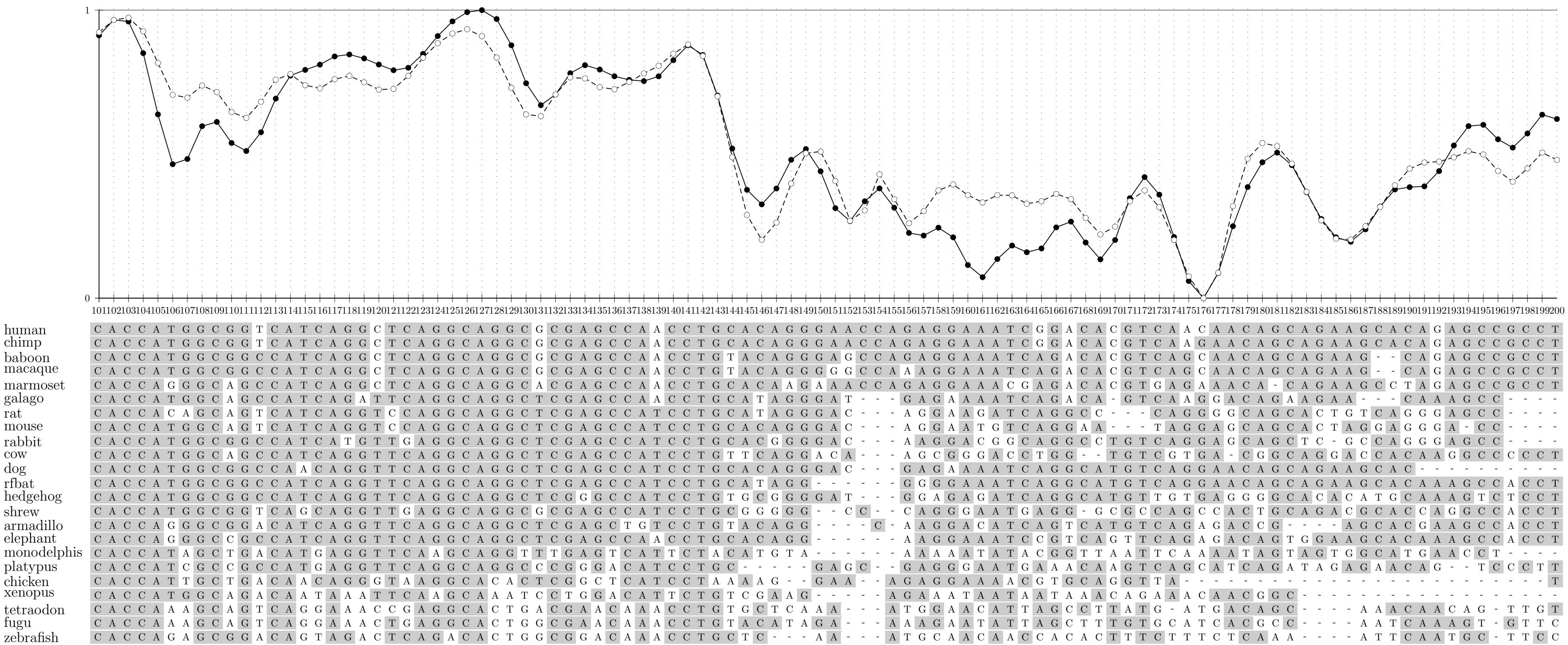


Figure C.2 continued

D

Update rules for branch length gradient

We derive the update rules for the iterative calculation of branch length gradients in a phylogenetic tree, given in Eq. (5.19), Chapter 5.

Given a phylogentic tree $\mathcal{T}$, suppose node u in that tree has children v and w and we are interested in computing the derivative of $l_{\boldsymbol{x}}(t_{uv})$ with respect to branchlength t_{uv} for a given alignment column $\boldsymbol{x}$:

$$\frac{\partial}{\partial t_{uv}} l_{\boldsymbol{x}}(t_{uv}) = \frac{\partial}{\partial t_{uv}} p(\boldsymbol{x}_{\ell(\mathfrak{r})}) \overset{(4.21)}{=} \frac{\partial}{\partial t_{uv}} \sum_{x_{\mathfrak{r}} \in \mathcal{A}} p(\boldsymbol{x}_{\ell(\mathfrak{r})} | x_{\mathfrak{r}}) \pi(x_{\mathfrak{r}}) \tag{D.1a}$$

$$= (\frac{\partial}{\partial t_{uv}} \boldsymbol{p}^{(\mathfrak{r})}) \boldsymbol{\pi}^T. \tag{D.1b}$$

Applying the matrix derivative rules

$$\frac{\partial}{\partial x}(\boldsymbol{A}\boldsymbol{B}) = (\frac{\partial}{\partial x}\boldsymbol{A})\boldsymbol{B} + \boldsymbol{A}(\frac{\partial}{\partial x}\boldsymbol{B}) \tag{D.2a}$$

$$\frac{\partial}{\partial x}(\boldsymbol{A} \odot \boldsymbol{B}) = (\frac{\partial}{\partial x}\boldsymbol{A}) \odot \boldsymbol{B} + \boldsymbol{A} \odot (\frac{\partial}{\partial x}\boldsymbol{B}), \tag{D.2b}$$

we calculate for each node u' with children v' and w':

$$\frac{\partial}{\partial t_{uv}} \boldsymbol{p}^{(u')} = \frac{\partial}{\partial t_{uv}} \left(\boldsymbol{p}^{(v')} \boldsymbol{P}^T_{v' \to u'} \right) \odot \left(\boldsymbol{p}^{(w')} \boldsymbol{P}^T_{w' \to u'} \right) \tag{D.3a}$$

$$= (\frac{\partial}{\partial t_{uv}} \boldsymbol{p}^{(v')} \boldsymbol{P}^T_{v' \to u'} + \boldsymbol{p}^{(v')} \frac{\partial}{\partial t_{uv}} \boldsymbol{P}^T_{v' \to u'}) \odot (\boldsymbol{p}^{(w')} \boldsymbol{P}^T_{w' \to u'}) \tag{D.3b}$$

$$+ (\boldsymbol{p}^{(v')} \boldsymbol{P}^T_{v' \to u'}) \odot \frac{\partial}{\partial t_{uv}} (\boldsymbol{p}^{(w')} \boldsymbol{P}^T_{w' \to u'}). \tag{D.3c}$$

We have to distinguish 3 cases:

(i) v' is ancestor of or equal to u ($\Rightarrow w'$ is not an ancestor nor equal to u):

$$\Rightarrow \frac{\partial}{\partial t_{uv}}(\boldsymbol{p}^{(w')}\boldsymbol{P}^T_{w'\to u'}) = \boldsymbol{0}$$
$$\Rightarrow \frac{\partial}{\partial t_{uv}}\boldsymbol{P}^T_{v'\to u'} = \boldsymbol{0}.$$

(ii) u' is a descendant of u

$$\Rightarrow \frac{\partial}{\partial t_{uv}}\boldsymbol{p}^{(u')} = \boldsymbol{0}$$

(iii) $u' = u$ and $v' = v$:

$$\Rightarrow \frac{\partial}{\partial t_{uv}}(\boldsymbol{p}^{(w')}\boldsymbol{P}^T_{w'\to u'}) = \boldsymbol{0}$$
$$\Rightarrow \frac{\partial}{\partial t_{uv}}\boldsymbol{P}^T_{v'\to u'} = \boldsymbol{R}^T\boldsymbol{P}^T_{v\to u}$$
$$\Rightarrow \frac{\partial}{\partial t_{uv}}\boldsymbol{p}^{(v')} = \boldsymbol{0}.$$

Messages on node u are therefore calculated as follows:

$$\frac{\partial}{\partial t_{uv}}\boldsymbol{p}^{(u')} = \begin{cases} (\frac{\partial}{\partial t_{uv}}\boldsymbol{p}^{(v')}\boldsymbol{P}^T_{v'\to u'}) \odot (\boldsymbol{p}^{(w')}\boldsymbol{P}^T_{w'\to u'}) & \text{if } v' \text{ is ancestor of or equal to } u \\ (\boldsymbol{p}^{(v')}\boldsymbol{R}^T\boldsymbol{P}^T_{v'\to u'}) \odot (\boldsymbol{p}^{(w')}\boldsymbol{P}^T_{w'\to u'}) & \text{if } u' = u, v' = v \\ (\boldsymbol{p}^{(v')}\boldsymbol{P}^T_{v'\to u'}) \odot (\boldsymbol{p}^{(v')}\boldsymbol{R}^T\boldsymbol{P}^T_{w'\to u'}) & \text{if } u' = u, w' = v \\ \boldsymbol{0} & \text{if } u' \text{is descendant of } u. \end{cases}$$

E

Non-binary boxplus

Denote the elements of a finite set $\mathcal{Z}$ of cardinality $|\mathcal{Z}| = q$ as $\{\alpha_0, \alpha_1, .., \alpha_{q-1}\}$, and let $c_m \in \mathcal{Z}$ be constant coefficients.

It is assumed that there is a zero element, which we denote by α_0. It is further assumed that addition and multiplication $(+, \cdot)$ are defined for the elements in $\mathcal{Z}$, and that $\mathcal{Z}$ is closed under addition and multiplication.

The log likelihood ratio (LLR) vector $\mathbf{L}(\mathrm{x}) \in \mathbb{R}^{\mathrm{q}-1}$ is defined as

$$\boldsymbol{L}(x) = \left[\log\left(\frac{P(x=\alpha_1)}{P(x=\alpha_0)}\right), ..., \log\left(\frac{P(x=\alpha_{q-1})}{P(x=\alpha_0)}\right)\right]. \tag{E.1}$$

where the following notation applies:

$$\boldsymbol{L}(x=\alpha_i) = \log\left(\frac{P(x=\alpha_i)}{P(x=\alpha_0)}\right), \; i = 1, .., q-1 \tag{E.2a}$$

$$\boldsymbol{L}(x=\alpha_0) = 0 \tag{E.2b}$$

Consider a set of random variables $\mathsf{X}_1, .., \mathsf{X}_M$ over $\mathcal{Z}$. The boxplus in the non-binary case is derived as follows:

$$\boldsymbol{L}(\sum_{m=1}^{M} c_m x_m = \alpha_i) = \log \left(\frac{\sum\limits_{\substack{\beta_1,..,\beta_M: \\ \sum_m^M c_m\beta_m = \alpha_i}} P(x_1 = \beta_1)...P(x_M = \beta_M)}{\sum\limits_{\substack{\beta_1,..,\beta_M: \\ \sum_m^M c_m\beta_m = \alpha_0}} P(x_1 = \beta_1)...P(x_M = \beta_M)} \right) \tag{E.3a}$$

$$= \log \left(\frac{\sum\limits_{\substack{\beta_1,..,\beta_M: \\ \sum_m^M c_m\beta_m = \alpha_i}} \frac{P(x_1 = \beta_1)...P(x_M = \beta_M)}{P(x_1 = \alpha_0)...P(x_M = \alpha_0)}}{\sum\limits_{\substack{\beta_1,..,\beta_M: \\ \sum_m^M c_m\beta_m = \alpha_0}} \frac{P(x_1 = \beta_1)...P(x_M = \beta_M)}{P(x_1 = \alpha_0)...P(x_M = \alpha_0)}} \right) \tag{E.3b}$$

$$= \log \left(\sum\limits_{\substack{\beta_1,..,\beta_M: \\ \sum_m^M c_m\beta_m = \alpha_i}} e^{\boldsymbol{L}(x_1=\beta_1)+\boldsymbol{L}(x_2=\beta_2)+...+\boldsymbol{L}(x_M=\beta_M)} \right)$$

$$- \log \left(\sum\limits_{\substack{\beta_1,..,\beta_M: \\ \sum_m^M c_m\beta_m = \alpha_0}} e^{\boldsymbol{L}(x_1=\beta_1)+\boldsymbol{L}(x_2=\beta_2)+...+\boldsymbol{L}(x_M=\beta_M)} \right). \tag{E.3c}$$

The derivative of Eq. (E.3c) w.r.t. $\boldsymbol{L}(x_m = \alpha_j)$ is given as

$$\frac{\partial \boldsymbol{L}(\sum_{m=1}^{M} c_m x_m = \alpha_i)}{\partial \boldsymbol{L}(x_m = \alpha_j)} = \frac{1}{c(\alpha_i)} \sum\limits_{\substack{\beta_1,..,\beta_m=\alpha_j,..\beta_M: \\ \sum_m^M c_m\beta_m = \alpha_i}} e^{\boldsymbol{L}(x_1=\beta_1)+..+\boldsymbol{L}(x_m=\alpha_j)+..+\boldsymbol{L}(x_M=\beta_M)}$$

$$- \frac{1}{c(\alpha_0)} \sum\limits_{\substack{\beta_1,..,\beta_m=\alpha_j,..\beta_M: \\ \sum_m^M c_m\beta_m = \alpha_0}} e^{\boldsymbol{L}(x_1=\beta_1)+..+\boldsymbol{L}(x_m=\alpha_j)+..+\boldsymbol{L}(x_M=\beta_M)},$$

where $c(\alpha_i)$ is

$$c(\alpha_i) = \sum\limits_{\substack{\beta_1,..,\beta_M: \\ \sum_m^M c_m\beta_m = \alpha_i}} e^{\boldsymbol{L}(x_1=\beta_1)+\boldsymbol{L}(x_2=\beta_2)+...+\boldsymbol{L}(x_M=\beta_M)}. \tag{E.5}$$

Nomenclature

Abbreviations

A adenine, page 38

BDC binary deletion channel, page 13

BIC binary insertion channel, page 13

BMD bounded minimum distance, page 16

BN Boolean network, page 126

C cytosine, page 38

cDNA complementary DNA, page 126

CME conditional mean estimator, page 81

CTMP continuous time Markov process, page 21

DMC discrete memoryless channel, page 10

DNA deoxyribonucleic acid, page 38

DTMP discrete time Markov process, page 21

E. coli *Escherichia coli*, page 38

ENCODE **ENC**yclopedia **O**f **D**NA **E**lements, page 1

F84 Felsentstein 1984 model of nucleotide substitions, page 56

FG factor graph, page 133

G guanine, page 38

GERP method introduced in [CSA$^+$05] for detecting conserved DNA sequences, page 76

GREV general reversible substitution model, page 55

GRN gene regulatory network, page 125

GS Guruswami-Sudan, page 139

HGP Human Genome Project, page 1

HKY Hasegawa, Kishino, and Yano substitution model, page 55

iff if and only if, page 6

iid independently and identically distributed, page 6

InDel insertion and deletion, page 43

JC Jukes-Cantor, page 56

JS Jensen-Shannon, page 77

K2P Kimura 2 parameter, page 56

KL Kullback-Leibler, page 9

KuLcons our method for detecting conserved DNA sequences, page 76

LLR log likelihood ratio, page 108

LS Laubenbacher-Stigler, page 135

MAP maximum a posteriori probability, page 28

MI mutual information, page 9

ML maximum likelihood, page 15

mRNA messenger RNA, page 40

MSA multiple sequence alignment, page 62

MSE mean squared error, page 81

ODE ordinary differential equation, page 129

PBN probabilistic Boolean network, page 126

pdf probability density function, page 6

PDS polynomial dynamical system, page 134

phylo-HMM phylogenetic hidden Markov model, page 77

pmf probability mass function, page 6

PW Pellikaan-Wu, page 139

r. v. random variable, page 6

RM Reed-Muller, page 138

RNA ribonucleic acid, page 40

RS Reed-Solomon, page 137

s.t. subject to, page 68

SCONE method introduced in [ARSS07] for detecting conserved DNA sequences, page 76

SME stochastic master equation, page 130

SPDS stochastic polynomial dynamical system, page 134

T thymine, page 38

U uracil, page 38

UCSC University of Santa Cruz, page 63

Notation

$(\cdot)^T$ transpose of argument, page 6

$\doteq$ "is defined as", page 21

$\equiv$ equivalence relation, page 69

$\frac{\partial}{\partial x}$ partial derivative with respect to x, page 70

$\mathrm{sign}(\cdot)$ returns sign of argument, page 111

$\boxplus$ boxplus operator, page 111

$\odot$ element-wise product of two matrices or vectors of the same dimension, page 60

$\oplus$ binary addition, page 17

$E\{\cdot\}$ expected value of a random variable, page 15

$\arg\max_x\{f(x)\}$ returns x maximizing $f(x)$, page 16

$\mathbb{F}_q$ Field of order q, page 14

$\mathbb{F}_q[x_1, x_2, ..., x_m]$ ring of multivariate polynomials in m variables over $\mathbb{F}_q$, page 134

$\deg(f)$ degree of polynomial f, page 137

$\mathrm{totdeg}(f)$ total degree of multivariate polynomial f, page 138

$P(\mathsf{X} = x)$,$P(x)$ probability of the event $\mathsf{X} = x$, page 5

$p_{\mathsf{X}}(x)$ probability density function (pdf) of random variable X with realization x, page 6

$p_{\mathsf{X}}(x; \theta)$ a pdf for X that depends on a deterministic parameter θ, page 29

$\mathbb{R}^+$ the non-negative real numbers including zero, page 21

$\mathcal{X}$ calligraphic symbols represent finite sets, page 7

$|\cdot|$ cardinality of a set, page 7

Θ parameter space, page 29

θ a parameter specifying a distribution, page 29

$\hat{\boldsymbol{x}}$ estimate of $\boldsymbol{x}$, page 14

X sans serif letters denote random variables, page 6

$\{\mathsf{X}_n\}_{n\geq 0}$ discrete time random process, page 19

$\{\mathsf{X}_t\}_{t\geq 0}$ continuous time random process, page 21

$\boldsymbol{X}$ boldface capital letters denote matrices, page 11

$\boldsymbol{x}$ boldface lowercase letters denote vectors, page 10

$[\boldsymbol{X}]_{ij}$ element in ith row and jth column of matrix $\boldsymbol{X}$., page 19

trace$\{\cdot\}$ sum of diagonal matrix elements, page 68

$[\boldsymbol{x}]_i$ ith element of the vector $\boldsymbol{x}$, page 20

$\boldsymbol{x}^n$ the sequence $x_0, x_1, ..., x_n$, page 28

$\boldsymbol{x}_m^n$ the sequence $x_m, x_1, ..., x_n$, defined for $m \leq n$, page 28

$d_H(\boldsymbol{x}, \boldsymbol{y})$ Hamming distance between vectors $\boldsymbol{y}$ and $\boldsymbol{x}$, page 14

$w_H(\boldsymbol{x})$ Hamming weight of vector $\boldsymbol{x}$, page 14

$\mathbb{Z}^+$ set of positive integers including 0, page 19

List of Symbols

$\boldsymbol{A}$ an alignment of two or multiple sequences, page 57

$\boldsymbol{a}_l$ lth column of an alignment, page 66

$\boldsymbol{a}_{[b]}$ fully conserved alignment column of nucleotide b, page 83

$\mathcal{A}$ nucleotide alphabet $\{A, C, G, T\}$, page 38

$\{\mathcal{A}, -\}$ extended symbol alphabet for sequence alignments, page 57

$-$ gap symbol in sequence alignments, page 57

$\boldsymbol{c}$ a codeword, page 15

$\boldsymbol{c}_\varepsilon$ noisy sample vector, page 138

$\mathcal{C}$ a collection of vectors forming a code, page 15

c a constant, page 68

$c_t^{(n)}$ coded output symbol of a convolutional encoder, page 108

C channel capacity, page 11

$\mathcal{D}(p_{\mathsf{X}} \,||\, p_{\mathsf{Y}})$ Kullback-Leibler (KL) distance, page 9

d deletion probability, page 13

d_{min} minimum Hamming distance of a code, page 15

$\boldsymbol{e}$ all one column vector $[1,\ 1, ..., 1]^T$, page 23

E set of edges in a graph, page 58

e number of errors, page 138

f_i function associated with gene v_i, page 128

$f_j(x)$ observed nucleotide frequencies for residue x in alignment column j, page 63

$\boldsymbol{g}^{(n)}$ vector of discrete convolutional encoder taps, page 108

$g_m^{(n)}$ tap in a convolutional encoder, page 108

$G_\Theta(\theta; \alpha, \beta)$ gamma distribution over θ with parameters α,β, page 85

$H(\mathsf{X} \mid \mathsf{Y})$ conditional entropy, page 8

$H(\mathsf{X})$ entropy of random variable X, page 7

$h(x)$ information content revealed by the event x, page 7

$\boldsymbol{I}$ identity matrix, page 21

$I(\mathsf{X}; \mathsf{Y})$ mutual information, page 9

$I(\theta)$ Fisher information, page 32

I_{ji} measure of influence from gene v_j on v_i, page 151

$\boldsymbol{L}(x)$ log likelihood ratio vector for non-binary random variable X, page 118

$\boldsymbol{L}$ log likelihood ratio encoder tap vector, page 111

$\boldsymbol{L}^{[k]}$ log likelihood ratio tap vector at iteration k, page 111

$L(x)$ log likelihood ratio of binary random variable X, page 111

$L_m^{(n)}$ log likelihood ratio of encoder tap, page 111

$\mathcal{L}_i$ list of functions or polynomials for network node i, page 132

$l_{\boldsymbol{x}}(\theta)$ likelihood function depending on θ given $\boldsymbol{x}$, page 30

$ll_{\boldsymbol{x}}(\theta)$ logarithm of the likelihood function depending on θ given data $\boldsymbol{x}$, page 30

$\mathcal{M}$ one of the sequence evolution models {GREV,HKY,K2P,F84,JC}, page 67

M memory of a convolutional encoder, page 17

$m(i)$ number of regulators of gene v_i, page 128

M_s number of species in multiple sequence alignment, page 62

$\mathcal{N}(\mu, \sigma^2)$ normal distribution with mean μ and variance σ^2, page 32

N length of vector or series., page 13

N_G number of genes in gene network G, page 128

$\boldsymbol{P}$ transition probability matrix, page 11

$\boldsymbol{P}(t)$ time dependend transition probability matrix of a CTMP, page 22

$\boldsymbol{p}(t)$ state vector of a continuous time Markov process, page 24

$\boldsymbol{p}^{(u)}$ probability vector of observing the leafs given the nucleotide at node u, page 61

$\boldsymbol{p}_n$ state vector of a discrete time Markov process, page 20

$\boldsymbol{P}_{u \to w}$ transition probability matrix between nodes u and v in phylogenetic tree, page 60

p_{ε} probability of ε being non zero, page 108

$p_{s_i s_j}$ transition probability from state s_i to s_j, page 19

$Q(\cdot, \cdot)$ Q-function in the EM algorithm, page 34

$\boldsymbol{R}$ rate matrix of a continuous time Markov process, page 21

$\boldsymbol{R}_s$ symmetric part of a rate matrix, page 55

$\mathcal{RM}_q(u, m)$ Reed-Muller code over $\mathbb{F}_q$ with parameters u, m, page 138

$\mathcal{RS}_q(n, u)$ Reed-Solomon code over $\mathbb{F}_q$ with parameters n, u, page 137

$\mathfrak{r}$ root node in a phylogenetic tree, page 59

R code rate in bits/symbol, page 15

$r_\alpha, r_\beta, r_\gamma, r_\delta, r_\epsilon, r_\zeta$ rate parameters in a substitution rate matrix, page 55

$r_{s_i s_i}$ escape rate from state s_i of a CTMP, page 21

$r_{s_i s_j}$ rate of a CTMP of going from state s_i to s_j, page 21

$\boldsymbol{s}_i$ state of the convolutional encoder at time i, page 17

$\mathfrak{s}_l$	conservation score for lth alignment column, page 76
s	number of erasures, page 139
$s(\cdot,\cdot)$	symmetric scoring function, page 52
$S(n)$	score at position n of two or multiple aligned sequences, page 52
$\mathcal{S}$	state space, page 19
s_i	state from state space $\mathcal{S}$, page 19
$\mathcal{T}$	phylogenetic tree, page 58
t_{uv}	positive real number assigning a distance to branch connecting u and v in a phylogenetic tree, page 59
$\boldsymbol{U}$	the matrix of eigenvectors of a rate matrix, page 22
$\boldsymbol{u}$	information stream, page 108
u_t	symbol fed into the convolutional encoder at time t, page 108
$\mathcal{U}_i$	set of unique transitions for gene v_i, page 143
$\mathcal{V}_i^{\varepsilon}$	set of noisy transitions for gene v_i, page 136
$\mathcal{V}_i$	time series of transitions for gene v_i, page 135
V	set of nodes in a graph, page 58
v_i	gene i in a gene regulatory network, page 128
$v_i(t)$	expression level of gene i at time t, page 128
$\boldsymbol{v}(t)$	state vector of gene network at time t, page 128
$\breve{\boldsymbol{v}}_i$	vector of regulators of gene v_i, page 128
$w[n]$	window function, page 82
$\bar{x}$	soft bit of random variable X, page 111
$\boldsymbol{x}_{\ell(u)}$	realizations at the leaves of the subtree of $\mathcal{T}$ rooted at u, page 60
$\boldsymbol{x}_{\ell^-}$	realizations at the leaves of $\mathcal{T}$, page 60
$\boldsymbol{x}_{\ell^-}$	realizations on the nodes $\mathcal{T}$ excluding the leaves, page 60
$y_t^{(n)}$	noisy output of a convolutional encoder, page 108
$\mathcal{Z}$	finite set with zero element and defined operations $(+,\cdot)$, page 117
δ	specifies size of sliding window, page 82

ε,$\boldsymbol{\varepsilon}$ realization of a noise random variable, page 15

$\mathbf{\Phi}$ diagonal matrix containing the eigenvalues of a rate matrix, page 25

ϕ_i ith eigenvector of a rate matrix, page 22

κ insertion probability, page 13

$\boldsymbol{\lambda}$ Initial distribution of a Markov process, page 19

μ transition probability, page 12

$\mathbf{\Pi}$ diagonal matrix with $\boldsymbol{\pi}$ in the diagonal and zero in off-diagonals, page 55

$\boldsymbol{\pi}$ stationary distribution of a Markov process, page 55

$\boldsymbol{\psi}$ set of parameters specifying a phylogenetic system, page 67

$\boldsymbol{\psi}^0$ evolutionary model of maximum conservation, page 83

ρ_θ autocorrelation of the rate variation process, page 86

σ^2_{X} variance of random variable X, page 84

τ set of distances assigned to the branches of a phylogenetic tree, page 59

ϑ adjustable multiplicity parameter in the Guruswami-Sudan algorithm, page 139

θ_l realization of rate variation at position l, page 73

$\{\Theta_l\}_{l\geq 0}$ rate variation random process, page 73

Bibliography

[AGM04] M. A. Aviñó, E. Green, and O. Moreno, "Applications of finite fields to dynamical systems and reverse engineering problems," in *Proc of the 2004 ACM Symposium on Applied Computing (SAC '04)*. New York, NY, USA: ACM, 2004, pp. 191–196.

[AKL+07] H. M. Aktulga, I. Kontoyiannis, L. A. Lyznik *et al.*, "Identifying statistical dependence in genomic sequences via mutual information estimates," *EURASIP Journal on Bioinformatics and Systems Biology*, vol. 2007, 2007, article ID 14741.

[Alo03] U. Alon, "Biological networks: The tinkerer as an engineer," *Science*, vol. 301, no. 5641, pp. 1866–1867, Sep 2003.

[Alo07] ——, *An Introduction To Systems Biology.* Chapman and Hall/CRC, 2007.

[Alt91] S. Altschul, "Amino acid substitution matrices from an information theoretic perspective," *J Mol Biol*, vol. 219, pp. 555–565, 1991.

[AMK00] T. Akutsu, S. Miyano, and S. Kuhara, "Algorithms for inferring qualitative models of biological networks." in *Pac Symp Biocomput*, 2000, pp. 293–304.

[ARSS07] S. Asthana, M. Roytberg, J. Stamatoyannopoulos *et al.*, "Analysis of sequence conservation at nucleotide resolution." *PLoS Comput Biol*, vol. 3, no. 12, p. e254, Dec 2007.

[Bar03] W. Bartens, "Clowns im Labor," Die Zeit, Nr. 9, Feb 2003, url: http://www.zeit.de/2003/09/DNA-Geschichte (March 2009).

[Bat06] G. Battail, *Information theory and error correcting codes in genetics and biological evolution.* Springer, Nov 2006, ch. 13.

[Bat08] ——, *An Outline of Informational Genetics*, ser. Synthesis Lectures on Biomedical Engineering. Morgan and Claypool Publishers, 2008.

[BDC+03] M. Brudno, C. Do, G. Cooper *et al.*, "LAGAN and Multi-LAGAN: efficient tools for large-scale multiple alignment of genomic DNA," *Genome Res*, vol. 13, no. 4, pp. 721–731, 2003.

[BE05] S. Bulashevska and R. Eils, "Inferring genetic regulatory logic from expression data," *Bioinformatics*, vol. 21, no. 11, pp. 2706–2713, 2005.

[Ber01] J. Berkmann, *Iterative decoding of binary block and convolutional codes*, ser. Fortschritt-Berichte VDI. VDI Verlag GmbH, Düsseldorf, 2001.

[BKML+08] D. A. Benson, I. Karsch-Mizrachi, D. J. Lipman *et al.*, "GenBank." *Nucleic Acids Res*, vol. 36, no. Database issue, pp. D25–30, Jan 2008.

[BKR+04] M. Blanchette, W. J. Kent, C. Riemer *et al.*, "Aligning multiple genomic sequences with the threaded blockset aligner." *Genome Res*, vol. 14, no. 4, pp. 708–15, Apr 2004.

[Bla01] M. Blanchette, "Algorithms for phylogenetic footprinting," in *Proc of the Fifth Annual International Conference on Computational Biology (RECOMB01).* Montréal, Québec, Canada: ACM Press, 2001, pp. 49–58.

[Bla03] ——, "A comparative analysis method for detecting binding sites in coding regions," in *Proc of the Seventh Annual International Conference on Research in Computational Molecular Biology (RECOMB03).* Berlin, Germany: ACM Press, 2003, pp. 57–66.

[Bla08] R. E. Blahut, *Algebraic codes on lines, planes and curves.* Cambridge University Press, 2008.

[BMG+04] E. Buratti, A. F. Muro, M. Giombi *et al.*, "RNA folding affects the recruitment of SR proteins by mouse and human polypurinic enhancer elements in the fibronectin EDA exon." *Mol Cell Biol*, vol. 24, no. 3, pp. 1387–400, Feb 2004.

[BP04] N. Bray and L. Pachter, "MAVID: Constrained ancestral alignment of multiple sequences," *Genome Res*, vol. 14, no. 4, pp. 693–699, 2004.

[BPM+04] G. Bejerano, M. Pheasant, I. Makunin *et al.*, "Ultraconserved elements in the human genome." *Science*, vol. 304, no. 5675, pp. 1321–5, May 2004.

[BRS+06] R. Bonneau, D. J. Reiss, P. Shannon *et al.*, "The inferelator: an algorithm for learning parsimonious regulatory networks from systems-biology data sets de novo." *Genome Biol*, vol. 7, no. 5, p. R36, 2006.

[BS03] A. Brazma and T. Schlitt, "Reverse engineering of gene regulatory networks: a finite state linear model," *Genome Biology*, vol. 4, no. 6, p. P5, 2003, this was the first version of this article to be made available publicly. [Online]. Available: http://genomebiology.com/2003/4/6/P5

[BST00] M. Blanchette, B. Schwikowski, and M. Tompa, "An exact algorithm to identify motifs in orthologous sequences from multiple species." *Proc Int Conf Intell Syst Mol Biol*, vol. 8, pp. 37–45, 2000.

[BVK07] T. Bollenbach, K. Vetsigian, and R. Kishony, "Evolution and multilevel optimization of the genetic code," *Genome Res*, vol. 17, no. 4, pp. 401–404, 2007.

[CBM98] O. Cappé, V. Buchoux, and E. Moulines, "Quasi-Newton method for maximum likelihood estimation of hidden Markov models," in *Proc of the 1998 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, vol. 4, 1998.

[CBS+04] G. M. Cooper, M. Brudno, E. A. Stone *et al.*, "Characterization of evolutionary rates and constraints in three mammalian genomes." *Genome Res*, vol. 14, no. 4, pp. 539–48, Apr 2004.

[CFR+01] S. Chung, G. Forney Jr, T. Richardson *et al.*, "On the design of low-density parity-check codes within 0.0045 dB of the Shannon limit," *IEEE Communications Letters*, vol. 5, no. 2, pp. 58–60, 2001.

[Che07] E. Check, "Genome project turns up evolutionary surprises," *Nature*, vol. 447, no. 14, pp. 760–761, Jun 2007.

[CHIW98] D. Costello, J. Hagenauer, H. Imai *et al.*, "Applications of error-control coding," *IEEE Trans on Information Theory*, vol. 44, no. 6, Oct 1998.

[CLO92] D. Cox, J. Little, and D. O'Shea, *Ideals, Varieties, and Algorithms.* New York: Springer, 1992.

[Clu06] M. Cluzeau, "Block code reconstruction using iterative decoding techniques," in *Proc of the IEEE International Symposium on Information Theory (ISIT06), Seattle*, Jul 2006, pp. 2269–2273.

[CS07] J. A. Capra and M. Singh, "Predicting functionally important residues from sequence conservation." *Bioinformatics*, vol. 23, no. 15, pp. 1875–82, Aug 2007.

[CSA+05] G. M. Cooper, E. A. Stone, G. Asimenos *et al.*, "Distribution and intensity of constraint in mammalian genomic sequence." *Genome Res*, vol. 15, no. 7, pp. 901–13, Jul 2005.

[CSK+03] R. Chenna, H. Sugawara, T. Koike *et al.*, "Multiple sequence alignment with the Clustal series of programs." *Nucleic Acids Res*, vol. 31, no. 13, p. 3497, 2003.

[CT91] T. M. Cover and J. A. Thomas, *Elements of Information Theory*, ser. Wiley Series In Telecommunications. Wiley-Interscience, 1991.

[CT06] B. Chor and T. Tuller, "Finding a maximum likelihood tree is hard," *J ACM*, vol. 53, no. 5, pp. 722–744, 2006.

[CW53] J. Crick and F. Watson, "A structure for deoxyribose nucleic acid," *Nature*, vol. 171, pp. 737–738, 1953.

[CW07] X. Cai and X. Wang, "Stochastic modeling and simulation of gene networks - a review of the state-of-the-art research on stochastic simulations," *IEEE Signal Processing Magazine*, vol. 24, no. 1, pp. 27–36, January 2007.

[DBN+06] J. A. Drake1, C. Bird, J. Nemesh *et al.*, "Conserved noncoding sequences are selectively constrained and not mutation cold spots," *Nat Genetics*, vol. 38, pp. 223–227, Feb 2006.

[de 02] H. de Jong, "Modeling and simulation of genetic regulatory systems: a literature review." *J Comput Biol*, vol. 9, no. 1, pp. 67–103, 2002.

[DE09] E. Delgado-Eckert, "Reverse engineering time discrete finite dynamical systems: A feasible undertaking?" *PLoS ONE*, vol. 4, no. 3, p. e4939, Mar 2009.

[DEKM98] R. Durbin, S. Eddy, A. Krogh *et al.*, *Biological Sequence Analysis - Probabilistic Models of Proteins and Nucleic Acids.* Cambridge University Press, 1998.

[dG05] H. de Jong and J. Geiselmann, *Modeling and simulation of genetic regulatory networks by ordinary differential equations*, ser. Signal Processing and Communications. Hindawi Publishing Corp. New York, NY, United States, 2005, ch. 6.

[DGH+06] Z. Dawy, B. Goebel, J. Hagenauer *et al.*, "Gene mapping and clustering using Shannon's mutual information," *IEEE/ACM Trans on Comput Bio and Bioinf*, vol. 3, no. 1, January - March 2006.

[DHHM05] Z. Dawy, J. Hagenauer, P. Hanus *et al.*, "Mutual information based distance measures for classification and content recognition with applications to genetics," in *Proc of the IEEE International Conference on Communications (ICC05)*, 2005, pp. 820–824.

[DJLS07] E. S. Dimitrova, A. S. Jarrah, R. Laubenbacher *et al.*, "A Gröbner fan method for biochemical network modeling," in *Proc of the 2007 International Symposium on Symbolic and Algebraic Computation (ISSAC '07)*, New York, NY, USA, 2007, pp. 122–126.

[DLR77] A. Dempster, N. Laird, and D. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *Journal of the Royal Statistical Society, Series B*, vol. 39, no. 1, pp. 1–38, 1977.

[DLS00] P. D'haeseleer, S. Liang, and R. Somogyi, "Genetic network inference: from co-expression clustering to reverse engineering." *Bioinformatics*, vol. 16, no. 8, pp. 707–26, Aug 2000.

[DM07] E. Drinea and M. Mitzenmacher, "Improved lower bounds for the capacity of i.i.d. deletion and duplication channels," *IEEE Trans on Information Theory*, vol. 53, no. 8, pp. 2693–2714, Aug. 2007.

[DMB06] A. Diallo, V. Makarenkov, and M. Blanchette, "Finding maximum likelihood indel scenarios," in *Proc of the fourth Recomb satellite conference on Comparative Genomics (RECOMB06)*, 2006, pp. 171–185.

[DML00] M. C. Davey, D. J. C. Mackay, and C. Laboratory, "Watermark codes: Reliable communication over insertion /deletion channels," in *Proc of the IEEE International Symposium on Information Theory (ISIT00)*, 2000, pp. 47–7.

[Dón03] D. Dónaill, "Why nature chose A, C, G, and U/T: An error-coding perspective of nucleotide alphabet composition," *Origins of Life and Evolution of the Biosphere*, vol. 33, pp. 433–455, 2003.

[Dón06] ——, "Digital parity and the composition of the nucleotide alphabet," *IEEE Engineering in Medicine and Biology*, vol. 25, no. 1, pp. 54–61, 2006.

[DRA05] E. Dermitzakis, A. Reymond, and S. Antonarakis, "Conserved non-genic sequences - an unexpected feature of mammalian genomes," *Nat Rev Genet*, vol. 6, pp. 151–157, 2005.

[DSCW05] E. R. Dougherty, I. Shmulevich, J. Chen *et al.*, Eds., *Genomic Signal Processing and Statistics*. Hindawi Publishing Corp. New York, NY, United States, 2005.

[DSJM05] Z. Dawy, M. Sarkis, J.Hagenauer *et al.*, "A novel gene mapping algorithm based on ica," in *Proc of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP05), Philadelphia*, Mar 2005, pp. v/381–v/384.

[DTA08] J. Dougherty, I. Tabus, and J. Astola, "Inference of gene regulatory networks based on a universal minimum description length," *EURASIP Journal on Bioinformatics and Systems Biology*, vol. 2008, 2008, article ID 482090, 11 pages.

[EG05] W. J. Ewens and G. R. Grant, *Statistical methods in Bioinformatics*, 2nd ed., ser. Statistics for Biology and Health. Springer, 2005.

[Eli55] P. Elias, "Coding for noisy channels," *IRE International Convention Record*, vol. 4, pp. 37–46, 1955.

[EM02] Y. Ephraim and N. Merhav, "Hidden Markov processes," *IEEE Trans on Information Theory*, vol. 48, no. 6, pp. 1518–1569, Jun 2002.

[FC96] J. Felsenstein and G. A. Churchill, "A hidden Markov model approach to variation among sites in rate of evolution." *Mol Biol Evol*, vol. 13, no. 1, pp. 93–104, Jan 1996.

[FDFe08] J. J. Faith, M. E. Driscoll, V. A. Fusaro *et al.*, "Many Microbe Microarrays database: uniformly normalized affymetrix compendia with structured experimental metadata." *Nucleic Acids Res*, vol. 36(Database issue), pp. D866–70, January 2008.

[Fel81] J. Felsenstein, "Evolutionary trees from DNA sequences: a maximum likelihood approach." *J Mol Evol*, vol. 17, no. 6, pp. 368–76, 1981.

[FH98] S. J. Freeland and L. D. Hurst, "The genetic code is one in a million." *J Mol Evol*, vol. 47, no. 3, pp. 238–48, Sep 1998.

[Fil97] E. Filiol, "Reconstruction of convolutional encoders over $GF(q)$," *Lecture Notes In Computer Science*, vol. 1355, pp. 101–109, 1997.

[Fil00] ——, "Reconstruction of punctured convolutional encoders," in *Proc of the IEEE International Symposium on Information Theory and Applications, Hawai*, Nov 2000.

[FLNP00] N. Friedman, M. Linial, I. Nachman *et al.*, "Using bayesian networks to analyze expression data." *J Comput Biol*, vol. 7, no. 3-4, pp. 601–20, 2000.

[For81] D. Forsdyke, "Are introns in-series error-detecting sequences?" *J Theor Biol*, vol. 93, pp. 861–866, 1981.

[FZSL07] X. Fan, J. Zhu, E. Schadt *et al.*, "Statistical power of phylo-HMM for evolutionarily conserved element detection," *BMC Bioinformatics*, vol. 8, p. 374, Oct 2007.

[GBR+07] M. Gerstein, C. Bruce, J. Rozowsky *et al.*, "What is a gene, post-ENCODE? History and updated definition," *Genome Res*, vol. 17, no. 6, p. 669, 2007.

[GCJJPG+08] S. Gama-Castro, V. Jiménez-Jacinto, M. Peralta-Gil *et al.*, "RegulonDB (version 6.0): gene regulation model of *Escherichia coli* k-12 beyond transcription, active (experimental) annotated promoters and textpresso navigation." *Nucleic Acids Res*, vol. 36, no. Database issue, pp. D120–4, Jan 2008.

[GDHM05] B. Goebel, Z. Dawy, J. Hagenauer *et al.*, "An approximation to the distribution of finite sample size mutual information estimates," in *Proc of the IEEE International Conference on Communications (ICC05), Seoul*, vol. 2, May 2005, pp. 1102–1106.

[GdLC03] T. S. Gardner, D. di Bernardo, D. Lorenz *et al.*, "Inferring genetic networks and identifying compound mode of action via expression profiling." *Science*, vol. 301, no. 5629, pp. 102–5, Jul 2003.

[GKS07] P. Gopalan, S. Khot, and R. Saket, "Hardness of reconstructing multivariate polynomials over finite fields," in *Proc of the 48th Annual IEEE Symposium on Foundations of Computer Science*, 2007, pp. 349–359.

[GKZ08] P. Gopalan, A. R. Klivans, and D. Zuckerman, "List-decoding Reed-Muller codes over small fields," in *Proc of the 40th annual ACM symposium on Theory of computing (STOC08)*, 2008, pp. 265–274.

[GLB+08] A. R. Gruber, R. Lorenz, S. H. Bernhart *et al.*, "The vienna RNA websuite." *Nucleic Acids Res*, vol. 36, no. Web Server issue, pp. W70–4, Jul 2008.

[GR06] P. Gaborit and O. Ruatta, "Efficient erasure list-decoding of Reed-Muller codes," in *Proc of the IEEE International Symposium on Information Theory (ISIT06)*, July 2006, pp. 148–152.

[Gru04] P. Grunwald, "A tutorial introduction to the minimum description length principle," 2004. [Online]. Available: http://www.citebase.org/abstract?id=oai:arXiv.org:math/0406077 (March09)

[GS99] V. Guruswami and M. Sudan, "Improved decoding of Reed-Solomon codes and algebraic geometry codes," *IEEE Trans on Information Theory*, vol. 45, no. 6, pp. 1757–1767, Sep 1999.

[GVTRS06] I. Gat-Viks, A. Tanay, D. Raijman *et al.*, "A probabilistic methodology for integrating knowledge and experiments on biological networks." *J Comput Biol*, vol. 13, no. 2, pp. 165–81, Mar 2006.

[HDE08] J. Hu, T. Duman, and M. Erden, "On the information rates of channels with insertion/deletion/substitution errors," in *Proc of the IEEE International Conference on Communications (ICC08)*, May 2008, pp. 956–960.

[Heg08] M. Hegele, "Detection of functional coherences of conserved elements using correlation and mutual information," Master's thesis, Technische Universität München, Mar 2008.

[HOP96] J. Hagenauer, E. Offer, and L. Papke, "Iterative decoding of binary block and convolutional codes," *IEEE Trans on Information Theory*, vol. 42, no. 2, pp. 429–445, Mar 1996.

[HRY+03] R. C. Hardison, K. M. Roskin, S. Yang *et al.*, "Covariation in frequencies of substitution, deletion, transposition, and recombination during eutherian evolution." *Genome Res*, vol. 13, no. 1, pp. 13–26, Jan 2003.

[IA07] S. Itzkovitz and U. Alon, "The genetic code is nearly optimal for allowing additional information within protein-coding sequences." *Genome Res*, vol. 17, no. 4, pp. 405–12, Apr 2007.

[ITSH00] T. Ideker, V. Thorsson, A. F. Siegel *et al.*, "Testing for differentially-expressed genes by maximum-likelihood analysis of microarray data." *J Comput Biol*, vol. 7, no. 6, pp. 805–17, 2000.

[JLSS07] A. S. Jarrah, R. Laubenbacher, B. Stigler *et al.*, “Reverse engineering of polynomial dynamical systems,” *Advances in Applied Mathematics*, vol. 39, pp. 477–489, Oct 2007.

[Jus06] W. Just, “Reverse engineering discrete dynamical systems from data sets with random input vectors.” *J Comput Biol*, vol. 13, no. 8, pp. 1435–56, Oct 2006.

[JZ99] R. Johannesson and K. S. Zigangirov, *Fundamentals of Convolutional Coding*, ser. Series on Digital & Mobile Communication, J. B. Anderson, Ed. IEEE Press, 1999.

[KA90] S. Karlin and S. Altschul, “Methods for assessing the statistical significance of molecular sequence features by using general scoring schemes,” *PNAS*, vol. 87, no. 6, pp. 2264–2268, 1990.

[Kau69] S. Kauffman, “Homeostasis and differentiation in random genetic control networks.” *Nature*, vol. 224, no. 5215, pp. 177–8, Oct 1969.

[Kay93] S. M. Kay, *Fundamentals of Statistical Signal Processing - Estimation Theory*, ser. Pretice Hall Signal Processing Series. Prentice Hall PTR, 1993.

[KBD+03] D. Karolchik, R. Baertsch, M. Diekhans *et al.*, “The UCSC genome browser database.” *Nucleic Acids Res*, vol. 31, no. 1, pp. 51–4, Jan 2003.

[Kel79] F. Kelly, “Reversibility and stochastic networks,” *New York*, 1979.

[Ker99] A. Kerlavage, “Computational genomics: Biological discovery in complete genomes.” in *Proc of the Seventh International Conference on Intelligent Systems for Molecular Biology.* Heidelberg, Germany: AAAI Press, 1999.

[KFL01] F. Kschischang, B. J. Frey, and H.-A. Loeliger, “Factor graphs and the sum-product algorithm,” *IEEE Trans on Information Theory*, vol. 47, no. 2, pp. 498–519, Feb 2001.

[KH08] V. Kuryshev and P. Hanus, “Compression based classification of primate endogenous retrovirus sequences,” in *Proc of the 5th International Workshop on Computational Systems Biology*, 2008, pp. 81–84.

[KLP68] T. Kasami, S. Lin, and W. W. Peterson, “New generalizations of the Reed-Muller codes part I: Primitive codes,” *IEEE Trans on Information Theory*, vol. 14, no. 2, pp. 189–199, Mar 1968.

[Kni06] R. Knippers, *Molekulare Genetik*, 9th ed. Georg Thieme Verlag, 2006.

[KS07] J. Kim and S. Sinha, “Indelign: a probabilistic framework for annotation of insertions and deletions in a multiple alignment.” *Bioinformatics*, vol. 23, no. 3, pp. 289–97, Feb 2007.

[KV94] G. K. Kaleh and R. Vallet, "Joint parameter estimation and symbol detection for linear or nonlinear unknown channels," *IEEE Trans on Communications*, vol. 42, no. 7, pp. 2406–2413, Jul 1994.

[KV03] R. Koetter and A. Vardy, "Algebraic soft-decision decoding of Reed-Solomon codes," *IEEE Trans on Information Theory*, vol. 49, no. 11, pp. 2809–2825, Nov. 2003.

[KXL06] M. Kamal, X. Xie, and E. S. Lander, "A large family of ancient repeat elements in the human genome is under strong selection." *PNAS*, vol. 103, no. 8, pp. 2740–5, Feb 2006.

[LC83] S. Lin and D. J. Costello, *Error Control Coding: Fundamentals and Applications.* Prentice-Hall Series in Computer Applications in Electrical Engineering, 1983.

[Lew04] B. Lewin, *Genes VIII*, international edition ed. Pearson Prentice Hall, 2004.

[LFS98] S. Liang, S. Fuhrman, and R. Somogyi, "Reveal, a general reverse engineering algorithm for inference of genetic network architectures." in *Proc of the Pac Symp Biocomput*, 1998, pp. 18–29.

[Lin91] J. Lin, "Divergence measures based on the Shannon entropy," *IEEE Trans on Information Theory*, vol. 37, no. 1, pp. 145–151, 1991.

[LLB$^+$01] E. Lander, L. Linton, B. Birren *et al.*, "Initial sequencing and analysis of the human genome," *Nature*, vol. 409, pp. 860–921, 2001.

[LM07] Z. Liu and M. Mitzenmacher, "Codes for deletion and insertion channels with segmented errors," in *Proc of the IEEE International Symposium on Information Theory (ISIT 2007)*, Jun 2007, pp. 846–850.

[LS04] R. Laubenbacher and B. Stigler, "A computational algebra approach to the reverse engineering of gene regulatory networks." *J Theor Biol*, vol. 229, no. 4, pp. 523–37, Aug 2004.

[LSM$^+$03] B. Lenhard, A. Sandelin, L. Mendoza *et al.*, "Identification of conserved regulatory elements by comparative genome analysis," *Journal of Biology*, vol. 2, no. 2, p. 13, 2003. [Online]. Available: http://jbiol.com/content/2/2/13

[LTTL96] L. S. Liebovitch, Y. Tao, A. T. Todorov *et al.*, "Is there an error correcting code in the base sequence in DNA," *Biophysical Journal*, vol. 71, no. 3, pp. 1539–1544, Sep 1996.

[LVYP05] S. J. Lolle, J. L. Victor, J. M. Young *et al.*, "Genome-wide non-mendelian inheritance of extra-genomic information in Arabidopsis," *Nature*, vol. 434, pp. 505–509, Mar 2005.

[Mat07] J. Mattick, "A new paradigm for developmental biology," *Journal of Experimental Biology*, vol. 210, no. 9, p. 1526, 2007.

[MBHG03] E. Margulies, M. Blanchette, D. Haussler *et al.*, "Identification and characterization of multi-species conserved sequences," *Genome Res*, vol. 13, pp. 2507–2518, 2003.

[MCA+07] E. H. Margulies, G. M. Cooper, G. Asimenos *et al.*, "Analyses of deep mammalian sequence alignments and constraint predictions for 1% of the human genome." *Genome Res*, vol. 17, no. 6, pp. 760–74, Jun 2007.

[McE03] R. J. McEliece, "The Guruswami-Sudan decoding algorithm for Reed-Solomon codes," *IPN Progress Report*, pp. 1–60, May 2003.

[Mey00] C. Meyer, *Matrix Analysis and Applied Linear Algebra.* Society for Industrial Mathematics, 2000.

[MFP05] I. Mayrose, N. Friedman, and T. Pupko, "A gamma mixture model better accounts for among site rate heterogeneity." *Bioinformatics*, vol. 21 Suppl 2, pp. ii151–8, Sep 2005.

[Mil06] O. Milenkovic, "Enumerating RNA motifs: A coding-theoretic approach," in *Proc of the Allerton Conference on Communications, Control and Computing*, Sep 2006.

[MM06] J. Mattick and I. Makunin, "Non-coding RNA," *Human Molecular Genetics*, vol. 15, no. 90001, pp. 17–29, 2006.

[Moo96] T. K. Moon, "The expectation-maximization algorithm," *IEEE Signal Processing Magazine*, pp. 47–60, Nov 1996.

[Moo02] ——, "Maximum-likelihood binary shift-register synthesis from noisy observations," *IEEE Trans on Information Theory*, vol. 48, no. 7, pp. 2096–2104, Jul 2002.

[Mor69] P. A. P. Moran, "Statistical inference with bivariate gamma distributions," *Biometrika*, vol. 56, no. 3, pp. 627–634, 1969.

[MPJ04] J. D. McAuliffe, L. Pachter, and M. I. Jordan, "Multiple-sequence functional annotation and the generalized hidden Markov phylogeny." *Bioinformatics*, vol. 20, no. 12, pp. 1850–60, Aug 2004.

[MS77] F. MacWilliams and N. Sloane, *The Theory of Error Correcting Codes.* New-York: North-Holland, 1977.

[MSY03] P. Mendes, W. Sha, and K. Ye, "Artificial gene networks for objective comparison of analysis algorithms." *Bioinformatics*, vol. 19 Suppl 2, pp. ii122–9, Oct 2003.

[MV04] O. Milenkovic and B. Vasic, "Information theory and coding problems in genetics," in *Proc of the International Workshop on Information Theory (ITW04)*, Oct 2004, pp. 60–65.

[MVH+07] D. M. McGaughey, R. M. Vinton, J. Huynh *et al.*, "Metrics of sequence constraint overlook regulatory sequences in an exhaustive analysis at *phox2b*." *Genome Res*, vol. 18, pp. 252–260, Dec 2007.

[MXD07] S. Marshall, Y. Xiao, and E. Dougherty, "Inference of a probabilistic Boolean network from a single observed temporal sequence," *EURASIP Journal on Bioinformatics and Systems Biology*, vol. 2007, no. 1, 2007.

[Nie97] R. Nielsen, "Site-by-site estimation of the rate of substitution and the correlation of rates in mitochondrial DNA." *Syst Biol*, vol. 46, no. 2, pp. 346–53, Jun 1997.

[Nie05] R. Nielsen, Ed., *Statistical Methods in Molecular Evolution*, ser. Statistics for Biology and Health. Springer, 2005.

[Nor97] J. Norris, *Markov chains*, ser. Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge University Press, 1997.

[NZPF+04] M. A. Nóbrega, Y. Zhu, I. Plajzer-Frick *et al.*, "Megabase deletions of gene deserts result in viable mice." *Nature*, vol. 431, no. 7011, pp. 988–93, Oct 2004.

[Pea82] J. Pearl, "Reverend bayes on inference engines: A distributed hierarchical approach," in *Proc of the AAAI National Conference on AI*, 1982, pp. 133–136.

[Pen07] E. Pennisi, "DNA study forces rethink of what it means to be a gene," *Science*, vol. 316, no. 5831, pp. 1556–1557, Jun 2007.

[Pev00] P. A. Pevzner, *Computational Biology: An Algorithmic Approach.* The MIT Press, 2000.

[Pha87] R. M. Phatarfod, "A linearly regressive gamma Markov process," *Stochastic Hydrology and Hydraulics*, vol. 1, pp. 155–160, 1987.

[PM07] M. Pheasant and J. S. Mattick, "Raising the estimate of functional human sequences." *Genome Res*, vol. 17, no. 9, pp. 1245–53, Sep 2007.

[PREF01] D. Pe'er, A. Regev, G. Elidan *et al.*, "Inferring subnetworks from perturbed expression profiles." *Bioinformatics*, vol. 17 Suppl 1, pp. S215–24, 2001.

[PW04] R. Pellikaan and X.-W. Wu, "List decoding of q-ary Reed-Muller codes," *IEEE Trans on Information Theory*, vol. 50, no. 4, pp. 679–682, Apr 2004.

[RGH+07] J. Rissanen, P. Grünwald, J. Heikkonen *et al.*, "Information theoretic methods for bioinformatics," *EURASIP Journal on Bioinformatics and Systems Biology*, vol. 2007, p. 2, 2007.

[Ric95] B. Rice, "Determining the parameters of a rate $\frac{1}{n}$ convolutional code over $GF(q)$," in *Proc of the Third International Conference on Finite Fields and Applications, Glasgow*, 1995.

[RIF+06] R. Redon, S. Ishikawa, K. R. Fitch *et al.*, "Global variation in copy number in the human genome," *Nature*, vol. 444, pp. 444–454, Nov 2006.

[Riv05] E. Rivas, "Evolutionary models for insertions and deletions in a probabilistic modeling framework." *BMC Bioinformatics*, vol. 6, p. 63, 2005.

[Roc06] S. Roch, "A short proof that phylogenetic tree reconstruction by maximum likelihood is hard." *IEEE/ACM Trans Comput Biol Bioinf*, vol. 3, no. 1, pp. 92–4, 2006.

[Ros06] G. L. Rosen, "Examining coding structure and redundancy in DNA," *IEEE Engineering in Medicine and Biology*, vol. 25, no. 1, pp. 62–68, Jan 2006.

[RU07] T. Richardson and R. Urbanke, *Modern Coding Theory*. Cambridge University Press, Mar 2007. [Online]. Available: http://lthcwww.epfl.ch/mct/index.php

[San07] N. Santhi, "On algebraic decoding of q-ary Reed-Muller and product Reed-Solomon codes," in *Proc of the IEEE International Symposium on Information Theory (ISIT07), Nice*, Jun 2007.

[SB07] T. Schlitt and A. Brazma, "Current approaches to gene regulatory network modelling." *BMC Bioinformatics*, vol. 8, no. Suppl 6, p. S9, 2007.

[SBP05a] A. Siepel, G. Bejerano, and J. S. Pedersen, "Evolutionarily conserved elements in vertebrate, insect, worm, and yeast genomes," *Genome Res*, vol. 15, no. 8, pp. 1034–1050, Aug 2005.

[SBP+05b] A. Siepel, G. Bejerano, J. S. Pedersen *et al.*, "Supplementary material for: evolutionarily conserved elements in vertebrate, insect, worm, and yeast genomes," *Genome Res*, vol. 15, no. 8, pp. 1034–1050, 2005. [Online]. Available: http://www.genome.org/cgi/content/abstract/15/8/1034

[SBT04] S. Sinha, M. Blanchette, and M. Tompa, "Phyme: A probabilistic algorithm for finding motifs in sets of orthologous sequences," *BMC Bioinformatics*, vol. 5, no. 1, Oct 2004. [Online]. Available: http://www.biomedcentral.com/1471-2105/5/170

[Sch97] S. Schäffler, *Decodierung binärer linearer Blockcodes durch globale Optimierung*, ser. Theorie und Forschung in den Ingenieurswissenschaften. S. Roderer Verlag, Regensburg, 1997.

[Sch06] S. Schober, "Stability of attractor cycles in random boolean networks," in *European Conference on Complex Systems (ECCS06)*, 2006.

[SD05] I. Shmulevich and E. R. Dougherty, *Modeling genetic regulatory neworks with probabilistic Boolean networks*, ser. Signal Processing and Communications. Hindawi Publishing Corp. New York, NY, United States, 2005, ch. 7.

[SDKZ02] I. Shmulevich, E. R. Dougherty, S. Kim *et al.*, "Probabilistic boolean networks: a rule-based uncertainty model for gene regulatory networks." *Bioinformatics*, vol. 18, no. 2, pp. 261–74, Feb 2002.

[SDM+05] J. A. Smagala, E. D. Dawson, M. Mehlmann *et al.*, "Confind: a robust tool for conserved sequence identification." *Bioinformatics*, vol. 21, no. 24, pp. 4420–2, Dec 2005.

[SDZ02] I. Shmulevich, E. R. Dougherty, and W. Zhang, "Gene perturbation and intervention in probabilistic boolean networks." *Bioinformatics*, vol. 18, no. 10, pp. 1319–31, Oct 2002.

[SFR+99] N. Stojanovic, L. Florea, C. Riemer *et al.*, "Comparison of five methods for finding conserved sequences in multiple alignments of gene regulatory regions," *Nucleic Acids Res*, vol. 27, no. 19, pp. 3899–3910, 1999. [Online]. Available: http://nar.oxfordjournals.org/cgi/content/abstract/27/19/3899

[SG04] D. Sorensen and D. Gianola, *Likelihood, Bayesian, and MCMC Methods in Quantitative Genetics*, ser. Statistics for Biology and Health. Springer, 2004.

[SH97] S. Schäffler and J. Hagenauer, "Soft decision MAP decoding of binary linear block codes via global optimization," in *Proc IEEE International Symposium on Information Theory (ISIT97), Ulm, Germany*, Jun 1997.

[SH04] A. Siepel and D. Haussler, "Combining phylogenetic and hidden Markov models in biosequence analysis." *J Comput Biol*, vol. 11, no. 2-3, pp. 413–28, 2004.

[SH05] ——, *Phylogenetic Hidden Markov Models*, ser. Statistics for Biology and Health. Springer, 2005, pp. 325–351.

[Sha48] C. Shannon, "A mathematical theory of communication," *Bell System Technical Journal*, vol. 27, pp. 379–423 and 623–656, July and October 1948.

[SL02] E. E. Schadt and K. Lange, "Codon and rate variation models in molecular phylogeny." *Mol Biol Evol*, vol. 19, no. 9, pp. 1534–49, Sep 2002.

[Slo00] N. J. A. Sloane, "On single-deletion-correcting codes," in *Ohio State University*, 2000, pp. 273–291.

[SMC07] G. Stolovitzky, D. Monroe, and A. Califano, "Dialogue on reverse-engineering assessment and methods: the dream of high-throughput pathway inference." *Ann N Y Acad Sci*, vol. 1115, pp. 1–22, Dec 2007.

[SP07] G. Sicot and R. Pyndiah, "Study on the genetic code: comparison with multiplexed codes," in *Proc of the IEEE International Symposium on Information Theory (ISIT07)*, Jun 2007, pp. 2666–2670.

[SPB07] M. L. Siegal, D. E. L. Promislow, and A. Bergman, "Functional and evolutionary inference in gene networks: does topology matter?" *Genetica*, vol. 129, no. 1, pp. 83–103, Jan 2007.

[Spe03] T. Speed, Ed., *Statistical Analysis of Gene Expression Microarray Data*. Chapman & Hall/CRC, 2003.

[SPH06] A. Siepel, K. S. Pollard, and D. Haussler, "New methods for detecting lineage-specific selection," in *Lect. Notes Comput. Sci.*, ser. 10th Annual International Conference on Research in Computational Molecular Biology (RECOMB06), vol. 3909 LNBI, U.C. Davis Genome Center, Davis, CA, 2006, pp. 190–205.

[SSGE86] T. D. Schneider, G. D. Stormo, L. Gold *et al.*, "Information content of binding sites on nucleotide sequences," *J Mol Biol*, vol. 188, pp. 415–431, 1986.

[Sud97] M. Sudan, "Decoding of Reed Solomon codes beyond the error-correction bound," *Journal of Complexity*, vol. 13, no. 1, pp. 180–193, 1997. [Online]. Available: citeseer.ist.psu.edu/article/sudan97decoding.html

[SZ02] I. Shmulevich and W. Zhang, "Binary analysis and optimization-based normalization of gene expression data." *Bioinformatics*, vol. 18, no. 4, pp. 555–65, Apr 2002.

[TB03] T. Tian and K. Burrage, "Stochastic neural network models for gene regulatory networks," in *Proc of the IEEE Congress on Evolutionary Computation (CEC)*, Dec 2003, pp. 162–169.

[TFV+08] C. Taccioli, E. Fabbri, R. Visone *et al.*, "UCbase & miRfunc: a database of ultraconserved sequences and microRNA function," *Nucleic Acids Research*, vol. 37(Database issue), pp. D41–D48, Sep 2008.

[The07] The ENCODE Project Consortium, "Identification and analysis of functional elements in 1% of the human genome by the ENCODE pilot project," *Nature*, vol. 447, no. 14, pp. 799–816, Jun 2007.

[TT98] D. Thieffry and R. Thomas, "Qualitative analysis of gene networks." in *Pac Symp Biocomput*, Jan 1998, pp. 77–88.

[Val01] A. Valembois, "Detection and recognition of a binary linear code," *Discrete Applied Mathematics*, vol. 111, pp. 199–218, 2001.

[Wil06] D. J. Wilkinson, *Stochastic Modelling for Systems Biology*, ser. Chapman & Hall/CRC Mathematical & Computational Biology. CRC Press, 2006, vol. 11.

[WLG01] S. Whelan, P. Liò, and N. Goldman, "Molecular phylogenetics: state-of-the-art methods for looking into the past." *Trends Genet*, vol. 17, no. 5, pp. 262–72, May 2001.

[WRT07] A. Wang, W. Ruzzo, and M. Tompa, "How accurately is ncRNA aligned within whole-genome multiple alignments?" *BMC Bioinformatics*, vol. 8, no. 417, Oct 2007.

[WSM04] H. Wymeersch, H. Steendam, and M. Moeneclaey, "Log-domain decoding of LDPC codes over GF(q)," in *Proc of the IEEE International Conference on Communications (ICC04)*, vol. 2, 2004, pp. 772–776.

[Yan93] Z. Yang, "Maximum-likelihood estimation of phylogeny from DNA sequences when substitution rates differ over sites." *Mol Biol Evol*, vol. 10, no. 6, pp. 1396–401, Nov 1993.

[Yan94] ——, "Maximum likelihood phylogenetic estimation from DNA sequences with variable rates over sites: approximate methods." *J Mol Evol*, vol. 39, no. 3, pp. 306–14, Sep 1994.

[Yan95] ——, "A space-time process model for the evolution of DNA sequences." *Genetics*, vol. 139, no. 2, pp. 993–1005, Feb 1995.

[Yan96] ——, "Among-site rate variation and its impact on phylogenetic analyses," *TREE*, vol. 11, no. 9, pp. 367–372, Sep 1996.

[Yan06] ——, *Computational Molecular Evolution*, ser. Oxford Series in Ecology and Evolution. Oxford University Press, 2006.

[Yan07] ——, "Paml 4: Phylogenetic analysis by maximum likelihood." *Mol Biol Evol*, vol. 24, no. 8, pp. 1586–91, Aug 2007.

[Yoc05] H. P. Yockey, *Information Theory, Evolution and The Origin of Life.* Cambridge University Press, 2005.

[YW95] Z. Yang and T. Wang, "Mixed model analysis of DNA sequence evolution," *Biometrics*, vol. 51, pp. 552–561, Jun 1995.