Computational Promoter Recognition in Eukaryotic Genomic DNA

Submitted to the

Technische Fakultät der
Universität Erlangen–Nürnberg

in fulfillment of the requirements for
the degree of

DOKTOR–INGENIEUR

from

Uwe Ohler

Erlangen — 2001

As dissertation accepted from the
Technische Fakultät der
Universität Erlangen–Nürnberg

Date of submission:	Nov 16th, 2001
Date of doctorate:	Apr 19th, 2002
Dean:	Prof. Dr. rer. nat. A. Winnacker
Reviewer:	Prof. Dr.–Ing. H. Niemann
	Prof. G. M. Rubin, Ph.D.

Dedicated to Christine (1972–1997)

If you asked me what I came into this world to do, I will tell you:
I came to live out loud.

— *Emile Zola*

And also to Christine.

Acknowledgments

Even though it (hopefully!) looks like one, this work has not been a straightforward effort. A number of "coincidences" lead to the rather unique possibility to pursue my own research ideas in two places at almost opposite points of the globe — at the University of Erlangen, Germany, and the University of California at Berkeley, USA. Not to mention that in one place it was at the Computer Science Department and in the other at the Department of Molecular and Cell Biology.

I am thus indebted to collegues and friends in both places: Martin Reese and Stefan Harbeck who helped to kick off the project in Berkeley and Erlangen, respectively. Experienced PhD students at the time, a rookie like me could learn a lot from them, according to the Manic Street Preachers' album title "This Is My Truth — Tell Me Yours". Joachim Hornegger was also essential at the beginning, but then unfortunately lost to industry. As part of their student's and diploma theses, three students implemented first drafts of a number of algorithms described in this work: Georg Stemmer, Andreas Kulicke, and Harald Köstler. Guochun Liao, Georg, and Christopher Drexler were there for good discussions and good code. Sima Misra helped with the promoter selection in the Drosophila Adh region, and Michael Winkler with the set of viral promoters. Martin, George Hartzell, Nomi Harris, and Suzi Lewis were all part of the Genome Annotation Assessment Project, one of the most interesting experiences connected to this work. Werner Obermayer, Fritz Popp, Frank Deinzer, and Erwin Frise kept the computer systems and other machines up and running. Dietrich Paulus and Elmar Nöth were the Wise Guys to get advice from. Professors Wolfgang Hillen and Günther Görz agreed to be members of the thesis committee. The latter was also the advisor of my student's thesis which can be regarded as the humble predecessor of this work. A big thank-you also to Professor Volker Strehl who substituted last-minute for one of the commitee members.

A large number of people read parts of the thesis while it came into existence (so I am not the only one to blame if you find spelling mistakes :), some reading only sections, others almost the whole thing: Catherine Nelson, Shane Hanson, Christine Oien, Ian Holmes, Georg, Jochen Schmidt, Martin, Elmar, Ben Berman. Christine and my parents provided me with food and shelter in Switzerland and Austria, over a period of five weeks in early 2001 in which the bulk of chapters 1 to 5 were written. Which brings me to the point that, of course, many people that I am glad to know had an indirect but probably not less important influence on the work. Whoever I mean will know it; there are too many names to write them all down in this place.

Finally, my thanks goes to the people responsible for the existence of this work in the first place: My advisors Prof. Heinrich Niemann in Erlangen and Prof. Gerald Rubin in Berkeley, whose groups were stimulating places to work in, and who always fully backed my project. And the Boehringer Ingelheim Fonds, namely Monika Beutelspacher and Hermann Fröhlich, whose money kept my brain going and helped me to survive in the costly Bay Area, but whose support went well beyond financial aid.

The following pages only deal with the “pure” science of bioinformatics and not with the socio-ethical issues that arise from it, be it the far too misty area of patent laws or the daunting questions of how the results of our work will actually be used by corporations and affect ordinary people’s lives. Be assured that many bioinformaticists think about them as very important issues.

Contents

List of Figures

List of Tables

List of Biological Abbreviations

bp	base pairs
cDNA	complementary DNA
CDS	coding sequence
DNA	deoxyribonucleic acid
DPE	downstream promoter element
EST	expressed sequence tag
Inr	initiator
LCR	locus control region
MAR	matrix attachment region
mRNA	messenger RNA
PIC	pre-initiation complex
pol-II	RNA polymerase II
RNA	ribonucleic acid
rRNA	ribosomal RNA
TAF	TBP associated factor
TBP	TATA binding protein
TF	transcription factor
tRNA	transfer RNA
TSS	transcription start site
UTR	untranslated region

List of Mathematical Abbreviations

ANN	artificial neural network
BW	Bühlmann-Wyner algorithm for VLMC training
CC	correlation coefficient
CE	cross-entropy
DP	dynamic programming
EM	expectation-maximization
ERR	equal recognition rate
FP	false positives
HMM	hidden Markov model
IMC	interpolated Markov chain
MC	Markov chain
MD	maximum discrimination
ML	maximum likelihood
MLP	multi-layer perceptron
MMI	maximum mutual information
MSE	mean square error
NN	neural network
PCA	principle component analysis
PFA	probabilistic finite automaton
PPX	perplexity
ROC	receiver operating characteristics
RST	Ron-Singer-Tishby algorithm for VLMC training
sn	sensitivity
sp	specificity
SSM	stochastic segment model
TP	true positives
VLMC	variable length Markov chain

Notation

Ω_k	class k
M	a (probabilistic) model
M_k	a model for class k
$P_k = P(\Omega_k)$	*a priori* probability for class k
Θ_k	set of parameters of model M_k
θ	parameter from the set Θ
W	set of training sequences $\boldsymbol{w}$
$\|W\| = n$	number of sequences in the set
W_k	subset of sequences from class k (with n_k sequences)
$\boldsymbol{w}$	discrete sequence of features
$\boldsymbol{w}_i^j$	subsequence $w_i \dots w_j$
$\|\boldsymbol{w}\| = T$	length of a sequence
V	vocabulary of words or symbols
v	one word or symbol from V
V^l	set of all sequences of length l with words from V
$P_k(\boldsymbol{w})$	conditional likelihood that a sequence belongs to class Ω_k
$\tilde{P}(\cdot)$	estimation of a probability $P(\cdot)$
$\hat{P}(\cdot)$	modified estimated probability (e. g. by smoothing)
$R_\Theta(W)$	objective function, evaluated on W using a model with parameters Θ
Q	set of states of a model or automaton; units in a neural network
X	set of feature vectors
$\boldsymbol{x}$	real-valued feature vector $\in \mathbb{R}^D$
$\boldsymbol{p}$	a *profile* (vector of physical property values)
$p_{i,k}$	profile value at position k within segment i

Chapter 1

Introduction

On February 11th, 2001, both the international publicly funded Human Genome Project and a private corporation (Celera Genomics) published the completed sequence of the human genome, the hereditary information in the form of DNA that is stored in every single cell (The Genome International Sequencing Consortium, 2001; Venter et al., 2001). This event marked the preliminary end of one of the most ambitious scientific projects that had ever been undertaken — deciphering the molecular plan of human life.

DNA sequencing is the process of determining the string of basic molecules that make up the desoxyribonucleic acid, the main carrier of hereditary information. It has become a factory process, and the speed and capacity of the sequencing machines has been immensely increased during the last 10 years. The sequencing of the human genome was thus finished five years earlier than scheduled in the original project outline, and it is not the only organism whose sequencing has been finished well ahead of time. Among the organisms whose complete DNA sequence was deposited in the public data bases in the year 2000 were important model organisms such as the fruit fly *Drosophila melanogaster* (Adams et al., 2000) and the first plant with a completely sequenced genome, *Arabidopsis thaliana* (The Arabidopsis Genome Initiative, 2000).

The arrival of these enormous amounts of data — the human genome, for example, consists of three billion basic units — has turned molecular biology into a computationally intensive discipline. This starts with the task of storing large amounts of data in different places, cross-linking them in an intelligent fashion and enabling access to it. But the application of computers goes far beyond that: Computers are essential to make sense of these data. The amount of information makes it often impossible to continue with biological research in the same labour-intensive way as it had been done before, at least not on the scale brought about by large-scale sequencing projects. A whole new field, *bioinformatics*, has therefore become an integral part of molecular biology research on a genome-wide level. It enables researchers to group the various types of data, filter them for interesting phenomena, learn about the functional background, and helps them to understand biological processes. Bioinformatics is a truly interdisciplinary field: an intersection of molecular biology and biochemistry, mathematics and computer science, and more. Typical topics of this discipline are sequence comparison and database searches for similar sequences; sequence analysis such as gene finding; phylogenetic analysis that deals with the evolutionary relations among organisms; 2-d and 3-d structure predictions of biological macromolecules; and simulation of regulatory or metabolic pathways. Excellent textbooks have been published throughout the last years; the ones that mainly deal with machine learning approaches to bioinformatics and are thus most relevant to this thesis were written by Durbin et al. (1998); Baldi and Brunak (1998); Salzberg et al. (1998b); Clote and Backofen (2000).

One example of the importance of bioinformatics concerns the interpretation of the complete DNA sequence of many organisms that the genome projects brought us. Without knowing about the important information such as the location of the genes, the direct impact of a completed genome would be limited to providing the sequence information for a few mapped genes — genes of which one already knew a rough location. Thus, an important scientific achievement that has accompanied the completed sequences is the automated computer-assisted annotation and analysis of the genomes: predicting where all the genes are and how they are regulated, finding genes similar to well-characterized ones from other species, and so forth. The publications on the human genome, for example, contained annotations of more than 20,000 of the estimated total number of 30,000–40,000 genes (The Genome International Sequencing Consortium, 2001; Venter et al., 2001). This is a necessary first step to a subsequent, more detailed analysis of genes and gene products that one is particularly interested in. Furthermore, it delivers preliminary results concerning the function of the gene products, based on similarity to already well-studied products from the same or a related organism. In the case of *Drosophila*, a more or less detailed function could be assigned to more than 50 % of the annotated genes (Adams et al., 2000), ranging from highly specific (“a DNA repair protein”) to rather general (“part of the cellular membrane”) statements.

An important part of computer-based annotation and analysis concerns regulatory DNA regions — parts of the sequence that have influence on how and when a gene is activated or *expressed*. Even though every single cell of a multicellular organism contains all genetic information at all times, only a fraction of it is active in a given tissue at any one time. The concerted and differentiated expression of genes is necessary for the existence of complex living beings with an intricate development that requires precise control on the expression of information. A beautiful example of this is an animal like the butterfly where the adult animal has an appearance completely different from its larval form, the caterpillar. Understanding the regulation of gene expression is therefore undoubtedly one of the most interesting challenges in molecular biology today. About 5–10 % of human and *Drosophila* genes are estimated to be used only to control the expression of other genes (Adams et al., 2000; Tupler et al., 2001). It is intuitively clear that errors occurring in this machinery, leading to mis-expression of genes, are a major cause of genetically based diseases. A current focus of research in bioinformatics is therefore the identification of regulatory regions and of the patterns in them that are responsible for specific regulation.

To enable subtle patterns of gene expression, control mechanisms appear at many different levels. One of the most important control levels is the first step of gene expression, the *transcription* of a gene. Here, the transcriptional machinery of the cell binds to a *promoter*, a DNA region that signals the start of a gene, and proceeds with the synthesis of a working copy of the gene. This thesis deals with the computer-based identification of promoters, and focuses on *eukaryotic* promoters of protein-encoding genes. Eukaryotes have compartmented cells with a nucleus that contains the DNA which is organized in a number of chromosomes. This distinguishes them from prokaryotes, mostly bacteria, which do not have a nucleus and whose genome mostly consists of a single coiled DNA loop. Protein-encoding genes are the largest group of genes, those whose transcription product is translated into a protein. The promoters of other gene groups that do not encode proteins, as well as the promoters of prokaryotic organisms, have a very different organization from those of protein-encoding genes, and are therefore not regarded in this thesis.

From a simplified computational point of view, the problem of annotation can be seen in the following abstract way: A channel transmitting large amounts of data in the form of symbol strings (DNA sequences) is continuously scanned, and each symbol is to be assigned to different classes: It can, for example, carry information (the genes), be junk (regions between genes that have no known function), or be a warning sign that information soon follows (the promoters).

1.1 Contributions and goals of this thesis

The main goal of this work is the development of the McPromoter system that can computationally recognize promoters in long, contiguous eukaryotic DNA sequences such as whole chromosomes. This is certainly important for DNA sequence annotation which should discover as much information as possible. But computational models can also help us to find out which information in a sequence is vital for a reliable recognition, and to increase our incomplete knowledge about the biology of gene expression.

I aim at a tool for *general* (as opposed to specific) promoter recognition: The model reflects the overall structure of promoter regions but does not deal with patterns that are specific for certain promoter subgroups. Also, the model is built to predict promoters *ab initio*, i. e. using the

DNA sequence of the organism of interest as only information.

A number of approaches dealing with eukaryotic promoter recognition have been published over the last ten years, but many of them have been largely heuristic, or have simplified some well-known facts about promoters. In contrast, my goal is to come up with a largely probabilistic model that reflects the underlying biology and is able to cope with the inherent variations of the data; every single promoter is different, which guarantees the specific expression of the gene under its control.

In particular, I take the following considerations into account:

- A framework is established where a promoter is subdivided in several parts, each of which is represented by a flexible sub-model.
- The potentially non-linear dependencies among the promoter parts are taken into account.
- To my best knowledge, this thesis describes for the first time how to use structural features of eukaryotic DNA to classify promoter sequences, and how to integrate them into a promoter prediction system.
- The model can be automatically trained for different organisms and is fast enough to be applicable on whole genomes.

From a computational point of view, the main features of the system are as follows:

- Stochastic segment models are used as model for a promoter sequence, where the states represent individual parts of a promoter. Evaluation and training algorithms are adapted from previously described generalizations of the well-known hidden Markov models.
- Interpolated and variable length Markov chains are used as efficient sub-models and trained with different objective functions. Because we use position-independent stationary chains, the model tolerates errors in the sequence.
- Non-linear dependencies among promoter parts are captured by a neural network that takes the likelihoods of the segment model as input.
- Structural, continuously valued DNA features are modeled with Gaussian densities and therefore easily integrated into the system.

1.2 Outline of this work

This work is structured as follows:

Chapter 2 provides a concise introduction to the underlying concepts of biology and bioinformatics. After describing the flow of information in the cell, I provide an overview of the computational annotation of genomes.

Chapter 3 describes gene regulation mechanisms, mainly transcriptional regulation of protein-encoding genes, and the state of the art in computational promoter finding. I discuss the basic approaches and some selected examples in more detail.

Chapter 4 gives an overview of the data sets that were used to train and evaluate the models developed throughout this thesis.

Chapter 5 is devoted to the discrete densities used to model eukaryotic promoter and background sequences. Particularly, I describe different variants of Markov chain models as well as stochastic segment models, a generalization of the well-known hidden Markov models.

Chapter 6 addresses the approach to model physico-chemical properties of the DNA in promoter regions with continuous probability densities.

Chapter 7 briefly describes the classification methods of the MCPROMOTER system and discusses evaluation criteria to measure the success of classification.

Chapter 8 turns to the description and evaluation of the MCPROMOTER system. Using the concepts introduced in previous chapters, I examine how an increasingly complex model improves on the problem of eukaryotic promoter recognition.

Chapter 9 discusses the major outcomes of this work and certain aspects of possible future work. I also comment briefly on computational methods to analyze promoters to reveal common regulatory sequence patterns.

Chapter 10 provides a summary of this work.

Chapter 2

Background in Molecular Biology and Bioinformatics

At the beginning of this work, I provide some of the necessary background in genetics and bioinformatics that will enable the reader to understand the context in which this work was carried out. I will explain the molecular structure and flow of information in the cell, and describe the computational pipeline used to annotate genes and their functions. In such a limited space, I cannot hope to provide all of the information on molecular biology that would constitute a thorough introduction; instead, the reader is referred to standard introductory works such as Knippers (1997); Alberts et al. (1994); Lewin (1999).

2.1 From DNA to proteins

Many tasks in bioinformatics deal with the analysis of sequences because the large macro-molecules that play an important role in the cell are *polymers*: sequences of linearly concatenated basic units. The most important biopolymers are nucleic acids and proteins, and the following sections describe how they are assembled, and how they relate to each other. I focus on mecha-nisms in eukaryotic cells and on the concepts that will be used in later chapters.

2.1.1 Structure of DNA and chromosomes

Hereditary information in the cell is stored in the form of DNA, *deoxyribonucleic acid*. DNA is usually present as a double-stranded molecule wherein the individual strands are wound around each other, forming a helix. The basic units of DNA are the *nucleotides* which consist of a sugar-phosphate backbone and one of the four bases adenine, cytosine, guanine and thymine. They are denoted by the letters A, C, G, and T; sometimes different letters are used to denote possible subsets from the set of all four, such as an N for “any nucleotide” (see appendix A). Adenine and guanine are purines; cytosine and thymine belong to the group of pyrimidine bases. Every turn of the double helix gives room for ten nucleotides. The bases are situated in the middle of the helix and form hydrogen bonds with the bases from the other strand, with the rule that only A and T as well as C and G can complement each other (see figure 2.1). That means that all necessary information is stored in the bases on *one* strand of the helix. This enables an easy mechanism to pass on hereditary information: in cell division, the DNA double helix separates, and afterwards each of the two daughter cells contains one strand of the original cell and one newly synthesized strand. Double-stranded molecules also have the advantage of increased stability.

DNA molecules are synthesized and read in a particular direction, from the 5’ to the 3’ end[1]. When a DNA sequence is written down, the strand which is read from left to right is called the *sense* strand whereas its counterpart in the double helix is the *anti-sense* strand because it is read in the opposite direction. A point located on the 5’ side of a reference point is said to be *upstream*, a location on the 3’ side *downstream*, and distances are denoted in bases or base pairs (bp).

DNA contains information on a multitude of levels. Already the simple relative frequency of A–T and G–C nucleotides plays an important role: A–T pairs have a weaker hydrogen bond than G–C pairs, and a separation of the double helix into single strands therefore requires less energy in AT-rich regions. Besides, the molecule does not only serve as a carrier of information but also contains sequences with no other function apart from the regulation of the expression of information contained in other parts.

Another variant of nucleic acids that can be found in cells is RNA, *ribonucleic acid*; this usually single-stranded molecule is very similar to DNA, the differences being a slightly modified sugar in the backbone and the base uracil (U) in place of thymine. Among other purposes, RNA can serve as a temporary transmitter of information or as a structural component of cell particles.

In prokaryotic organisms that do not have a nucleus, the DNA is present as a naked double stranded helix. In eukaryotes, the DNA is divided into several molecules, the *chromosomes*, and wrapped up in *chromatin*. One reason for this is the limited space in the cell: A linear double helix of DNA of the entire chromosome set of a human cell, for example, would be two meters in length. Chromatin consists of the DNA itself and protein complexes, mainly *histones*, around which the DNA is coiled up forming *nucleosomes*. This structure is subsequently folded into a more compact *solenoid*, where specific histones seal the DNA around one nucleosome and associate with each other (see figure 2.2). This tight packing is able to regulate the accessibility

[1]The numbers 5 and 3 denote the locations on a nucleotide molecule where the previous and next one in a chain are attached.

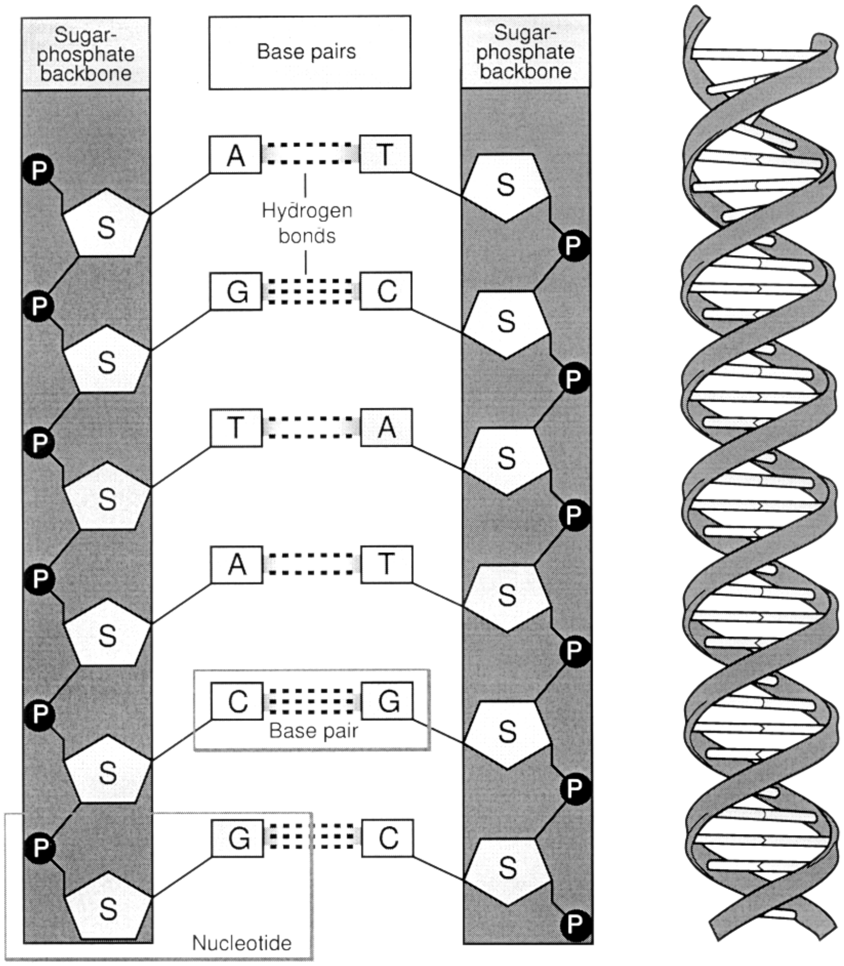


Figure 2.1: **Structure of DNA.** The left side shows the double stranded composition, the right side the famous double helix of the molecule (from The National Human Genome Research Institute (2002)).

of regions in the genome on a high level and is therefore important for gene regulation (see section 3.2.2).

During cell division, the solenoids are further compacted by extensive looping, thus forming the well-known structure of the chromosomes (figure 2.2). As opposed to prokaryotes which are *haploid*, i. e. they own only one copy of their genes, eukaryotes are usually polyploid and own multiple copies of their chromosomes. Vertebrates and *Drosophila* are both diploid; one set of chromosomes is inherited from the male and one from the female ancestor. The ensemble of molecules bearing hereditary information is called the *genome*. Table 2.1 shows the genome sizes

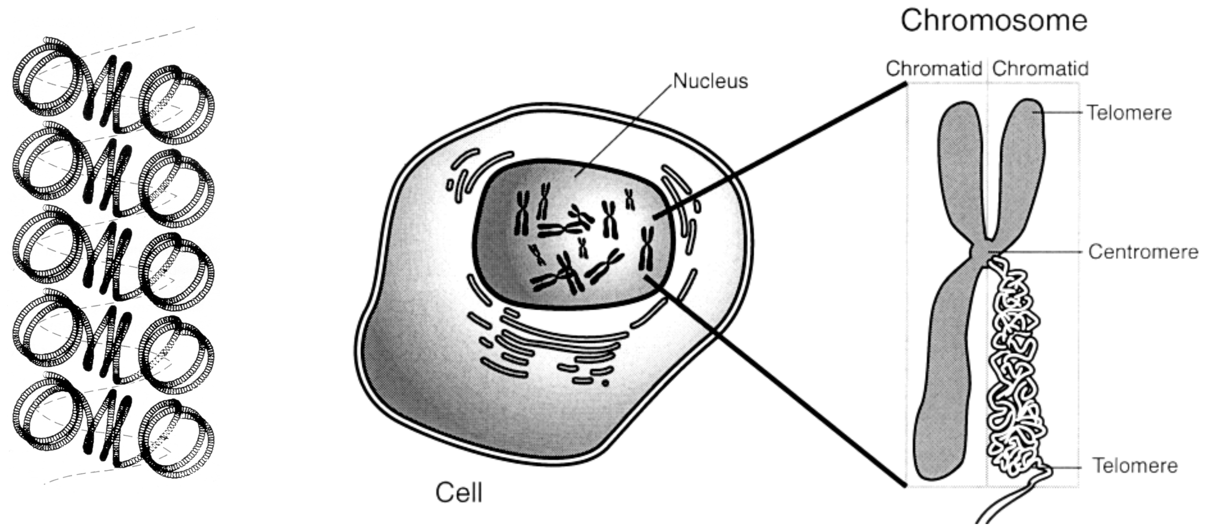


Figure 2.2: **Structure of solenoids and chromosomes.** (Left) Schematic solenoid structure, from Latchman (1998). For clarity, histones are not depicted, but one can see how the DNA loops around them (each "row" contains three nucleosomes). The dotted line denotes the higher order wrapping into solenoids. (Right) Wrapping of solenoids during cell division results in the well-known chromosomal structure of DNA (from The National Human Genome Research Institute (2002)).

of some organisms whose DNA has already been sequenced.

2.1.2 Proteins, transcription and translation

Another important class of biopolymers, proteins, are macromolecules made up by polymerization of basic units, the *amino acids*. In general and throughout all species, a cell uses 20 different amino acids, although there are additional rare ones. The relationship between DNA and proteins is as follows: A *gene* denotes a discrete segment of DNA which encodes the sequence of one (or possibly more than one) protein. Three nucleotides within the coding part of a gene, a *codon* or *triplet*, encode one particular amino acid. As there are 64 different triplets and only 20 different amino acids, this is a many-to-one relationship: the genetic code is degenerate. Three codons (UAG, UAA, UGA) serve as a stop signal without an amino acid counterpart, and the codon AUG encoding methionine always starts a protein.

Proteins are active components of a cell and serve different purposes: For example, they can form part of the cell membrane, serve as catalytic compounds (enzymes), or influence the expression of genes. The function of a protein is determined by its three-dimensional structure

Organism	**No. chromos.**	**Total size (bp)**	**No. chromos. sets**	**Est. no. genes**
Simian Virus 40	1	$5,243$	1	6
E. coli	1	$4.6 \cdot 10^6$	1	4,400
S. cerevisiae	16	$12 \cdot 10^6$	1 or 2	6,000
D. melanogaster	4	$140 \cdot 10^6$	2	13,000
A. thaliana	5	$120 \cdot 10^6$	2	25,000
H. sapiens	23	$3 \cdot 10^9$	2	35,000

Table 2.1: **Genome sizes of selected organisms.** The genomes of the simian virus 40 and the prokaryote *E. coli* contain one double-stranded DNA molecule; the eukaryote genomes are organized into a number of chromosomes. The yeast *S. cerevisiae* has a different number of chromosome sets depending on the state of proliferation. In contrast to the model plant organism *A. thaliana*, many other plants are polyploid, i. e. they contain more than two chromosome copies. Note the decreasing gene density in higher eukaryotes.

which occurs when the linear sequence is folded into its most energetically favorable state. The *secondary structure* of a protein describes the arrangement of some of its amino acids into basic three-dimensional units such as α-helices or β-sheets. The *tertiary* structure then refers to the three-dimensional conformation of the whole protein.

Proteins which are derived from one common ancestor and thus serve the same purpose are called *homologous*. As the sequence determines the structure, and similar structure implies similar function, homologous proteins show considerable sequence conservation, i. e. a large number of residues[2] have the same or chemically related amino acids. A set of proteins that serve the same purpose is called a *protein family*. Homology is also observed on the level of protein *domains* — protein parts which carry out the same function, for example, interaction with DNA or integration into a membrane. A recurrent sequence pattern such as a domain is called a *motif* and often described by means of a *consensus sequence* which shows the most frequent residue(s) at every position.

The information within a gene is used to synthesize a protein in the following way:

1. During *transcription*, the so-called messenger RNA is generated — an RNA copy of the gene sequence. The enzyme which generates the copy of protein encoding genes is called RNA polymerase II; polymerase I and III transcribe RNA genes that do not encode proteins. The polymerase recognizes the start of the gene by means of a specific promoter sequence, starts its work at the transcription start site, and stops it at terminator sites about which

[2]The term *residue* denotes a basic unit in a biopolymer.

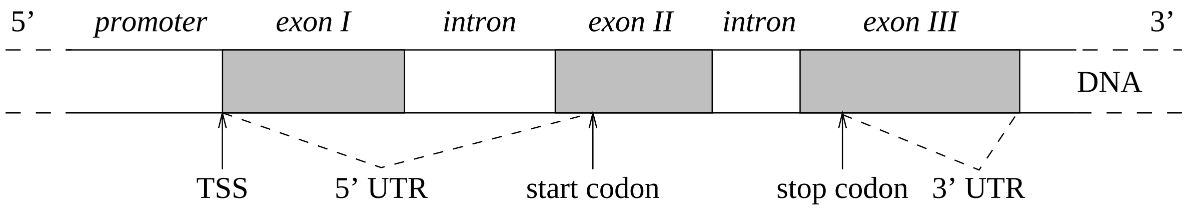


Figure 2.3: **A multi-exon gene structure.** In this artificial example, the start codon is contained in the second exon, so the 5' untranslated region (UTR) spans the complete first (non-coding) exon, the first intron, and a part of the second exon. The location of the start and stop codon, the promoter region and the transcription start site (TSS) is depicted.

hardly anything is known so far.

2. Eukaryotic *translation* takes place in the cytoplasm outside of the nucleus and synthesizes a sequence of amino acids using the sequence of nucleotides in an mRNA molecule as a template. This process takes place at active cell components called *ribosomes*. In this process, the non-translated RNA gene products play essential roles: ribosomal RNA sequences are structural components of the ribosomes, and transfer RNAs serve to guide the amino acids to the ribosomes and the right nucleotide triplet.

The first AUG in an mRNA does not necessarily serve as a start codon, and only the mRNA part between start and stop codon encodes a protein sequence; thus, the mRNA contains untranslated regions (UTRs) on both ends. In eukaryotes, the story is even more complicated: Stretches of coding nucleotides (the *exons*) are interrupted by stretches of non-coding nucleotides (the *introns*). At the beginning and end of intron sequences, so-called *splice sites* are found, characteristic sequence patterns of about 15 base pairs. Figure 2.3 shows an example gene containing two introns.

After transcription, the introns are spliced out of the pre-mRNA, and the ribosome only sees an mRNA made up from exon sequences. There is not always a unique way to splice a gene, and therefore one gene is able to encode more than one protein. Alternative splicing becomes more the rule than the exception when we look at a highly complex organism (see the recent review by Graveley (2001)). In humans, at least one third of the genes are alternatively spliced, and at the moment this is believed to be the main reason why the number of genes does not grow linearly with the complexity of an organism. On the contrary, the relatively complicated organism of the fruit fly contains considerably fewer genes than the simple worm *C. elegans*.

2.2 Computer-based annotation of genomes

As mentioned above, the raw DNA sequence data that are determined in the course of a sequencing project offer few new insights. During the process of annotation, these raw data are interpreted into useful biological information (see the reviews by Rouzé et al. (1999) for plant genomes and Lewis et al. (2000) for a more general introduction). In a large-scale sequencing project for a model organism such as *Drosophila*, annotation integrates computational analyses with a lot of biological knowledge about specific genes. It therefore is a semi-automated process in which the results of many algorithms are integrated and presented to a human curator, who then decides on the final annotation. Currently, one can identify two steps in the annotation task: Structural annotation, which deals with the identification of biologically relevant sites in the sequence, and functional annotation, which attributes specific biological information to the genome as a whole and to the sites found in the first step. Annotation provides a broad overview and description of the features contained in a genome. A deep analysis is still left to the biologist working in the lab.

Annotation has become a daunting task for large genomes because it is begun while the sequencing process is far from being finished. It must deal with a constantly changing target, update and re-annotate new or changed sequences, and track the changes over time. An example for such a project is the ensEMBL pipeline to annotate the public version of the human genome (Birney et al., 2001).

2.2.1 Structural annotation

Structural annotation usually comprises locating protein and RNA-encoding genes along with their control elements, translating the putative genes into proteins, and identifying global genomic features important for chromosome organization such as matrix attachment regions (Singh et al., 1997) or CpG islands (see section 3.2.3).

The most crucial step in structural annotation is gene finding. This is a complex task and involves the identification of patterns in the sequence as well as grouping these putative patterns to meaningful interpretations. The most recent reviews were written by Stormo (2000b); Haussler (1998); Burge and Karlin (1998), but they focus on the first of the following three different approaches to gene finding: *ab initio*, alignment, and homology based methods.

***Ab initio* gene finding.** This group of gene finders uses no information but the genomic sequence itself to find a gene. According to Burge (1997), there are four generations of *ab initio* gene finders. The first generation used statistics on coding and non-coding regions to approx-

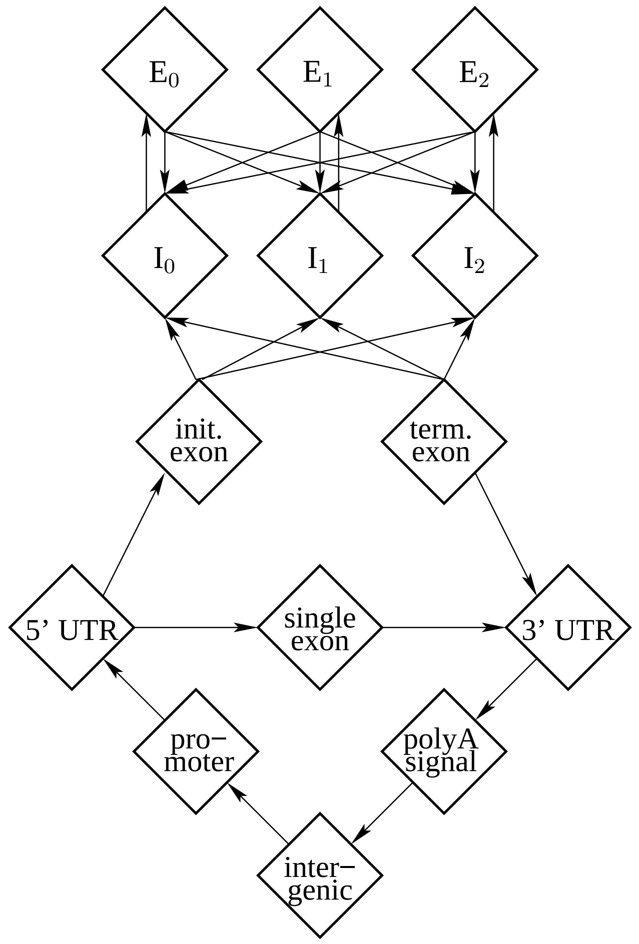


Figure 2.4: **A probabilistic *ab initio* gene finder.** The picture shows the model structure for the forward strand that is used in the GenScan system by Burge and Karlin (1997). A comparison with figure 2.3 reveals that each state represents a particular pattern or region of a gene. GenScan contains a specific state for single-exon genes and for initial and terminal exons, as their length distributions differ considerably from internal exons. The three states each for internal exons and introns are necessary to ensure that the total length of the coding sequence is a multitude of three.

imately locate exons; the second generation combined these statistics with models for splice sites to exactly locate exons; the third generation combined multiple exons in a model for a single gene; and the fourth generation was finally able to predict multiple and partial genes on both sides of a long genomic sequence. Recent gene finders thus consist of *signal sensors* that identify patterns with positionally conserved nucleotides such as the splice sites found at the intron/exon boundaries, and of *content sensors* that identify regions with a statistically significant composition such as coding exons. Exon sequences have distinct oligonucleotide statistics because of the three-periodicity caused by the triplets, and also because of a bias in codon usage: not all codons are used equally frequent. A model then describes admissible combinations of these patterns and serves to calculate an optimal parse of a DNA sequence. As such, almost all current gene finders use a framework of hidden Markov models (HMMs) or generalized HMMs (see section 5.3.1). The first simple HMM based gene finder was developed by Krogh et al. (1994b) to analyze the bacterial genome of *E. coli*; later, Kulp et al. (1996) and Burge and Karlin (1997) pioneered the application of so-called generalized HMMs for eukaryotic genomes. In contrast to simple HMMs, where a state emits a *single* symbol each time it is visited, a state in a generalized HMM models a *sequence* of symbols. This formalism is perfectly suited for gene structures, as it allows arbitrary sub-models in the states and therefore integrates both content and signal sensors within a probabilistic model (see figure 2.4).

Alignment gene finding. An *alignment* generally refers to the local or global matching of two biopolymer sequences. Usually, the residues are superimposed in such a way as to minimize the distance between the two sequences, measured by (possibly negative) scores for matches, mismatches, and insertions/deletions. Alignments can be efficiently computed by dynamic programming algorithms; Durbin et al. (1998) provide an excellent introduction (cf. also section 5.3.2). It is possible to find genes and their structure by means of an alignment of complementary DNAs (cDNAs) with a genomic sequence. A cDNA is obtained by reversely transcribing an mRNA found in the cytoplasm, and its complete sequence is assembled from short, sequenced segments called expressed sequence tags (ESTs). The set of all cDNAs from a certain tissue is called a library. Because of the low quality of cDNA sequences, wrong nucleotides as well as insertions and deletions are likely to occur. Therefore, gene finding based on cDNAs employs dynamic programming for the alignment to the genomic sequence, along with suitable gap costs for the intronic sequences that are not present in the cDNA, and a model for splice sites (see the sim4 program by Florea et al. (1998) as a widely used example). Theoretically, cDNA alignments provide the best way to find genes — *ab initio* gene finders are only able to find the coding part of genes along with the intervening introns, and usually miss the pattern-less non-coding exons and UTRs. Nevertheless, there are a number of pitfalls: Apart from contamination, a cDNA library contains only sequences of genes that were actively transcribed under certain conditions; otherwise, no mRNA would be found. Besides, cDNA sequencing does hardly ever span the complete mRNA; the single-stranded RNA is easily disrupted or digested before the reverse transcription has reached the opposite end. So-called *full-length* cDNAs try to circumvent this problem by selecting the longest cDNA of a whole set that all refer to the same gene, or by selecting cDNAs that contain the cap structure (section 3.1).

Homology gene finding. Two variants of homology based gene prediction exist. The first one attempts to identify a gene in a DNA sequence based on known proteins. This can be done by a modified dynamic programming approach which takes the one-to-many relationship between a protein sequence and all DNA sequences that can be translated into that particular protein into account. Different algorithms use either data bases of known proteins (TBLASTX, a variant of the popular BLAST alignment algorithm by Gish and States (1993)), or a library of protein family HMMs (the GeneWise approach by Birney and Durbin (2000)).

The second approach to homology based gene prediction makes use of genomic sequences of two species, known to contain homologous genes, and aligns them taking possibly different splicing into account (Bafna and Huson, 2000; Batzoglou et al., 2000). This follows the observation that coding sequences are usually conserved across species because they are translated into a protein with similar function, whereas the intervening non-coding sequences may accumulate mutations without affecting the product.

Homology based gene finding has the disadvantage that the homology might not span the complete gene but could be limited to a part of the protein. A similar observation is true for partial cDNAs, too. On the other hand, false positives are hardly ever made with these approaches, and if something about the function of one gene product is already known, the other one can be assumed to serve the same purpose. Therefore, recent gene finders usually integrate *ab initio* with alignment and/or homology approaches (Reese et al., 2000b; Krogh, 2000; Yeh et al., 2001).

Promoter recognition. The recognition of regulatory regions, namely of promoters, also belongs to the first step of annotation. Similar to the different approaches for gene finding, we can also distinguish between *ab initio* and homology based methods. Because eukaryotic mRNAs usually contain only one transcribed gene at once, it is tempting to use a suitable promoter model as one state of a probabilistic model for *ab initio* gene finding, as in figure 2.4. In this way, the admissible search region is restricted to upstream regions of detected genes, and on the other hand, a reliable promoter recognition could help to recognize the border between two neighboring genes.

In practice, this idea is hampered because of the lacking ability of gene finders to predict the non-coding exons at the 5' and the 3' end of a gene which do not contain specific patterns. It has also turned out that promoter recognition is a problem that equals if not exceeds the complexity of gene recognition. This does not come as a real surprise if one considers that promoters are located within double stranded DNA in chromatin, whereas the patterns used in gene finding are still present in linear single stranded mRNAs. Therefore, a simple promoter recognition module as it is used in the GenScan system in figure 2.4 (see section 3.3 for details) is much less reliable than the other modules. Gene finders with integrated cDNA alignment are able to considerably restrict the admissible region of promoter predictions and are therefore more successful (Reese et al., 2000b).

A complex system for promoter recognition such as the one introduced in this thesis does not rely exclusively on linear sequence dependencies but also takes non-linear dependencies as well as other sources of information into account. Adding such a model to a generalized HMM and still obeying the underlying theory is not straightforward and poses a challenging conceptual problem that has not yet been dealt with. In big annotation projects, the algorithms are thus applied independently of each other, and a human curator deals with the combination of the results. A comprehensive literature survey on promoter recognition algorithms is given in section 3.3.

Why promoter recognition is important. From the above discussion, several reasons emerge why a reliable stand-alone promoter recognition system is useful. Gene regulation is one of the most important research topics in molecular biology, but one in which many things are still unclear. It is therefore important to exactly find the regulatory regions to be able to examine them in detail, either computationally or by experiments, and learn the mechanisms that control the expression of genes. An evaluation which features improve the quality of promoter predictions can also help to understand the mechanisms how promoters are actually recognized in the cell.

From the annotation point of view, promoter identification can help gene finding algorithms to identify the 5’ UTRs that can span up to tens of thousands of kilobases. In the genomic test set that is used to evaluate the performance of the *Drosophila* promoter predictor, the average UTR length is about 2,000 base pairs, and some examples have UTRs that extend over more than 30,000 bp. It can also help to detect genes in the first place, namely those which are rarely expressed and thus not part of a cDNA library, short genes which are easily missed altogether, and non-coding RNA genes which do not show codon statistics at all.

2.2.2 Functional annotation

Once the genes and other functional parts of the DNA sequence have been identified, the next step consists of the functional annotation of those features. Teichmann et al. (1999) provide a recent overview of this field.

The first step is the assignment of an isolated function to each individual gene, either by pair-wise sequence similarity or similarity to a model of a protein family — this is implicitly carried out in gene finding by homology. The currently best way to do this appears to be a bootstrap approach: using an up-to-date protein database, similar sequences to the query sequence are pulled out, and a model is constructed from this new set. This step can be iterated and will thus find more and more distant homologues (Park et al., 1998).

If no homologue to a protein arising from a complete gene structure can be found, it may still be possible to provide some information on the domain level. For this task, large databases of domain models have been collected (e. g. InterPro (Fleischmann et al., 1999)) that integrate different resources. Also, some properties of a protein, such as its integration into a membrane, can be predicted reliably (Krogh et al., 2001), and secondary structures such as helices and sheets can be assigned to some protein regions. This annotation then enables us to look at cross-species conservation on a genome-wide level: how many protein families and which of the protein domains are present in all species, how many of the proteins contain transmembrane components, and so forth. The Gene Ontology Consortium (2000), among others, aims to provide a controlled vocabulary for such large-scale annotations to ease comparisons of annotation results for different species.

DNA and protein microarrays provide a completely different view of the genome (DeRisi et al., 1997; Haab et al., 2001). This technology monitors the activity of many, up to all known, genes of an organism under certain experimental conditions, either on the mRNA or protein level. Analyses such as the activity of genes related to the cell cycle (Spellman et al., 1998) provide a wealth of information. Recently, Shoemaker et al. (2001) have even released an annotation of the draft human genome based on microarray data. From a number of different experiments, correlations between the expression of several genes may become visible, and may thus serve to reconstruct genetic networks depicting the flow of information in different metabolic or regulatory pathways of the cell (Bower and Bolouri, 2001; Friedman et al., 2000).

The outcome of microarray analysis also enables further functional annotation. The promoter regions of co-regulated genes can be analyzed for common patterns (see chapter 9); on the other hand, such common patterns may also serve to provide functional annotation based on the promoter regions of genes, especially for those for which no homologous sequences could be found (Pavlidis et al., 2001). If the proteins interacting with regulatory patterns are known, this approach is a promising way to elucidate regulatory networks.

As a summary for the annotation process, figure 2.5 gives a schematic overview of the flow of information in an annotation pipeline.

2.2.3 Assessment of genome annotations

Annotation is a crucial task to make full use of the large volume of genomic sequence. Therefore, it is of great importance to objectively assess the accuracy of predictions made in the annotation process, both to know how well an automated annotation generally works, and to know which method is best for a certain problem.

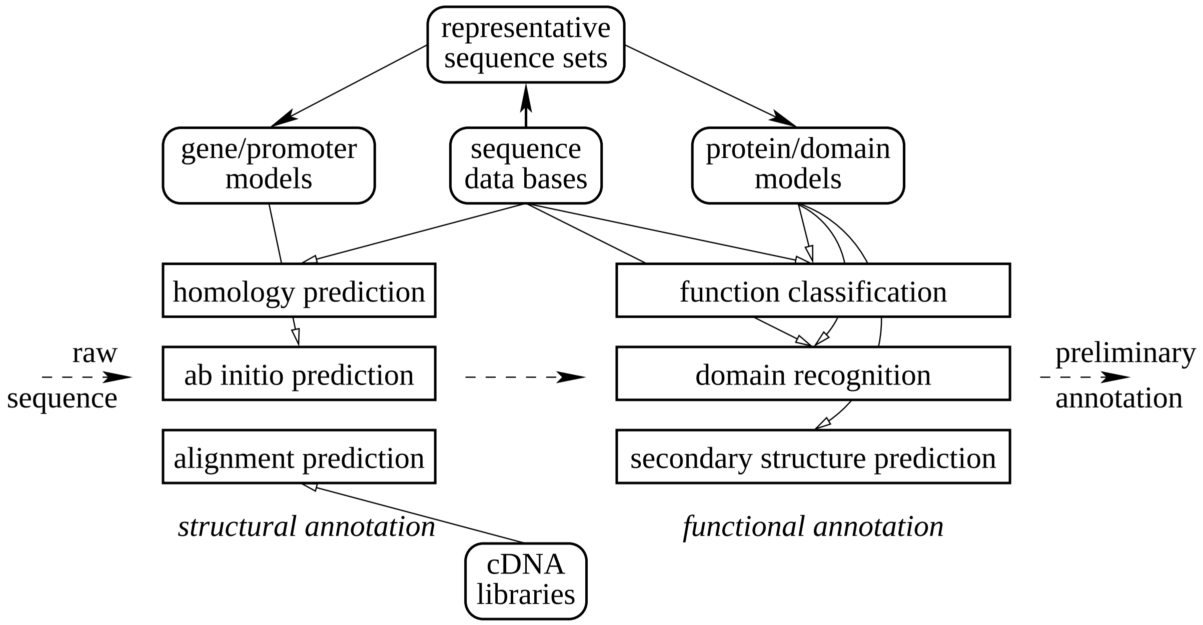


Figure 2.5: **Schematic overview of sequence annotation.** The picture shows only some important tasks of structural and functional annotation to exemplify the relations between data bases, models, and the sequence to be annotated. The preliminary annotation is in many cases validated by human curation.

The prerequisite to and most important factor in an objective assessment is the standard data set used to evaluate solutions. The standard must be well-studied, but also correct, fair and appropriate in the community's eyes. The correctness should have been established by methods independent of the methods being assessed; the fairness is guaranteed if no predictor had any prior knowledge of it; the appropriateness is given if the standard is representative and large enough for drawing meaningful conclusions.

Because sequences are usually almost immediately released to the public data bases, a data set that is large enough and still fulfills the fairness criterion is almost impossible to obtain. Assessments are thus often carried out on either too small (Burset and Guigo, 1996; Fickett and Hatzigeorgiou, 1997) or partially simulated data sets (Guigo et al., 2000). The genome annotation assessment project (GASP) by Reese et al. (2000a) therefore turned out to be an unprecedented opportunity to assess predictions on a realistic scale: An almost 3 million base pair long and well-studied contiguous piece of genomic *Drosophila* DNA had been sequenced but not yet published. This enabled the authors to tackle with the criteria mentioned above: For an assessment of the *sensitivity*, i. e., of how many of the known genes were found, they used a set of full-length cDNA sequences of hitherto unknown genes. As this incomplete gene set was clearly not suited to assess the *specificity*, i. e., how many of all predictions turned out to be true, a second set of complete annotations was compiled which also included previously known genes and homology based predictions. The first set contained 43, the second 222 genes, and were therefore large enough for a thorough comparison. Twelve groups participated in the project, making submissions for gene finding, promoter recognition, repeat finding, and protein domain classification. The best gene finding programs achieved a sensitivity of 95% and a specificity of 90% on the base level, but only 60% respectively 40% on the level of complete gene structures[3]. For the assessment of promoter recognition, a representative subset of the 222 genes was compiled (see section 4.1). The promoter finders had a sensitivity of about 30–35%, with different specificity depending on whether they were used *ab initio* or in combination with a gene finder. Domain classifications and repeat identification were not evaluated because a correct answer was not known for these categories. The overall results showed that the algorithms performed worse than expected from the assessments on smaller sets; genes are sparsely arranged in the DNA of higher eukaryotes, and false predictions are therefore likely. Consequently, the annotation of the draft human genome by Birney et al. (2001) relied only on gene finders that also use homology or alignment information.

So far, I have discussed the basic biological concepts and the process of computer based annotation of DNA sequences, and have also shown where promoter recognition fits into the larger framework. We can therefore now turn to the specific biology of promoter eukaryotic promoters, and to previously published algorithms to recognize them.

[3]Only the coding parts of genes were considered; UTRs cannot be predicted by most current gene finders.

Chapter 3

Promoters and Promoter Recognition

The topic of gene regulation has always received great attention because the key for the development of complex organisms does not lie as much in the mere number of genes but rather in their specific regulation and interaction. In the following, I will give a brief description of the biology of gene regulation, particularly of DNA transcription control and the organization of eukaryotic promoter regions. Again, this text cannot go into all necessary details but will focus on the concepts relevant to this thesis. A comprehensive yet easy to read introduction to this fascinating topic was written by Latchman (1998) from which much of the following description is inspired. Other, mostly more recent references are cited throughout where appropriate. In the final section, I will turn to computational approaches for promoter recognition published so far and discuss what aspects of promoters are taken into account in current algorithms.

3.1 Gene regulation in eukaryotes

It was observed rather early that a loss of DNA content occurs only in some notable exceptions and therefore cannot offer a general explanation for the individual protein levels found in different developmental stages or tissues. Rather, the process whereby DNA produces mRNA (and subsequently proteins, see section 2.1) must be responsible for the regulation of gene expression in eukaryotes. A number of stages leads from the initial transcription to the final protein product (see figure 3.1 for a schematic overview). In theory, any of these stages could be used to regulate the expression of a gene, and it has been shown that indeed all of them are targeted under one condition or another. I will explain the stages shown in figure 3.1 and indicate how they can be regulated, before I turn to a more detailed description of transcription control. Even though this description suggests that all steps have to be performed in a rigorous order, evidence shows that they are at least partly concurrent.

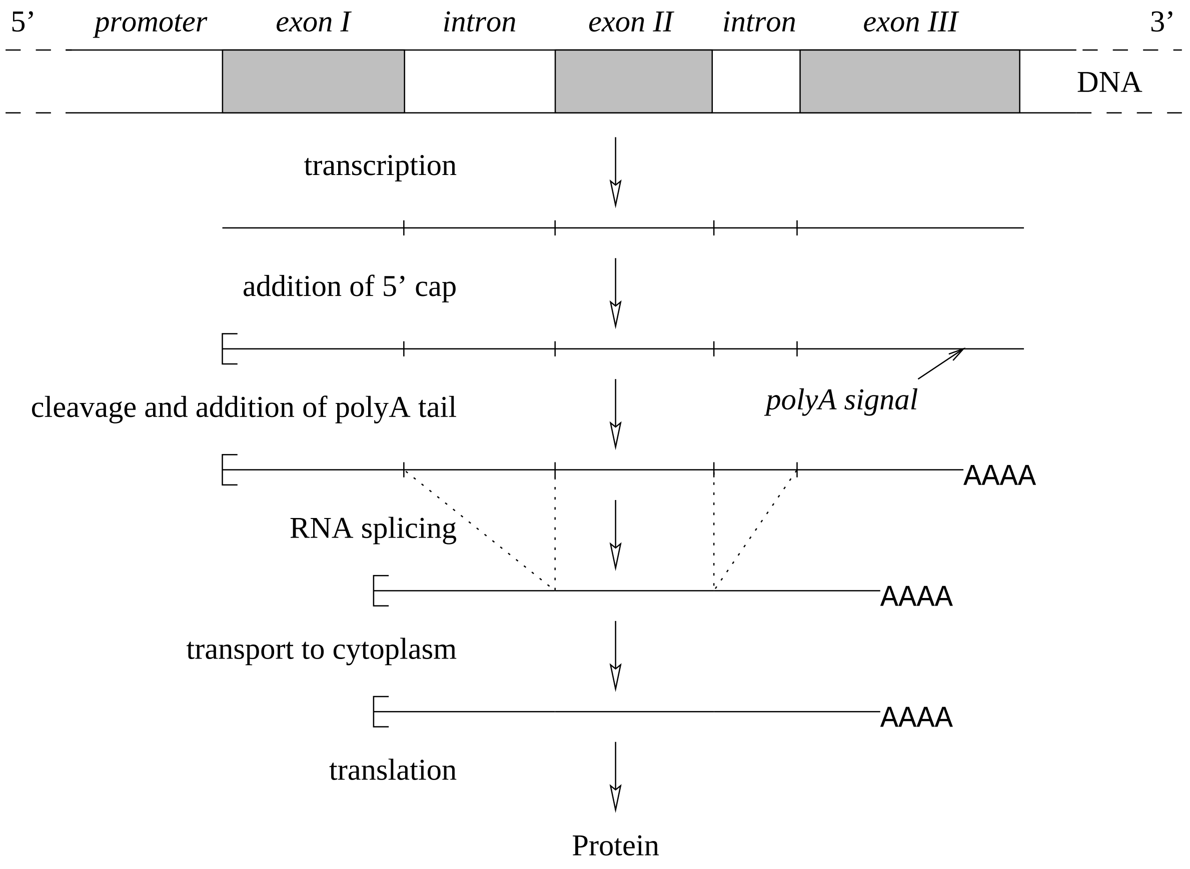


Figure 3.1: **Stages of gene regulation**, after Latchman (1998). See the text for details.

1. The *transcription* of protein encoding genes is done by the RNA polymerase II enzyme, and control at this level involves guiding the polymerase to the right places as well as inhibiting its activity (see section 3.2). The result of transcription is the pre-mRNA or *primary transcript*. A single gene can be transcribed starting from different promoters that are active only under specific conditions, giving rise to two or more (partially) different gene products.

2. The *post-transcriptional events*, which lead from the primary transcript to the final mRNA serving as a template for translation, start with the *capping* of the pre-mRNA. A cap structure consists of a guanosine residue linked in an unusual way to the 5' end of the RNA. The cap is the place where a ribosome binds to the RNA and is also necessary to protect an RNA from degradation enzymes.

3. Contrary to the modification at the 5' end which involves the adding of a single nucleotide, the 3' end is cleaved, a large RNA is stretch removed, and up to 200 adenosines are added.

The site of this *poly-adenylation* is flanked by two conserved sequence patterns where two protein factors bind, interact, and finally cut the mRNA. Similar to the cap at the 5’ end, the polyA tail serves as protection against degradation, and it appears to have an effect on the translation efficiency of the mRNA. Also, more than one polyA signal can be present, leading to different possibilities to truncate an mRNA on its 3’ end.

4. The next step in RNA processing is *splicing*. Apart from the splice sites at both ends of an intron (see section 2.1), an additional less well-conserved pattern around the *branch point* can be found close to the splice site at the 3’ end of an intron. Splicing occurs in a complex structure known as the spliceosome which involves a number of RNA and protein components and holds the upstream and downstream parts of the mRNA in the correct place while cutting out the intron. Alternative splicing as discussed in section 2.1 has emerged to be a crucial regulatory step, complementing transcription control and serving to deliver variants of a single protein needed under specific conditions. Alternative splicing is regulated by tissue specific factors promoting a certain splice site as well as by the ratio balance of several proteins belonging to the spliceosome.[1]

5. After the mRNA has been brought in its final shape, it is *transported* from the nucleus through the nuclear membrane into the right place in the cytoplasm. A number of proteins have been identified that are believed to mediate this transport. It appears that a nuclear export signal in such a protein is crucial to guarantee export of itself and its associated mRNA. A few examples show that regulation may also happen at this stage, e. g. promoting the transport of a certain splice variant of a viral mRNA in HIV infected cells.

6. In the cytoplasm, *translation* takes place at the organelles known as ribosomes. It is initiated by the binding of a ribosome at the cap structure on the 5’ end. A subunit of this ribosome then migrates along the mRNA until it finds an appropriate start codon. In rare cases, an mRNA may contain more than one functional start codon. A key role in the subsequent translation is played by transfer RNA (tRNA) molecules which deliver the correct amino acid to the currently considered nucleotide triplet. tRNAs have a common characteristic secondary structure and are bound to the mRNA by means of *anti-codons* complementary to the triplet for which they carry the appropriate amino acid. Subsequently, one tRNA after another is recruited, and a polypeptide is synthesized until the first stop codon is encountered. General control at this stage is possible by inhibiting components of the ribosomes; specific mechanisms interact with patterns in the 5’ and 3’ UTR of the mRNA that form characteristic secondary structures, the latter sometimes also preventing poly-adenylation

[1]Different proteins that are derived from the same primary transcript can also be a cause of *RNA editing* which modifies single bases, thus replacing one amino acid by another or introducing a stop codon.

which is necessary for a correct translation.

Translation is influenced by the *stability* of the mRNA which determines the number of times that it is translated. A working model of this mechanism involves digestion enzymes attached to the ribosome that either recognize the beginning of the synthesized polypeptide or short regions in the 3' UTR that fold into a secondary structure. RNA stability is an effective means to control the rate of protein synthesis, especially in cases where a rapid and transient change of a specific protein level is necessary, and is often accompanied by a change in transcription rate.

To summarize, gene expression controls which genes are used, which modifications are carried out to the transcript, and how efficiently the final product is synthesized. A clear point should be made that gene regulation, and therefore transcription, is not a yes/no activity: Genes which are "switched on" do not all produce the same amount of mRNA. Although analogies from the terminology of engineering might suggest it, a cell is not a simple machine, not even at the level of individual genes. It is a viable precondition for the correct development of an organism that a subtle control is possible for every single product of biosynthesis. Also, if a gene is found to be "active" under certain *in vitro* conditions, its activity might be dramatically enhanced or suppressed by interactions only observable *in vivo*[2].

3.2 Regulation at the transcriptional level

Even though regulation occurs at all stages of protein synthesis, the control on the transcriptional level is clearly the most important. This intuitively makes sense: Why should a cell generally sacrifice valuable energy to synthesize products whose activity is subsequently repressed, again under the consumption of energy?

The RNA polymerase II (pol-II) enzyme has 12 subunits; but it despite its structural complexity, it cannot carry out transcription by itself. It requires auxiliary factors to recognize its target promoters, and to modulate production to react on specific environmental conditions.

The promoters of protein encoding genes can be seen to consist of a core promoter, a proximal promoter region, and distal enhancers, all of which contain *transcription elements*, short DNA sequence patterns that are targeted by specific auxiliary proteins called transcription factors. Transcription initiation by pol-II is regulated by those factors interacting with transcription elements, pol-II, and also with each other, and by an open chromatin structure that enables the factors to access the DNA.

[2] "*In vitro*" usually means "simplified conditions in experiments at a lab bench", whereas "*in vivo*" refers to the living cell.

3.2.1 The basal transcription machinery

The common and best characterized part of promoters is the *core* promoter which is responsible for guiding the polymerase to the correct transcription start site (TSS). Accurate initiation of transcription depends on assembling a pre-initiation complex (PIC) containing pol-II and at least six transcription factors, the *general* initiation factors, which have been identified over the past 20 years (see the general review by Roeder (1996), and the one by Nikolov and Burley (1997) focusing on a detailed view on the protein structures). This complex machinery is immensely well preserved throughout all species.

Inspection of the sequences immediately upstream of the transcription start sites showed that a large group of eukaryotic promoters share an AT-rich sequence element around position -30.[3] This so-called TATA box is the most prominent sequence element in eukaryotic promoters. Detailed *in vitro* studies have elucidated the fundamentals of PIC assembly, deriving a minimal set of factors that suffice for transcription from a strong viral promoter containing a well-conserved TATA box.

The TATA box is a target of transcription factor (TF) IID, or more specifically, of one component of TFIID, the TATA binding protein (TBP). TFIID contains at least a dozen other components known as TAFs (TBP associated factors) which also interact, directly or indirectly, with other sequence elements. Upon binding of TBP, the DNA is strongly distorted, and sequences up- and downstream of the TATA box are brought in close proximity. Transcription factor IIA stabilizes this complex, even though it is not essential in all cases as originally thought and only vital for promoters with weaker TATA boxes.

After binding of TFIID (and possibly TFIIA), this complex is recognized by transcription factor IIB. This orients the growing complex towards the transcription start site and maybe guides the polymerase to the exact start position. TFIIB also recruits the pre-formed TFIIF–pol-II complex through direct interactions with both components. The binding of transcription factors IIE and IIH to the polymerase completes the assembly of the pre-initiation complex. With the exception of TFIID and possibly TFIIB (see below), all TFs are recruited by protein-protein interactions, and no interactions with specific DNA motifs has been observed so far. TFIIH finally triggers the start of transcription by modification of a pol-II subunit, and unwinds the DNA double helix in a 10 base pair long stretch downstream of the TSS. This step-wise assembly is summarized in figure 3.2.

While the polymerase moves off down the gene, TFIIF remains associated with the polymerase and TFIID remains bound to the core promoter, alleviating further cycles of PIC as-

[3]Positions upstream of the TSS are counted backwards starting at -1, and positions downstream, including the TSS itself, are started counting at +1.

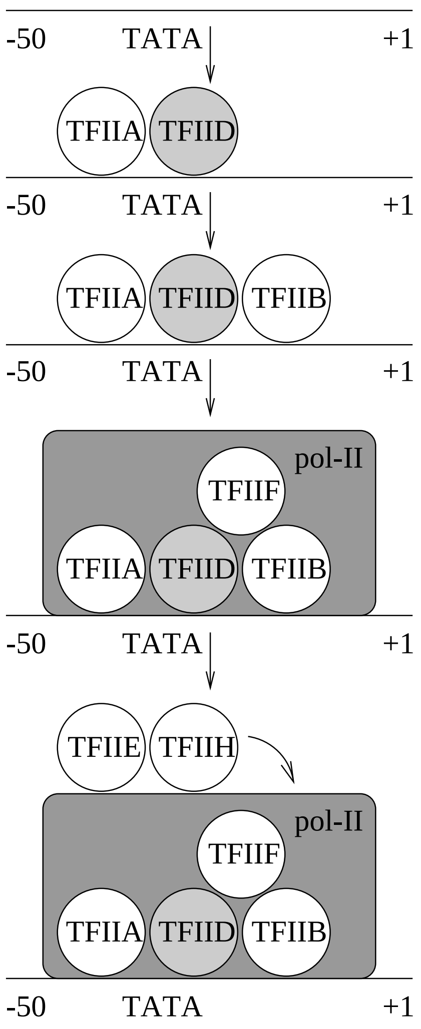


Figure 3.2: **Step-wise assembly of the pre-initiation complex**, after Latchman (1998).

sembly. Large multi-protein complexes containing pol-II and some of the transcription factors have been reported from purification experiments. This suggests the existence of a so-called holo-enzyme in which much of the PIC is already pre-assembled, which allows the process of transcription initiation to happen much faster than an individual step-wise assembly would require.

As tempting as it sounds, the above description is by far a general mechanism of pol-II recruitment. For example, the promoters of house-keeping genes (i. e. genes that are always “switched on”) do not contain anything resembling the TATA box. In these cases, the binding of TFIID is mediated by the *initiator* sequence element right at the TSS, but which is not only present in TATA-less promoters. Chalkley and Verrijzer (1999) recently reported that some of

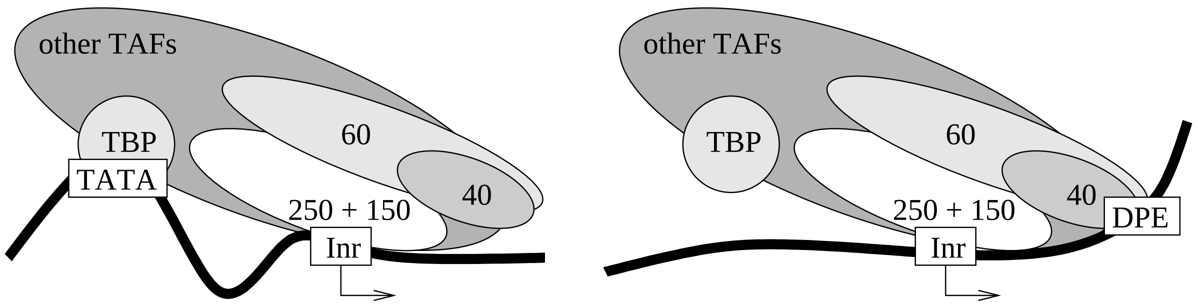


Figure 3.3: **Interaction of TFIID with the core promoter elements.** Two distinct interactions with TATA-driven (by TBP) and DPE-driven (by TAFs 60 and 40) promoters are shown in this model after Kutach and Kadonaga (2000) (Inr: initiator).

the TAFs are able to directly recognize this element.

In *Drosophila* as well as vertebrates, sequences downstream of the initiator were also found to have influence on basal transcription activity. Arkhipova (1995) showed that a number of short sequence patterns are significantly over-represented in downstream sequences of *Drosophila*, but found considerably weaker conservation in vertebrates. According to recent findings by Kutach and Kadonaga (2000), a specific *downstream promoter element* (DPE) appears to be as widely used as the TATA box but is less well-conserved. Its core motif is located exactly from 28 to 33 base pairs downstream of the TSS and was earlier shown to be recognized by two factors of the TFIID enzyme (Burke and Kadonaga, 1997). A striking preference for the initiator consensus in promoters that contain a DPE suggests a strong co-dependency of both elements. Although evidence for downstream vertebrate elements exists, current knowledge suggests that DPEs play a less important role in these organisms. It should finally be noted that in TATA-less promoters, different transcription starts from several neighboring bases have been observed, and a transcription start “site” as such does not exist. If the promoter elements are not well conserved, it possibly is a general rule that the transcription start differs within a small range.

Sequence patterns in the core promoter. To summarize, the main sequence patterns by which interactions with transcription factors occur in the core promoter, and which could be exploited in a computational promoter finding system, are the TATA box, the initiator, and the downstream promoter element (see figure 3.3). These are all known to be directly targeted by TFIID components. Bucher (1990) was the first to systematically study the patterns of TATA box and initiator in vertebrates, and Arkhipova (1995) extended this to the sequence elements of *Drosophila*, including DPE. On the one hand, she found that the TATA box is present in at most 50% of the

Drosophila promoters which is less frequent than in vertebrates. On the other hand, the initiator is better conserved in fly promoters. Also, as stated above, the DPE is much more frequent in *Drosophila* and appears to play the role as a downstream counterpart of the TATA box. Hence, the machinery of transcription is well conserved throughout the whole eukaryotic kingdom, but the ways in which it is employed in transcription regulation are not. This makes it vital to use different models for the prediction of promoters in different organisms. It shall also be noted that a working binding site such as the TATA box is not defined by some absolute strength but also by the context in which it appears: Using the best hit of a TATA box model within each promoter, instead of all above a threshold, results in a much better sensitivity and specificity of the detection of known TATA boxes (Audic and Claverie, 1998).

TFIIA and TFIIB also have direct contact with DNA, but it has been widely believed that these contacts are not sequence-specific. Recent experiments (Lagrange et al., 1998) that were published during the course of this thesis suggest that TFIIB binding in humans is at least partly influenced by a sequence motif directly upstream of the TATA box, but this is not well characterized so far. Detailed studies of sequence motifs in *Drosophila* (Arkhipova, 1995; Kutach and Kadonaga, 2000) revealed the TATA, initiator, and DPE motifs, but failed to detect any motif resembling the TFIIB response element. So even if it might be present in a small number of cases, it does not play an overall important role, at least in *Drosophila*. It is striking that this putative sequence pattern consists almost exclusively of guanines and cytosines: Human promoters have a very high overall GC content. This gives rise to the suspicion that the TFIIB element is to some extent reflecting the overall human promoter sequence composition. Thus, the proven *in vitro* binding of TFIIB to a sequence pattern might not play a specific role *in vivo*.

3.2.2 Chromatin structure in promoter regions

The large number of genes found in eukaryote genomes would render it very impractical should all of them compete for the components of the basal transcription machinery at the same time. Most of them are transcribed only inside a specific tissue or under rarely occurring circumstances. Evolution has therefore found a way to effectively shut down large regions of the genome that are not needed within a certain tissue. This also guarantees that all cells of a tissue stay committed to expressing the same genes without actually losing parts of the genome.

Experiments have shown that even transcribed genes are still wrapped up around nucleosomes (cf. figure 2.2), but that the higher order condensation into solenoids is lost in active or potentially active genes. Such genes exhibit a heightened sensitivity to a DNA digestion enzyme that even extends for some distance up- and downstream of the transcribed regions. These less condensed regions are not dependent on the act of transcribing but remain stably established and thus reflect the ability to be transcribed.

Methylation and CpG islands. In vertebrates, the open solenoid structure is closely associated with *DNA methylation*: some cytosines are chemically modified and bear an additional methyl group. In 90% of the cases, the methylation occurs in cytosines that are part of the di-nucleotide CG.[4] It was found that some CG sites are always methylated whereas for others, this pattern keeps changing in a tissue-specific manner, and active genes appear to be un-methylated. Furthermore, Antequera and Bird (1993) postulated that the upstream regions of all constitutively (i. e., constantly) expressed genes, and also a substantial portion of other genes, are correlated with clusters of CG dinucleotides, so-called CpG islands (Gardiner-Garden and Frommer, 1987). Methylated CG dinucleotides are a hot spot for mutations in which the cytosine is wrongly replaced by a thymine, which over the course of time leads to CG depleted regions. Indeed, the CG di-nucleotide occurs much less frequently in vertebrate genomes than expected from the mono-nucleotide composition. CpG islands with a high number of CG di-nucleotides therefore hint at generally low methylated regions.

DNA methylation has no direct effect on the chromatin structure, and no direct evidence for specific protein interactions with methylated regions that are associated with chromatin structure has been reported. On the other hand, DNA methylation is known to be stable during cell division because the CG di-nucleotide on the opposite strand of a methylated one is also methylated. Methylation can therefore explain the stable commission to certain groups of active genes within specific tissues.

Histone modification. Methylation can possibly explain the majority of cases of tissue-specific commitment in vertebrates, but in invertebrates such as *Drosophila* it hardly occurs (Lyko, 2001). A number of vertebrate cases are also known where differences in methylation between expressing and non-expressing tissues cannot be detected. Other features of active chromatin structure concern chemical modifications of the histones. Histone modifications either affect histone association with each other or the DNA, or proteins interacting with histones. It is known that one component of the TFIID enzyme as well as other transcription factors have the ability to acetylate histones which leads to an opening of chromatin. The opposite case of de-acetylation and therefore a negative effect on regulation is also observed. In either way, chromatin structure could thus be changed.

[4] The notation “CpG” for a CG di-nucleotide is used to resemble the phosphate bridge between adjacent bases. This avoids the possible mis-interpretation of CG as a complementary pair in the double helix.

Chromatin structure in regulatory regions. Following the discovery that a change in the chromatin structure of genes is necessary for their (potential) activation, further studies showed that the DNA in the regulatory regions of active genes is even more sensitive to DNA digestion. These hypersensitive sites are a result of either loss or modification of nucleosomes and hint at less tightly packed DNA compared to active genes. Hypersensitive sites are furthermore not only concentrated in the core promoter region, but exist also in other regulatory regions described in section 3.2.3. Widely used transcription factors that bind to those regions associate with specific proteins that indeed have the capability of displacing or modifying the nucleosomes. A common mechanism in gene regulation is therefore the attraction of nucleosome displacing factors which enables the binding of other factors and finally the PIC itself. The observation that nucleosome displacing proteins have also been found in some of the holo-enzymes of pol-II perfectly fits in that picture.

The DNA in promoter regions is furthermore likely to exist in an alternative super-coiled conformation, the so-called *Z-DNA*. This conformation occurs in DNA with alternating purine and pyrimidine nucleotides and offers an increased accessibility to the single strands of DNA, which means that it is easier for proteins to interact with Z-DNA than with normal DNA.

3.2.3 Specific gene regulation: Sequence elements and transcription factors

So far, I have dealt with the basal transcription machinery, describing how the transcription start site is recognized and which proteins are involved in this process, and with the chromatin structure that enables the access to genes in the first place. In eukaryotes, where each mRNA that is transcribed encodes for only one gene[5], this cannot explain how genes whose protein products are needed in parallel are co-regulated. Very often, coordinately expressed genes do not even reside at close positions in the genome, but rather on different chromosomes. Such a system reflects the greater need for flexibility in eukaryotes; for example, human α-globins on chromosome 16 are expressed at the same time as γ-globins on chromosome 11 to form working globins in the fetus, but in adults the γ-globins are replaced by β-globins which also reside on chromosome 11.

Britten and Davidson (1969) published an early working model of such coordinated gene expression that, at an abstract level, still holds (see figure 3.4). They proposed that genes regulated in parallel, in response to a particular signal, would contain a common regulatory element which would cause the activation of these genes. Moreover, genes could contain more than one element, each shared with a different group of genes. A signal would then act by stimulating a specific

[5] There is no rule without exception: Note the *Drosophila Adh/AdhR* genes that are transcribed on one mRNA.

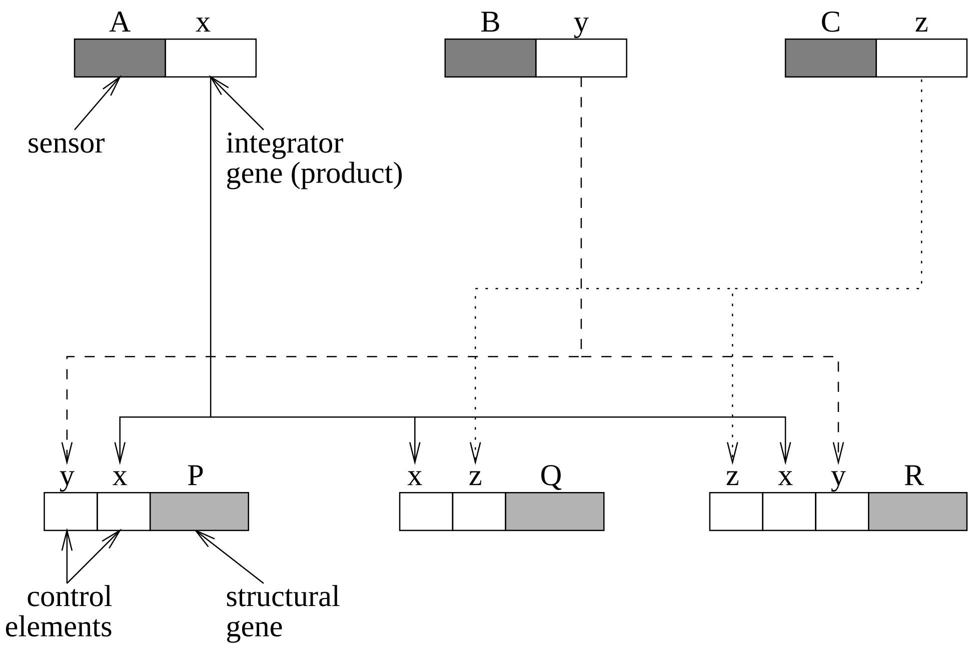


Figure 3.4: **The Britten and Davidson model** for coordinated gene regulation, after Latchman (1998). Sensor elements A, B, and C detect changes that require a different expression and therefore switch on appropriate integrator genes x, y, and z. The products of genes x, y, and z then interact with control elements, coordinately switching on appropriate genes P, Q, and R. Alternatively, x, y, and z can be proteins undergoing a conformational change under the presence of specific signals which enables them to interact with the control elements.

"integrator gene" whose product would interact with a specific sequence element in several genes at once. A gene would finally be activated if all its sequence elements had been "switched on" by integrator gene products. Using current terminology, the integrator gene is considered as encoding a transcription factor which binds to regulatory sequence elements, the transcription factor binding sites, and activates or suppresses a specific group of genes. Supplementing the original theory, a transcription factor can be activated not only by *de novo* synthesis but also by changing the inactive state of the pre-existing protein into an active one, often by means of post-transcriptional regulation (see section 3.1). The latter possibility is actually the more frequent one, as a regulation of transcription factors by transcriptional control simply pushes the problem onto a higher level.

The proximal promoter region. Many of the regulatory elements serving as transcription factor targets are located in the proximal promoter region, i. e. directly upstream of the core pro-

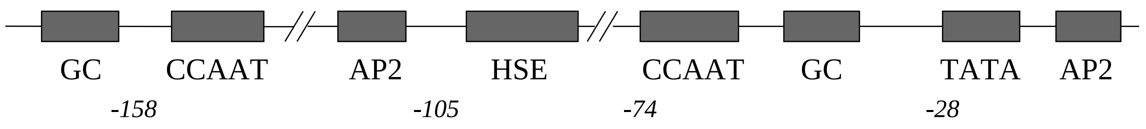


Figure 3.5: **The promoter of the human hsp70 gene.** As an example, this promoter contains non-specific (CCAAT, GC, AP2 boxes) as well as specific (HSE, heat shock element) control elements in its sequence (Latchman, 1998). The numbers in this schematic structure refer to the position relative to the transcription start site.

moter. These factors can either influence (both suppress or alleviate) the binding of the core promoter components, or the chromatin structure (see section 3.2.2), or both at the same time. The first group thus interacts with components of the general initiation factors, such as the TATA box binding protein or its associated factors, or alleviates the binding of other factors which then interact with the basal machinery. This only works when considering the 2-d or 3-d DNA structure — in a linear DNA sequence, the binding sites are too far away from each other to enable direct contacts of their TFs. An example for a synergistic interaction of two transcription factors with two TAFs is reported by Verrijzer and Tjian (1996): The two *Drosophila* factors bicoid and hunchback interact with specific TAFs and lead independently to an already improved transcription activation, which is nonlinearly increased when both factors are present. Therefore, the complex structure of TFIID is not necessary to bind to the DNA and recruit the polymerase, but rather serves as a modular machinery that offers a vast number of possibilities to interact with.

Some transcription factors work in a non-specific way, i. e. they merely serve to increase the production rate of the basal machinery and can thus be found in a variety of genes. Sequence elements that interact with these factors are the CCAAT and GC boxes in vertebrates, or the GAGA box in *Drosophila*. Other factors work in a very specific way and are contained in only a small number of promoters (see figure 3.5 as an example for a human promoter). Transcription elements can be present in several copies in one promoter and are very often organized in dyad symmetry, i. e. one of two identical sequence parts is contained on the sense and the other on the anti-sense strand, thus forming a palindrome. Orientation therefore does not matter, but this is also true for many non-palindromic patterns. In some cases, elements responding on related stimuli are also related on the sequence level, such as in the case of hormone receptor binding sites, some of which are made up by the same repeat of the sequence GGTCA, but with variable spacing and either as a direct or palindromic repeat (see the examples in table 3.1).

Enhancers and silencers. Apart from the proximal promoter regions, it has been discovered that sequences as far away as several kilobases have a major influence on transcription. Although

Signal	Regulatory element
Palindromic repeats	
Oestrogen	RGGTCAN^{3}TGACCY
Glucocorticoid	RGRACAN^{3}TGTYCY
Direct repeats	
Vitamin D3	AGGTCAN^{3}AGGTCA
Thyroid hormone	AGGTCAN^{4}AGGTCA

Table 3.1: **Various hormone response elements.** An N indicates any base; R indicates a purine, Y a pyrimidine (see appendix A). Note that these are consensus sequences — i. e. the most frequent base is given at each position — but that individual binding sites may have mutations differing from the consensus.

these sequences cannot act as promoters on their own, they are able to enhance or suppress the activity of transcription up to three orders of magnitude. Interestingly, such a sequence cannot only be far away from the promoter it affects, but also both upstream or downstream and even within an intron of the gene which promoter it enhances. As with many transcription factors, the orientation of the sequence, i. e. whether it is on the sense or anti-sense strand, is also not important for its functionality. These *enhancers* or *silencers* often exhibit a tissue-specific activity, and they are often composed of the same sequence elements found in (proximal) promoters that mediate tissue-specific expression. Like transcription factors binding to promoters, factors binding to enhancer elements influence gene expression both by changing the chromatin structure and by interaction with proteins of the transcription apparatus. Because of the very large distance of the enhancers from the affected promoters, the second mechanism is especially puzzling, and the most commonly accepted explanation in concordance with experimental results involves the looping out of intervening DNA. A particular enhancer can affect more than one promoter, and can exert its influence also on the transcription of other genes when transferred into their neighborhood.

Locus control regions. A high level of control of the expression of several genes at once is achieved by so-called *locus control regions* (LCRs). These regions were found to be crucial for the activity of all the genes in a cluster, e. g. the α- or β-globin genes. They act independently of their position and over a large distance, and without them no single promoter in a cluster can attract the polymerase *in vivo*. As with enhancers, some elements that are present in promoter regions are also found in LCRs, and they are likely to have a long-range influence on the chromatin structure. Several LCRs were also found to contain sequences which are involved in the attachment of chromatin domains to a protein scaffold, the so-called nuclear matrix. An LCR controlled region may therefore constitute one solenoid loop, the structure of which — and therefore general accessibility to transcription factors — is regulated as a single unit. It also serves as an insulator to block the activity of outside enhancers.

3.3 Approaches for computational promoter recognition

The previous section introduced the underlying biology of transcription control in eukaryotes, and the following pages now turn to the description of approaches that deal with the identification of regulatory DNA sequences by computational methods. The first description of common patterns in eukaryotic promoters, in the form of *weight matrices* which are equivalent to linear hidden Markov models, can be found in the ground-breaking publication by Bucher (1990). In this thesis, I specifically concentrate on the *general* identification of proximal promoter *regions*, although I will shortly discuss related approaches to model specific sub-groups of promoters. For more information on models for single transcription factor binding sites, the reader is referred to the reviews by Werner (1999); Stormo (2000a). I also do not discuss prokaryotic promoter recognition because promoters in lower organisms have a different, somewhat less complex structure.

Similar to gene finding approaches, existing methods for general promoter prediction can be classified into two different categories, *ab initio* and homology based. Fickett and Hatzigeorgiou (1997) wrote a noteworthy review on *ab initio* predictors and compared their performance on a set of independent sequences. Even though the set they used is too small to allow for precise conclusions, and even though the paper is outdated by now, it is still of great influence because it judged a large number of systems in an unbiased and sound way. One of the first reviews on homology based methods was written by Duret and Bucher (1997); this approach is also termed phylogenetic footprinting.

3.3.1 *Ab initio* prediction

Computational methods that aim at the identification of promoters *ab initio* tackle the task by establishing a model of promoters — and possibly non-promoters as well —, and then use this model to search for an unknown number of promoters in a contiguous DNA sequence. Depending on how the model captures promoter features, different sub-groups of *ab initio* predictors can be distinguished:

- *Search-by-content* algorithms identify regulatory regions by using measures based on the sequence composition of promoter and non-promoter examples.
- *Search-by-signal* algorithms make predictions based on the detection of core promoter elements such as the TATA box or the initiator, and/or transcription factor binding sites outside the core.

There are also methods that combine both ideas – looking for signals and for regions of specific composition. To achieve an exact promoter localization, a system output should include the prediction of the transcription start site. Search-by-content methods do not provide good TSS predictions because they do not consider any positionally conserved signals. To enable the comparison of different algorithms, and to account for possible multiple start sites, predictions are usually counted as correct if they are made within a window around an experimentally verified start site.

In the following, I will discuss the most important publications. For a description of the underlying algorithms, the reader is referred to the text books such as Durbin et al. (1998); Baldi and Brunak (1998), and also to chapters 5 and 7 of this work. All of the approaches deal with primate or vertebrate promoters; even though some of them were certainly applied to data from other eukaryotic organisms, none was specificly re-trained.

Search-by-content. This group of approaches considers features derived from long promoter and non-promoter sequences, and uses the established model to calculate scores on moving sequence windows. For example, Audic and Claverie (1997) train Markov chain models of different orders on promoter and non-promoter sequences (see section 5.2), and classify a sliding window of 250 bp using Bayes' rule. They use only one Markov chain model trained on intron and exon sequences as background, and take the non-promoter sequences of the data base records from which the promoters were derived as negative samples. There are several shortcomings in this approach: First, the training sequences for non-promoters are not checked for redundancy, and both coding and non-coding sequences are represented by the same model. Also, only sequences from the initiator to -250 are included, and downstream promoter sequences are neglected. The reason that most promoter sets use regions of 250–300 bases upstream of the TSS lies in the observation that the greatest increase in transcription factor binding site density is observed in the region from -200 on (Prestridge and Burks, 1993).

A similar approach based on oligomers[6] of length six is proposed by Hutchinson (1996). His model distinguishes between background classes for coding and non-coding sequences and are trained on 300 base pair long sequences, the promoters ranging from -300 to the start site. Instead

[6]An oligomer refers to a short biopolymer sequence of fixed length.

of modeling the sequence classes by Markov chains, two discriminative measures D_1 and D_2 are defined as follows:

$$D_i(w) = \frac{F_f(w)}{F_f(w) + F_{b_i}(w)} \tag{3.1}$$

$F_f(w)$ denotes the absolute count of word w in the foreground promoter sequences and $F_{b_i}(w)$ the absolute count in background sequences ($i = 1$: non-coding; $i = 2$: coding). Hutchinson uses words of length six, so w denotes one of the 4096 possible hexamers. 196 out of the 200 top rating hexamers according to the D_1 measure contain at least one CG di-nucleotide, showing the strong bias for non-methylated regions in the promoters (see section 3.2.2). To apply the system on unknown sequences, a 300 base pair window is shifted in steps of 10 bases, and identifies the window with the highest average D_1 value that also exceed a threshold on D_2. As an additional restriction, the sequence is assumed to contain exactly one promoter. Because no apparent correlation of scores and true and false positives is seen, Hutchinson also suspects that a relative score maximum is important for a working promoter. The underlying restriction of one promoter per sequence, though, is not justified any longer in the post-genomic age where whole genomes are scanned for putative promoters.

The idea of discriminative counts is also employed by Scherf et al. (2000). They construct "classifiers" for two classes by exclusively assigning certain sequence groups to one class if their occurrence ratio in the sequences of this particular class versus the other class exceeds a certain threshold. Sequence groups contain a motif of a certain width plus a limited spacer of arbitrary nucleotides that may occur within the motif. These classifiers are then used together to classify a sequence window of 100 bases in the following way: Each classifier determines how many groups in each class are hit, and the window is regarded as a promoter if the promoter class has the largest number of group hits for all classifiers, in this case for promoters against the background classes coding, non-coding, and 3' UTR. In a post-processing step, neighboring windows that are assigned to the promoter class are merged, and a region is finally reported if it extends for at least 200 bases. The promoter training sequences range from -500 to +50, and redundant sequences in the training set are removed (cf. chapter 4). Because the authors noted that their approach consistently causes "shadow predictions", i. e. predictions at the same position on the opposite strand, the detected regions are not strand specific and also quite large — roughly between 500 and 2,000 base pairs. In a recent publication where they applied the system on the complete sequence of human chromosome 22, they additionally report that, as in the case of the approach by Hutchinson (1996), the predictions are highly correlated with CpG islands. This means that their approach is very helpful to narrow down the search region, but misses a large portion of non-CpG island associated promoters.

A different way of search-by-content has recently been proposed by Ioshikhes and Zhang (2000). They concentrate on CpG island associated promoters and come up with features that help to distinguish between promoter-associated and other CpG islands (see section 6.1). A CpG island is generally defined by a minimum length of 200 bp, a GC content of at least 50 %, and a ratio of observed to expected CG di-nucleotide frequency of more than 0.6. CpG islands containing a TSS are found to have a greater average length, higher GC content and CG ratio than other CpG islands. The combination of the three feature variables with quadratic discriminant analysis leads to successful recognition.

Search-by-signal. In contrast to search-by-content methods which work with overall features derived from sequence classes, search-by-signal approaches are based on models of specific patterns in promoters, i. e. models of transcription factor binding sites. Thereby, either extensive lists of binding sites from transcription factor databases are used, or an exact modeling of the core promoter is attempted.

After it had been found that the detection of single patterns such as the TATA box is far from being both sensitive and specific enough, the combination of several patterns was pioneered by Prestridge (1995). First, he determined the occurrence of known transcription factor binding sites that were compiled from the literature. The hit ratio within promoters and non-promoters is then used as a measure of reliability. To look for promoters, the ratio scores of binding site hits within a window of 250 bases are summed up, and this sum is increased by a somewhat arbitrary value if a TATA box is observed within the last 50 bases of the window. A promoter is predicted if the sum finally exceeds a pre-defined threshold. The main problem of this approach lies in the way the transcription factor binding sites are used: they are represented as strings and therefore need an exact match to count, certainly too inflexible for many degenerate binding sites.

Owing to the ideas of the previous approach, and somewhat similar to the system of Scherf et al. (2000), Chen et al. (1997) build their recognition system on both over-represented oligonucleotides and a collection of weight matrices. They try to overcome the problem of Prestridge's approach by determining the over-representation of all oligonucleotides, by non-strand-specific occurrence ratio or χ^2 significance of promoters with respect to non-promoters. Additionally, weight matrix models for frequent transcription factor binding sites are added to the model if they have a considerably larger amount of hits within promoters than in non-promoters. Scanning for promoters is done in a similar way to Prestridge, but the scores are based on absolute counts and not on the occurrence ratios. The thresholds to decide whether the oligonucleotides and weight matrices are to be added to the model, and how many hits are to be encountered to regard a sequence as promoter, are chosen *ad hoc*. If a sequence is believed to be a promoter, an attempt to detect a TATA box is made. If no TATA box is present, the orientation of the promoter cannot be determined.

The exact modeling of core promoters is in many cases based on artificial neural networks. Reese (2001) trains two time-delay networks for the TATA box and the core promoter element which are able to detect a pattern even if it does not occur at a fixed position within the considered input windows. This is achieved by receptive fields in which the nodes use the same shared weights. The two sub-nets are then combined in another time-delay net to allow for non-linear weighting of the two patterns. The networks are trained on large sets of representative positive and negative samples, and extensive weight pruning keeps the networks from over-adapting to the data.

The architecture of Knudsen (1999) is based on an ensemble of multi-layer perceptrons for binding sites. In contrast to the previous approach, the individual networks are supposed to learn the most prominent sequence patterns in an un-supervised way. Apart from input nodes for sequence patterns, each of the networks has an additional input node which is fed with the strongest activation of the other networks on the same input. This prevents the networks from modeling the same sequence motif. Knudsen compares several approaches with four subnets. In the first one, all nets are trained with a randomized algorithm that aims at the maximization of the correlation coefficient of promoter and non-promoter sequences. After the training, three of the four networks recognize TATA-box-resembling features, whereas the fourth does not show a significant preference for known patterns. In a second experiment, the subnets are initialized with the four common motifs that were described by Bucher (1990) — TATA box, initiator, CCAAT and GC elements. A third experiment uses the parameters from Bucher (1990) and only trains the inhibitory weights on the input nodes from other subnets, which surprisingly turns out to be the most successful approach. This is possibly due to the choice of downstream promoter regions as negative examples which leads to bad results in the two approaches where parameters are indeed trained from the data.

Bucher's weight matrices for TATA box and initiator are also used within the built-in promoter state of the GenScan system (Burge and Karlin, 1997), along with an additional spacer state. The first eukaryotic promoter recognition module included in a gene finding system was described by Matis et al. (1996), but it depends on other signals such as a start codon and a coding region in a reasonable distance downstream, and is restricted to TATA box containing promoters. In contrast to the GenScan system which is able to partially parse a DNA sequence and therefore also to provide isolated promoter predictions, the latter approach relies on complete gene structures and is therefore not an *ab initio* predictor *per se*. Finally, studies by Ohler (1995) and Pedersen et al. (1996) describe the usage of hidden Markov models trained on either core elements or the longer upstream promoter sequences, but no complete systems for promoter recognition are reported.

Combined methods. Solovyev and Salamov (1997) developed a promoter finding system that combines a TATA box weight matrix, triplet preferences in the initiator, hexamer frequencies in three regions -1 to -100, -101 to -200, and -201 to -300, and hits of transcription factor binding sites by linear discriminant analysis. It therefore integrates specific models for binding sites inside and outside the core promoter with general statistics that are also used by search-by-content approaches.

Zhang (1998) divides a promoter region in eight consecutive segments of 30 bases, ranging from -160 to +80, and in 5 segments of 45 bases, ranging from -145 to +80. For each segment, he uses discriminative word counts analogous to equation 3.1: The sequence within the considered window makes up the foreground, the sequences within the left and right neighbor window the background counts. The feature variables consist of the mean discriminative count within each window, and are combined using quadratic discriminant analysis. Zhang does not use non-promoter sequences in his model and describes his predictor as a tool to locate the core promoter within a window of 1,000–2,000 base pairs.

Shortcomings of ab initio approaches. After the description of previous approaches, this section continues with a discussion on the limitations and differences of them.

First of all, an exact prediction of the TSS can only be made if the model is based on elements within the core promoter. This is one of the notable differences between search-by-content and search-by-signal methods: Even though some of the first group predict TSSs, based on a maximum in the prediction score, the mean distance from the annotated transcription start sites is usually larger than with signal based approaches. A modeling of the core promoter alone, though, is not specific enough, and these approaches (Reese, 2001; Knudsen, 1999) have high false positive rates. On the other hand, methods that look for exact string matches of TF binding have low true positive rates: most binding sites are degenerate, and it is not very common that a new instance looks exactly like the few ones collected in the data bases. In this light, an algorithm that combines a core promoter model with statistics of extended regions appears to be a promising way.

It is also important to look at the way how dependencies both within and across the signals are captured. For example, it can be expected that the GenScan approach to linearly combine the TATA box and initiator weight matrices is not as successful as the time-delay network approach: Neural networks are able to represent joint distributions on the whole patterns and therefore take dependencies among distant nucleotides in the same signal into account. The additional combination of several sub-networks also allows for a non-linear combination of the signals, as does the quadratic discriminant analysis.

To some extent, shortcomings are certainly due to inappropriate background training and

author	core promoter signals	TF binding sites	content features	CpG island features	> 1 non-prom. classes	repres. non-prom. sets	non-linear correl.
Audic			x				—
Hutchinson			x		x		—
Scherf			x		x	x	—
Ioshikhes				x	—	—	x
Prestridge	x	x				x	
Chen	x	x				x	
Reese	x				x	x	x
Knudsen	x	(x)					x
Solovyev	x	x	x		o	o	
Zhang			x		—	—	x

Table 3.2: **Properties of promoter finding approaches.** An “x” indicates an existing property, a blank that the property is missing. An “o” tells that the information could not be extracted from the literature. A dash means that the property does not make sense in the context of the program; for example, the approach by Zhang looks for start sites within promoters and does therefore not contain any non-promoter classes.

data: distinctive background classes are represented by only one model (Audic and Claverie, 1997), and the negative set is not cleaned for redundancy or simply badly chosen (e. g. downstream sequences in the approach by Knudsen (1999)). Table 3.2 compares the discussed approaches by means of the discussed properties.

The assessment of eight promoter finders by Fickett and Hatzigeorgiou (1997) provided a unique source of information on the real-scale application of the systems. It showed that most of the publications had reported too reliable a performance; in fact, many authors had simply extrapolated the false prediction rate from the number of predictions obtained on a small negative set. Although the number of promoters in the assessment test set was too small to allow for an exact ranking, some tendencies were noteworthy: A weight matrix for the TATA box alone provided a recognition rate hardly above chance, and although there was a large difference in the number of total predictions, the ratio of false and true predictions was very similar for all methods. The algorithms correctly predicted 13–54 % of the promoters and had false positive rates of 1/460 up to 1/5,520 base pairs, where the predictor with the highest true positive rate also had the highest false positive rate and vice versa. False positive rates might in some cases be caused by enhancers that were recognized as promoters because they often contain the same binding sites found in promoters. No publication so far attempted at an explicit distinction between enhancers and promoters.

The performance of recent approaches is better: Ioshikhes and Zhang (2000) achieved a recognition rate of 93 % of CpG island associated promoters (i. e., a CpG island within -500 up to +1500), which translates to an overall sensitivity of 47 % and a specificity of 34 % on a test set of 135 promoters. This is similar to the results of (Scherf et al., 2000) who reported 43 % sensitivity and specificity, albeit on a different set. In their recent analysis of human chromosome 22 (Scherf et al., 2001), about 45 % of the previously known genes had a hit in the region of -2000 up to $+500$, with a total specificity of about 40 %. Again, these programs do not attempt to predict an exact TSS, but report non-strand-specific regions of roughly 0.5–2 kilobases in size. It also appears that these approaches are successful on a large scale because of the restriction to or correlation with the subset of CpG-island-associated promoters, even if the modeling did not start from this assumption in the case of Scherf et al. (2000).

Physical properties. From the discussion above, it can be seen that almost all approaches are exclusively based on sequence properties of DNA and do not exploit physical properties to reflect the distinct chromatin structure observed within promoters. Merely the promoter-associated CpG island identification by Ioshikhes and Zhang (2000) can be somewhat regarded as an exception. The only approach known to the author to classify promoters based on physical properties of DNA so far has been published by Lisser and Margalit (1994) and deals with *E. coli*, i. e. prokaryotic, sequences. For a set of promoters and non-promoters, profiles for four different properties were computed (see chapter 6). The profiles were divided in five non-overlapping consecutive segments defined by the two sequence elements present in prokaryotic promoters: the -35 and -10 box. The mean values of each property and each segment then served as feature variables for a linear discriminant analysis. Although the authors did achieve a good classification performance on a set of promoters and coding sequences, they did neither use their system to look for new promoters in long sequences, nor did they integrate sequence and profile features. Profiles of eukaryotic promoter regions were calculated for a number of properties (Pedersen et al., 1998; Babenko et al., 1999), but no attempt to classify eukaryotic promoters by means of profile features has been previously published.

3.3.2 Prediction by homology

A completely different approach to identify promoters, or regulatory elements in general, is the identification of regulatory sequences in the upstream regions of related genes that are conserved across different species. Here, the idea is that evolutionary pressure keeps the regulatory patterns free of mutations, whereas the surrounding DNA without specific function will accumulate mutations. In analogy to "footprinting" experiments in the wet lab which digest the bulk of DNA but leave intact the sequences in regions where proteins bind, this approach has been dubbed *phylogenetic footprinting*. Its popularity increases with the advent of the complete sequences from several model organisms. Duret and Bucher (1997) describe phylogenetic footprinting applications, and excellent recent reviews on these approaches in the context of genome wide analyses were written by Fickett and Wasserman (2000) and Pennacchio and Rubin (2001).

Phylogenetic footprinting is carried out by specialized alignment algorithms. Standard dynamic programming approaches do not obey the additional biologically meaningful restrictions that can be imposed: Sequences shall contain conserved blocks that represent one or more binding sites, and very low similarity otherwise. How much this corresponds to reality depends on the organisms from which the sequences are used. Organisms that are too closely related will show too much overall similarity; in distant organisms the underlying regulatory pathway might have changed, in which case the sequences contain different binding sites.

Two practical examples where human-mouse phylogenetic footprinting was one step leading to the identification of regulatory elements have recently been reported (Wasserman et al., 2000; Hardison, 2000).

Shortcomings of homology approaches. Phylogenetic footprinting is a completely different way to look for regulatory regions; it does not employ a model of what to look for, but rather identifies the common patterns found in two or more sequences. As with content based *ab initio* approaches, homology based approaches cannot be expected to deliver exact predictions of the transcription start site. If sequences from two organisms in the right evolutionary distance are chosen, it is an effective means to approximately locate regulatory regions, not necessarily restricted to the proximal promoter region. It is therefore best used as a means to narrow down the search region, either for *ab initio* promoter finders or for methods that identify binding sites common to a group of sequences (see chapter 9).

3.3.3 Models for promoter sub-classes

The first publication ever describing a computational prediction of eukaryotic promoters dealt with specific models for heat-shock and glucocorticoid elements (Claverie and Sauvaget, 1985). They consist of two consensus sequences separated by a fixed length spacer sequence. This pioneered the usage of specific models for certain promoter sub-classes. Whereas a single binding site is not specific enough to reliably detect real promoters, such a combination of two elements can already greatly increase the specificity (see Werner (1999) for a review of these simple approaches). Wagner (1999) describes a clean statistical framework for the detection of significant clusters containing multiple binding sites of one or two factors, regardless of their number or distance relative to each other.

In many cases, though, the story is more complicated. Wasserman and Fickett (1998) examined the promoter sub-group of muscle-specific genes. They found that most promoters contain some out of a group of specific binding sites, but their number, occurrence and distance is apparently not hard-wired. In this case, simple models will fail. Wasserman and Fickett (1998) therefore propose a logistic regression function that uses the best hits of weight matrices for commonly observed factors within a specific window.

The Bayesian approach by Crowley et al. (1997) could also be a suitable way to detect a sub-group of promoters. They start from a group of binding site motifs and raise the assumption that their occurrence in regulatory regions is remarkably different from the one in non-regulatory DNA sequences. Placing priors on the number and length of regulatory regions that are expected in a sequence, they calculate the a posteriori probability that a certain position is within or outside a regulatory region by a sampling approach.

Pavlidis et al. (2001) use another way to combine frequent patterns, in their case within a modular hidden Markov model with sub-models for the individual binding sites. They do not use this model to search for promoters, but rather classify the function of genes based on their promoter regions, using the best path through the model as a feature vector for a support vector machine. This appears to be a promising approach in combination with data from gene expression experiments. In contrast, constructing specific models to look for promoters of a sub-group within genomic sequences will, to some extent, remain a special case because of the frequent lack of the necessary amount of data. Specific models are certainly a good way to identify enhancers that simply consist of a cluster of specific binding sites, but they cannot satisfy the need for reliable models for general recognition of eukaryotic pol-II promoters.

Chapter 4

Data Sets

Over the past two decades, publicly accessible data bases have accumulated an incredible wealth of information about biological sequences. One group of data bases aims to generally collect all DNA and protein sequences; examples include the international nucleotide sequence database collaboration of Genbank, European Molecular Biology Laboratory (EMBL) database and DNA Database of Japan where the publicly funded genome projects deposit their sequences (Wheeler et al., 2001; Stoesser et al., 2001; Tateno et al., 2000) or the SwissProt collection of confirmed proteins (Bairoch and Apweiler, 2000). Another group collects sequences related to specific problems.

The notorious difficulty of promoter recognition is partly due to the limited amount of reliably annotated training material. The experimental mapping of a TSS is a laborious process and therefore not routinely carried out, even if the gene itself is studied extensively. In the field of transcriptional regulation, the Eukaryotic Promoter Database (EPD) maintained by Perier et al. (2000) and available at http://www.epd.isb-sib.ch is therefore of great interest. The database curators extract experimentally proven pol-II promoter sequences of higher eukaryotes and their viruses from the literature and cross-link them to EMBL data base entries. If possible, entries contain the sequence from -500 to +100 relative to the TSS, but a minimum of only 45 bases between -49 and +10 is required. EPD also contains a *representative subset* which marks only one representative for each group of highly homologous sequences ($>$ 50 % identical between -79 and +20) and therefore avoids the danger of over-representation of closely related entries. This subset is especially suited for computational analyses: If data sets are not checked for redundant entries, the machine learning algorithms are inevitably biased to favor sequences similar to the over-represented ones, which is generally not desired. In April 2001, EPD release 66 contained a total of 1390 entries, 905 of which belonged to the representative subset. Another important collection related to eukaryotic promoters is TRANSFAC (Wingender et al., 2001), which contains both protein entries of transcription factors as well as DNA entries of their binding sites.

As described in chapter 2, much of the annotation of genomes is based on computational tools which need to be trained on reliable data sets. The data bases mentioned above provide a good source from which one can start to construct representative sets of confirmed genes and regulatory sequences. The goal when generating these data sets is to make them relatively “clean” and to ensure that each sequence conforms to specific criteria such as minimum length or non-ambiguous annotations. At the Berkeley Drosophila Genome Project we therefore used restrictive filters which were run on the data bases to create high-quality data sets for gene finding and the recognition of promoters and splice signals (Reese et al., 2000a). Each set is divided into a number of disjoint parts which can be used for a cross-validated evaluation, i. e. repeated runs of training and testing on different sets to obtain meaningful results.

The aim was to provide common sets to be shared among various research groups as a stable basis for the evaluation and comparison of different methods for the analysis of human and *D. melanogaster* DNA sequences. Therefore, we made our ready-to-use training and test sets available and encouraged researchers in the community to use these common datasets for the development of their methods. Common data sets allow also a fair and rigorous scientific comparison between different methods, as it was done in the Genome Annotation Assessment Project (Reese et al., 2000a, see also section 2.2.3) where these sets were widely used by participating groups. The sets are available at http://www.fruitfly.org/sequence/human-datasets.html and http://www.fruitfly.org/sequence/drosophila-datasets.html.

4.1 *Drosophila* data sets

Our sequence sets for the training of promoter recognition models consist of three parts: promoter sequences, coding sequences (CDSs), i. e. exon sequences outside the untranslated regions, and non-coding (intron) sequences. All sequences were to have the same size of 300 bp. The goal was a set of three disjoint and equally sized parts to allow for cross-validation experiments.

Starting point for the non-promoter sequences was a set of 275 multi-exon genes collected from Genbank version 109 (1999). Genbank was searched for sequences containing single Drosophila genes, i. e. ranging at least from start through stop codon. The sequences were obtained on the genomic level rather than by cDNA sequencing. Only one CDS annotation was allowed to discard known alternatively spliced genes. No in-frame stop codons were accepted, and the splice sites were required to follow the minimal consensus (GT at the 5’ end and AG at the 3’ end). Pseudo genes or entries marked as putative or predicted were excluded. Finally, the set was checked for multiple closely related entries, and those with more than 80 % identity as computed by the program BLAST (Altschul et al., 1990) were discarded. This set was used to train the GENIE gene finder (Reese et al., 2000b) for the annotation of the *Drosophila* genome.

The coding sequences were extracted as follows:

- The whole GENIE set of 1999 with genes containing at least two exons was split into three subsets with an equal number of genes each. By doing so, sequences from one gene are guaranteed to be part of only one subset.
- The exons were concatenated to form contiguous coding sequences.
- These complete CDSs were cut consecutively into 300 base pair long non-overlapping sequences. Shorter sequences and remaining sequences at the end were discarded.
- Because of the different length of the contiguous sequences, the files contained a different number of sequences (433–492). To ensure an equal amount of training/validation/test data for every cross-validation experiment, the number of sequences in each file was reduced to the smallest number (433). This was done by skipping parts of long sequences not to run into the danger of missing whole genes. This ensures that one can average over the results from different cross-validation experiments; especially the cross-correlation coefficient that is used as one criterion of success (see section 7.3) is known to depend on the ratio of positive and negative sample size (Baldi et al., 2000).

Some single ambiguous (i. e., non-ACGT characters) in the sequences were randomly replaced, as well as in the intron sequences that were extracted in the following way:

- Starting point was again the complete 1999 GENIE multiple exon genes set. Again, it was split in three subsets, each containing the introns from an equal number of genes.
- All introns were cut consecutively into 300 bp long non-overlapping sequences. Shorter sequences and remaining sequences at the end were discarded.
- Because of the different length and number of the intron sequences, the files contained a different number of sequences (121–136). Like above, the number of sequences in each file was reduced to the smallest number (121).

The promoter data was generated as follows:

- All sequences of the Drosophila Promoter Database (Arkhipova, 1995, Harvard University) were taken as a starting point.
- The subset marked as EPD drosophila entries were discarded, and all *Drosophila* entries of EPD release 63 independent subset (August 2000) were added.

- Because many entries did not contain sufficiently long sequences, they were aligned to the complete *Drosophila* genome (Altschul et al., 1990), and 300 base pair long sequences from -250 to +50 were taken from the genomic sequence. This left 247 entries. Some ambiguous nucleotide symbols were randomly replaced.
- These sequences were split into three cross validation files with 82 promoter sequences each.

Each part therefore contains three sequence files to be used for threefold cross-validation. The complements of coding and non-coding sequences are also used as data sets for non-promoters. As a whole, I refer to these data as *Drosophila* training set.

Genomic test sets. Apart from a high-quality set of sequences suited for training, a large set of promoter annotations in contiguous DNA sequences has to be collected. Only this enables us to assess the results of predictors within a real-world application. However, building such a set for the evaluation of transcription start site predictions or, more generally, for promoter recognition, is difficult. Here, the publication of high-quality annotations of a large contiguous genomic sequence, the *Adh* region (Ashburner et al., 1999), provided a most welcome opportunity to generate a high number of reliable annotations. Even for this well-studied region, almost no experimentally confirmed annotation of transcription start sites existed. As the 5' UTR regions in Drosophila can extend up to several kilobases, we could not simply use the region directly upstream of the start codon. To obtain the best possible approximation, we took the 5' ends of annotations from (Ashburner et al., 1999) where the upstream region relied on experimental evidence (the 5' ends of full-length cDNAs) and for which the alignment of the cDNA to the genomic sequence included a consistent annotation of the gene structure. The resulting set contains 92 genes out of the 222 original annotations. The 5' UTR of the 92 selected genes has an average length of 1,860 base pairs, a minimum length of 0 base pairs (when the start codon was annotated at the beginning, due to the lack of any further cDNA alignment information; this is very likely to be only a partial 5' UTR) and a maximum length of 36,392 base pairs. 17 genes had UTRs longer than 1000 base pairs. Two of the promoters were contained in our collection of training sequences described above. The set was used within the Genome Annotation Assessment Project (Reese et al., 2000a).

4.2 Vertebrate data sets

The vertebrate set was assembled in the same manner as the Drosophila set, but comprises a larger amount of sequences. It also consists of subsets of promoter, coding/exon, and non-coding/intron sequences, each of which is split in five equal-sized parts to be used for cross-validation.

For the CDS sequences, starting point were five out of seven files (with 330 out of 462 genes) from the 1998 GENIE human multiple exon genes set, extracted from Genbank version 105 according to the same criteria as the Drosophila set. The original idea was to leave some sequences aside for an independent evaluation of an integration into a gene finding system. The concatenated exons were again cut into 300 bp long consecutive non-overlapping sequences, and shorter sequences and remaining sequences at the end were discarded. Again, the parts contained a different number of sequences (178–192). The number of sequences in each file was reduced to the smallest number to have an equal amount of sequences in each part.

The intron sequences were also generated from five out of seven files from the 1998 GENIE multiple exon genes set by cutting them into 300 base pair long non-overlapping sequences. Shorter sequences and remaining sequences at the end were discarded. Because of the different length and number of the intron sequences, each part contained once more a different number of sequences (869–1722) and was therefore reduced to the smallest number.

The promoter data was exclusively taken from EPD, extracting all vertebrate sequences (except retroviruses) of the independent subset out of release 50 (575 sequences)[1]. Taking not only human but all vertebrate sequences is justified because of the highly similar transcription machinery. About half of the sequences were of human origin. Entries with less than 40 bp upstream and/or 5 bp downstream were discarded, leaving 565 entries. Out of these, 250 bases upstream and 50 bases downstream were extracted, resulting in 300 bases long sequences with flanking ambiguous nucleotides in some cases because of lacking data in the beginning and/or end of the promoter region. These ambiguous symbols were randomly replaced. The set was split in five subsets with 113 sequences each. As a whole, I refer to these data as human training set.

Genomic test sets. As test sets for the vertebrate model, two collections of well-mapped promoters were taken. One was used by Fickett and Hatzigeorgiou (1997) for an assessment of promoter prediction programs and contains 24 promoters in 18 rather short but well studied vertebrate sequences, ranging in length from 565 to 5,663 bp. The other one comprises 20 exactly mapped transcription start sites of the human cytomegalovirus (HCMV, Genbank entry X17403, 229,354 bp) and was kindly provided by Dr. Michael Winkler, then at the Institute of Clinical and Molecular Virology of the Universität Erlangen. Because viruses exploit the transcriptional machinery of the host cell, the vertebrate models can also be used to analyze the genomes of vertebrate viruses.

Additionally, the complete human chromosome 22 was scanned for promoters. The original

[1]Release 66 in April 2001 contained the only slightly larger number of 593 sequences.

Organism	**promoter**	**CDS**	**intron**
Drosophila	74,100	389,700	108,900
human	169,500	267,000	1,303,500

Table 4.1: **Amount of available training data** for the human and *Drosophila* models. Given is the total length in bases of all sequences for each of the three sequence groups.

Set	**Length**	**No. promoters**
Drosophila Adh	853,180	92
Human CMV	458,708	$<$ 202 (20 exact)
Fickett	66,240	24
Chromosome 22 known genes	12,348,750	339

Table 4.2: **Amount of available test data** for the human and *Drosophila* models. Given is the total length in bases of all sequences for each of the test sets, along with the number of promoters. The number for human CMV refers to the total number of genes, but the number of promoters is presumably lower as a considerable number of transcripts contains more than one gene. (The Fickett and HCMV sequences are evaluated on both strands, therefore the number is two times the number of base pairs mentioned in the text.)

annotation (Dunham et al., 1999) contains 545 genes, grouped into 247 known, 150 related, and 148 predicted genes, according to the information the annotation was based on. The exact TSS is known for only 20 genes, and two promoters were reported to be contained in EPD (Scherf et al., 2001). We used the revised annotation version 2.3 of May 2001, with 339 known, 112 related, and 109 predicted genes. Similar to the genomic set in *Drosophila*, we also evaluate only the known genes, which at least partly rely on cDNA alignments. Nevertheless, this set is of less quality, as we did not check whether the cDNA aligns to the 5’ end of the gene and includes the start codon. Tables 4.1 and 4.2 give a concise overview on all data sets.

This chapter concludes the first part of this thesis, which dealt with the background in biology and bioinformatics. Next, we turn to the computational part and start with the probabilistic models used to represent DNA sequence classes.

Chapter 5

Discrete Densities for Biopolymer Sequences

Sequence data has a number of inherent variations, even if the sequences are supposed to serve for the same biological purpose. Partly, this is a simple consequence of the way the data is generated: DNA sequencing is an error-prone process, and even though the sequencing projects circumvent this problem largely by providing a manifold coverage of the same sequence (Adams et al., 2000), errors cannot be ruled out completely. More important variations are related to the biological function. The general transcription machinery might be used throughout all species, and pol-II is involved in the transcription of all protein encoding genes, but every single gene needs to be activated in its very own specific manner.

It is therefore not surprising that a probabilistic modeling proved to be advantageous for pattern recognition problems in bioinformatics (Durbin et al., 1998; Baldi and Brunak, 1998). This chapter introduces suitable discrete models to represent promoter sequences: Markov chain models and stochastic segment models. Markov chains have been shown to be useful to model DNA sequences as a whole, without *positionally* conserved patterns. They are based on the occurrence of short symbol strings (typically, three to eight), which makes them a good model to capture transcription factor binding sites scattered throughout a eukaryotic promoter region. Following the Markov chains, stochastic segment models are described which provide a framework to combine several sub-models in a probabilistic way. This is especially suitable for modeling distinct parts of the core promoter (see section 3.2.3). The next chapter then discusses how continuous features related to the structure of DNA are computed and modeled by Gaussian densities. Throughout, I assume that the reader is familiar with the basics of probability calculus, and is referred to standard textbooks otherwise (Krengel, 2000).

5.1 Introduction

A probabilistic model M_k provides a representation of a class Ω_k, with K classes of interest, i. e. $k = 1 \dots K$. Given a feature vector as input, in our case a DNA sequence $\boldsymbol{w}$ with the nucleotides as features, it can be used to compute the likelihood P_k that the sequence belongs to class Ω_k, with

$$P_k(\boldsymbol{w}) := P(\boldsymbol{w}|\Omega_k). \tag{5.1}$$

For example, in this thesis models for promoters and non-promoters, e. g. coding and non-coding sequences, are considered.

A model contains a set of parameters Θ_k which are adjusted during training, using a set W of n samples. In the case of supervised training, the samples are known to belong to one class each, so W consists of disjoint subsets for each of the K classes ($W = \cup_k W^k$). Establishing a model therefore involves a choice of parameters, the structure or *topology* of the model, followed by the training of the parameters. Thereby, our goal is to obtain a set of parameters which results in the best possible recognition rate on the K classes under consideration. The choice of topology should lead to a model which is a good compromise between generalization and adaptation. Intuitively, a model with many parameters will give a good performance on the training set but will generalize poorly because it has over-adapted to the training samples. A small model, however, will not be able to fully capture the problem and therefore also perform poorly on a test set of unseen samples.

It is often not possible to train the parameters in such a way that the recognition rate is optimized directly. Therefore, an objective function R is used instead, and the parameters are set to values that lead to an optimum of this function applied to the training set. One well-known objective function is *Maximum Likelihood* (ML):

$$R^{ML}_{\Theta_k}(W^k) = \prod_{i=1}^{n_k} P_k(\boldsymbol{w}_{\boldsymbol{i}}^{\boldsymbol{k}}), \tag{5.2}$$

where n_k is the number of sequences in the training set W^k for class k ($\sum_k n_k = n$).

The ML objective function regards each class as independent of the others and aims at the maximization of the probability that the given training sample was generated, knowing to which class each sequence belongs. In the case that the models will be used for classification, it can be advantageous to employ *discriminative* objective functions where the emphasis is not put on the exact modeling of a class but on the correct classification of the samples. The Maximum Mutual Information (MMI) or Conditional Maximum Likelihood (CML) objective function (Bahl et al., 1986),

$$\begin{aligned} R_{\Theta}^{MMI}(W) &= \prod_{i=1}^{n} P(\Omega_{q_i}|\boldsymbol{w_i}) \\ &= \prod_{i=1}^{n} \frac{P(\boldsymbol{w_i}|\Omega_{q_i})P(\Omega_{q_i})}{\sum_j P(\boldsymbol{w_i}|\Omega_j)P(\Omega_j)}, \end{aligned} \tag{5.3}$$

is an example for such a discriminative function and maximizes the *a posteriori* probability of a class under the assumption that a pattern belonging to this class was observed. Here, q_i gives the number of the correct class for sequence $\boldsymbol{w_i}$, and $P(\Omega_j)$ denotes the *a priori* probability of class Ω_j. The objective function is not maximized for each class independently as with ML; instead, all samples are regarded together, and the likelihood of the model for the correct class is put in relation to the total likelihood obtained by all models together. Nadas et al. (1988) proved that MMI leads to better recognition results than ML in the case of limited sample size and wrong model assumptions.

Eddy et al. (1995) described a special case of MMI called Maximum Discrimination (MD) where models are trained according to MMI, but using only the samples from each class:

$$\begin{aligned} R_{\Theta_k}^{MD}(W^k) &= \prod_{i=1}^{n_k} P(\Omega_k|\boldsymbol{w_i^k}) \\ &= \prod_{i=1}^{n_k} \frac{P(\boldsymbol{w_i^k}|\Omega_k)P(\Omega_k)}{\sum_j P(\boldsymbol{w_i^k}|\Omega_j)P(\Omega_j)}, \end{aligned} \tag{5.4}$$

They describe an application on hidden Markov models for the classification of protein sequences; an application of MMI on the recognition of genes in DNA sequences was reported by Krogh (1997). The main reason why such discriminative functions have not replaced the ML estimation in general lies in the greatly increased complexity of the training procedure. Computation of the MMI objective function requires the application of *all* models on the *complete* sample set, and MD needs the application of *all* models on the subset of the respective class, whereas ML only requires to apply *one* model on its respective sample subset. When discriminative functions shall be applied to a complex task, a fixed background model is therefore often used instead, or the number of models to evaluate is limited to those concurrent models which are the most similar.

For ML, MMI, and MD, it is often possible to derive a closed or iterative solution to the parameter estimation problem for a given model; the choice of topology is however not straightforward, and in many cases solid heuristics are the only way to tackle this problem.

5.2 Stationary Markov chains

5.2.1 Basics

A particular type of probabilistic model that has turned out to be suited for a variety of applications in bioinformatics is the Markov chain model (for example, see the applications of Borodovsky et al. (1994); Audic and Claverie (1997); Ohler et al. (1999b)). This type of model is also popular in the field of speech recognition where it is usually referred to as *language model* and used to judge the reliability of word chains which are the result of acoustic speech recognition. Many of the literature references below come from this field, and good introductions were written for instance by Jelinek (1990) and Schukat-Talamazzini (1995).

Markov chains are motivated by the observation that the likelihood of a sequence $\boldsymbol{w} = w_1 \ldots w_T$, with w_i equal to a word v from a finite vocabulary V, can be decomposed into a product of conditional likelihoods as follows[1]:

$$P(\boldsymbol{w}) = P(w_1) \prod_{t=2}^{T} P(w_t | \underbrace{w_1 \ldots w_{t-1}}_{\text{context}}). \tag{5.5}$$

This equation shows that one symbol in a sequence is depending on all its predecessors, i. e. on the context of symbols observed so far. In a model with a fixed size, such a context of arbitrary length can certainly not be handled. A possible approximation of the probability $P(\boldsymbol{w})$ is therefore made by limiting the context length to N which is the basic idea of an Nth order Markov chain:

$$P(\boldsymbol{w}) \approx P(w_1) \prod_{t=2}^{T} P(w_t | w_{t-N} \ldots w_{t-1}) \tag{5.6}$$

In speech recognition, this Nth order Markov chain is called an $(N+1)$-gram language model. The model contains $|V|^{N+1}$ parameters — one for each possible word after each possible context of length N. The parameters for each context $\hat{\boldsymbol{v}} = v_1 \ldots v_N$ have to constitute a discrete probability distribution, following the constraints

$$\sum_{v \in V} P(v|\hat{\boldsymbol{v}}) = 1, \quad \forall v : P(v|\hat{\boldsymbol{v}}) \geq 0. \tag{5.7}$$

These parameters stay the same for the whole observation and do not change along the sequence; hence the name stationary Markov chain. Figure 5.1 gives an example of a first order Markov chain model for DNA sequences.

[1]For brevity and clarity, I will identify the outcome of an experiment with the random variable representing it.

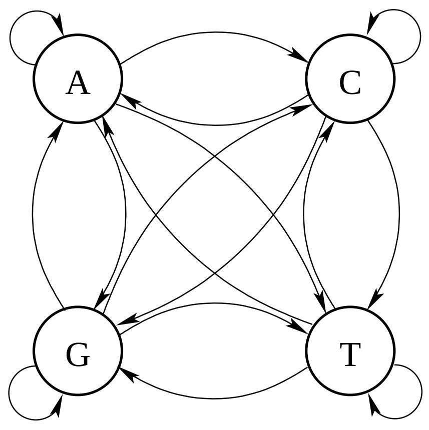


Figure 5.1: **Example of a first-order Markov chain model** for DNA sequences (after Durbin et al., 1998, p. 48).

For the ML estimation of the conditional probabilities, the relation $P(v|\hat{\boldsymbol{v}}) = P(\hat{\boldsymbol{v}}v)/P(\hat{\boldsymbol{v}})$ can be exploited. Given one training sequence $w_1 \dots w_T$, ML estimations of the probabilities on the right hand side are given by

$$\tilde{P}(v_1 \dots v_N) = \frac{\#(v_1 \dots v_N)}{T - N}, \tag{5.8}$$

with # denoting the absolute count of its argument in the sequence (Schukat-Talamazzini, 1995). An approximation of $P(v|\hat{\boldsymbol{v}})$ is then given by

$$\tilde{P}(v|\hat{\boldsymbol{v}}) = \frac{\tilde{P}(\hat{\boldsymbol{v}}v)}{\tilde{P}(\hat{\boldsymbol{v}})} \approx \frac{\#(\hat{\boldsymbol{v}}v)}{\#(\hat{\boldsymbol{v}})}, \tag{5.9}$$

which means that it is carried out by simply counting the substrings of length N and $(N+1)$ in the set of training sequences W. If a context appears at the very end of a sequence, it is not extended by any symbol. This means that the count for the context might be larger than the sum of all extensions by the letters from V, and it is the reason for equation 5.9 only being an approximation. To avoid the case that some distributions might not sum up to one, an alternative estimation therefore replaces the denominator by the sum over all counts with the same context:

$$\tilde{P}(v|\hat{\boldsymbol{v}}) = \frac{\#(\hat{\boldsymbol{v}}v)}{\sum_{v' \in V} \#(\hat{\boldsymbol{v}}v')} \tag{5.10}$$

A pitfall that needs to be avoided is that some of the parameters might be undefined or zero if a certain context $\hat{\boldsymbol{v}}$ or its extension by v is never observed in the training set. This has the consequence that the probability of a sequence in which such a problematic subsequence occurs

	CpG island				*non-CpG island*			
context	A	C	G	T	A	C	G	T
A	0.180	0.274	0.426	0.120	0.300	0.205	0.285	0.210
C	0.171	0.368	0.274	0.188	0.322	0.298	0.078	0.302
G	0.161	0.339	0.375	0.125	0.248	0.246	0.298	0.208
T	0.079	0.355	0.384	0.182	0.177	0.239	0.292	0.292

Table 5.1: **First order Markov chains of CpG islands and non-CpG islands**, parameters from Durbin et al. (1998).

is immediately set to zero. It can be avoided by smoothing techniques, the easiest of which is Jeffrey's discounting (also known as Laplace's rule, Schukat-Talamazzini, 1995; Durbin et al., 1998), where the counts of all parameters are increased by one. More sophisticated approaches modify the estimates either relative to the sample size or the non-smoothed parameter value. Finally, a less heuristic way replaces the discounting by a Dirichlet prior distribution (Krogh et al., 1994a):

$$\tilde{P}(v|\hat{\boldsymbol{v}}) = \frac{\#(\hat{\boldsymbol{v}}v) + \alpha \cdot m_v}{\sum_{v' \in V} \#(\hat{\boldsymbol{v}}v') + \alpha \cdot m_{v'}}, \tag{5.11}$$

with m_v the expected mean frequency of symbol v, and α the strength of the prior. Thus, prior information is explicitly included and the ML estimate becomes a maximum *a posteriori* (MAP) estimate. It is easy to see that Jeffrey's discounting, for example, is a special case of this approach with uniform expected frequencies and a prior weight equal to the size of the vocabulary.

Considering the model in figure 5.1, a simple application consists in the detection of CpG islands (see section 3.2.2). Durbin et al. (1998) give the ML probabilities for two Markov chains of first order, trained on 48 CpG island and other negative examples (table 5.1). Each row contains the probabilities conditioned on the same context — the base in the leftmost column — and the values therefore sum up to one. From the parameters in table 5.1, it becomes apparent that even at the level of first-order statistics, the parameters are considerably different. Note especially the remarkable under-representation of $P(G|C)$ in non-CpG islands.

For the first N symbols of a sequence, the context is not yet fully available. Therefore, a vector of probabilities to start with a certain context has to be provided. Alternatively, Markov chains of order zero to $N-1$ can be estimated on the same training set and then used to provide the probabilities with shorter context. Markov chains describe a probability distribution over sequences of the same length, i. e.

$$\sum_{\boldsymbol{w} \in V^T} P(\boldsymbol{w}) = 1, \quad T \in \mathbb{N}. \tag{5.12}$$

This also means that they describe a distinct distribution for *any* given sequence length, which will be advantageous in our application context. In an application such as a language model in speech recognition where the model is used to judge the likelihood of word chains of different length, a single distribution over all sequences of any length, i. e. over V^*, can be achieved by adding a distinct end symbol (see, e. g., (Durbin et al., 1998, chapter 3.1) or (Schukat-Talamazzini, 1995, chapter 7.2)).

As easy as the parameter estimation might be, Markov chains have a big disadvantage: The number of parameters increases exponentially when the context is extended. Therefore, this quickly leads to an over-fitting even though some of the parameters with extended context could possibly still be reliably estimated.

5.2.2 Interpolated Markov chains

One way to deal with parameter over-fitting, caused by increasing context size and parameter number, is an *interpolation* between Markov chains of different order. The basic idea of applying interpolation methods is to fall back on the probability estimation of subsequences shorter than $N+1$ if the frequencies of an oligomer $\hat{\boldsymbol{v}}v$ of size $(N+1)$ cannot be reliably estimated. In principle, interpolation leads us to a re-estimation of the initial parameter values (equation 5.9). Here, we will consider two different interpolation techniques. The first one is the *linear interpolation* (Jelinek, 1990) between all conditional probabilities with increasing context length up to N:

$$\begin{aligned} \hat{P}(v|\hat{\boldsymbol{v}}_1^N) := \rho_0 \frac{1}{|V|} \\ + \quad \rho_1 \tilde{P}(v) \\ + \rho_2 \tilde{P}(v|\hat{v}_N) \\ \vdots \\ + \rho_{N+1} \tilde{P}(v|\hat{\boldsymbol{v}}_1^N) \end{aligned} \tag{5.13}$$

The fraction $(1/|V|)$ accounts for unseen events and ensures that no probability is set to zero. Therefore, no additional parameter smoothing is needed as it is the case with Nth order Markov chains. $\hat{\boldsymbol{v}}_i^j$ is a shortcut for $\hat{v}_i \ldots \hat{v}_j$, and denotes the empty string if $i > j$. The coefficients ρ_i are non-negative values which sum up to one to guarantee that the new parameter values $\hat{P}(\cdot|\hat{\boldsymbol{v}})$ again form a probability distribution.

Setting all the weights $\rho_0 \dots \rho_N$ to zero and ρ_{N+1} to one is equal to an Nth order Markov chain. The models with linear interpolation are thus a straightforward generalization combining oligomer counts of different length. The advantage of interpolation is that the model can take into account statistics of a higher order without running into the danger of over-fitting the model to the training data.

There are some shortcomings of linear interpolation: Equation 5.13 contains only *one* vector of interpolation coefficients, whether all the subsequences up to length N really occur in the training data or not. Additionally, all parameters are treated equally, whereas the interpolation coefficient assigned to a parameter with a frequently occurring context should be larger than the coefficient for a rare event. By introducing an additional function $g(\hat{\boldsymbol{v}})$ which scores the reliability of the context $\hat{\boldsymbol{v}}$ monotonically, the linear interpolation can be extended to handle this problem accurately:

$$\hat{P}(v|\hat{\boldsymbol{v}}) := \frac{\sum_{i=0}^{N+1} \rho_i \cdot g(\hat{\boldsymbol{v}}_{N-i+2}^{N}) \cdot \ \tilde{P}(v|\hat{\boldsymbol{v}}_{N-i+2}^{N})}{\sum_{i=0}^{N+1} \rho_i \cdot g(\hat{\boldsymbol{v}}_{N-i+2}^{N})}, \tag{5.14}$$

This interpolation scheme is called *rational interpolation* (Schukat-Talamazzini et al., 1997). It overcomes the problems of linear interpolation by using the function $g(\boldsymbol{v'})$, which we chose to be a sigmoid function dependent of the frequency of $\boldsymbol{v'}$:

$$g(\boldsymbol{v'}) = \frac{\#(\boldsymbol{v'})}{\#(\boldsymbol{v'}) + C} \tag{5.15}$$

The shape of the sigmoid function is dependent on the constant bias C. In the case of $C = 0$, the function g is always equal to one and equation 5.14 becomes equivalent to linear interpolation. Also, with an increasing amount of training data, the bias C becomes less and less important; the rational interpolation thus has the largest impact if the training sample size is small.

Estimation of interpolation coefficients. We still lack the means to specify appropriate coefficients ρ_i for both linear and rational interpolation. To avoid over-fitting, optimal coefficients according to the ML objective function are calculated on a second disjoint part of the training sample. This step is called *validation* and is carried out after the initial estimation of the conditional probabilities (equation 5.9).

There is no closed solution for a maximum of the ML objective function in the case of interpolated Markov chains, but for the coefficients used in linear interpolation a local optimum can be found by application of the iterative Expectation Maximization (EM) algorithm. The Markov chain model parameters are left constant, and the coefficients are seen as *hidden variables*. Afterwards, a large weight will be assigned to those contexts which are reliable; if only sparse data are at hand, the weights belonging to short contexts will be increased. A more detailed treatment was given for example in (Schukat-Talamazzini, 1995, section 7.3.3) or (Hendrych, 1995).

For rational interpolation, the EM algorithm cannot be applied and the computation of locally optimal interpolation weights is carried out with a gradient descent algorithm instead. As the interpolated Markov chains are not in the main focus of this work, the detailed re-estimation formulas are omitted at this point and can be found in (Schukat-Talamazzini et al., 1997).

A different approach to the interpolation of Markov chains was applied on the parsing of microbial sequences by Salzberg et al. (1998a); in their case, the coefficients are calculated using a predefined function based on the χ^2 statistical test. Other objective functions such as MMI can also be used to estimate the coefficients (Warnke et al., 1999); but since the estimation of the Markov chain parameters themselves is still done according to ML (equation 5.9), different objective functions are used for different groups of model parameters.

5.2.3 Variable length Markov chains

A different way of facing the problem of exponential parameter growth is offered by the concept of variable length Markov chains (VLMCs), a generalization of the fixed-order Markov chains (Ron et al., 1996; Bühlmann and Wyner, 1999). VLMCs allow for parameters with variable context length to capture all significant symbol sequences in the most accurate way.

VLMCs can be represented as stochastic automata with one state per context, or in the form of *context trees*. A context tree is an acyclic graph whose nodes vary in degree between zero and $|V|$. The arcs leading from a node are labeled with the words $v \in V$, and each symbol is allowed to label at most one arc. The nodes are defined by pairs $(\hat{\boldsymbol{v}}, P(\cdot|\hat{\boldsymbol{v}}))$. $\hat{\boldsymbol{v}}$ is the string of labels on the path from the node *up* to a special empty root node e, and represents a context that is contained in the model. $P(\cdot|\hat{\boldsymbol{v}})$ is the probability distribution over the vocabulary, conditioned on that context.

Represented as a VLMC, a Markov chain of Nth order is a full tree of depth N. A small example of a more general context tree is given in figure 5.2.

Probability of a sequence. To illustrate how a VLMC is used to calculate the probability of a sequence, we use the tree in figure 5.2 to evaluate the sequence 01011. The probability is computed by applying the chain rule given in equation 5.5. For each symbol in the sequence, we determine the maximal context by going left in the sequence and down in the tree at the same time, according to the symbols we traverse. We stop when a leaf is reached — or when the first symbol of the sequence is hit — and look up the probability for the current symbol in this leaf. For each symbol, we have to start at the root again to determine the proper context. In our

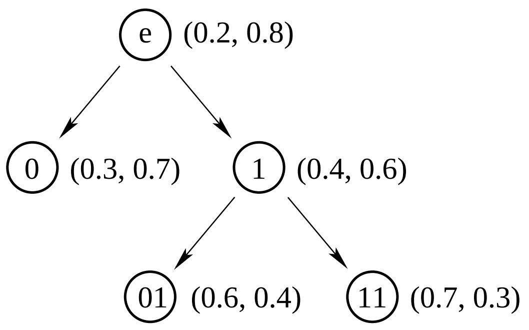


Figure 5.2: **Example of a context tree on the alphabet** $\{0, 1\}$, from Kulicke (2000). The probabilities of the symbols are given in parentheses next to the node where they are stored. An arc leading left corresponds to a transition with 0, one leading right to a transition with 1.

example, this leads to the following computation:

$$\begin{aligned} P_T(01011) &= P(0|e) \cdot P(1|0) \cdot P(0|01) \cdot P(1|0) \cdot P(1|01) \\ P_T(01011) &= 0.2 \cdot 0.7 \cdot 0.6 \cdot 0.7 \cdot 0.4 \end{aligned}$$

For the tree in figure 5.2, the context 0101 is equivalent to 01. This hints at an alternative view on context trees, namely that a given tree of maximum depth N defines a projection function c on the maximal context $\hat{\boldsymbol{v}}_1^{t-1}$ at any point t in a sequence with

$$c : \begin{cases} V^* \longrightarrow \cup_{m=0}^{N} V^m \\ \hat{\boldsymbol{v}}_1^{t-1} \mapsto \hat{\boldsymbol{v}}_{t-l}^{t-1} \end{cases}, \tag{5.16}$$
$$l = \min \left\{ k ; P(v_t|\hat{\boldsymbol{v}}_1^{t-1}) = P(v_t|\hat{\boldsymbol{v}}_{t-k}^{t-1}), \forall v_t \in V, 0 \leq k \leq N \right\},$$

Using c, the probability of a sequence generated by a VLMC can now be written as

$$P(\boldsymbol{w}) \approx \prod_{t=1}^{T} P(w_t|c(\boldsymbol{w}_1^{t-1})), \tag{5.17}$$

in analogy to the Nth order Markov chain in equation 5.6. As the distributions are not only contained in a leaf, but also in all inner nodes, no special care has to be taken for the beginning of sequences where the maximum context might not yet be available.

A representation equivalent to context trees are probabilistic suffix automata (PSA), a subclass of probabilistic finite automata (PFA). Here, the context is represented by a state of the automaton, with the restriction that no state label is a suffix of any other label, and that out of each state, a transition exists for all $v \in V$. For every context tree, there is an equivalent representation as PFA, and a PSA can be constructed if an additional property holds. Appendix B gives the detailed algorithm for conversion of a context tree into a PFA; as the probability calculation using a PFA is up to N times faster[2], such a conversion is advisable once a context tree has been established.

Estimation of VLMCs. In contrast to fixed-order Markov chains, VLMCs enable us to include important patterns with long contexts without an exponential growth of the number of parameters. The problem then remains how to decide in an automated manner which contexts should be included in the model, which means nothing less than a training of the model *structure*. To find the best among all possible model structures would leave us again with an exponential search problem, which is clearly not feasible. Ron et al. (1996) showed that the algorithm in figure 5.3 (Ron-Singer-Tishby (RST) algorithm) is able to learn a context tree according to the PAC (probably approximately correct) learning paradigm (Mitchell, 1997): With a likelihood of $1 - \delta$, the distribution of the learned VLMC M^L will have a Kullback-Leibler (KL) divergence D_{KL} from the correct underlying VLMC M^C of at most ϵ ($0 \leq \epsilon, \delta \leq 1$):

$$\frac{1}{T} D_{\text{KL}}[M^L][M^C] \leq \epsilon, \quad T > 0 \tag{5.18}$$

Thereby, the inequality holds for all possible sequence lengths T and is normalized by the length, as the likelihood generally decreases with increasing sequence length. The KL divergence between two (discrete) distributions P and Q on a set of observations X is defined by

$$D_{\text{KL}}[P][Q] := \sum_{x \in X} P(x) \log \frac{P(x)}{Q(x)}; \tag{5.19}$$

here, the set of observations is V^T.

The algorithm successively grows a tree up to the maximum depth N. Nodes depicting a context $\hat{\boldsymbol{v}}$ are added if they fulfill the following conditions:

1. The probability estimate $\tilde{P}(\hat{\boldsymbol{v}})$ is above a minimum probability.
2. For at least one symbol v, the probability conditioned on $\hat{\boldsymbol{v}}$ is significantly different from the father node with context $\text{suffix}(\hat{\boldsymbol{v}})$, or $\hat{\boldsymbol{v}}$ is a node on the path from $\boldsymbol{e}$ to another node which is significantly different.

The first condition keeps the tree from an exponential growth, as only nodes with a certain minimal probability are considered. The last condition assures that all nodes which are significantly different are indeed added to the tree, even if some ancestral nodes have almost the same

[2]The tree has to be traversed from root to leaf for every single symbol, which can take up to N accessions depending on the context.

```
Initialization: $\bar{M} = \{\boldsymbol{e}\}$, $\bar{S} = \{v \in V \mid \tilde{P}(v) \geq P_{\min}\}$
Building the context tree: WHILE $\bar{S} \neq \emptyset$
    Choose and delete an arbitrary context $\hat{\boldsymbol{v}}$ from $\bar{S}$
    IF $\exists\, v \in V: \quad \tilde{P}(v|\hat{\boldsymbol{v}}) \geq (1+\alpha)\gamma_{\min}$ and $\frac{\tilde{P}(v|\hat{\boldsymbol{v}})}{\tilde{P}(v|\text{suffix}(\hat{\boldsymbol{v}}))} \geq r$ or $\leq 1/r$
    THEN Add $\hat{\boldsymbol{v}}$ to $\bar{M}$, together with all $\hat{\boldsymbol{v}}'$ that lie on the path from $\boldsymbol{e}$ to $\hat{\boldsymbol{v}}$
    IF $|\hat{\boldsymbol{v}}| < N$
    THEN $\forall\, v' \in V$
        IF $\tilde{P}(v'\hat{\boldsymbol{v}}) \geq P_{\min}$
        THEN $\bar{S} = \bar{S} \cup \{v'\hat{\boldsymbol{v}}\}$
Smoothing: $\forall\, \hat{\boldsymbol{v}} \in \bar{M}$
    $\hat{P}(v|\hat{\boldsymbol{v}}) = (1 - |V|\gamma_{\min})\tilde{P}(v|\hat{\boldsymbol{v}}) + \gamma_{\min}$
```

Figure 5.3: **The RST algorithm** for the training of a context tree $\bar{M}$, after Kulicke (2000). For any context $\hat{\boldsymbol{v}} = \hat{v}_1 \ldots \hat{v}_l$, $\text{suffix}(\hat{\boldsymbol{v}}) = \hat{v}_2 \ldots \hat{v}_l$. Details see text.

probability distribution. As difference measure, the ratio between one child and its corresponding parent node parameter value is used. The algorithm has the following five parameters:

N :	maximal depth of the tree,
$P_{\min}$:	minimal probability of a context to be considered,
r :	measure for the difference of parent and child node probabilities,
$\gamma_{\min}$:	smoothing factor,
α :	minimal difference of any $P(v|\hat{\boldsymbol{v}})$ from the smoothing factor.

Ron et al. (1996) prove that the algorithm indeed fulfills the PAC criterion described above and runs in time polynomial in N, $|V|$, $1/\epsilon$, $1/\delta$, and the amount of observations in the training data T_{all}. In practice, though, the *exact* amount of training data needed is not known, and even if so, one could most probably not provide it, at least in the case of DNA sequence analysis. Nevertheless, Bejerano and Yona (2001) show that a reasonable choice of parameters can lead to good results in protein sequence classification. They also extend the algorithm to consider application specific background knowledge on proteins. In their application, α is set to zero; in addition to this simplification, we pursue Jeffrey's discounting as a smoothing approach, which leaves an effective number of three adjustable parameters: depth, cutoff, and minimal probability.

An alternative approach for the training of VLMCs was introduced by Bühlmann and Wyner

```
**Initialization:** $\bar{T} = \{\boldsymbol{e}\}$, $\bar{S} = \{v \in V \mid \tilde{P}(v) \geq P_{\min}\}$
**Building the context tree:** WHILE $\bar{S} \neq \emptyset$
    Choose and delete an arbitrary context $\hat{\boldsymbol{v}}$ from $\bar{S}$
    IF    $\Delta(\hat{\boldsymbol{v}}) \geq K$
    THEN  Add $\hat{\boldsymbol{v}}$ to $\bar{T}$, together with all $\hat{\boldsymbol{v}}'$ that lie on the path from $\boldsymbol{e}$ to $\hat{\boldsymbol{v}}$
    IF    $|\hat{\boldsymbol{v}}| < N$
    THEN  $\forall\, v' \in V$
              IF    $\tilde{P}(v'\hat{\boldsymbol{v}}) \geq P_{\min}$
              THEN  $\bar{S} = \bar{S} \cup \{v'\hat{\boldsymbol{v}}\}$
**Smoothing:** $\forall\, \hat{\boldsymbol{v}} \in \bar{T}$
    $\hat{P}(v|\hat{\boldsymbol{v}}) = \frac{1+\#(\hat{\boldsymbol{v}}v)}{|V|+\sum_{v' \in V} \#(\hat{\boldsymbol{v}}v')}$
```

Figure 5.4: **The BW algorithm** for the training of a context tree, after Kulicke (2000); compare with figure 5.3. Details see text.

(1999). They propose a training algorithm where the decision whether to extend the tree or not is not based on the difference of a single parameter, but on the whole distribution. The measure Δ employs the Kullback-Leibler divergence (equation 5.19), weighted by the absolute count of the context (see figure 5.4, the Bühlmann-Wyner (BW) algorithm):

$$\Delta(\hat{\boldsymbol{v}}) = D_{\text{KL}}[P(\cdot|\hat{\boldsymbol{v}})][P(\cdot|\text{suffix}(\hat{\boldsymbol{v}}))] \cdot \#(\hat{\boldsymbol{v}}) \tag{5.20}$$

Bühlmann and Wyner (1999) show that the algorithm is consistent if the cutoff value K on the distance Δ follows

$$K \sim C \cdot \ln(T_{\text{all}}), \quad \text{with} \quad C > 2 \cdot |V| + 4 \tag{5.21}$$

which means that the cutoff should be chosen proportional to the amount of training data at hand and the size of the vocabulary. Again, there is no closed solution for the optimal value for K; as a rule of thumb for the magnitude of K, Bühlmann (2000) uses the $\chi^2/2$ quantiles $\chi^2_{|V|-1;0.9}/2$ and $\chi^2_{|V|-1;0.8}/2$. The parameter $P_{\min}$ is only crucial for preventing the algorithm to examine too many nodes; as the count of the context appears in the distance measure, the minimal probability does not have a large influence on the final topology of the tree. This is not the case for the distance measure r used in the RST algorithm.

Cross-validation estimation of optimal cut-off. Both the BW and the RST algorithm do not provide an estimate of the cutoff r respectively K that delivers the "best" VLMC fitted to the training observations. But as Bühlmann (2000) points out, the model that comes closest to the true distribution might not even be the one that is desired. Often, the aim is to find a model that optimizes a global risk or an objective function, such as ML and MMI described above in section 5.1. A feasible way to find such a (sub)optimal tree is to restrict the search to the trees generated by a number of different cut-offs K, and choose the one for which the optimal value is achieved. Note that this algorithm is not guaranteed to deliver the best tree as there might be better trees that are not generated by one overall cut-off for all leaves of the tree.

Therefore, we are left with the problem of estimating the objective function for different K (or r and $P_{\min}$) values. Bühlmann (2000) addresses the problem by means of a Metropolis sampling scheme: He estimates an initial tree, uses it to generate a large number of independent samples, trains trees on each of those samples and computes the average risk over the trees within a certain range of K. In this work, a computationally less demanding cross-validation scheme is used instead. The training data is randomly divided into m equal parts, and $m-1$ of them are used to estimate the tree. The objective function is then evaluated on the remaining disjoint part for different values of K. This is repeated m times, and the ML or MMI estimate is given by the mean value over the m experiments. The final tree is then constructed using the complete set of training samples and the cut-off for which the optimal value was found. Figure 5.5 summarizes this. As mentioned above, the χ^2 quantiles are suitable as estimates for $K_{\min}$ and $K_{\max}$. To apply the algorithm on a discriminative measure, background models such as Markov chains of fixed order can be used. Alternatively, optimal cutoff values for all models together could be determined, at the possible cost of a multidimensional exhaustive search. This is certainly only practical for small numbers of data and classes. Note that the algorithm uses K as an example parameter, but can be also employed for other parameters such as the upper context length or the ratio r.

5.2.4 Discriminative estimation of Markov chain parameters

By now, we have discussed two approaches to improve the modeling of sequences with Markov chains: Interpolation of Markov chains of different order, and the explicit representation of variable context. Both address the problems caused by the exponential parameter growth with increasing context length, and in the case of VLMCs explicitly deal with the topology of a model.

The following section concerns the estimation of parameters themselves. As mentioned in the beginning of this chapter, parameters can be estimated according to different objective functions. The ML estimate of Markov chain parameters was already given above in equation 5.9, and we

```
divide the training sample into equally sized disjoint parts $W[0], W[1] \dots W[m-1]$
FOR $K = K_{\min}$ TO $K_{\max}$
    set $b = 0$
    FOR $i = 0$ TO $m - 1$
        $X = W[i \bmod m]$
        $W = W[(i+1) \bmod m] + \dots + W[(i+m-1) \bmod m]$
        Build the context tree using $K$ and the samples in $W$
        $b = b + R(X)$
    estimate of objective function: $R[K] = b/m$
$K_{\text{opt}} = \text{argmax}_K \, R[K]$
```

Figure 5.5: **Algorithm for the estimation of objective functions** R by means of cross-validation.

will now derive the parameter estimation for MMI and MD as well (Ohler et al., 1999a).

Assuming that we have one training sequence $\boldsymbol{w_i}$, the partial derivation of the logarithm of the MMI objective function (equation 5.3) with respect to a parameter $P_k(v|\hat{\boldsymbol{v}})$ of (variable length) Markov chain M_k for class Ω_k leads us to

$$\begin{aligned}
\frac{\partial \log R_{\Theta}^{MMI}(\boldsymbol{w_i})}{\partial P_k(v|\hat{\boldsymbol{v}})} &= \frac{\partial}{\partial P_k(v|\hat{\boldsymbol{v}})} (\log P(\boldsymbol{w_i}|\Omega_{q_i}) P(\Omega_{q_i}) - \log \sum_j P(\boldsymbol{w_i}|\Omega_j) P(\Omega_j)) \\
&= \frac{\#(\hat{\boldsymbol{v}}v)}{P_k(v|\hat{\boldsymbol{v}})} \delta_{k,q_i} - \frac{\#(\hat{\boldsymbol{v}}v)}{P_k(v|\hat{\boldsymbol{v}})} \cdot \frac{P(\boldsymbol{w_i}|\Omega_k) P(\Omega_k)}{\sum_j P(\boldsymbol{w_i}|\Omega_j) P(\Omega_j)} \\
&=: \frac{1}{P_k(v|\hat{\boldsymbol{v}})} \left(\#_{k,q_i}(\hat{\boldsymbol{v}}v) - \#'(\hat{\boldsymbol{v}}v)\right) \qquad (5.22)
\end{aligned}$$

where δ_{k,q_i} is equal to one if $q_i = k$ and zero otherwise. $\#'$ is a weighted counting function, and $\#_{k,q_i}$ is a function which counts only if $q_i = k$. Setting the right hand side to zero and solving for $P_k(v|\hat{\boldsymbol{v}})$ does not lead to a closed solution as in the case of the ML estimation. Instead, we are left with the task to iteratively optimize the objective function using the gradient given above.

We do not use a standard gradient descent technique but rather follow the approach described by Normandin and Morgera (1991) who use MMI to train hidden Markov models for spoken digit recognition. They carry out the parameter optimization with a re-estimation formula for rational objective functions such as MMI:

$$\tilde{P}_k(v|\hat{\boldsymbol{v}}) = \frac{P_k(v|\hat{\boldsymbol{v}})\left(\frac{\partial \log R_\Theta^{MMI}(W)}{\partial P_k(v|\hat{\boldsymbol{v}})} + D\right)}{\sum\limits_{v_j \in V} P_k(v_j|\hat{\boldsymbol{v}})\left(\frac{\partial \log R_\Theta^{MMI}(W)}{\partial P_k(v_j|\hat{\boldsymbol{v}})} + D\right)} \tag{5.23}$$

For a sufficiently large constant D, the convergence to a local optimum was proven by Gopalakrishnan et al. (1989). Because the theoretical bound on D leads to a slow convergence in practice, we choose D to be equal to

$$D = \max_{v_j \in V}\left\{-\frac{\partial \log R_\Theta^{MMI}(W)}{\partial P_k(v_j|\hat{\boldsymbol{v}})}, 0\right\} + \epsilon \tag{5.24}$$

which then guarantees that the new parameters fulfill the conditions of a probability distribution. ϵ is a small positive constant. The division by parameter $P_k(v|\hat{\boldsymbol{v}})$ in the MMI derivation (equation 5.22) are an inherent cause for numerical instability for low-valued parameters. Incidentally, these are also parameters which are likely to be unreliably estimated. Thus, Merialdo (1988) replaced the original value of the partial derivation by

$$\frac{\partial \log R_\Theta^{MMI}(W)}{\partial P_k(v|\hat{\boldsymbol{v}})} \approx \frac{\#_{k,q_i}(\hat{\boldsymbol{v}}v)}{\sum\limits_{v_j \in \mathcal{V}} \#_{k,q_i}(\hat{\boldsymbol{v}}v_j)} - \frac{\#'(\hat{\boldsymbol{v}}v)}{\sum\limits_{v_j \in \mathcal{V}} \#'(\hat{\boldsymbol{v}}v_j)} \tag{5.25}$$

to remove emphasis from low-valued parameters, concentrate on the important high-valued parameters and thus achieve a more stable convergence.

Maximum Discrimination estimation. Eddy et al. (1995) derive estimates for HMM parameters according to the MD objective. When we calculate the derivation of the MD objective function (equation 5.4) for a Markov chain M_k, we get a special case of the MMI derivative:

$$\frac{\partial \log R_{\Theta_k}^{MD}(\boldsymbol{w_i})}{\partial P_k(v|\hat{\boldsymbol{v}})} = \frac{\#(\hat{\boldsymbol{v}}v)}{P_k(v|\hat{\boldsymbol{v}})}\left(1 - R_\Theta^{MMI}(\boldsymbol{w_i})\right), \tag{5.26}$$

because $\delta_{k,q}$ (equation 5.22) is always equal to one, and the negative weight term is equal to the MMI objective function. We follow the approach of Eddy et al. (1995) and introduce the condition that all parameters belonging to the same distribution (i. e., to the same context) must sum up to one with the help of Lagrange multipliers. This leads us to an Expectation-Maximization style re-estimation formula for the parameters:

$$\tilde{P}_k(v|\hat{\boldsymbol{v}}) = \frac{\#(\hat{\boldsymbol{v}}v)(1 - R_\Theta^{MMI}(\boldsymbol{w_i}))}{\sum\limits_{v_j \in \mathcal{V}} \#(\hat{\boldsymbol{v}}v_j)(1 - R_\Theta^{MMI}(\boldsymbol{w_i}))} \tag{5.27}$$

The values on the right side are calculated using the parameters of the last iteration. If we have n training sequences, the numerator and denominator sum up over all of them. Once initialized with values greater than zero, the parameters will always be greater than or equal to zero, thus fulfilling all characteristics of a probability distribution. To ensure that no models parameters are set to zero during the iterations, the counts on the right hand side are modified *ad hoc* by Dirichlet priors (see equation 5.11 above), for example using the estimated *a priori* probabilities of $P(v), v \in V$.

A closer look at equation (5.27) shows that MD can be regarded as nothing else than a weighted version of ML estimation where the training sequences have weights dependent on how bad they are recognized by the correct model.

Corrective training, validation, and model interpolation. MMI and MD lead to iterative formulas for parameter estimation, which leaves us with the problem of appropriate start values. Throughout this thesis, we use the standard ML estimates for the initialization. The iterative estimation worsens the already mentioned higher complexity of the discriminative approaches. Normandin and Morgera (1991) therefore propose *corrective training*, i. e. a training where only the misclassified sequences of the last iteration are part of the actual training set. This is justified by the observation that well recognized sequences do not contribute much to the derivation (equation 5.22) and can thus be left away without much harm. Corrective training improves drastically on the speed of an iteration, as only a fraction of the sequences has to be taken into account.

To avoid oscillatory effects during the course of training, it is necessary for both MMI and MD to perform an interpolation between the model before and after an estimation iteration. One possible way to do so is to assign a class-dependent weight to the updated parameters which depends on the classification performance of the old model (Normandin and Morgera, 1991). Additionally, this weight might decline logarithmically with the number of iterations (Ohler et al., 1999a). Another possibility is to use large uniform weights (such as 0.98 or 0.99) for the old model (Eddy et al., 1995). For MMI, the oscillatory effects are certainly partly due to the deviation from the cases with proven convergence (value of D in equation 5.23, approximation of the derivation in equation 5.25). For MD, the convergence of the re-estimation formula has not been explicitly shown so far. In any case, the corrective training might add to oscillatory effects, as the training set and therefore the counts which are considered might drastically change from iteration to iteration.

Finally, instead of iterating the training process until convergence is reached, we use a disjoint validation set to determine a suitable point to stop. As soon as the goal function does not improve on the validation set any longer, we finish the estimation to prevent the models from over-adaptation.

5.3 Stochastic segment models

The structure of eukaryotic promoters as described in section 3.2.1 is too complex to be adequately reflected in a single Markov chain model. The nucleotide statistics within the upstream region is obviously very different from the TATA box or the initiator site. The formalism of *stochastic segment models,* sometimes also called *generalized hidden Markov models,* which is described in this section, provides us with the necessary means to model promoters as a sequence of segments generated by different probability distributions.

5.3.1 From hidden Markov to segment models

A (discrete) hidden Markov model (HMM) generates a symbol sequence by a double process, instead of the single stochastic process like the Markov chains described above. It consists of a finite set of states $Q = \left\{q^1, \dots q^L\right\}$, each of which contains a probability distribution over the set of symbols V.

The first process generates a sequence of states $q_1 \dots q_T$, following the *Markov property* that the next state is only depending on the current state:

$$P(q_t | q_1 \dots q_{t-1}) = P(q_t | q_{t-1})$$

These probabilities $P(q_t | q_{t-1})$ are called *transition probabilities,* and constitute a stationary Markov chain of first order on the sequence of states. By convention, they are given in the $L \times L$ matrix $\boldsymbol{A}$, with $a_{ij} := P(q^j | q^i)$. Because we lack the necessary context at the beginning of a sequence, a vector of *start probabilities* is also needed. This vector is usually denoted by $\boldsymbol{\pi}$, with $\pi_i := P(q_1 = q^i)$. The sequence of states is not visible to the observer, and we therefore speak of a *hidden* Markov model.

The second process generates the symbols w_t from the vocabulary that are visible to the outside. Each state contains a distinct probability distribution on the vocabulary, and the probability is only conditioned on the current state:

$$P(w_t | w_1 \dots w_{t-1}; q_1 \dots q_t) = P(w_t | q_t)$$

These probabilities are provided by the distribution $b_{q_t}(w_t)$ and are arranged in the $L \times |V|$ matrix $\boldsymbol{B}$, with $b_{ij} = P(v^j | q^i)$. Therefore, an HMM M is fully specified by

$$M = (\boldsymbol{\pi}, \boldsymbol{A}, \boldsymbol{B})$$

If we assume a specific path $\boldsymbol{s} = s_1 \dots s_T, s_i \in \{1, \dots, |Q|\}$, through the model states q^{s_i}, the probability that it produces a sequence $\boldsymbol{w} = w_1 \dots w_T$ on this path is

$$P(\boldsymbol{w}, \boldsymbol{s}) = \pi_{s_1} \prod_{t=1}^{T-1} b_{s_t}(w_t) a_{s_t s_{t+1}} \cdot b_{s_T}(w_T), \tag{5.28}$$

and the total probability for a sequence under the model is obtained by marginalization, i. e. summation over all paths:

$$P(\boldsymbol{w}) = \sum_{\boldsymbol{s}=s_1 \ldots s_T} P(\boldsymbol{w}, \boldsymbol{s}) \tag{5.29}$$

Like a Markov chain, an HMM represents a probability distribution over all sequences of the same length. Over the last 20 or so years, HMMs have become the state-of-the-art models for acoustic speech recognition (Rabiner, 1989; Niemann, 1990). Starting in 1992, their application for biosequence analysis, namely of protein sequence families, was pioneered by a number of independent research groups (Krogh et al., 1994a; Baldi et al., 1994; Eddy, 1995). Shortly after, they were also employed to model eukaryotic promoters (Ohler, 1995; Pedersen et al., 1996), although no complete system for prediction in genomic sequences was built. Figure 5.6 shows a popular HMM topology for protein families, using match, insert, and delete states: The most common amino acids are modeled with match states; insert states allow for the addition of amino acids between match states; and delete states are silent states (i. .e., without an output distribution) that serve to leave out single amino acids or whole protein domains. Libraries such as Pfam (Bateman et al., 2000) contain a large collection of HMMs for protein families; if a new protein sequence with unknown function is determined, its probability to belong to one of the families can be computed with the models from these libraries. As another example, the two Markov chain models for CpG islands and non-CpG islands given in table 5.1 can be combined into one HMM which can be used to parse a DNA sequence into the classes CpG or non-CpG island (see the Viterbi algorithm below). Such a model consists of two sub-models as in figure 5.1, with additional transitions from all states within one sub model to all within the other one. Depending on which state is traversed when parsing a DNA sequence, each nucleotide can be labeled as belonging or not belonging to a CpG island.

HMMs are an adequate representation for positionally conserved patterns. They can be regarded as a straightforward extension of weight matrix models for patterns: Every position has a discrete distribution on its own, but HMMs allow for arbitrary state transitions, whereas weight matrices correspond to a strictly linear model topology. Markov chains, in comparison, are suited to model whole regions and do not capture the position of a pattern. On the other hand, they do not assume independence among the observed symbols in the pattern like HMMs do. Interestingly, each model can be transformed into the other one, though at the price of a much larger model: A Markov chain can be represented as a special HMM where each context in the MC is

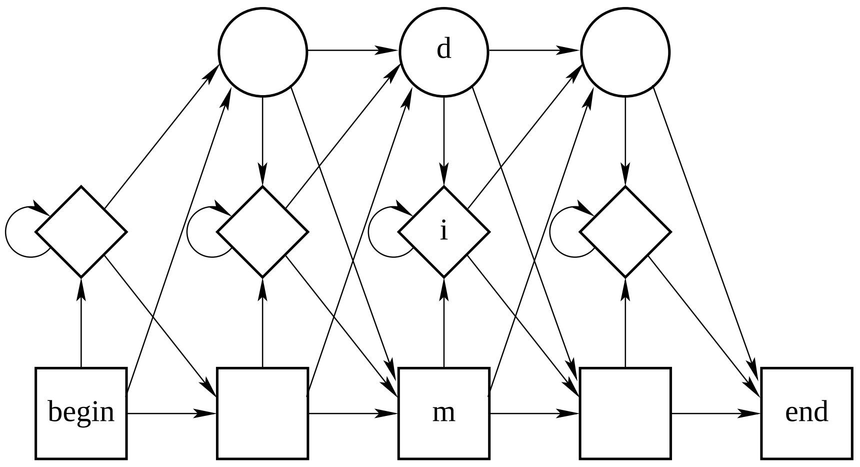


Figure 5.6: **Structure of HMMs for protein families.** Match states are at the bottom, diamonds are used to indicate insert states, and circles for delete states; after Krogh et al. (1994a).

identified with a state in the HMM, the distribution on the context is replaced by the corresponding transition probabilities, and the output distribution is zero for all entries but the last letter of the context, which is set to one. In the other direction, an HMM can be modeled by an MC which encodes paths through the model as contexts.

An HMM state emits only a single symbol every time it is visited. Of course, more than one symbol can be emitted from the same state if it has a transition probability to itself which is larger than zero. This has the direct consequence that the probability $d_i(\tau)$ to stay in state q^i for τ time points underlies the geometrical distribution

$$d_i(\tau) = a_{ii}^{\tau-1} \cdot (1 - a_{ii}) \tag{5.30}$$

If this is known to be not the case, an explicit duration distribution $d_i(\tau)$ can be included in every state of an HMM. This model topology is known as *hidden semi-Markov model* (Rabiner and Juang, 1993). To be practically useful, the (discrete) distribution d_i has a lower and upper limit on the non-negative probabilities. The probability to generate a partial sequence $\boldsymbol{w_i} = w_1 \dots w_\tau$ of length τ by state q^j is then given by

$$P_j(\boldsymbol{w_i}, \tau) = d_j(\tau) \prod_{t=1}^{\tau} b_j(w_t) \tag{5.31}$$

With this extension, the limitation of geometrically distributed durations is abolished, but at a computationally higher price: The computation of the total likelihood in equation 5.29 also has to take all possible combinations of durations into account. In the following, we omit the length

τ on the left-hand side, as it is implicitly captured by the length of the sequence w_i.

The consecutive symbols that are produced by the same state of a hidden semi-Markov model are still independent from each other. If known dependencies among symbols shall be taken into account, the modeling approach can be extended further: The discrete output distribution $b_i(v)$ of state q^i which is defined over the vocabulary V is replaced by an arbitrarily complex distribution $b_i(\boldsymbol{w}|\tau)$ that generates a number of consecutive symbols, a whole *segment*, and is conditioned on the length of the segment. The probability $P_j(\boldsymbol{w_i})$ that a state produces a partial sequence $\boldsymbol{w_i}$ of length τ_i is given by

$$P_j(\boldsymbol{w_i}) = d_j(\tau_i) \cdot b_j(\boldsymbol{w_i}|\tau_i). \tag{5.32}$$

With a given valid segmentation $(\boldsymbol{s}, \boldsymbol{\tau}) = ((q^{s_1}, \tau_1) \ldots (q^{s_m}, \tau_m))$ of sequence $\boldsymbol{w}$ into segments $\boldsymbol{w_i}$, $\sum_i \tau_i = |\boldsymbol{w}|$, the probability of the sequence can be expressed as

$$P(\boldsymbol{w}, \boldsymbol{s}, \boldsymbol{\tau}) = \pi_{s_1} \prod_{i=1}^{m-1} P_{s_i}(\boldsymbol{w_i}) a_{s_i s_{i+1}} \cdot P_{s_m}(\boldsymbol{w_m}), \tag{5.33}$$

in analogy to the HMM equation 5.28. A walk through such a *stochastic segment model* (SSM) therefore looks as follows:

1. Choose an initial state q_1, according to the start probability vector $\boldsymbol{\pi}$.
2. Decide on the length τ_i of the current segment $\boldsymbol{w_i}$, according to the duration distribution d_j of state q_j.
3. Generate the sequence segment $\boldsymbol{w_i}$ using the output distribution b_j.
4. Choose the next state according to matrix $\boldsymbol{A}$.
5. Repeat steps 2–4 until the sequence length T is reached.

Because of the arbitrary duration distributions, an SSM does *not* constitute a probability distribution on the sequences of the same length any more, as HMMs and MCs do. Burge (1997) notes that an SSM defines a probability measure on the *joint* probability of sequence and parse, as given in equation 5.33.

The output distribution b_j can itself be arbitrarily complex and take into account dependencies between symbols within a segment. Depending on the field of application, different distributions such as Markov chains or HMMs may be suitable. Because the output distribution is conditioned on the duration, we either have to provide an individual distribution for each possible segment length, a mapping function from various segment lengths to a limited number, or the distributions have to be able to generate sequences of all valid lengths. This is one of the advantages of Markov chain and hidden Markov models (without explicit end state, see section 5.2): they constitute a probability distribution on all sequences of the same length; therefore, only one model has to be provided no matter which value τ might be set to. Also, the complex sub-models lead to a further increased computational complexity: Apart from the summation over all possible segment lengths, the computation of the probability of a segment by b_j has to be carried out for each of these segment lengths.

HMM model extensions such as SSMs were pioneered in the field of speech recognition, where the sub-models are usually employed to take correlations among neighboring feature vectors into account. Depending on the sub-models and the way how feature sequences of varying length are mapped onto them, a large number of slightly different approaches were published; Stemmer (1999) and Ostendorf et al. (1996) provide concise overviews. Here, we use the term *stochastic segment models* (SSMs) pioneered by Ostendorf et al. (1996).

SSMs are not new to the field of DNA sequence analysis – gene finding systems which make use of stochastic models mostly fit into the framework of SSMs. Especially the GenScan system (Burge and Karlin, 1997) uses a model structure as described above[3] (see the GenScan model in figure 2.4 as an example for an SSM). Promoter modeling with SSMs was first described in Stemmer (1999); Ohler et al. (2000).

5.3.2 Algorithms for training and evaluation

After the outline of HMMs and their extension to SSMs, I now describe algorithms to train these models and use them to calculate the likelihoods of sequences. All algorithms are specified for SSMs; the corresponding algorithms of HMMs are simpler versions without duration distributions and can be taken from the literature (Niemann, 1990; Rabiner and Juang, 1993).

If we have an advance annotation of the training material, a supervised and individual learning of each output, duration and transition distribution is possible. This is the case e. g. for gene finding systems, where the data base entries contain exon and intron locations. We therefore assume at first that the model is already given, and describe the algorithms to compute the total likelihood as well as the most likely state sequence.

The forward algorithm. The probability of generating sequence $\boldsymbol{w}$ with a segment model is equal to the sum of all possible segmentations over which the sequence can be produced. Thus, using equation 5.33 from above, we have

[3]Burge uses the term hidden semi-Markov model; according to the terminology used in this work, it corresponds to an SSM.

```
$\boldsymbol{F} := 0$
FOR $t := 1$ TO $T$
    FOR $j := 1$ TO $L$
        FOR $t' = 0$ TO $t-1$
            IF $t' = 0$
            THEN $sum := \pi_j$
            ELSE $sum := 0$
                 FOR $i := 1$ TO $n$
                     $sum := sum + F_{t',i} \cdot a_{ij}$
            $F_{t,j} := F_{t,j} + sum \cdot P_j(w_{t'+1}, .., w_t)$
$P(\boldsymbol{w}) := \sum_{j=1}^{n} F_{T,j}$
```

Figure 5.7: **Forward algorithm for segment models**, from Ohler et al. (2000). The input is $\boldsymbol{w} = w_1, .., w_T$. The matrix $\boldsymbol{F}$ contains the forward variables, L is the number of states. t is the current time, j the current state, and t' is the time where the state transition from state q_i to q_j takes place.

$$P(\boldsymbol{w}) = \sum_{\boldsymbol{s}} \sum_{\boldsymbol{\tau}} P(\boldsymbol{w}, \boldsymbol{s}, \boldsymbol{\tau}) \tag{5.34}$$

Instead of literally summing over all individual summands that are computed independently from each other, the corresponding probability can be computed efficiently by the *forward algorithm*, given in figure 5.7. This algorithm calculates the forward variables $\alpha_{t,j}$ which contain the probability that the model is in state q_j at time t and has so far produced the symbol chain $w_1 \ldots w_j$. The summation over all states at the end of the observation then gives us the total likelihood over all possible segmentations.

The matrix containing the forward variables is initialized with the vector of start probabilities times the probabilities to generate the first symbol. The algorithm then fills this matrix from the beginning to the end of the sequence, regarding each of the states at each position. In contrast to HMMs where a state transition happens after each symbol, the state duration in SSMs is variable, so we have to sum up over all preceding time points where a state transition was possible. Figure 5.8 exemplifies this.

The complexity of the forward algorithm for segment models involves $\frac{T \cdot (T-1)}{2} \cdot L$ calls to the segment probability P_j, which is the part consuming most of the runtime. In comparison, an HMM only needs $T \cdot L$ calls, as it does not require to take all possible segment lengths into

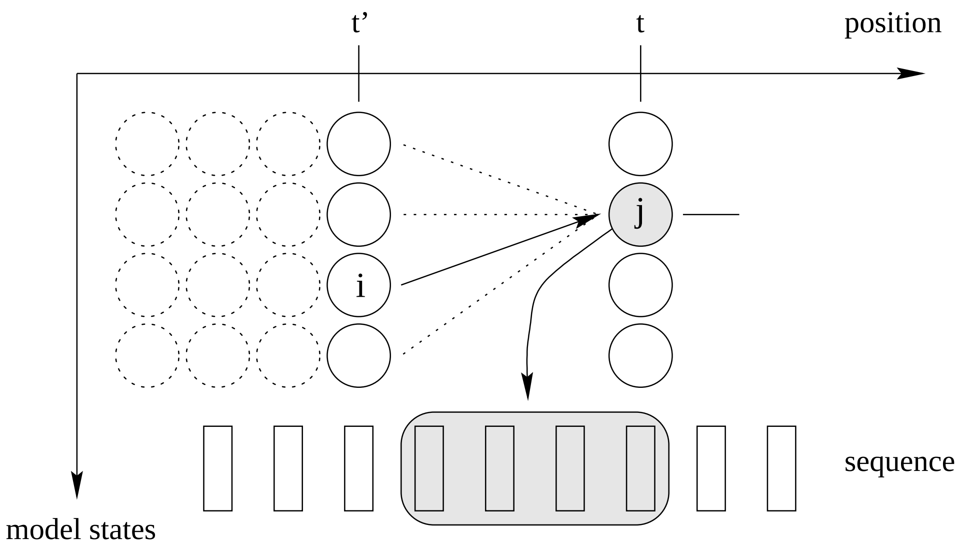


Figure 5.8: **Schematic view on the forward algorithm**, from Stemmer (1999). The figure exemplifies how the matrix containing the forward variables is computed. At the current time t, state q^j is considered. To obtain the forward variable, an iteration over all possible state durations of state q^j is carried out, summing up over all possible previous states q^i.

account[4]. The evaluation of the forward algorithm thus involves many computations of the output distributions b_j, and has the consequence that we can only make use of distributions which can be computed efficiently. The next section describes this further.

The Viterbi algorithm. The forward algorithm computes the likelihood over *all* possible segmentations, but the underlying application often calls for the *best* segmentation,

$$\hat{\boldsymbol{s}}, \hat{\boldsymbol{\tau}} = \operatorname*{argmax}_{\boldsymbol{s},\boldsymbol{\tau}} P(\boldsymbol{s}, \boldsymbol{\tau} | \boldsymbol{w}), \tag{5.35}$$

as a result, plus the likelihood obtained on it. For example, the user of GenScan is not interested in the total likelihood of a complete genomic sequence under the model; instead, he wants to know where the exons and introns are most possibly located.

For this goal, we can replace equation 5.35 using Bayes' theorem:

$$\hat{\boldsymbol{s}}, \hat{\boldsymbol{\tau}} = \operatorname*{argmax}_{\boldsymbol{s},\boldsymbol{\tau}} P(\boldsymbol{s}, \boldsymbol{\tau} | \boldsymbol{w}) = \operatorname*{argmax}_{\boldsymbol{s},\boldsymbol{\tau}} \frac{P(\boldsymbol{s}, \boldsymbol{\tau}, \boldsymbol{w})}{P(\boldsymbol{w})} = \operatorname*{argmax}_{\boldsymbol{s},\boldsymbol{\tau}} P(\boldsymbol{s}, \boldsymbol{\tau}, \boldsymbol{w}) \tag{5.36}$$

[4]Very often, the complexity of the HMM forward algorithm is given as $T \cdot L^2$ operations which refers to the matrix accesses instead of calls to the probability function.

This most likely segmentation can be computed using the Viterbi algorithm, in which the sum over all possible segmentations that is computed by the forward algorithm is replaced by its maximum. It is given in figure 5.9 and is a realization of the efficient *dynamic programming* (DP) algorithm. DP can be applied to find the best path in a graph if the principle of optimality holds. It states that each sub-path on the optimal path through the graph will also be optimal if a monotonous and separable cost function is applied. In our case, the graph is given by the matrix of states and symbols. The principle holds for segment models because the probability calculation fulfills the following condition (Stemmer, 1999):

$$\max_{\substack{\boldsymbol{s}=s_1,..,s_n\\ \boldsymbol{\tau}=\tau_1,..,\tau_n}} P(\boldsymbol{w}|\boldsymbol{s},\boldsymbol{\tau}) = \max_{\substack{\boldsymbol{s}'=s_1,..,s_{n-1}\\ \boldsymbol{\tau}'=\tau_1,..,\tau_{n-1}}} P(\boldsymbol{w}'|\boldsymbol{s}';\boldsymbol{\tau}')\cdot\max_{\substack{s_n\\ \tau_n}}\{a_{s_{n-1}s_n}\cdot P_{s_n}(\boldsymbol{w}'')\} \quad (5.37)$$

In this equation, $\boldsymbol{w}$ is the complete sequence. $\boldsymbol{w}'$ is the sequence generated by the model while traversing the segments $\boldsymbol{s}',\boldsymbol{\tau}'$. $\boldsymbol{w}''$ then represents the rest of the sequence $\boldsymbol{w}$ which is not part of $\boldsymbol{w}'$. $\boldsymbol{w}''$ is generated by model state s_n.

The likelihoods of the optimal sub-paths are stored in the matrix $\boldsymbol{\Delta}$, just as the total likelihoods were stored in the forward matrix $\boldsymbol{F}$ (see figure 5.7). Additionally, we need to keep track of the best previous state and its duration for every state and position. These matrices $\boldsymbol{\Phi}^{state}$ and $\boldsymbol{\Phi}^{dur}$ are used to backtrack the optimal segmentation once the matrix $\boldsymbol{\Delta}$ has been filled and we therefore know in which state the overall best path ends. Apart from this backtracking, the complexity is essentially identical to that of the forward algorithm.

Viterbi training. What remains to be specified, are algorithms to train the segment models, i. e. the start and transition probabilities as well as the duration and output distributions. For all practically useful goal functions such as the ones discussed in section 5.1, no closed solution for the model parameters can be given, not even in the case of ML which still has a closed solution in the case of Markov chains (cf. section 5.2.1). The obvious reason for this is the second stochastic process which generates the hidden segmentation.

An efficient algorithm to train an SSM uses the Viterbi algorithm inside an iterative two-step learning process: First, we determine the most likely state/duration sequence for each training sequence, then we treat this segmentation as the correct annotation. The resulting training material for each state is used to estimate the output and duration distribution; the probabilities of the state transitions and initial states are modified as well. This is known as *Viterbi* or *decision supervised* training. The algorithm aims at the maximization of the Viterbi score of the model, i. e., the score $P(\boldsymbol{w},\hat{\boldsymbol{s}},\hat{\boldsymbol{\tau}})$ obtained on the most probable segmentation $\hat{\boldsymbol{s}},\hat{\boldsymbol{\tau}}$ of sequence $\boldsymbol{w}$. This

```
Initialize $\mathbf{\Phi}^{state}, \mathbf{\Phi}^{dur}, \mathbf{\Delta}$
FOR $t := 1$ TO $T$
    FOR $j := 1$ TO $L$
        Initialize $m$
        FOR $t' = 0$ TO $t$
            IF $t' = 0$
            THEN $m' := \pi_j$
                 $m'_{index} := -1$
            ELSE FOR $i := 1$ TO $N$
                     $h := \mathbf{\Delta}_{t',i} \cdot A_{ij}$
                     IF $h > m'$
                     THEN $m' := h$
                          $m'_{index} := i$
            $a := m' \cdot P_j(w_{t'+1}, .., w_t)$
            IF $a > m$
            THEN $m := a$
                 $m_{index} := m'_{index}$
                 $m_{dur} := t - t'$
        $\Delta_{t,j} := m$
        $\Phi^{state}_{t,j} := m_{index}$
        $\Phi^{dur}_{t,j} := m_{dur}$
$s_n := \text{argmax}_{j \in N}\{\Delta_{T,j}\}$
$\tau_n := \Phi^{dur}_{T,n}$
```

Figure 5.9: **Viterbi algorithm for segment models**, after Stemmer (1999). Given is a sequence $\boldsymbol{w} = w_1, .., w_T$. $\mathbf{\Phi}^{state}$ is a matrix that contains pointers back to the previous states; $\mathbf{\Phi}^{dur}$ contains the corresponding segment durations. These are needed to recover the optimal segmentation. $\mathbf{\Delta}$ stores the probability of the respective optimal state sub-sequences and is the counterpart of the forward matrix $\boldsymbol{F}$. The probability of the best parse is contained in Δ_{T,s_n}. $\boldsymbol{s}, \boldsymbol{\tau}$ result from the back pointers stored in the matrices $\mathbf{\Phi}$.

is a goal function which is different from Maximum Likelihood and also the other ones discussed in section 5.1.

For HMM model estimation, Schukat-Talamazzini (1995) notes that the following inequality holds: If M denotes the model before and M' the model after the current training iteration, and

$\boldsymbol{q}$ a path through the model,

$$\begin{aligned} P^*_M(\boldsymbol{w}) = P_M(\boldsymbol{w}, \hat{\boldsymbol{q}}) \quad &\leq \quad P_{M'}(\boldsymbol{w}, \hat{\boldsymbol{q}}) \leq P^*_{M'}(\boldsymbol{w}) \\ \text{with} \quad & P^*_M(\boldsymbol{w}) = P_M(\boldsymbol{w}, \operatorname*{argmax}_{\boldsymbol{q}} P_M(\boldsymbol{q}|\boldsymbol{w})). \end{aligned} \tag{5.38}$$

The goal function evaluated on M' is therefore guaranteed to be at least as large as the one on M; re-estimation of the parameters according to the Viterbi training algorithm is based on the optimal path which in principle could also be used by the updated model. But because the best path obtained by the old model might not be the best under the new one, the last part of the equation is an inequality. In the case of SSMs, though, a proof of convergence is deferred to the sub-models: as they might be trained with any goal function, the overall convergence according to the Viterbi goal function cannot be guaranteed in general. When Markov chain models of fixed order are used as sub-models, the ML estimation corresponds to the Viterbi goal function, and the inequality holds. But already in the case of Maximum Likelihood estimation of interpolated Markov chains, this does not hold any more, as they perform a local optimization after the initial parameter estimation. In practical experiments, though, convergence is usually observed.

Viterbi training has the complexity of the Viterbi algorithm, plus the complexity of the training algorithms for the sub-models, and usually results in a fast convergence. Other popular training algorithms for HMMs could also be generalized to the extended model structure of the SSMs. For example, there are expectation-maximization (EM) or gradient descent algorithms to (locally) optimize the ML goal function (Niemann, 1990; Baldi and Brunak, 1998). The EM based approach is known as *Baum-Welch algorithm* and is based on the forward algorithm and a corresponding backward algorithm that fills a matrix of backward variables starting at the end of a sequence. This algorithm takes all possible paths through the model instead of only the best one into account which results in a much higher complexity: first, all posterior probabilities for transitions and durations are calculated from the forward and backward matrices, and then the sub-models have to be trained; for each state, either one sub-model with partial sequences weighted with their duration probability, or several sub-model for all durations that had a probability greater than zero. In either way, this obviously results in a much larger runtime, which is not even justified when one is only interested in the application of the Viterbi algorithm to obtain the best segmentations. This is the case for gene finding, where, apart from alternative splicing, only one gene structure needs to be recovered, and also for promoter analysis, where e. g. only at most one TATA box is present at a specific location within the sequence.

Numerical considerations and scaling. Computing with probabilities often leads to numerical problems which have to be accounted for. Simply multiplying all probabilities for the individual

```
Initialize model $M$
WHILE not converged or FOR a predefined number of cycles
    $\hat{\boldsymbol{s}}, \hat{\boldsymbol{\tau}} := \mathrm{argmax}_{\boldsymbol{s},\boldsymbol{\tau}} P_M(\boldsymbol{s}, \boldsymbol{\tau} | \boldsymbol{w})$    (Viterbi algorithm)
    $\forall i :$      $\bar{\pi}_i := \#(\hat{s}_1 = i)$
    $\forall i, j :$   $\bar{a}_{ij} := \#(\hat{s}_t = i \wedge \hat{s}_{t+1} = j)$
    $\forall i :$      $\hat{\pi}_i := \frac{\bar{\pi}_i}{\sum_i \bar{\pi}_i}$
    $\forall i, j :$   $\hat{a}_{ij} := \frac{\bar{a}_{ij}}{\sum_j \bar{a}_{ij}}$
    $\forall i :$      Estimation of $P_i$ including $d_i$ and $b_i$
    $M := (\hat{\boldsymbol{\pi}}; \hat{\boldsymbol{A}}; \hat{P})$
```

Figure 5.10: **Viterbi training for segment models**, from Ohler et al. (2000). $\#$ is a function which counts the occurrence of its argument.

components, e. g. as given in the chain rule (equation 5.5), will lead to an underflow even if sequences of only moderate length are considered.

A natural way to avoid underflows is to compute the sum of the logarithms of the probabilities instead of the product of the probabilities themselves. As the logarithm is a strictly monotonous function, the maximum of the sum of logarithms is at the same point as the maximum of the product of probabilities. For the Viterbi algorithm, where only the respective maximum is considered, this is an easy solution. It also leads to a faster run-time because the time consuming products are replaced by sums.

In the case of the forward algorithm, the situation is more complex, as it involves the computation of sums of probabilities. Converting back and forth between logarithmized and normal probabilities is definitely no solution, but the equation of Kingsbury and Rayner (1971) provides a way out. It states that the logarithmized sum of two probabilities can be computed as follows:

$$\log_u(p_1 + p_2) = \log_u p_1 + \log_u(1 + u^{\log_u p_2 - log_u p_1}) \tag{5.39}$$

Incidentally, we also need this equation to compute the sum over the likelihoods in the denominator of the MMI goal function (equation 5.3).

An alternative solution is provided by scaling of the forward variables. In the case of HMMs, we simply re-scale the entries with a column-dependent factor; very often, as a factor the sum of all column entries is used:

$$\alpha'_{t,j} = \frac{1}{\delta_t} \cdot \alpha_{t,j} = \frac{\alpha_{t,j}}{\sum_j \alpha_{t,j}} \tag{5.40}$$

In the course of the forward algorithm, we are now able to use the re-scaled variables because the following condition holds:

$$\alpha_{t,j} = \sum_i \alpha'_{t-1,i} \cdot \delta_{t-1} \cdot a_{ij} \cdot b_j(w_t) = \delta_{t-1} \cdot \sum_i \alpha'_{t-1,i} \cdot a_{ij} \cdot b_j(w_t) \tag{5.41}$$

The real forward variables can now be reconstructed at any time point via:

$$\alpha_{t,i} = \delta_1 \cdots \delta_t \cdot \alpha'_{t,i}. \tag{5.42}$$

If we want to use the re-scaled variables for segment models, we run into problems because the SSM equation corresponding to equation 5.41 looks as follows (Stemmer, 1999):

$$\alpha_{t,j} = \sum_{t'=0}^{t-1} P_j(w_{t'+1} \dots w_t) \cdot \sum_i \alpha'_{t',i} \cdot \delta_{t'} \cdot a_{ij} \tag{5.43}$$

In this case, the re-scaling factor $\delta_{t'}$ cannot be put before the summation over t'. The only simplification is to put this factor before the sum over the previous states i:

$$\alpha_{t,j} = \sum_{t'=0}^{t-1} \delta_{t'} \cdot P_j(w_{t'+1} \dots w_t) \cdot \sum_i \alpha'_{t',i} \cdot a_{ij} \tag{5.44}$$

A more heuristic but much easier way that was pointed out by Burge (1997) consists of a re-scaling with a constant factor; in this work, the size of the alphabet was used as such a factor, and no numerical problems were encountered.

5.3.3 Runtime considerations

The algorithms above are derived from the efficient DP paradigm, but their practical usefulness will be severely limited if no additional restrictions or simplifications can be made. Even if we assume a complexity of the submodels which is linear in the sequence length (as is the case for Markov chains), the Viterbi and forward algorithm have to make $\frac{1}{2}(T-1)$ times more calls to the output distribution to take the variable segment lengths into account. As an example, consider a segment model with three fully connected states and an observation of length 300. In this case, about 134,000 calls to the output distribution have to be carried out. Luckily, assumptions on the model topology and the distributions lead us back to a practical scale.

Duration distributions. Discrete duration distributions are represented as histograms of the relative frequencies. One way to reduce the number of calculations drastically is to provide minimum and maximum durations τ_{min} and τ_{max} for each state, which is obviously application dependent. If we only have D values for which the duration distributions are greater than zero, the complexity is reduced to $T \cdot D \cdot N$ from $\frac{T\cdot(T-1)}{2} \cdot N$. The training is then started with a uniform distribution over $[\tau_{min}, \tau_{max}]$. In the case of Viterbi training that only considers the most probable length, the values are smoothed with their left and right neighbors after each iteration.

In some applications, a restriction to subsets of the set of possible durations $\{\tau_{min}, \ldots, \tau_{max}\}$ can help to further reduce the runtime.

Model topologies. One of the most efficient restrictions concerns the model topology. If we assume a sequence of states which is restricted, the set of previous states is reduced from an average 50% in the case of a left-right model (i. e. , there is an order on the states such that possible state transitions only go from lower to higher numbers) up to a single state in the case of a strictly linear model. In the latter case, the complexity is thus equal to $\frac{T\cdot(T-1)}{2} \cdot 1$, and we also have to start filling the matrix only after the sum over all τ_{min} from the first up to the considered state has been exceeded.

Advance calculation and factorization of likelihoods. Without modifications, the forward or Viterbi algorithm will call the same output density with the same segment several times to fill the entries of different states. It is therefore much faster to apply the output distributions ahead and store the likelihoods in a table indexed by position and length. Thereby, the complexity of density evaluation and computing the optimal likelihood are decoupled. For some densities, this approach can lead to further savings in runtime:

- In the case of an HMM output distribution, the HMM has to be called only once for τ_{max}, and the other likelihoods for segment lengths down to τ_{min} can be retrieved from the previous columns in the forward or Viterbi matrix.
- With an MC, the total probability of a sequence can be factorized into single conditional probabilities per base. If we store the cumulative sum of the log probabilities along the whole sequence for each model state in advance, the calculation of a segment probability will be reduced to two table accesses and a subtraction. We therefore do not need to explicitly calculate and/or store the density values for different segment durations.

  This calculation will assume a full available context even at the beginning of a segment after a state transition has been observed. In reality, though, the context was generated by a different distribution. To obtain the correct probabilities, we therefore have to call the distribution again for the first N bases, if N denotes the context of the Markov chain, and retrieve the probabilities for the rest of the segment from the table of log likelihoods.

Chapter 6

Extraction and Modeling of Continuous Features

The approach for a computational modeling of eukaryotic promoters presented in this work has so far been based on specific features of the DNA promoter sequence: Binding sites of transcription factors, or the base composition in different segments. But as we have seen in section 3.2.2, eukaryotic promoters do not only contain specific sequence elements that serve as targets for interacting proteins; they also exhibit distinct physical properties. For example, the DNA of an actively transcribed promoter has to be accessible and must not be wrapped up in nucleosomes.

In this chapter, I will therefore describe continuously valued features that can be calculated from a DNA sequence and relate to physico-chemical properties of DNA. The first, short section deals with CpG islands features, the rest of the chapter with the calculation of property profiles and features that can be extracted from these profiles.

6.1 CpG island features

CpG islands hint at regions of generally low methylation and therefore an open chromatin structure (section 3.2.2). They are associated with an estimated 50 % of vertebrate promoters, but do not exist in non-vertebrate eukaryotes such as *D. melanogaster* (Lyko, 2001). In the fruit fly, the level of methylation is generally very low. Furthermore, it is not the cytosine which is methylated, and therefore the characteristic under-representation of CG dinucleotides, which is a cause of the mutations from methylated cytosine to thymine, is not observed. CpG island features can thus not be exploited for eukaryotic promoter finding in general, but can be used for the modeling of vertebrate promoters.

Following the definition of CpG islands (Gardiner-Garden and Frommer, 1987), three features are distinct for them:

1. GC content. For a sequence $\boldsymbol{w}$ of length T, this is

$$\text{gc\_content}(\boldsymbol{w}) = \frac{\#(\text{G}) + \#(\text{C})}{T} \tag{6.1}$$

   In the literature, a CpG island has a minimum GC content of 0.5.

2. The ratio of expected to observed CG di-nucleotides cg_ratio. This is defined by

$$\text{cg\_ratio}(\boldsymbol{w}) = \frac{\frac{\#(\text{CG})}{T-1}}{\frac{\#(\text{C})}{T} \cdot \frac{\#(\text{G})}{T}} \tag{6.2}$$

   Here, the minimum value is given as 0.6.

3. A minimum length of 200 bases for which content and ratio as defined above must be above the given threshold.

As we will see later (chapter 8), the core of the promoter recognition system is a classifier which labels sequences of fixed length as promoters or non-promoters. These sequences are of 300 bases length and consequently above the minimum length threshold. We will therefore use two feature variables for GC content and CG di-nucleotide ratio, calculated on windows of 300 bases. It is possible to reduce these values to binary features, assigning a zero if a value is below the given threshold and a one if it is above, but this would reduce valuable information at an early stage: Ioshikhes and Zhang (2000) found that CpG islands associated with promoters have different average feature values than those found at other places in the genome.

6.2 Calculation of property profiles

CpG islands are specific for vertebrate organisms only. But studies on different pro- and eukaryotic organisms (Pedersen et al., 1998; Babenko et al., 1999; Pedersen et al., 2000; Ohler et al., 2001), showed that the DNA sequence in promoters is distinct for a wide variety of physico-chemical properties that e. g. relate to the chromatin structure and therefore their accessibility (cf. section 3.2.2).

For a large number of these properties, parameters have been published that generally refer to di- or tri-nucleotides. They are symmetric, i. e. , an oligonucleotide has the same parameter value as its reverse complement. One simple example of such a property is the GC content based on tri-nucleotides which simply counts how many guanines and cytosines are present in each tri-nucleotide. For other properties, parameter sets have been experimentally derived, and the

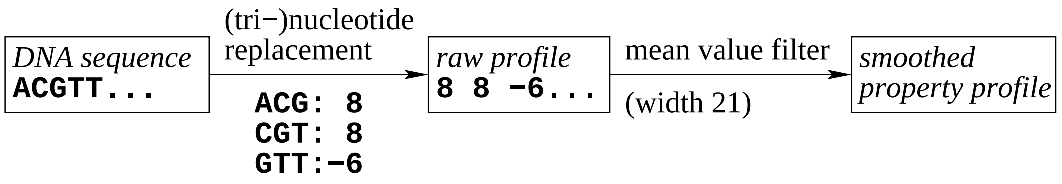


Figure 6.1: **Conversion of a sequence into a profile.** The figure shows the exemplary conversion of a short DNA sequence into a nucleosome positioning preference profile. In this case, the experimentally derived parameters refer to tri-nucleotides; the full set is given in appendix C.

values relate to properties such as the sensitivity regarding DNA digestion enzymes (high values pointing at low bendability), the preference to be located at nucleosomes, or the distortion angles observed in protein-DNA-interactions (the protein-DNA-twist). The parameters of a physical property can be used to calculate a *profile* of this property. A profile consists of the corresponding values from the chosen parameter set in place of each overlapping di- or tri-nucleotide within a given DNA sequence.

An example where the target of a DNA interacting protein is largely defined by DNA physical properties is the P transposable element insertion site in Drosophila. Here, no clear sequence consensus can be seen, but for a large variety of properties, the profiles at the insertion site show distinct peaks. I explored the 14 different parameter sets of physical DNA properties compiled by Liao et al. (2000) for this P element study; the parameter tables for all these properties are listed in appendix C.

Because the parameters refer to di- or tri-nucleotides only, the profiles generally appear to be very noisy. Therefore, they are smoothed with a mean value filter of a certain fixed width, usually 20–30 base pairs (Pedersen et al., 1998; Liao et al., 2000). Figure 6.1 summarizes the mapping between sequence and profile, and examples for different property profiles of the same sequence can be seen in figures 6.2 and 6.3.

Excursion: Filter methods. In an ideal case, a signal is observed without any distortions or noise, and in many applications we implicitly assume that this is the case. Therefore, the noise that we encounter in real examples (such as in figure 6.3) should be eliminated as reliably as possible before any further processing or an extraction of features. Filtering is seen as transformation of a signal into another signal which is hopefully easier to process (for an introduction, see Paulus and Hornegger, 2001; Niemann, 1983).

The mean filter used above is a simple example of a *linear* filter, a linear transformation where the new signal can be expressed as a convolution of the original signal with a mask or *window*.

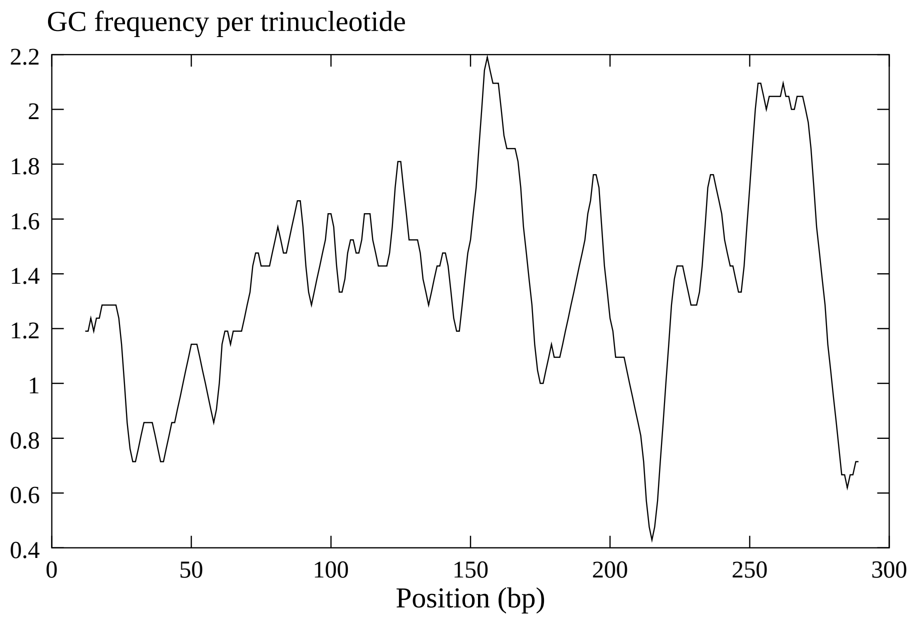


Figure 6.2: **Property profile of trinucleotide GC content.** This picture shows the profile of a *Drosophila* promoter. The transcription start site is at position 250, and one can clearly locate the TATA box at the point of the distinct valley upstream of the TSS.

A current value v_i of the signal, centered in the middle of the window, is set to the average of all values within the window of size $n+1$:

$$\hat{v}_i = \left(\frac{1}{n+1}, \ldots, \frac{1}{n+1}\right) \cdot \begin{pmatrix} v_{i-n/2} \\ \cdots \\ v_i \\ \cdots \\ v_{i+n/2} \end{pmatrix} \tag{6.3}$$

A mean filter eliminates rough changes in the signal by removing high frequencies, which usually results in a blurred transformed signal.

Examples of non-linear filters, which will be used later to smooth the *output* of the promoter prediction system, are the *median* and the *hysteresis* filter. The median filter is a so-called rank order filter: We first determine the order of all values within a window around the current position in a signal, in our case using the $\geq$-relation on real numbers. Then we set the value at the current position to the middle value in the list of ordered values. In contrast to the mean filter, the median filter preserves sharp edges and does not lead to blurred results.

The hysteresis threshold filter (Duda et al., 2000) shifts a smoothing cursor of a chosen height

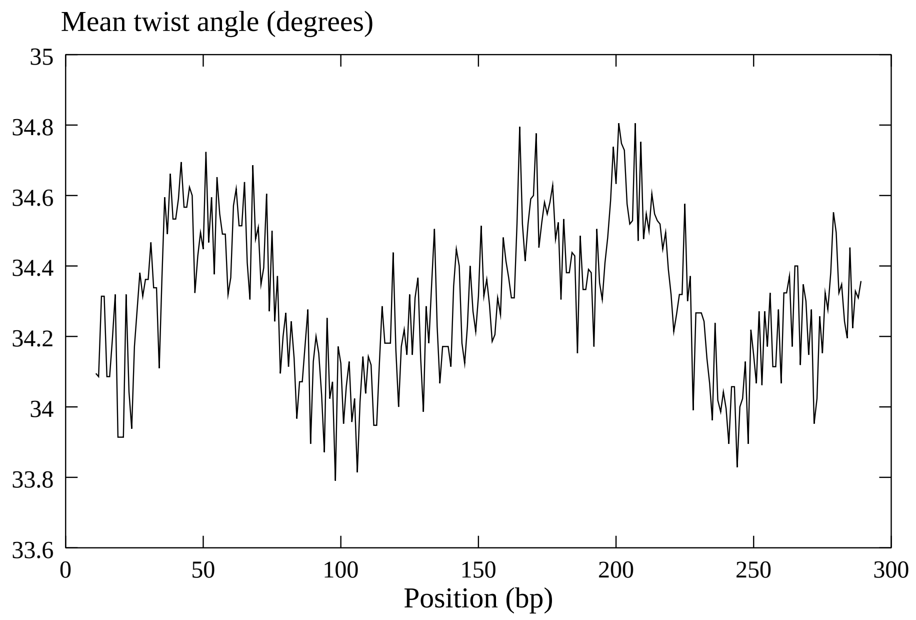


Figure 6.3: **Property profile of protein-DNA-twist.** This picture shows the profile of a different property of the same *Drosophila* promoter as in figure 6.2. In contrast to the GC tri-nucleotide content, the protein-DNA-twist parameters refer to di-nucleotides only, and the profile appears to be more noisy.

over the signal from left to right, and the middle position of the cursor is always emitted as new output. As long as the next considered value lies within the cursor area, the cursor position is not moved vertically. If the next value lies above the cursor, it is moved up so that the upper rim corresponds with the value; if it lies below the cursor, it is moved down in an analogous way. With increasing cursor width, the curve is thus smoothed more and more.

Figure 6.2 through 6.5 are smoothed with a mean filter of 21 bases window width; this width is used for all profile calculations throughout this work. An example of hysteresis filtering is shown in figure 6.4 which contains the protein-DNA-twist profile of figure 6.3 before and after hysteresis smoothing.

6.3 Features for property profiles

Figure 6.5 shows the GC content profiles of the three sequence classes within the *Drosophila* training set, namely coding and non-coding sequences as well as promoters whose transcription start site is aligned at position 250. The profiles are averaged over all sequences in the set.

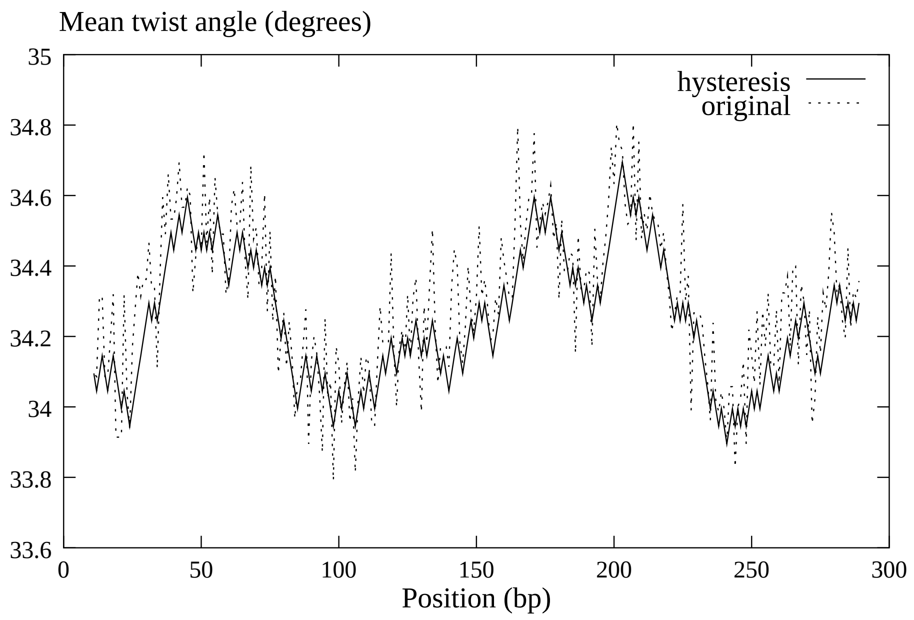


Figure 6.4: **Example of hysteresis filtering.** This picture shows the original profile of figure 6.3 as well as the profile after smoothing by a hysteresis filter of cursor width 0.1.

Coding and non-coding sequences, as expected, show rather uniform values with no positional preferences. In contrast, the promoters have a distinct profile with drops in the areas of TATA box around position 220 and the initiator at position 250. So, in the case of GC content profiles, a distinction between the considered classes is visually detectable. But the figure gives an inaccurate impression because the profile shown is averaged over a large set of sequences and does not reflect that some promoters lack distinct profile features such as the TATA box valley. Moreover, even in the case where a TATA box is present, individual profiles show a high degree of variation from the average profile (cf. figure 6.2), resulting from the unique underlying sequence. Even after smoothing with a mean filter, the profiles of single sequences appear rather noisy as can be seen from figure 6.3.

I therefore decided to use features that do not relate to single positions within the profile, but rather approximate the profile slope with a number of simple functions for distinct parts of the promoter, such as the TATA box or the initiator area — corresponding to the segments generated by the states of the promoter sequence model (see section 5.3.1).

The first set of features consists of the mean values x_i of the profiles $\boldsymbol{p_i}$ corresponding to segments $\boldsymbol{s_i}$ of length τ_i, calculated according to some profile parameter set:

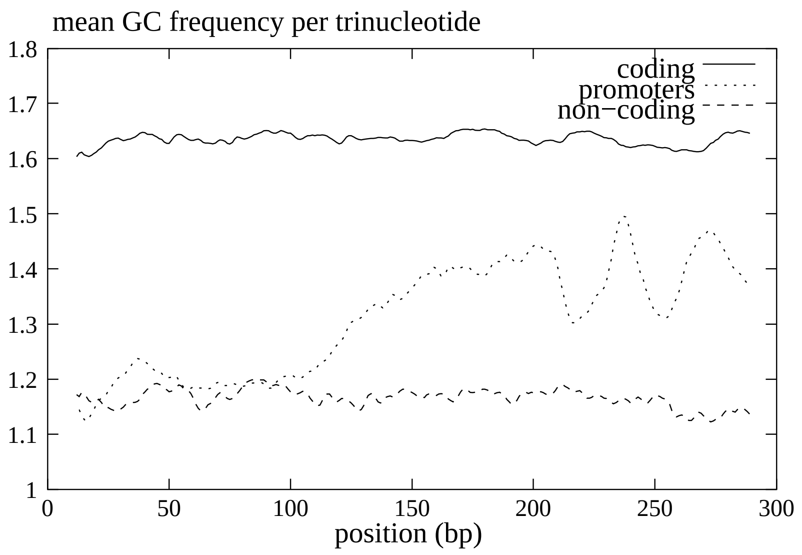


Figure 6.5: **GC content of promoters, coding, and non-coding sequences in *Drosophila***, from Ohler et al. (2001). The transcription start sites of the promoters are aligned at position 250. The profiles were smoothed with a mean filter of width 21 and averaged position-wise over *all* sequence profiles for the *Drosophila* training set.

$$x_i = \frac{1}{\tau_i} \sum_{k=1}^{\tau_i} p_{i,k}. \tag{6.4}$$

This leads us to an approximation by constant functions, i. e. polynoms of order zero.

Even though the individual values do not properly reflect it, a distinct ascent or descent such as the increase in GC content before the TATA box (see figure 6.5) might be visible from a regression line. Adding the slope coefficient of a straight regression line for $\boldsymbol{p_i}$ as additional feature thus leads us to polynoms of first order. If we assume that we minimize the mean quadratic error, the coefficients are given as (see Press et al. (1993))

$$a_i = \frac{\tau_i \sum_{k=1}^{\tau_i} k p_{i,k} - \sum_{k=1}^{\tau_i} p_{i,k} \sum_{k=1}^{\tau_i} k}{\tau_i \sum_{k=1}^{\tau_i} k^2 (\sum_{k=1}^{\tau_i} k)^2}. \tag{6.5}$$

Because of the different range of the parameter sets, the coefficients a_i are normalized by

$$\hat{a}_i = a_i \frac{|\boldsymbol{p}|}{p_{\max} - p_{\min}}.$$

$p_{\max}, p_{\min}$ is the largest respectively smallest value of the property parameter set that the profile was computed for (the range on the y axis), and $|\boldsymbol{p}|$ is the length of the whole profile in base pairs (the range on the x axis; here 300 bases).

Feature transformation: principal component analysis. We are not restricted to one profile, but can in principle use features for a different number of physical properties such as the whole set listed in appendix C. For the small data sets that we have at our hands, though, we will quickly encounter the problem of over-adaptation of models if we explicitly use parameters for every single feature that we can possibly extract. Many of the 14 parameter sets are also highly correlated (see Liao et al. (2000) and the web supplement at http://www.fruitfly.org/~guochun/pins.html), and even if features delivered a good classification rate when used on their own, the overall classification will not improve much when they are correlated too closely.

An elegant solution to this problem is provided by principal component analysis. The underlying idea is to provide a projection of the high dimensional space of all features to a subspace of lower dimensionality which largely conserves the properties of the original space.

Starting from an orthonormal basis $\mathbf{\Phi}$ of the original space $\mathrm{I\!R}^D$, we can express each D-dimensional feature vector $\boldsymbol{x}$ in terms of the base vectors $\boldsymbol{\phi_i}$:

$$\boldsymbol{x} = \sum_{i=1}^{D} y_i \boldsymbol{\phi_i} \tag{6.6}$$

An approximation $\hat{\boldsymbol{x}}$ of $\boldsymbol{x}$ is then given by a summation of only the first $d, d < D$, vectors instead of the full set, which causes an expected quadratic error of

$$\epsilon_d = \mathcal{E}[\|\boldsymbol{x} - \hat{\boldsymbol{x}}\|^2] = \mathcal{E}[\| \sum_{i=d+1}^{D} y_i \boldsymbol{\phi_i} \|^2] = \sum_{i=d+1}^{D} \boldsymbol{\phi_i}^{\mathrm{T}} \mathcal{E}[\boldsymbol{x}\boldsymbol{x}^{\mathrm{T}}] \boldsymbol{\phi_i}, \tag{6.7}$$

where $\boldsymbol{S} := \mathcal{E}[\boldsymbol{x}\boldsymbol{x}^{\mathrm{T}}]$ can be estimated from the training set of feature vectors. Minimizing the error ϵ_d with respect to the basis $\mathbf{\Phi}$ leads to the eigenvalue problem $\boldsymbol{S}\boldsymbol{\phi_i} = \lambda_i \boldsymbol{\phi_i}$, with the result that $\epsilon_d = \sum_{i=d+1}^{D} \lambda_i$ (see standard textbooks such as the ones by Niemann (1983); Schukat-Talamazzini (1995) for more details). Selecting the d basis vectors that correspond to the largest eigenvalues therefore leads to the smallest error. To account for the quite diverse range of feature values due to the different profile parameters, we normalize the original feature values to have mean value zero and a variance of one and set

$$\boldsymbol{y} = \Phi^{\mathrm{T}} (\boldsymbol{x} - \boldsymbol{\mu}) \mathbf{\Sigma}^{-1}, \tag{6.8}$$

with $\mathbf{\Sigma}$ being the diagonal matrix containing the variance values σ_i for the individual features.

The PCA transformed features do not necessarily lead to an improvement of classification. For example, if a feature is not well suited for classification, the PCA will still regard it as one of the most important features if it has a large variance.

6.4 Continuous densities for profile features

To allow for an integration into the existing probabilistic framework of the promoter sequence model, we use a probabilistic modeling approach for features computed from the profiles. If nothing else is known about the data, it is very common to assume a Gaussian distribution for continuously valued features. Using the notation from the previous section, we therefore have a distribution c_j for a profile $\boldsymbol{p}$ within a segment as follows:

$$\begin{aligned} c_j(\boldsymbol{p}[\boldsymbol{w}, \tau]) &:= \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu_j}, \boldsymbol{\Sigma_j}) = \\ &\frac{1}{\sqrt{|2\pi\boldsymbol{\Sigma_j}|}} exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu_j})^{\mathrm{T}}\boldsymbol{\Sigma_j}^{-1}(\boldsymbol{x}-\boldsymbol{\mu_j})\right), \end{aligned} \tag{6.9}$$

with the mean vector $\boldsymbol{\mu_j}$ and the symmetric covariance matrix $\boldsymbol{\Sigma_j}$ as parameters of the distribution. The profile is thereby represented by a set of features which are gathered in the vector $\boldsymbol{x}$, computed as described in the last section.

For a data set containing N_j samples, the maximum likelihood parameter estimation of a Gaussian distribution has the following closed solution (Niemann, 1983; Schukat-Talamazzini, 1995):

$$\hat{\boldsymbol{\mu}}_{\boldsymbol{j}} = \frac{1}{N_j}\sum_{i=1}^{N_j} \boldsymbol{x_i} \tag{6.10}$$

$$\hat{\boldsymbol{\Sigma}}_{\boldsymbol{j}} = \frac{1}{N_j}\sum_{i=1}^{N_j} (\boldsymbol{x_i}-\hat{\boldsymbol{\mu}}_{\boldsymbol{j}})(\boldsymbol{x_i}-\hat{\boldsymbol{\mu}}_{\boldsymbol{j}})^{\mathrm{T}} \tag{6.11}$$

A more general approach models a profile with a *mixture* distribution; looking at GC content as an example, this should account for different GC isochores, i. e. for regions with a different overall GC frequency, or for TATA-box containing versus TATA-less promoters. In the case of the profile features, a mixture of Gaussians with m components is given by

$$c_j(\boldsymbol{p}[\boldsymbol{w}, \tau]) := \sum_{\nu=1}^{m} a_{j\nu}\mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu_{j\nu}}, \boldsymbol{\Sigma_{j\nu}}). \tag{6.12}$$

As additional parameters, we have the mixture coefficients a, with the condition that $\sum_{\nu=1}^{m} a_{j\nu} = 1$. If the number of mixture components is large enough, this mixture distribution can approximate any distribution. There is no general closed solution for the ML parameter estimation any more, but we can regard the mixture coefficients as hidden variables and apply the Expectation Maximization algorithm. We then have the following iterative parameter estimation scheme (Schukat-Talamazzini, 1995; Hornegger, 1996): First we use the m components to calculate the *a posteriori* probabilities $\gamma_{j\nu}$ for each of the N_j samples to belong to component ν:

$$\gamma_{j\nu}^{i} = \frac{a_{j\nu}\mathcal{N}(\boldsymbol{x_i}|\boldsymbol{\mu_{j\nu}}, \boldsymbol{\Sigma_{j\nu}})}{\sum_{\nu=1}^{m} a_{j\nu}\mathcal{N}(\boldsymbol{x_i}|\boldsymbol{\mu_{j\nu}}, \boldsymbol{\Sigma_{j\nu}})}, \tag{6.13}$$

then we use them to re-estimate the parameters:

$$\hat{a}_{j\nu} = \frac{1}{N_j}\sum_{i=1}^{N_j} \gamma_{j\nu}^{i} \tag{6.14}$$

$$\boldsymbol{\hat{\mu}_{j\nu}} = \frac{1}{\sum_i \gamma_{j\nu}^{i}}\sum_{i=1}^{N_j} \gamma_{j\nu}^{i}\boldsymbol{x_i} \tag{6.15}$$

$$\boldsymbol{\hat{\Sigma}_{j\nu}} = \frac{1}{\sum_i \gamma_{j\nu}^{i}}\sum_{i=1}^{N_j} \gamma_{j\nu}^{i} \cdot (\boldsymbol{x_i} - \boldsymbol{\hat{\mu}_{j\nu}})(\boldsymbol{x_i} - \boldsymbol{\hat{\mu}_{j\nu}})^{\mathrm{T}} \tag{6.16}$$

As an example initialization, the mixture coefficients can be set to uniform values, the mean to randomly chosen samples $\boldsymbol{x_i}$, and the covariance matrix to a diagonal matrix with the nonzero entries set to the variance calculated over the whole sample set.

At this point, I have presented probabilistic models for sequence and physical properties of DNA sequences. The following chapter will now look at the problem of classification: How can we use these models to decide which class a sequence belongs to?

Chapter 7

Classification and Evaluation

Classification of a pattern means that we assign one of $k, 1 \leq k \leq K$ classes to a feature vector derived from the input pattern. Thus, the classes represent a partition of the feature vector space and might be explicitly given by the application — as is the case with promoter and non-promoter — or derived from the data in an un-supervised way; the estimation of mixture distributions as presented in the previous chapter can be considered as un-supervised learning. Note that in the case of DNA sequence-based classification, the feature vector consists of the nucleotides of the sequence, and no feature extraction step is necessary. A classification based on physical properties represents the case where we first extract features from a pattern, and then decide on the class of the pattern based on these features.

In the literature, three different approaches for classification are usually considered: Statistical, distribution-free, and non-parametric classifiers (Niemann, 1990; Duda et al., 2000). In this work, we will refer to the first two approaches only, where the classification is based on parameters that are learned from the data, either representing a distribution of the features themselves or a function that separates the different classes in feature space. In contrast, non-parametric classification is directly based on the whole or a representative part of the training set.

The following two sections discuss statistical and distribution-free approaches for classification. The last section turns to different aspects of assessing the quality of the predictions that result from classification.

7.1 Bayes classifier

Statistical classification is based on the assumption that feature vectors $\boldsymbol{x} = (x_1 \ldots x_D) \in \mathrm{I\!R}^D$ are generated by a two-step random process. First, the class which the vector belongs to is chosen, according to the *a priori* probabilities $P_k = P(\Omega_k)$, with $1 \leq k \leq K, \sum_k P_k = 1$. Then, the vector itself is generated by a probability density function $P(\boldsymbol{x}|\Omega_k)$, conditioned on the class that was chosen in the first step. We saw examples of such densities in the previous chapters: continuous densities such as Gaussian distributions in the case of the continuous-valued features derived from profiles, but also discrete densities such as Markov chains in the case where the nucleotides of a sequence were directly considered as features.

How can we use the densities to make a classification as successful as possible? From a formal point of view, our goal in classification is the identification of a decision function

$$\delta(\Omega_k|\boldsymbol{x}), \quad \sum_{k=1}^{K} \delta(\Omega_k|\boldsymbol{x}) = 1 \quad \forall \boldsymbol{x} \in \mathrm{I\!R}^D. \tag{7.1}$$

This is a randomized decision rule where one vector is assigned to each class with a certain probability, and the decision might thus change for two subsequent evaluations of the same input vector. We want to choose the function in such a way that the overall *cost* or *risk* is minimized. A detailed derivation of such an optimal classifier can be taken e. g. from (Niemann, 1983); here, I only sketch the outline and the main result.

The risk is quantified by a matrix $\boldsymbol{R}$, with entries r_{jk} that denote the cost of a mis-classification of a vector which belongs to class Ω_k but is put into class Ω_j. The expected risk R associated with a particular decision function δ is then given by

$$R(\delta) = \sum_{k=1}^{K} P_k \sum_{j=1}^{K} r_{jk} \int_{\mathrm{I\!R}^D} \delta(\Omega_j|\boldsymbol{x}) P(\boldsymbol{x}|\Omega_k) d\boldsymbol{x} \tag{7.2}$$

A minimization of this risk follows the consideration that the value of the integral is minimized if the value of the integral term is minimal for every possible feature vector $\boldsymbol{x}$. In the end, this leads to the deterministic decision function:

$$\begin{aligned} \delta^*(\Omega_k|\boldsymbol{x}) &= 1 \qquad \text{if} \quad u_k(\boldsymbol{x}) = \min_j u_j(\boldsymbol{x}) \\ \delta^*(\Omega_j|\boldsymbol{x}) &= 0 \qquad \forall j, j \neq k \end{aligned} \tag{7.3}$$

with the test variables

$$u_j(\boldsymbol{x}) = \sum_k r_{jk} P_k P(\boldsymbol{x}|\Omega_k). \tag{7.4}$$

If we assume uniform mis-classification costs, more specifically of $r_{jk} = 1, j \neq k$, and $r_{kk} = 0$, the test variables simplify to

$$u_j(\boldsymbol{x}) = \sum_{k, k \neq j} P_k P(\boldsymbol{x}|\Omega_k). \tag{7.5}$$

In this case, the decision rule is equivalent to choosing the class with the largest *a posteriori* probability $P(\Omega_k|\boldsymbol{x})$. This is known as the *Bayes classifier* and has the smallest possible misclassification rate under the assumption of the zero-one-cost function, provided that the *correct* conditional densities are known. In reality, these densities can only be estimated from data, and apart from errors in parameter estimation due to limited data, it is often unknown whether the data is indeed generated by a member of the density family of our choice.

In this work, we use a slightly modified Bayesian classification rule:

- Even though we have a two-class problem — promoter or non-promoter — the non-promoter class consists of two distinct sub-classes, namely coding and non-coding, non-regulatory sequences. We keep the models for these two classes separate, perform the decision on the best background class against the promoter class, and do not care whether the correct background class was chosen. By doing so, it is easier to compare different classifiers, some of which use explicit models for coding and non-coding sequences. An alternative way would be to model all non-promoter models with a more complex mixture distribution.
- Now that the full genomic sequence and an estimate on the length and number of genes are available for both human and *Drosophila*, we would in principle be able to estimate the *a priori* probabilities for promoters and non-promoters. But instead of a specification of fixed *a priori* probabilities for the individual classes, we rather perform the classification based on a (variable) threshold of promoter against best non-promoter class, and set the *a priori* probabilities to uniform values. There are two reasons to proceed in such a way:
  1. The Bayes classifier is based on the assumption of a zero-one loss function. However, it is unclear if this is indeed the case in practical applications, and might also change from application to application. One user might be interested in finding the true promoter at all cost, possibly tolerating a large number of misclassifications which are then eliminated by subsequent wet lab experiments. On the other hand, another user might only want to know whether one reliable prediction is made on his sequence of interest.
  2. Even though genome-wide *a priori* probabilities can be calculated, they differ considerably from the ones in small fractions. The gene (and therefore also promoter) density is different on different chromosomes, and even more within chromosomes. Furthermore, researchers that are interested where a promoter of a specific gene is might have additional information about it. For example, if they already know where the coding part is, it is very likely that they will search for promoters in the sequences upstream from the coding part only, which again results in very different *a priori* probabilities.

The final decision rule therefore looks as follows if the promoters are assigned to class Ω_1 and the background sequences to $\Omega_2 \ldots \Omega_K$:

$$\begin{aligned} \delta(\Omega_1|\boldsymbol{x}) &= 1 \quad \text{if} \quad (\log P(\boldsymbol{x}|\Omega_1))/|\boldsymbol{x}| - \max_{i,2\leq i\leq K}(\log P(\boldsymbol{x}|\Omega_i))/|\boldsymbol{x}| \geq \nu \\ \delta(\Omega_1|\boldsymbol{x}) &= 0 \quad \text{otherwise,} \end{aligned} \tag{7.6}$$

with a threshold ν on the difference between the length normalized promoter and best non-promoter log likelihood.

7.2 Neural networks

Instead of modeling the classes in an appropriate way, such as by densities in the context of statistical classification, one can also learn a parametric function that describes the boundaries of the classes in the input feature space.

One approach is to identify the best out of a given class of functions, for example of linear or quadratic shape. General, nonlinear discriminative functions that are able to separate classes whose samples cover arbitrarily shaped areas in the input space are often learned in the form of artificial neural networks (ANNs). A wide variety of literature has been published on this topic, and Bishop (1995) gives an excellent overview from the pattern recognition perspective, with an emphasis on the relation to statistical classification.

7.2.1 Architecture

ANNs were originally motivated by the structure of the central nervous system of living organisms, which consists of a large number of simple computational units that achieve their power by strong interconnection. The basic unit of an ANN is called a *neuron* or *node*. Each neuron q_j is connected to a number of other nodes: a set I_j from which it receives input signals, and a set O_j to which it sends its computed output activity. The connections are weighted, and these weights θ_{ij} constitute the set of parameters that have to be learned.

The processing steps inside each neuron q_j are:

1. Computation of the input activity a_j. The signals s_i from all neurons q_i that belong to the set I_j are combined with the weights associated with the connections, e. g. by a scalar product. An additional bias or threshold value θ_j is used to shift the overall input and can be conceptually integrated by an additional neuron in the input set whose activity is always set to one.

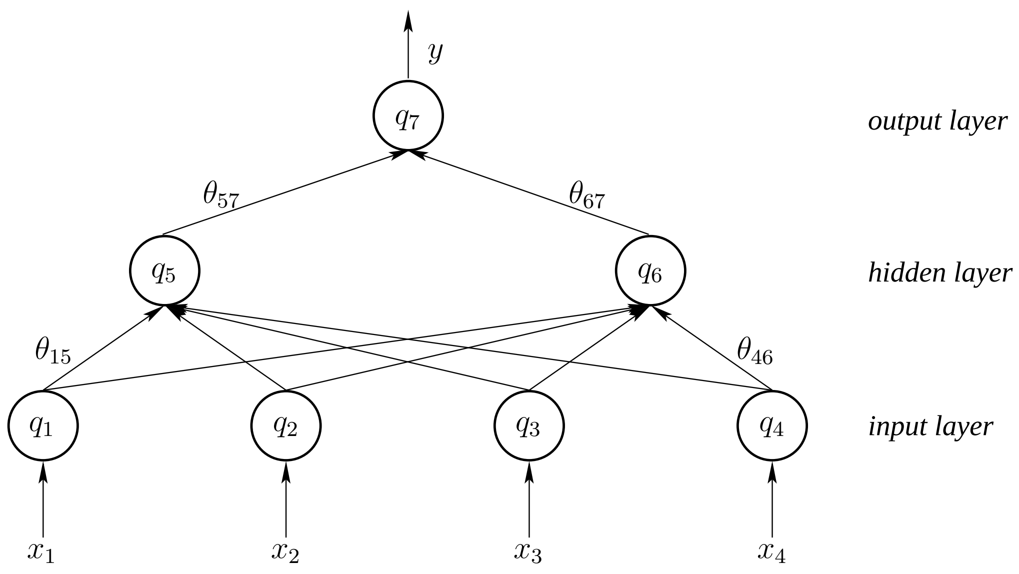


Figure 7.1: **Example for a multi-layer perceptron** with two layers of weights. This example has four input nodes which receive the feature vector, two nodes in the hidden layer, and one output node which emits the network output. Note that there are no recurrent connections; every node sends signals only to nodes in the next layer. Some weight labels are omitted for clarity.

$$a_j = \sum_{i,q_i \in I_j} s_i \theta_{ij} + \theta_j \tag{7.7}$$

2. The input signal is transformed by means of an activation function $f(a_j)$ which is then sent as signal s_j to other neurons in the set O_j. The underlying idea is that if the input activity is strong enough, the neuron is activated and *fires*, i. e. sends a signal to the nodes in set O_j. Therefore, the activation function is often realized as a step function or a mathematically better tractable differentiable approximation such as the sigmoid function $f(a_j) = 1/(1 + e^{-a_j})$. For some problems, a linear activation function is better suited to ensure an unrestricted range of values.

A common topology of neural networks consists of a multi-layer architecture where the nodes in one layer receive inputs exclusively from the previous layer and send their activity only to neurons in the next one. ANNs of this particular topology are known as feed-forward networks or multi-layer perceptrons (MLPs). The units in the input layer are directly fed with the components of the input vector; one or more hidden layers perform the computations; and the activity of the output layer neuron(s) constitute the response of the network. In neural networks which are applied on classification tasks, the output layer often contains one node per class, and the network is trained on a target vector where the vector component of the correct class is set to one and the others to zero. For a two-class problem, one output node is used, and samples from the one class are trained on target values of one and the others on target values of zero. Figure 7.1 shows an example of an MLP network with two layers of weights and a single output node.

By the following intuitive reasoning, MLPs with three layers of adaptable weights are able to learn arbitrarily shaped decision functions, even if samples of the same class populate separate areas in feature space: The first layer learns linear decision functions, the second one combines them to closed areas, and the third provides a combination of several areas. It was shown that even two-layer networks have this property. For a practical application, though, it is not known how *large* the layers have to be, i. e. how many nodes are needed. Neural networks also underly the bias/variance tradeoff which means that we cannot simply make the network arbitrarily large without the risk to over-adapt to the training data and lose generality on unseen samples.

The ability of neural networks to approximate any function makes them not only suitable for classification purposes, but also for regression problems of any kind — classification can be seen as a special regression problem which refers to the approximation of the unknown decision function. In turn, ANNs can therefore also be used to estimate arbitrary posterior probability functions which are then plugged into a Bayesian classifier. This is not further examined in this work, and the reader is once more referred to the book by Bishop (1995).

7.2.2 Learning the weights

We now turn to the problem to determine a set of optimal weights Θ for a multi-layer network of given topology in a supervised way. We assume that for every input vector $\boldsymbol{x}$, a target vector $\boldsymbol{t}(\boldsymbol{x})$ is given; the output that is in fact computed by the network is denoted by $\boldsymbol{y}(\boldsymbol{x}, \Theta)$. We follow the outline of Bishop (1995) where the error function R, which the weights shall be optimized for, is only assumed to be differentiable and additive:

$$R_\Theta(X) = \sum_{\boldsymbol{x}} R_\Theta(\boldsymbol{x})$$

Finding optimal weights corresponds to a minimization of the error, and because of the additive error function, we can compute the derivative of R with respect to the weights for each of the patterns independently. We further assume that the input activity a_j of node q_j is computed with a scalar product as in equation 7.7.

We start by processing each pattern through the network — the forward propagation — to compute the error function and the activation of each neuron. The derivative of the error with respect to a particular weight θ_{ij} depends only on the summed activity a_j of the node the connection leads to:

$$\frac{\partial R}{\partial \theta_{ij}} = \frac{\partial R}{\partial a_j}\frac{\partial a_j}{\partial \theta_{ij}} \tag{7.8}$$

The derivation of the activation a_j with respect to weight θ_{ij} is equal to the signal s_i coming from node q_i. Setting $\delta_j \equiv \frac{\partial R}{\partial a_j}$, we have the derivation in the general form

$$\frac{\partial R}{\partial \theta_{ij}} = \delta_j s_i. \tag{7.9}$$

In the output layer, s_j is equal to y_j, and we have

$$\delta_j = \frac{\partial R}{\partial y_j} f'(a_j). \tag{7.10}$$

For the hidden layers, δ_j can be written as follows using the chain rule:

$$\delta_j \equiv \frac{\partial R}{\partial a_j} = \sum_k \frac{\partial R}{\partial a_k}\frac{\partial a_k}{\partial a_j},$$

with a sum over all nodes to which node q_j sends a signal to. This can be finally written as

$$\delta_j = \sum_k \theta_{jk}\delta_k f'(a_j). \tag{7.11}$$

Thus, starting from the derivation for the output layer, we can propagate the error back through the network, recursively updating the layers one by one. This is why this algorithm is called *error back-propagation*. The concrete form of the update equations depend on the individual error and activation functions.

In its simplest form, back-propagation uses a fixed-step gradient descent technique to subsequently change the weight values. For the first weight layer, with s_i equal to the input x_i, this leads to

$$\Delta\theta_{ij} = -\eta\delta_j x_i, \tag{7.12}$$

where η is a pre-specified fixed learning rate, and with analogous updates for the other layers.

The probably widest spread error function is *mean square error* (MSE). If $\epsilon_{\boldsymbol{x}}$ denotes the error observed for input pattern $\boldsymbol{x}$,

$$\epsilon_{\boldsymbol{x}} = \boldsymbol{y}(\boldsymbol{x}, \boldsymbol{w}) - \boldsymbol{t}(\boldsymbol{x}),$$

MSE is given by

$$R^{MSE}(X) = \frac{1}{2}\sum_{\boldsymbol{x}} \|\epsilon_{\boldsymbol{x}}\|^2. \tag{7.13}$$

For classification purposes, the use of *cross-entropy* is better suited, as the error measure is directly related to the classification performance. For a two-class problem with target value 1 for class Ω_1 and 0 for class Ω_2 , we have

$$R^{CE}(X) = -\sum_{\boldsymbol{x}\in\Omega_1} \ln(1+\epsilon_{\boldsymbol{x}}) - \sum_{\boldsymbol{x}\in\Omega_2} \ln(1-\epsilon_{\boldsymbol{x}}) \tag{7.14}$$

Interestingly, this is equivalent to an optimization of the maximum mutual information objective function for a two-class problem and equal *a priori* probabilities (see equation 5.3).

There are a number of more sophisticated algorithms to train the weights of a neural network. Many of them start from the idea of back-propagation, but include a momentum term to quickly traverse the error surface, or take the second derivation and therefore the curvature of the error surface into account. None of them is considered in this work, as the standard back-propagation algorithm is good enough for the problem that the networks are supposed to deal with.

Early stopping. During the training of a neural network, crucial attention has to be paid to over-adaptation. The back-propagation algorithm is a gradient descent approach which leads to a locally optimal reproduction of the desired output for the training data. Nothing guarantees, though, that the performance on unseen data will be equally good. Therefore, it is common practice to set aside an independent part of the training sample on which the net is evaluated throughout the training. It is expected that the error on this validation set will at first decrease in the course of the training, but will increase again after a certain number of iterations when the net starts to over-adapt to the training data. At this point, the training is ended before the local optimum is reached on the training data ("early stopping"), but at the benefit of a better generalization. An alternative to early stopping is to provide Bayesian regularizers on the weights (Bishop, 1995, chapter 10).

The back-propagation algorithm provides another means of reducing over-adaptation: Apart from the learning rate η, an additional parameter can be provided that constitutes a threshold on the error back-propagation: If the net output for a certain input vector has an error of less than ν, it is not back-propagated any longer. This is especially useful for classification problems, where one is not interested in a reproduction of the class labeling, but rather in a correct classification, i. e. that the output is close enough to the value associated with the correct class.

7.2.3 Feature pre-processing

For practical applications, a pre-processing of the input data can considerably help to improve the performance of an ANN classifier. Here, I mention two pre-processing steps which are generally useful independent of the application.

If the amount of data in our training set is not representative for the *a priori* probabilities of the classes, back-propagation will lead to an over-adaptation to the more frequent classes. The first step in pre-processing is therefore an equalization of the amount of training data that is available for the distinct classes. It is usually accomplished by propagating the samples of the less frequent classes more than once. If enough data is at hand, one can also alternatively eliminate samples from the more frequent classes.

The second step is input normalization. In some cases, the input vector to a neural network consists of components whose continuous values might differ significantly, but are independent of their relative importance. By normalizing the data to have means zero and variance one, all the components x_i of an input vector $\boldsymbol{x}$ are scaled by

$$\hat{x_i} = \frac{x_i - \mu_i}{\sigma_i}, \tag{7.15}$$

where μ_i and σ_i are the mean value and variance of component i, computed on the whole training set. This ensures that all input vector components have the same order of magnitude, and are also comparable to the network weights which are randomly initialized to values between zero and one.

7.3 Criteria of success

In the description of classifiers, I have so far left aside the problem how to compare the performance of different approaches. I finally discuss some measures that concisely describe the success of a two-class classification. In this case, members of one class are often referred to as “positives” — here, the promoter sequences —, whereas the members of the other class are called “negatives” — here, the non-promoters.

The outcome of a classification experiment leaves us with a number of correctly classified samples from both classes, the “true positives (TP)” and “true negatives” (TN). In the same sense, the mis-classified samples are called “false positives” (FP; members of the negative class that were put into the positive one) respectively “false negatives” (FN; samples from the positive class believed to be from the negative class). The *correlation coefficient* (CC) of an experiment is calculated using these four numbers; it is defined by

$$\mathrm{CC} = \frac{(\mathrm{TP}\cdot\mathrm{TN}) - (\mathrm{FN}\cdot\mathrm{FP})}{\sqrt{(\mathrm{TP}+\mathrm{FN})\cdot(\mathrm{TN}+\mathrm{FP})\cdot(\mathrm{TP}+\mathrm{FP})\cdot(\mathrm{TN}+\mathrm{FN})}}. \tag{7.16}$$

The values range between -1 and 1; a CC of one means perfect prediction, a CC of zero occurs for random predictions, and a CC of -1 shows perfect anti-prediction. The CC value depends on the proportion of negative and positive samples, which means that it cannot be compared across different data sets in general.

The sensitivity sn and specificity sp of a classification are defined as

$$\text{sn} = \frac{\text{TP}}{\text{TP} + \text{FN}} \cdot 100, \qquad \text{sp} = \frac{\text{TP}}{\text{TP} + \text{FP}} \cdot 100. \tag{7.17}$$

The specificity shows which part of the set of all predictions is actually true, and sensitivity describes how many of all positives were successfully identified by the classifier. A synonym of sensitivity therefore is “true positive rate” and we can define the rate of false positives, true negatives, and false negatives in the same manner.

For both neural network and Bayesian classifier, we can tune the classification towards the one or the other class, either by putting different thresholds on the neural network output, or by expecting a minimum distance between the *a posteriori* probabilities for the different classes. A single number that describes the performance of a classifier is the equal recognition rate (ERR); we tune the threshold to the value that leads to the same recognition rate for both classes, i. e. where the rate of true positives equals the rate of true negatives. Obviously, this threshold is arbitrary and might not correspond to the one that is finally used in a system.

To judge the performance in a more global manner, we can calculate the true positive rate for any given rate of false positives. A graph where the true positive rate is plotted on the y-axis against the false positive rate on the x-axis is called *receiver operating characteristics*. In our case, we plot the ROC curve from 0 to 100 percent in one-percent steps. After this, we apply the trapezoid rule to numerically compute the integral of the ROC curve. This leaves us again with a single number, but one that was computed over the full range of the classifier performance. The highest achievable value is 10,000 ($100 \cdot 100$, i.e. perfect recognition for all rates of false positives); a random classification results in a value of 5,000.

These are just some out of many different evaluation criteria; the reader is referred to Baldi et al. (2000) for a recent discussion.

Chapter 8

McPromoter: System, Experiments and Results

The focus of this thesis lies on computational methods for the identification of proximal promoter regions — and the transcription start sites contained in them — in eukaryotic genomic DNA. In the following pages, I describe the McPromoter system that I developed to solve this task. Several models of promoters are presented, owing to the ideas and concepts of a probabilistic modeling of biopolymer sequences introduced in the previous chapters. These models are now evaluated on real human and *Drosophila* data. At the beginning, though, stands the general design of the system.

8.1 General remarks

8.1.1 Outline of the system

The general outline can be seen in figure 8.1. Both strands of a contiguous DNA sequence are analyzed independently for promoter occurrences. A window of 300 base pairs is moved along the sequence in the 5'–3' direction, in steps of 10 base pairs. This sequence window is then evaluated by probability density functions of promoters, coding, and non-coding sequences. The results of the density functions are fed into and scored by a classifier. The output of the classifier then constitutes a graph along the sequence, which is smoothed by a simple filter method. Finally, a list of promoter predictions is delivered which corresponds to local maxima of the smoothed result score graph.

The following sections describe the application of different probability density functions and classifiers for both human and *Drosophila* promoter finding. I start with rather simple density

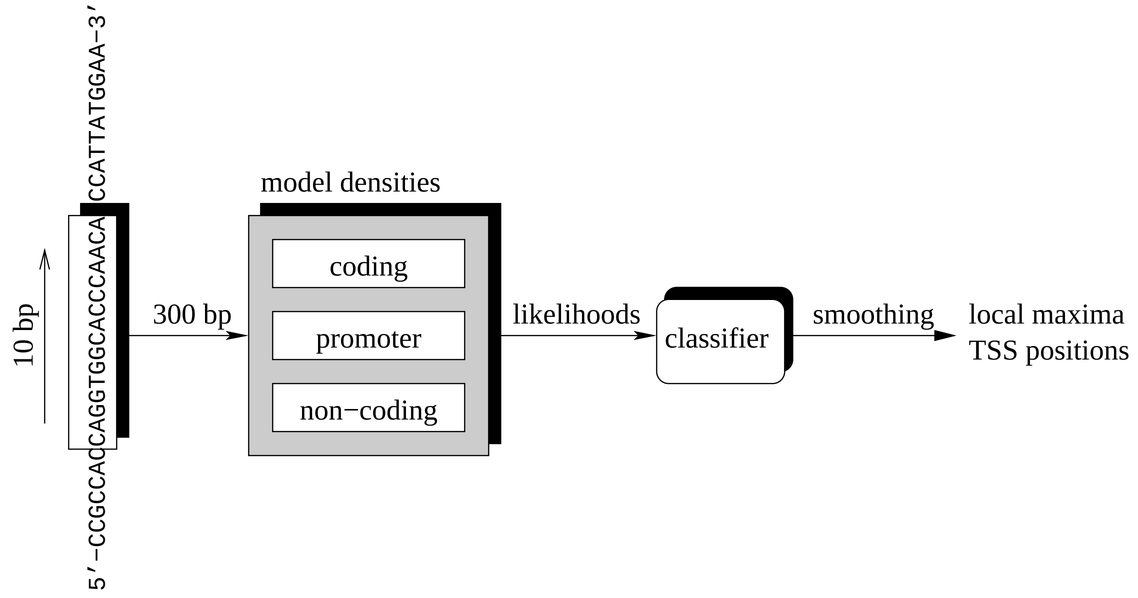


Figure 8.1: **Outline of the** McPromoter **system.** See text for explanation.

functions, and we will see whether and how more complex density functions, which represent the knowledge about the underlying biology more appropriately, improve on the problem of eukaryotic promoter recognition. Also, differences between the performance of *Drosophila* and human models will be pointed out. Before that, I discuss the design and evaluation of these experiments.

8.1.2 Experimental design and evaluation

The central part of the system is the classification of a 300 base pair long sequence into promoter or non-promoter. Therefore, two experimental setups are useful: The first evaluates the performance of the classifier on these fixed-length sequences, and the second one the entire system for promoter recognition in long genomic DNA sequences. In analogy to image processing problems, the first application can be seen as classification, the second as localization of an object against a variable background.

Evaluation of the classification. The parameters of the density functions and the classifier are estimated on the data sets of 300 bp long sequences described in chapter 4. To assess the performance of an approach, a large part of the data is used to estimate the parameters, and a different part serves as an independent test set. To obtain reliable results that are not biased by this arbitrary division in training and test set, I use cross-validation, i. e., repeat the experiment several times with different divisions in training and test set such that every sequence has been in the test set exactly once. Then the average over all experiments is computed and used as a result for comparison. For the human data, the experiments are carried out five times; in the case of *Drosophila*, where less data is available, only three times. If nothing else is stated, classification results always refer to these averaged cross-validation results. Measures that are used to describe the success of classification were introduced in section 7.3. For the assessment in this work, ROC curves and integral values will be used throughout to compare different approaches because they provide a global judgement of the quality of a given classifier. Nevertheless, I also provide the still popular equal recognition rates and correlation coefficients.

Evaluation of the promoter recognition system. The situation is different when we turn to the problem of identifying transcription start sites in genomic sequences: Here, we scan along a contiguous sequence which might contain one or more promoters at unknown locations, each of which might cause several neighboring windows to be classified as promoters. I therefore adopted the measures proposed by Fickett and Hatzigeorgiou (1997). They evaluated the success of promoter predictors by giving the percentage of correctly identified transcription start sites, the true positives, versus the false positive rate. A TSS is regarded as identified if a program makes one or more predictions within a certain “likely” region around the annotated site. In contrast to the definition for classification problems (see section 7.3), the false positive rate for localization problems is defined slightly different: It refers to the number of predictions within the “unlikely” regions outside the likely regions, divided by the total number of bases on both strands contained in the unlikely set. The FP rates are thus given per base; in other publications, they are sometimes given per base pair. In cases where the whole sequences are evaluated on both strands, the FP rate per base pair is thus twice the FP rate per base. Which region is regarded as likely depends on the knowledge about the annotation of the TSS that is available: For exactly annotated TSSs, Fickett and Hatzigeorgiou (1997) used a region from -200 to +100 around the annotated TSS, and the remaining sequence parts are regarded as unlikely.

On the large genomic Adh region from *Drosophila*, the TSSs are not experimentally confirmed but based on 5’ cDNA alignments. Here, I chose a larger region of 500 bases upstream and 50 bases downstream of the annotated TSS as the “likely” region. A similar scoring was proposed for the evaluation of human chromosome 22 by Scherf et al. (2001); but as the 5’ UTR regions can be very large in humans, they considered a larger region of -2000 to +500 as likely. The upstream region is always taken as the “likely” region, even if it could possibly overlap with a neighboring gene annotation on the same strand. The “unlikely” region for each gene then consists of the rest of the gene annotation, from the end of the likely region downstream of the TSS to the end of the final exon. Predictions in other parts of the sequences are ignored. Finally, the average distance of predictions from the annotated transcription start sites is used to assess whether the positional accuracy of TSS predictions changes for the different models under consideration.

8.2 Sequence-based models of promoters

The first group of promoter finding approaches presented in this work attempts to identify promoters with models for different classes of sequences. All the previously published systems discussed in section 3.3 belong to that category.

At first, promoter and background sequences are represented by different Markov chain models (cf. section 5.2): full-order, interpolated, and variable length Markov chains. I also discuss the benefits of different objective functions. The next section turns towards a modeling of promoter sequences by stochastic segment models (cf. section 5.3). Both of these approaches use a modified Bayesian classifier. In the final section, a neural network taking the output of the density functions as feature variables, replaces the Bayesian classifier to allow for non-linear dependencies. The different classification approaches are evaluated on human and *Drosophila* data throughout, and the section is closed by a comparison of the results obtained on genomic data.

8.2.1 Markov chain models

The general picture for the Markov chain model system is as follows:

- One Markov chain model each is used for promoter, coding (exon), and non-coding (intron) sequences (following the general Markov chain equation 5.6).
- A modified Bayesian classifier as described in section 7.1 is used as classifier (equation 7.6).

As the Markov chain models are not position-specific but stationary, we can expect that a promoter sequence causes a good score in a number of consecutive windows. The output of the classifier is therefore post-processed by a hysteresis filter (see section 6.2) that smoothes local maxima which are separated by only a shallow valley. The cursor width is chosen once by visual inspection (0.015), and then left constant for all applications.

Furthermore, each Markov chain for a background class is evaluated on both sense and anti-sense strand of a sequence, and the likelihood is a *mixture* of sense and anti-sense likelihood. The underlying reason for this is that we do not expect a promoter on the forward strand if we detect a coding or non-coding region on the reverse strand. The mixture weights are set to 0.5

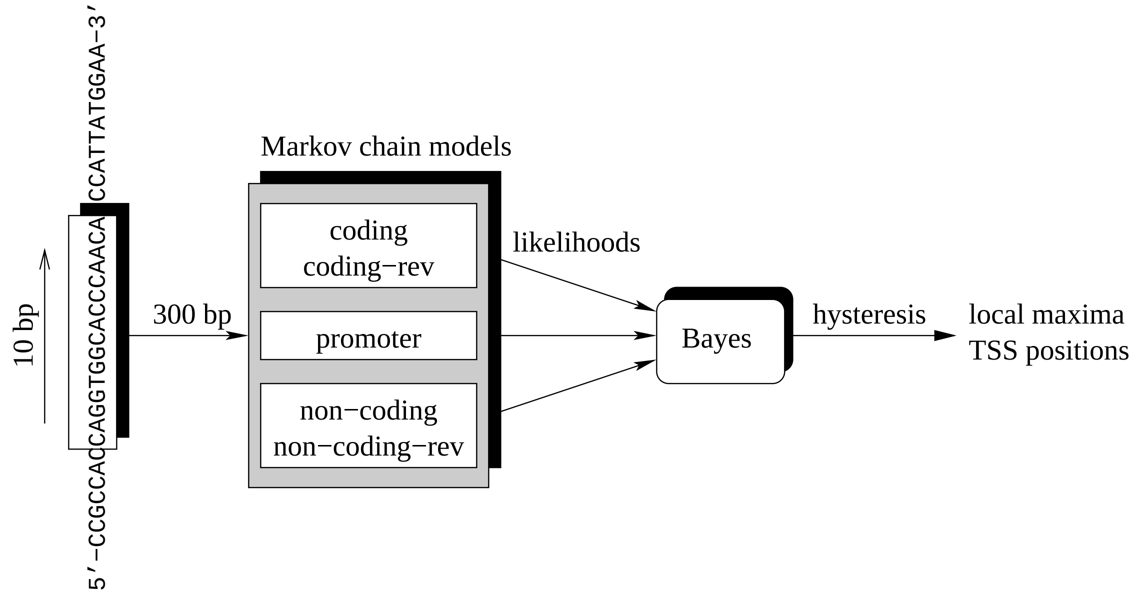


Figure 8.2: **The Markov chain promoter finding system.**

each; there is no general strand specific bias for the location of a gene. From a practical point of view, this leads to a strand-invariant background model which has to be evaluated only once even if we look for promoters on both strands of a sequence.

Figure 8.2 (cf. the system outline in figure 8.1) gives the refined system for MCPROMOTER — the Markov chain promoter finder.

Full-order Markov chains. The left part of figure 8.3 shows the receiver operating characteristics of the modified Bayes classifier with full-order Markov chain models and human sequences (section 5.2.1). The MC parameters are estimated with the Maximum Likelihood objective function (equation 5.10), discounted by one to ensure non-zero probabilities. All three models — promoter, coding, and non-coding sequences — have the same order. The ROC curve is averaged over all five cross-validation experiments on human sequences. We can see that the best result is obtained for fifth-order Markov chains; MCs of shorter order do not capture all the sequence characteristics, and MCs of higher order over-adapt to the training data and perform worse on unseen data.

Next, I examined the influence of an additional Maximum Mutual Information Training of the parameters (section 5.2.4, equation 5.25): After the initial ML estimation as above, the model parameters are refined with up to 30 iterations of the corrective training algorithm. 20 % of the training data are set aside as validation set on which the MMI function (equation 5.3) is evaluated

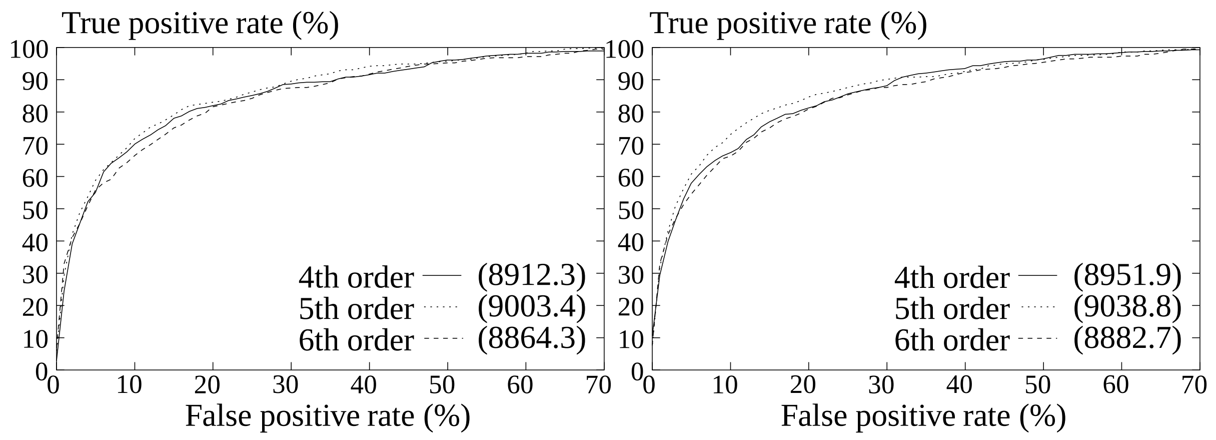


Figure 8.3: **Classification of human sequences with full-order Markov chain models.** Markov chains of different order are compared; the left picture shows the ROC results from ML estimation, and the right picture from an additional MMI estimation of the MC parameters. The best performing fifth order models achieve an ERR of 82.2 % (ML) and 82.4 % (MMI).

after every round. As soon as the objective function starts to get worse on the validation set, the estimation process is stopped. Figure 8.4 shows the convergence of the estimation process for the 5th order Markov chain models of one cross-validation experiment. With the help of corrective training and model interpolation using a constant weight of 0.98 for the old model and 0.02 for the new one, a steady convergence behaviour is achieved.

When we compare the MMI estimated MCs (figure 8.3, right panel) with the ones trained only by the ML objective function (left panel), we observe a small but clear improvement in classification: The ROC integral value for the fifth order models, which performed best, increases from 9003.4 to 9038.8, and the best correlation coefficient rises from 0.52 to 0.54. More notable than these small improvements is the observation that better results are achieved for Markov chains of every examined order, pinpointing the general usefulness of MMI estimation on human data.

The overall picture is the same when we turn to the ML training on *Drosophila* data (figure 8.5, left panel). But here, the MMI training does not lead to improved classification; the results are almost identical to ML estimated Markov chains. First of all, the MMI estimation is not guaranteed to deliver better results than ML. Second, the iterative nature of the MMI estimation carries an inherent danger to over-adaptation. Because the data sets are quite small, I use the same data for the ML initialization and the subsequent iterative MMI estimation. Therefore, the disjoint validation set, which is used to decide when to stop the iterative MMI estimation, has been used before in the ML estimation and is less suited than a completely independent

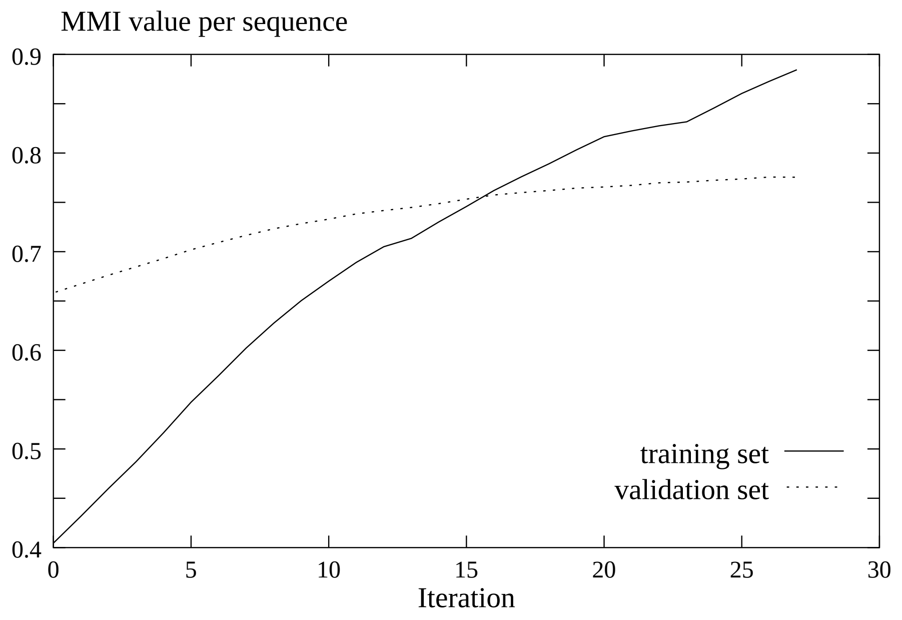


Figure 8.4: **Optimization of the MMI objective function.** This figure shows the average MMI value per sequence during a run of the iterative MMI parameter estimation on the 5th order Markov chain models for human sequences.

set. Besides, the ML estimated parameters of higher order MCs for *Drosophila* are so strongly adapted to the training set that they are able to perfectly classify it. Therefore, no training data are left for the corrective MMI training, which only looks at wrongly classified sequences.

So far, the order of the models is chosen after the evaluation on unseen data. It would be better if we could directly estimate the optimal order of a model from the training data itself, before it is applied on test data. One way to do so is to use the algorithm in figure 5.5: This algorithm performs a cross-validation on the *training data* to obtain estimates for the objective function applied on models of different order, and chooses the order that delivers the best estimate. Again, different objective functions are possible; estimates for the ML and MMI functions on human data obtained by five cross-validation rounds are given in table 8.1. ML selects fourth order models for exons and promoters and a sixth order model for introns — this reflects the fact that we have more intron data available than exon or promoter sequences. MMI model selection leads to fourth order models for exons and introns and a fifth order model for promoters. Both approaches are slightly worse than the best full-order models from above: ML delivers a CC of 0.51 and a ROC integral of 8915.3 with an ERR of 80.8 %, MMI a CC of 0.52 and a ROC integral of 8950.9 with an ERR of 81.3 %. The optimal ML models are used as concurring models in the evaluation of the MMI function.

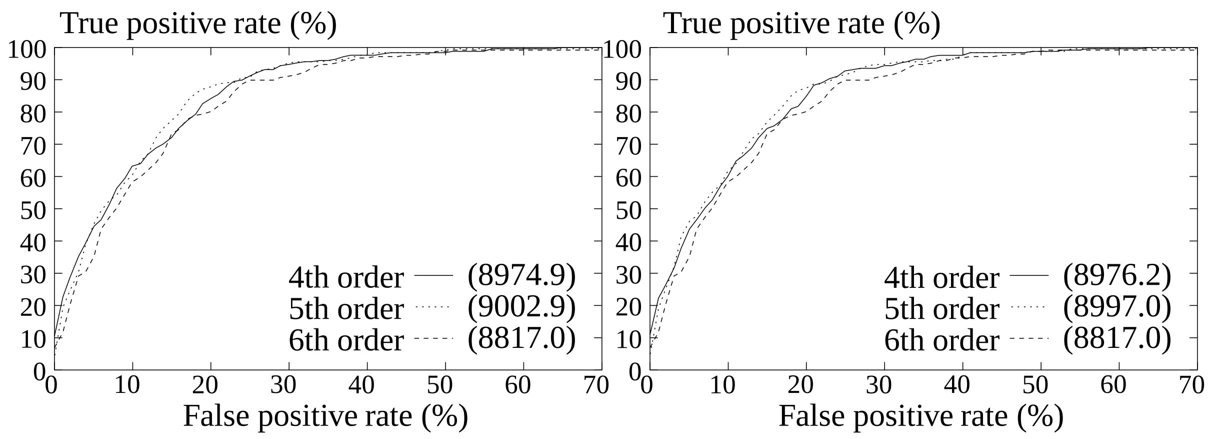


Figure 8.5: **Classification of fruit fly sequences with full-order Markov chain models.** Markov chains of different order are compared; the left picture shows the ROC results from ML estimation, and the right picture from an additional MMI estimation of the MC parameters. The ERR for the 5th order chains are 83.2 % (ML) and 82.9 % (MMI). The results for 6th order are identical for ML and MMI, as the corrective MMI training is no longer carried out due to perfect classification of the training set.

	ML/PPX			**MMI**		
MC	promoter	exon	intron	promoter	exon	intron
4	3.857–3.864	3.758–3.764	3.773–3.776	1.454–1.543	1.637–1.771	1.158–1.222
5	3.874–3.882	3.770–3.777	3.740–3.745	1.367–1.437	1.667–1.819	1.191–1.254
6	3.965–3.980	3.791–3.820	3.693–3.706	1.625–1.757	1.862–2.070	1.454–1.543

Table 8.1: **Optimal full-order model choice.** The table shows the range of the objective function estimates that was obtained on the five cross-validation experiments on human data, using full-order Markov chains. For the ML estimation, I compute the average perplexity per symbol, defined as $\mathrm{PPX}(\boldsymbol{w}) := e^{-1/T \cdot \ln R^{ML}(\boldsymbol{w})}$. The number of the MMI function refers to the average negative log MMI value per sequence, and the corresponding optimal ML full-order models were used as concurring models. For both ML and MMI, smaller numbers in the table thus refer to models that capture the data characteristics in a better way.

Interpolated Markov chains. Starting from the order which delivered the best results for standard MC models, I examined how interpolated models (section 5.2.2) of higher order performed on the classification task. For human data, the left side of figure 8.6 shows results with linear interpolation (equation 5.13), and the right side with rational interpolation (equation 5.14). In each

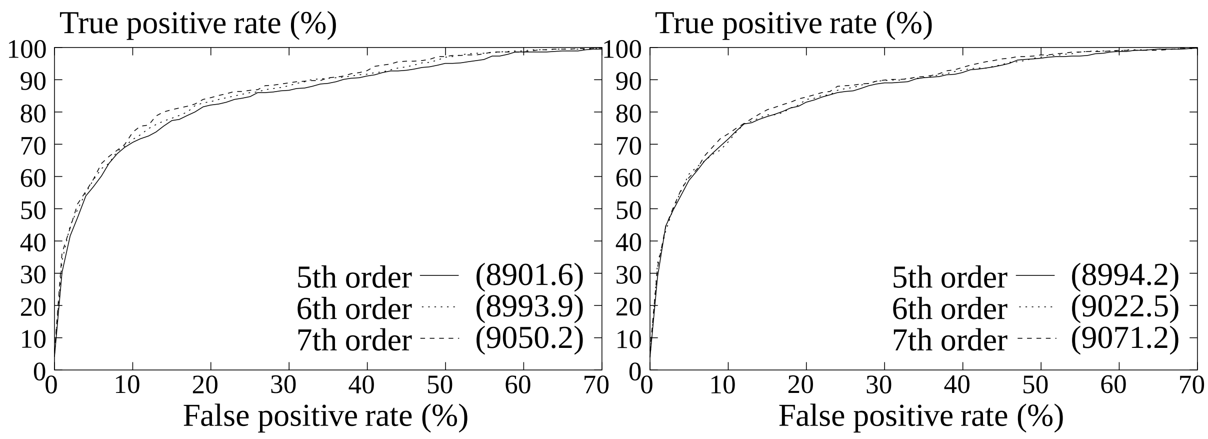


Figure 8.6: **Interpolated Markov chains for human promoter classification.** The left panel gives the ROC using linearly interpolated Markov chains, the right panel for IMCs using rational interpolation. The corresponding ERRs for the 7th order chains are 82.4 % for linear and 82.6 % for rational interpolation.

run, 20 % of the training set were set aside to calculate the interpolation coefficients. For fifth order models, we can see that the interpolated models perform worse than the full-order Markov chains. But the performance of interpolated models does not decrease with higher order as in the case of full-order MCs where over-adaptation is observed, and the overall performance gets better. Rational interpolation, which takes the reliability of each individual likelihood into account, delivers better results than the simpler linear approach. The same is observed for *Drosophila*, as can be seen from figure 8.7. Here we can actually see that the ROC measure delivers a different view on the quality of a classifier than the ERR measure: The ERR values are slightly lower for all IMCs when compared to the fifth order full Markov chain, but the ROC integral values, which judge the global performance independent of the desired false or true positive rate, rise significantly.

Using rational IMCs of 7th order as an example, I looked at classification results of human promoter versus only one non-promoter class at a time. A classification of promoters versus exon sequences is much more successful than a classification of promoters versus intron sequences: Promoter/exon classification delivers an equal recognition rate of 92.5 % (ROC integral: 9745.6), whereas promoter/intron leads to a significantly worse ERR of only 81.8 % (ROC integral: 9026.6). The combined classification has an ERR of 82.6 % and a ROC value of 9071.2, which is only slightly above the promoter/intron classification because the number of intron sequences in the set is much larger than the number of exon sequences. This underlines the fact that promoter and non-coding sequences contain only few significant patterns and are thus harder to

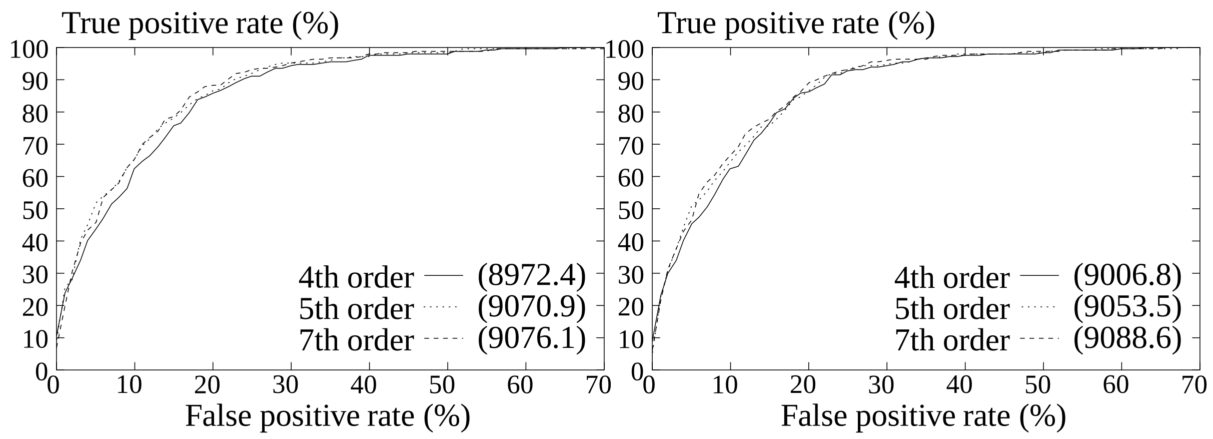


Figure 8.7: **Interpolated Markov chains for *Drosophila* promoter classification.** The left panel gives the ROC using linearly interpolated Markov chains, the right panel for IMCs using rational interpolation. The 6th order chains deliver only slightly different results than 5th order chains and are left out for clarity. The corresponding ERRs for the 7th order chains are 83.0 % for linear and 82.8 % for rational interpolation.

distinguish from each other, as opposed to coding sequences which display the well-known bias in codon usage and the periodicity of three.

Variable length Markov chains. An approach that copes with the exponential growth of the number of parameters with increasing model order was introduced in section 5.2.3 – Markov chains of variable context length. Learning VLMCs corresponds to a model structure optimization, and two similar algorithms were described that select model structures based on a local decision criterion. I first compare the VLMC models that are chosen by either the BW or the RST algorithm (see figures 5.4 and 5.3), using the MMI objective function. To evaluate the MMI function, models of concurring classes are needed; here, I chose the optimal full-order models according to an ML cross-validation (see table 8.1), which avoids the danger of over-fitted concurring models. With this approach, it is feasible to optimize one model at a time, as the concurring models are left constant throughout the optimization.

Figure 8.8 shows an example optimization of the exon and promoter VLMCs of one cross-validation experiment with the BW algorithm. The optimization runs over the local cut-off parameter K, which is increased from 0 to 7 in steps of 0.25. The objective function is estimated by repeated cross-validation estimation on the training data (see figure 5.5). For a fifth order chain, a smaller cut-off value is obtained than for a sixth order chain, as we do not have to cut back the tree as much. The sixth order models show clearly that low values of K at first lead to

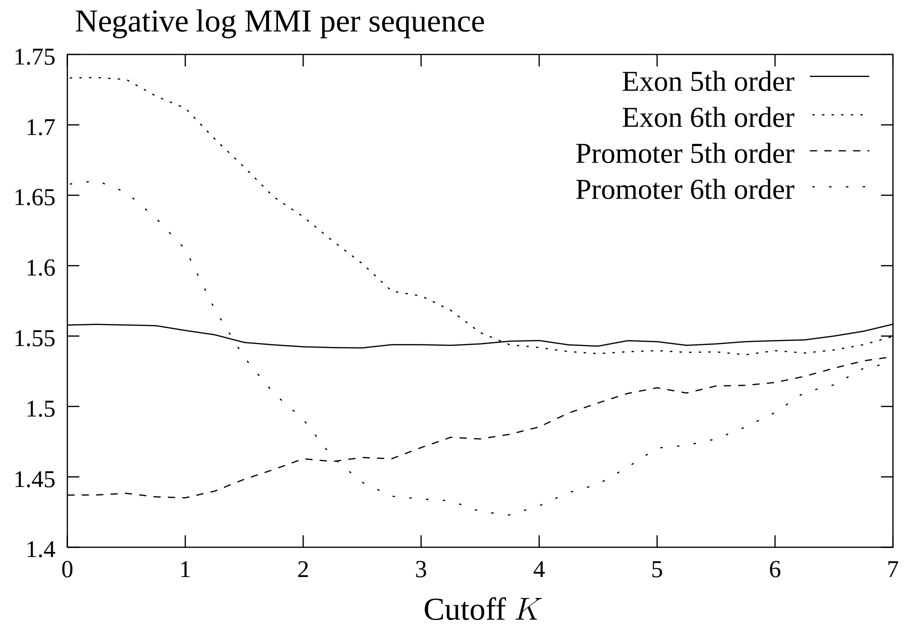


Figure 8.8: **Optimization of the BW algorithm cut-off value.** For human promoter and exon VLMC models of 5th and 6th order, the MMI objective function is plotted against the cut-off value K that is used in the BW algorithm.

over-adapted trees, then to a certain optimal K value, and finally to small and non-descriptive trees at large values of K. The minimum number of occurrences that a context must have to be included in the tree is constantly set to 10. For the RST algorithm, we face a two-dimensional optimization problem over the minimum probability, which is varied from 0 to $2 \cdot 10^{-3}$ in steps of 10^{-4}, and the difference ratio r which is varied from 1.05 to 1.25 in steps of 0.05. As mentioned in section 5.2.3, I used the same simple discounting in the final parameter smoothing step that was also used in the BW algorithm (figure 5.4). This eliminates the smoothing parameter $\gamma_{\min}$ which is used in the original RST algorithm (figure 5.3).

The number of nodes of the context trees built by the VLMC training algorithms are given in table 8.2. Some tendencies are noteworthy: First, a tree of low order such as 4th order is hardly pruned; second, a tree representing a class for which a larger data set is at hand is pruned less than those for classes with less data, as can be seen when comparing the BW trees for introns with those for exons. With the exception of the 6th order promoter models, the RST algorithm delivers smaller trees.

Altogether, the optimization does not completely behave as expected: For example, the number of nodes should never decrease with the model order (as is the case for promoter models of fifth and sixth order in the BW algorithm) — if no additional information can be gained, it should

	full	**BW**			**RST**		
order	model	promoter	exon	intron	promoter	exon	intron
4	341	214–336	332–341	313–338	185–338	297–329	281–309
5	1365	979–1365	363–1194	1272–1365	524–1136	257–997	780–1123
6	5461	544–768	413–1074	2021–2846	1254–1831	274–1167	561–1052

Table 8.2: **Size of VLMCs of different maximum order.** For both BW and RST learning algorithm, the range of node number of the optimal trees that were found in five cross-validation runs on human data is shown.

	BW			**RST**		
	promoter	exon	intron	promoter	exon	intron
MMI val.	1.4351	1.5416	1.2513	1.4323	1.5334	1.2179
# nodes	979	559	1273	1105	996	796

Table 8.3: **Comparison of VLMCs learned by different algorithms.** For one cross-validation experiment on human data, the average negative log MMI values of the optimal 5th order VLMCs for the classes promoter, exon, and intron are given, along with number of nodes in the trees.

stay the same. It must be noted, though, that trees of considerably different sizes can lead to similar values of the objective function: Table 8.3 compares the BW and RST algorithm related to the optima of the objective function and the corresponding trees. This behaviour might result from the current MMI training approach: The concurring models are left constant throughout the optimization, which is only an approximation of the real MMI objective function. But when the models are finally used for classification, an MMI optimized model is used together with the other optimized models. It would thus be advantageous to explore a *simultaneous* optimization of all models. This leads to a combinatorial optimization problem over possible model structures, i. e. over the presence of parameters and not over their values. There is no derivation of the objective function with respect to the presence or absence of parameters, and efficient gradient descent approaches cannot be used. It was therefore not studied further in this work.

Figure 8.9 finally shows the results for the BW and RST algorithm. We can see that BW and RST perform about the same; the VLMC trees of higher maximum order are indeed better than the full-order trees (cf. figure 8.3). On the other hand, we still observe that trees of higher order start to over-adapt to the data, i. e. have lower ROC integral values than models of smaller order. A fully compensatory effect, as we saw for the application of interpolated Markov chains, is not

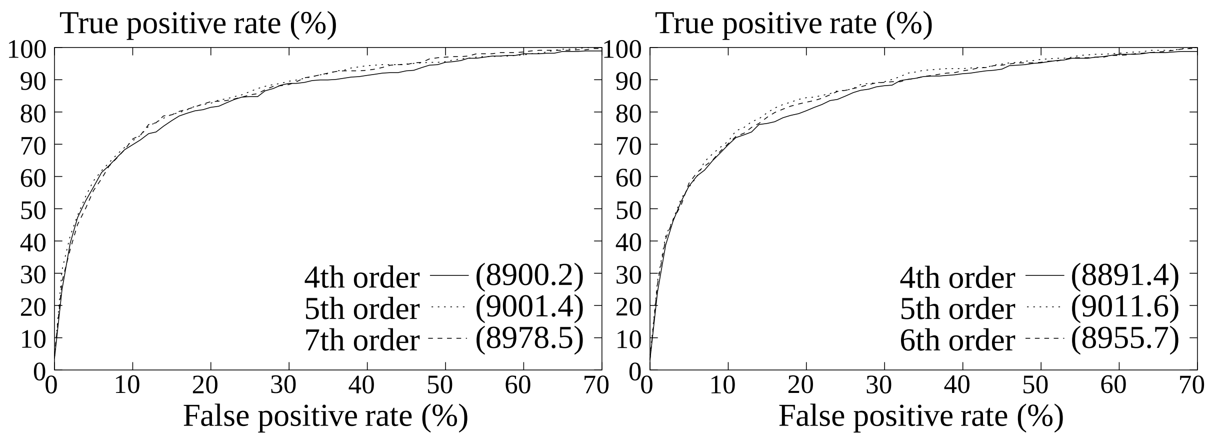


Figure 8.9: **Variable length Markov chains for vertebrate promoter classification.** For both experiments, the MMI objective function was used. The left panel gives the ROC using the BW training algorithm, the right panel for the RST algorithm. In the case of BW, we show the results obtained on 7th order chains. They deliver only slightly different results than 6th order chains which are left out for clarity. For the best-performing 5th order models, an ERR of 81.9 % (BW) respectively 82.6 % (RST) was observed.

yet obtained.

The same holds for the application on fruit fly data. In the left panel of figure 8.10, the application of the BW algorithm using the MMI objective function is depicted. Again, the VLMCs perform better than full-order trees, but show a tendency to over-adaptation. The right panel shows the BW algorithm using the ML objective function; it can be seen that ML performs worse than MMI, which was observed in all other experiments on VLMC structure optimization as well.

Summary. The classification of promoter sequences using Markov chain models is already quite successful, considering that no information about the structure of promoter regions is incorporated into the models yet: The best models achieve an equal recognition rate of 82.6 % for human and 83.2 % for *Drosophila* sequences.

As expected, there is an optimal order for full-order Markov chains for which they are well adapted to the training data without losing generality. An MMI training of parameter values leads to an improved classification result on human data; on fruit fly data, the overall results for ML and MMI estimation are the same.

Interpolated Markov chain models lead to the desired effect that we can increase the model order without over-adaptation to the data. In some few cross-validation runs, a decline in clas-

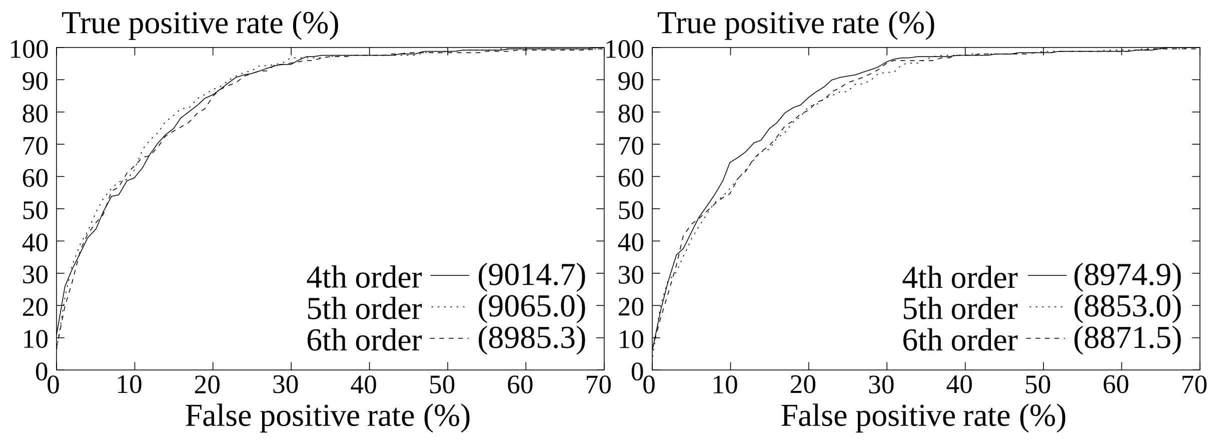


Figure 8.10: **Variable length Markov chains for *Drosophila* promoter classification.** For both experiments, the BW algorithm was used. The left panel gives the ROC using the MMI objective function, the right panel for ML. The best MMI optimized result (5th order) achieves an ERR of 82.6 %, the ML optimization (4th order) an ERR of 81.7 %.

sification performance is observed, which is due to the local optimization of interpolation coefficients and the fact that better ML values do not necessarily lead to an improved classification. Nevertheless, the average classification results are considerably better when compared to full-order Markov chains — for human sequences, we observe a ROC integral of 9071.2 instead of 9003.4.

This is also the case for variable length Markov chains: they lead to equal or better results when compared to full models of the same order in all cases. However, over-adaptation effects are still observed, and the classification is less successful than with interpolated Markov chains. This does not agree with the VLMC results reported by Bejerano and Yona (2001), where the authors achieve an increasing classification rate when increasing the maximum model order. However, the authors use manually selected parameters of the RST algorithm, and they work on protein domains, which contain well-conserved and class-specific subsequences of up to 30 amino acids. This is different from DNA sequence classes, where we do not encounter patterns as large as in proteins, and where the relatively short patterns are thus still likely to occur in sequences of other classes simply because the size of the alphabet is very small.

8.2.2 Stochastic segment models

A eukaryotic promoter does not consist of one large region with the same nucleotide distribution at every position — this is assumed in the modeling approach presented in the last section. Rather,

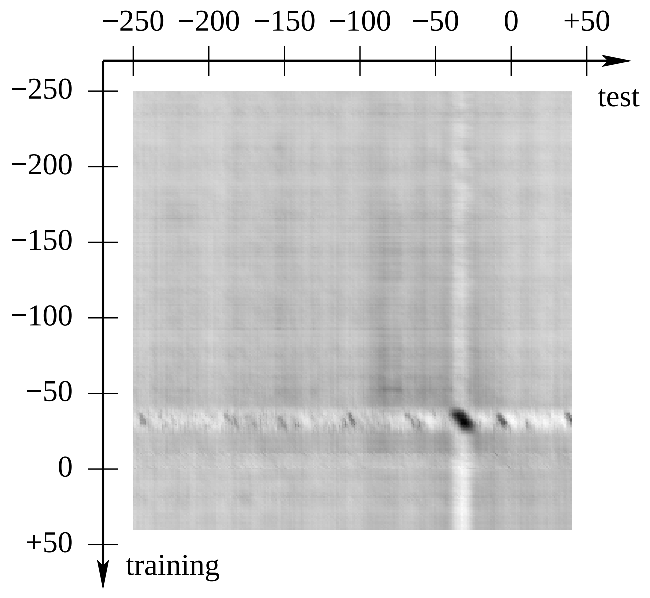


Figure 8.11: **Detection of conserved regions in human promoters.** On the vertical axis, the starting position of the window on which a model was trained is given. The horizontal axis depicts how well the model trained on a certain window position performed in all windows. See the text for further explanation.

it can be divided into segments (see section 3.2): the region upstream from the transcription start site, the core promoter where the main initiation complex binds, and a region downstream from the start site. The core promoter can be further split into the TATA box and the initiator region, separated by a spacer of approximately 15 bp. We use this broad segmentation of a pol-II promoter region to pursue an approach for promoter recognition based on a stochastic modeling of promoter segments.

To determine an initial promoter model structure, I performed the following experiment (Ohler et al., 2000). A window of 10 bases was shifted along the human promoter sequences in the training set of a cross-validation experiment. At each position, a second-order interpolated Markov chain was trained with the window content of all sequences. This model setting ensures that the model is not over-adapted, but is able to capture specific sequence elements. The model was then evaluated at every position of the remaining sequences, again within a window size of 10 bases. All the scores were summed up for each window, normalized and plotted against the position on which the window was trained (figure 8.11). High scoring windows appear in a dark color, and if a dark region appears on the diagonal, it indicates a position specific signal within the promoters which can be detected by the model. The only clearly visible position-specific

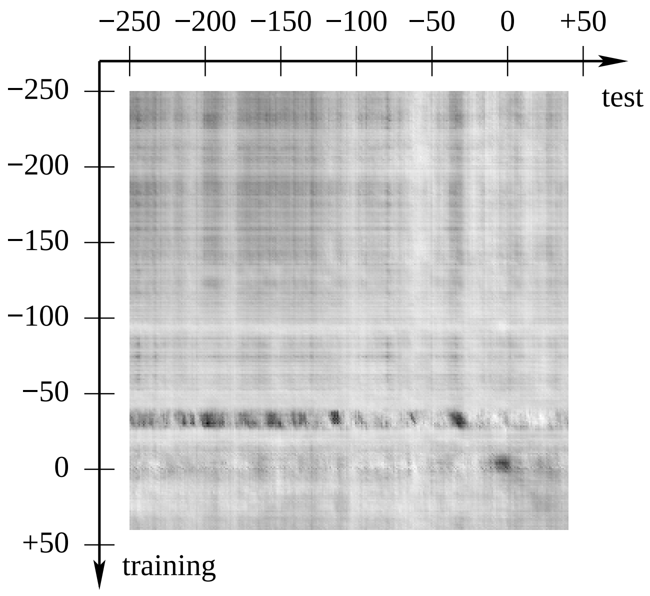


Figure 8.12: **Detection of conserved regions in fruit fly promoters.** On the vertical axis, the starting position of the window on which a model was trained is given. The horizontal axis depicts how well the model trained on a certain window position performed in all windows. See the text for further explanation.

signal is the TATA box region. Even at the TSS itself, there is no clear sign that the models trained on this region perform better than models trained on a different part of the promoter. This is somewhat surprising, but in accordance with the results of Zhang (1998), who found that TATAAA is the only clear position specific six-tuple within promoters. Obviously, the window size of 10 bases is too small to detect *region*-specific signals, such as transcription factor binding sites which occur more frequently in specific parts of the upstream region.

This effect is also observed on the *Drosophila* dataset (figure 8.12), although we can see that the rather GC-poor region upstream of -100 (cf. figure 6.5) scores better under the TATA box model than other parts outside the TATA box region. In fruit fly promoters, we can also detect a clear conservation of the initiator pattern, which appears as a second conserved black box downstream of the TATA box region. These results suggest a modeling of promoters with a stochastic segment model as introduced in section 5.3. For the human model, at least three states — upstream, TATA, downstream — are suggested by figure 8.11; in the case of the *Drosophila* model, at least five states — upstream, TATA, spacer, initiator, and downstream.

Interpolated Markov chain sub-models. To determine a suitable model topology, I compared fruit fly and human promoter models with one, three, five and also six states. Looking at the GC content in promoter sequences (figure 6.5), an additional sixth state, which comes into play by splitting the upstream region into two separate equally sized states, can account for the quite different base composition from -250 to -150 and from -150 to -50. It is possible to provide rough upper and lower bounds on the segment lengths: the position of the initiator is known from the data, and the overall range of spacer lengths between TATA and initiator can be taken from the literature. Also, Bucher (1990) examined the size of the core promoter patterns. The length distributions are initialized uniformly. As submodels, I use rational interpolated Markov chains since they performed slightly better than the alternative approaches discussed in the previous sections. The IMC order is chosen by hand — smaller order for smaller segments — and the parameters are initialized on rather large subsequences containing the promoter parts which they stand for. The SSM is trained with four iterations of the Viterbi training (figure 5.10), and the likelihood is therefore also computed using the Viterbi algorithm (figure 5.9) and not the forward algorithm (figure 5.7). Using the likelihood of the best path only also makes sense from a biological point of view, where we expect that a pattern such as the TATA box is present at a particular position within the sequence. The background models stay the same for all experiments, i. e. an IMC for coding and non-coding sequences each, 7th order for the more extensive human set and 5th order for *Drosophila*. Higher orders do not increase the performance.

We can see in figure 8.13 that the greatest leap forward is made by splitting up the promoters into three regions, as figures 8.11 and 8.12 suggested. A further split does not change the overall results considerably — for *Drosophila*, a slight improvement is observed, for human, a slight deterioration. In the following, I proceed with six segments for both species, as this will give us more flexibility when we turn to more complex models. Figure 8.14 shows the resulting promoter model that replaces the single Markov chain in the system of figure 8.2, leading to the changed system of figure 8.15 where an SSM has replaced the simple promoter Markov chain, and a median of width three is taken instead of hysteresis filtering. As the SSM contains explicit promoter states, we expect that not so many neighboring windows as with MC models score high, and the hysteresis approach is not suitable any longer (Ohler, 2000).

Full-order Markov chain sub-models. As we saw in the previous section (for example in figure 8.6), interpolated Markov chains are not prone to over-adapt with increasing context length. When we use full-order Markov chains, we have to estimate the optimal model order by a cross-validation estimation of an objective function on the training data to prevent over-adaptation (table 8.1). On using this approach to train the segment model on the ML objective function, we obtain a structure of the models as given in table 8.5. This optimization is part of the Viterbi

model	**state**	τ_{min}	τ_{max}	**order**
3-state	upstream	205	230	5
	TATA	10	20	3
	downstream	50	85	4
5-state	upstream	205	230	5
	TATA	10	20	3
	spacer	10	20	2
	Inr	5	15	3
	downstream	35	50	4
6-state	upstream 1	100	115	4
	upstream 2	105	115	4
	TATA	10	20	3
	spacer	10	20	2
	Inr	5	15	3
	downstream	35	50	4

Table 8.4: **Structure of stochastic segment promoter models with different numbers of states.** Shown are the minimum and maximum for each length and the Markov order of each output distribution (Markov chains with rational interpolation).

training algorithm, i. e. the model order is optimized after each iteration when the submodels are re-estimated. As the segment lengths have rather tight upper and lower bounds, though, the optimal model orders do not change in course of the training.

At this point, we can also look at the strength of the patterns that are represented by the states of the segment model, corresponding to the perplexity values that were determined by the ML cross-validation during the last training iteration. Table 8.5 shows these perplexity values of the six promoter states for human and *Drosophila*. We can see that initiator and TATA box models are clearly stronger, i. e. they have smaller perplexity values than the other sub-models, and that the initiator is considerably stronger in *Drosophila* than in human, especially in comparison with the TATA box. The overall perplexity values are lower in human, as we have more training data at hand and are able to use higher order models.

When we turn to the MMI objective function to estimate Markov chain orders, we encounter a potential problem because of the segmental model structure. Can we regard the submodels in other segments as concurring models, just like the models for the other classes? Figure 8.16 shows the results that are obtained when only the exon and intron models are used as constant

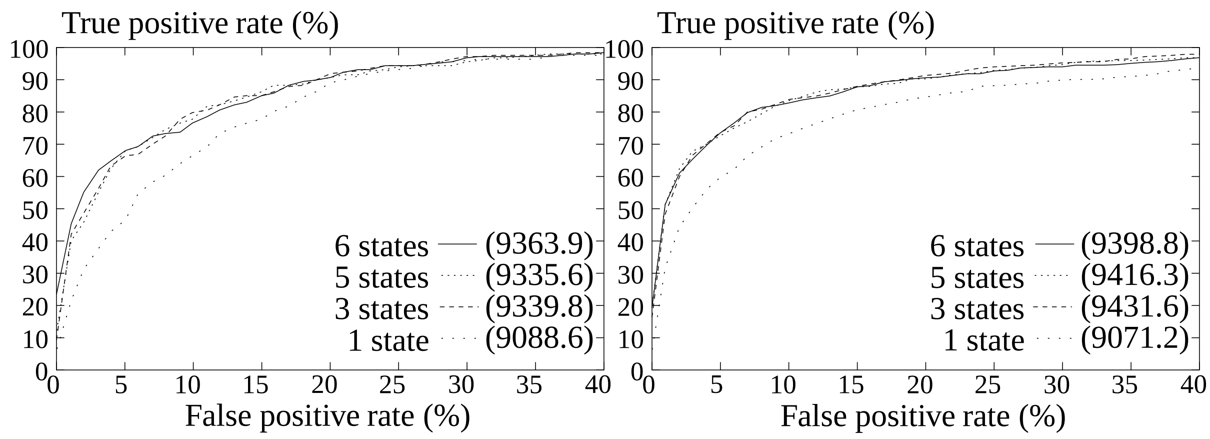


Figure 8.13: **Stochastic segment models for *Drosophila* (left) and human promoter (right) classification.** For both experiments, rational interpolated Markov chains were used; the model specifications are given in table 8.4. The 1-state promoter model corresponds to the 7th-order interpolated Markov chain experiments (figures 8.6 and 8.7). The best ERRs are achieved for the 5-state models: 85.6 % for *Drosophila* and 86.9 % for humans. For the 6-state models, the ERRs decrease slightly to 85.0 % and 86.1 %, respectively. The best CC is 0.64 for *Drosophila* (6-state model) and 0.67 for human (5-state model).

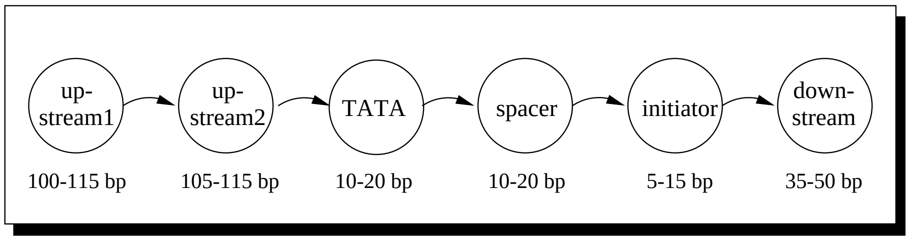


Figure 8.14: **The stochastic segment model for promoters** used in the McPromoter system, from Ohler et al. (2001).

background models for each of the segments. After that, the whole segment model is used as constant concurring promoter model when the exon and intron models are optimized. As always, the optimal full-order models in the sense of ML are used as background models. Figure 8.16 shows that this MC order optimization according to MMI does not deliver better results in the context of SSMs. As in the case of VLMCs, a simultaneous optimization of the models might be better than using constant background models. Moreover, the Viterbi training is only guaranteed to optimize the objective function when the submodels are estimated using ML (equation 5.38).

We can also see from figure 8.16 that the results for ML-optimized full-order MC submodels

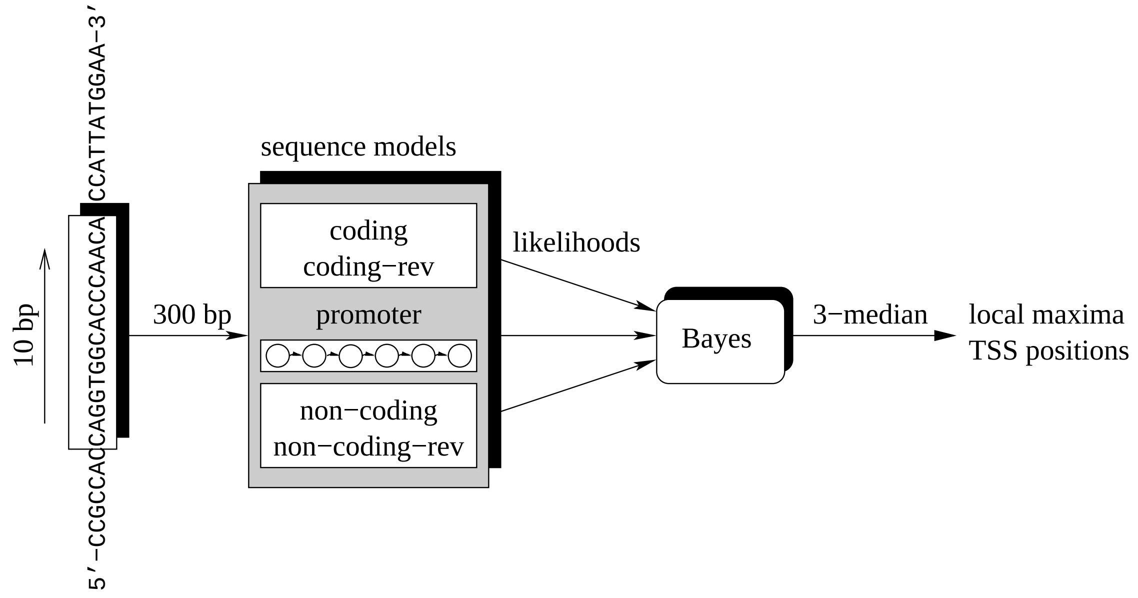


Figure 8.15: **The Markov chain promoter finding system with a stochastic segment promoter model.**

model	**Drosophila**		**Human**	
state	*order*	PPX	*order*	PPX
upstream 1	2	3.858	3	3.900
upstream 2	3	3.917	4	3.826
TATA	2	3.356	3	3.063
spacer	2	3.869	2	3.629
Inr	2	3.276	2	3.442
downstream	2	3.900	3	3.793

Table 8.5: **Structure of the optimal full-order stochastic segment promoter models.** Shown are the optimal Markov order of each output distribution and the perplexity values estimated during the last training iteration of one cross-validation experiment.

are competitive with the IMC submodels (cf. figure 8.13): for *Drosophila*, slightly better ERR and ROC integral values are obtained, for human, slightly worse. I therefore use the full-order Markov chain submodels on the evaluation of the large human data sets, especially because the runtime for interpolated Markov chains is considerably larger than the current implementation for variable-length Markov chains. The IMC software was developed for speech recognition

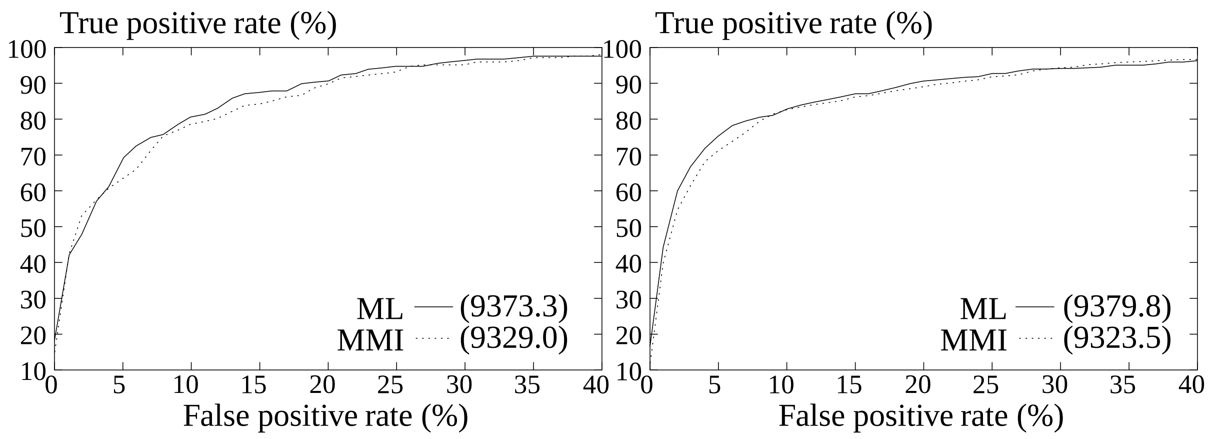


Figure 8.16: **Six-state Stochastic segment models with full-order Markov chain submodels.** On the left, the results with optimized full-order Markov chains for *Drosophila* are given; on the right, the same for human promoter sequences. The model specifications resulting from the ML estimation are given in table 8.5. The ERRs for these models are 86.5 % (*Drosophila*) and 86.1 % (human).

applications, where we usually encounter a large vocabulary and a rather small context — typical values are 5,000 words and second-order models. In this case, we cannot use a tree structure to store the likelihoods, as each node in a context tree contains the values for all the words in the vocabulary, even if the word was never observed in the particular context. The implementation of interpolated Markov chains that is used in this application (Schukat-Talamazzini et al., 1997) therefore stores the counts of all observed word chains with a length up to the context size plus one in a hash table, and calculates the conditional probability anew for each symbol in a sequence. This is clearly hazardous when we have large contexts and large sequences like in our application. On the other hand, the context tree of the full-order Markov chains stores the conditional log likelihoods and is therefore much faster. In table 8.6, user time in seconds is compared for the human models, both for Markov chain models of sixth order and segment models with six states, with sub-model orders as in table 8.5. We look at the likelihood calculation of one 300 base pair sequence versus a 10,000 base pair sequence evaluated with a window of 300 base pairs moving in steps of 10 base pairs. All runtime considerations of section 5.3.3 were taken into account. The VLMC implementation is faster by almost an order of magnitude, even before a conversion into an automaton (see appendix B) which would lead to an additional speed-up.

sub-model	**MC model**		**SSM model**	
	300 bp	*10,000 bp*	*300 bp*	*10,000 bp*
IMC	0.06	1.79	0.45	322.76
VLMC	0.008	0.23	0.08	58.96

Table 8.6: **Runtime for Markov chains and segment models.** We compare the user time (in seconds) that is needed on a 400 MHz Pentium II PC run under the Linux operating system.

MMI parameter optimization. After an initial ML training of the segment model, we can re-estimate the model parameters according to MMI. As a derivation of the MMI re-estimation equations has not yet been described for stochastic segment models, we simultaneously optimize all Markov chain models, i. e. the promoter sub-models and the background models. Instead of discriminating between the three classes promoter, exon and intron, the aim is therefore a correct classification of the six promoter sub-models plus the two background models. On the one hand, this could be advantageous to localize appropriate patterns for the sub-models; on the other hand, it might “distract” the algorithm from the more important discrimination between promoter and non-promoter sequences. The results confirm this ambiguity: For human sequences, the ROC integral increases slightly from 9379.8 (ML) to 9383.4 (MMI); for fruit fly sequences, it decreases from 9373.3 to 9361.8.

Summary. The classification with stochastic segment models improves significantly on the classification with any kind of Markov chain model. The apparent reason is that different density functions now represent different parts of the promoters which leads to a more exact modeling of the promoter structure. Both full-order Markov chains and interpolated Markov chains deliver comparable results, but the application of the MMI objective function leads to conceptual problems because the promoter is represented by the more complicated segment model. Altogether, the classification results could be improved from ROC integral values of close to 9,100 to values of 9,350 and more. The best ERRs using interpolated Markov chains are 85.6 % for *Drosophila* and 86.9 % for human sequences, an improvement of more than 2.5 percent points for *Drosophila* and 4 percent points for human when compared to the best single state interpolated Markov chain promoter models. The best correlation coefficient rises from 0.54 to 0.67 for the human and from 0.52 to 0.64 for the *Drosophila* sets.

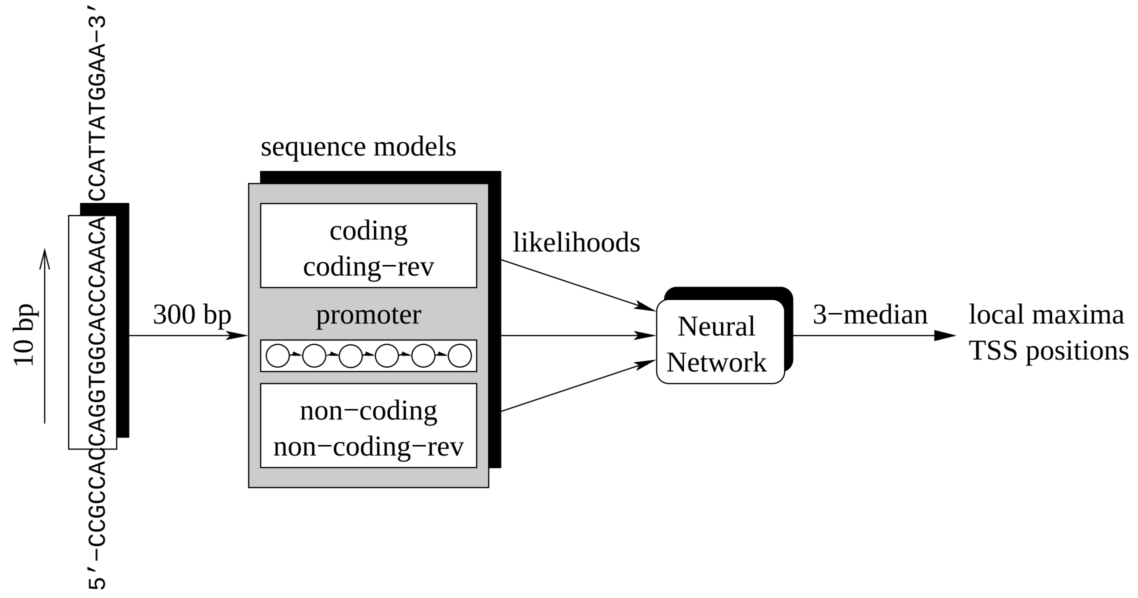


Figure 8.17: **The promoter finding system with neural network classifier.**

8.2.3 Neural network classifier

So far, the modeling of promoters allows for dependencies within a model state, but conditional independence is assumed between the states. This might not reflect the biological reality — studies have shown that at least in *Drosophila*, there are dependencies among the states, namely between TATA box and initiator or TATA box and downstream promoter element (Kutach and Kadonaga, 2000). If one of them is weakly conserved, it is much more likely that the other one is strong and will obtain a good score under the model.

One way to account for these dependencies is to use a different classifier. A stochastic segment model combines the log likelihoods obtained for each sub-model in an additive and unweighted fashion (equation 5.33), and this final likelihood is used by the modified Bayesian classifier of equation 7.6. Instead, a multi-layer perceptron is used from now on. It takes the promoter and background likelihoods and the likelihoods produced by each state as input, and is therefore able to respect arbitrary dependencies between the promoter parts (figure 8.17).

Figure 8.18 shows the resulting feed-forward network. It has nine input nodes, nine (human) respectively six (*Drosophila*) hidden nodes, and one output node. The network is trained on a different part of the training set than the segment models: first, I take half of the training data to establish the probabilistic promoter and background models, then these models calculate the likelihoods of the sequences in the other half. For each of the two background classes, the models are again evaluated on both strands, and mixtures of forward and backward likelihood with uni-

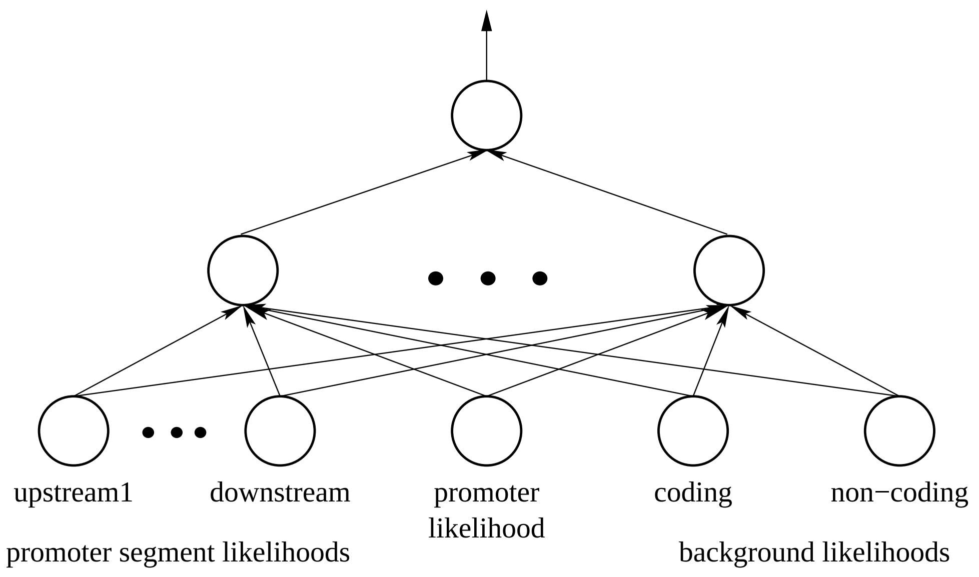


Figure 8.18: **The neural network classifier for sequence model densities.** The feed-forward NN takes the likelihoods of each promoter state as well as the total likelihoods of promoter and background models as input, leading to a total of nine input nodes. The human network uses nine nodes in the hidden layer, the *Drosophila* network six.

form weights of 0.5 are presented to the coding and non-coding input nodes. All likelihoods are normalized by sequence respectively segment length and then presented to the neural network. At this step, the data are pre-processed as described in section 7.2.3.

To train the neural network, I used standard online back-propagation, i. e. the weights are updated after each training sample (equation 7.12. The learning rate η is set to 0.005, and a threshold ν of 0.05 is used to decide whether a sample is back-propagated or already close enough to the desired output value. The SNNS package that was used to train the networks (see appendix D) only provides mean square error as error function for back-propagation; however, I also used conditional entropy (equation 7.14) to evaluate the network on the validation set and stopped when the best network according to CE was obtained. A third of the network training data was set aside as validation set. The number of nodes in the hidden layer as well as the parameters of the training algorithm were varied to make sure that small changes in the topology and in the algorithm settings also resulted in small changes of performance.

Figure 8.19 compares the results of the neural network classifier with the results of the best interpolated Markov chain and segment models. Even though the segment model is now trained on only half of the data, an overall improvement is visible. For *Drosophila*, the available training data is more limited, and the two-step training process runs into problems for the cross-validation

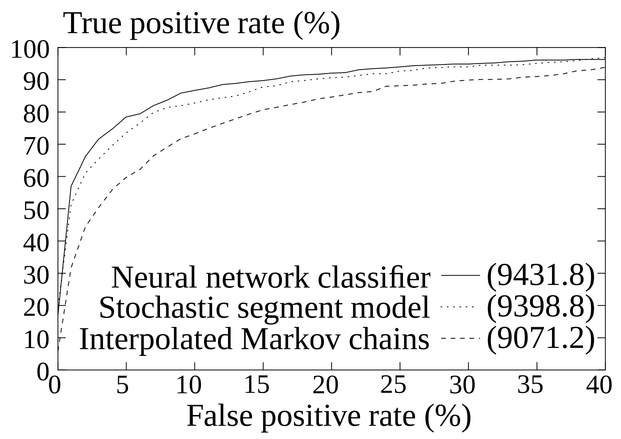


Figure 8.19: **Results of the neural network classifier on human data.** The picture compares the five-fold cross-validation results of the neural network that takes the output of the state densities as input variables with the ones of the Bayes classifier using the likelihoods computed by the six-state segment model and the interpolated Markov chains. The ERR obtained with the NN classifier is 88.3 %, the CC is 0.69.

experiments: The segment model is trained on 82 sequences only, which makes it much less reliable as before. For the training of the neural network, the remaining training set of 82 promoters is used, a third of which is set aside for an independent validation and early stopping. Not surprisingly, this limited set did not result in better classification results. Nevertheless, we will see that an SSM/NN model improves on the evaluation of *Drosophila* genomic data where we can make use all data for model training and do not leave aside sequences for the cross-validation test of the classifier.

Summary. Instead of a Bayesian classifier, a multi-layer perceptron is now used to combine the promoter state and the background likelihoods in a nonlinear fashion. The training sets are therefore split in two parts, to train the parameters of network and segment model independently. For *Drosophila*, the resulting data sets are too small to obtain reliable cross-validation results. For human, an ROC integral improvement from 9398.8 to 9431.8 is observed for the six-state model (equal recognition rate of 88.3 % instead of 86.1 %).

8.2.4 Localization of promoters with sequence models

To assess the quality of the densities and classifiers presented up to now, different genomic data sets were collected and described in chapter 4. On these data sets, the best interpolated Markov chain (IMC, section 8.2.1) and promoter segment model (SSM, section 8.2.2) with the Bayesian classifier as well as the neural network taking the likelihoods as features (NN, section 8.2.3) are compared. At the beginning, we look at the evaluation on human sets with known transcription start sites.

The first one is the set of 24 exactly mapped transcription start sites in 18 vertebrate sequences, collected by Fickett and Hatzigeorgiou (1997). Of these 24 TSS, some cannot be expected to be detected by the McPromoter system at all:

- McPromoter uses an input window of 300 bases. Two TSS are too close to the beginning (28 respectively 143 bases) so that we do not have a full 300 bases input window. The argument of Fickett that all predictors have to cope with this situation is not valid any more now that we have completely assembled chromosomes.
- McPromoter only retains the best of possibly multiple hits within 300 bases. In one sequence, there are three TSSs within 295 bases, and in the case of an exact localization, only the best would thus be kept. Because a hit is counted if it falls within the region from -200 to +100, there might be cases where two of these TSSs can be distinguished from each other if their predictions are separated by more than 300 bases.

As a “real” positive set, we can therefore only consider 21 out of the 24 annotated TSSs. Table 8.7 shows the results for the different approaches described above, each evaluated on a number of thresholds. The table shows the number of true and false positive predictions, as well as the number of close hits for one threshold; these are predictions less than 20 bases up- or downstream from the annotated TSS. The false positive rate is given per base and not per base pair, as the predictor is applied independently on both strands. For the approaches that provide an exact start site prediction, SSM and NN, the fraction of exact predictions among correct ones is larger than for the IMC model, where position 250 within the highest scoring window is assumed to be the start site.

The next set contains the 5’ start points of all 202 genes within the human cytomegalovirus (table 8.8). As the exact positions of the TSSs might be some distance upstream of the start codon in many cases, the regions from -300 to +50 are considered as likely to contain the TSSs. The real number of TSSs is most probably lower than 202 because poly-cistronic transcripts, i. e. transcripts containing several genes in a row, are frequently observed in viruses. We also look at a subset of 20 exactly mapped transcription start sites and how much of them have a close hit. The fraction of hits regarding these exact TSSs is consistently higher than the fraction of overall hits, giving evidence to the assumption of a substantial fraction of polycistronic transcripts.

After these tests, a number of qualitative observations are possible. First, the stochastic segment models are more successful than the Markov chain models, at least for larger numbers of true positives. It is also clear that the number of exact predictions is higher for the segment model

model	threshold	TP	sn	FP	FP rate	sp	close hits
IMC	0	12	57.1	63	1/1,051	16.0	3
	0.01	11	52.4	44	1/1,505	20.0	
	0.03	9	42.9	13	1/5,095	40.9	
SSM	0	11	52.4	29	1/2,284	27.5	6
	0.01	9	42.9	15	1/4,416	37.5	
	0.015	7	33.3	9	1/7,360	43.8	
NN	0.9	10	47.6	19	1/3,486	34.5	6
	0.92	8	38.1	14	1/4,731	36.4	

Table 8.7: **Results of** M<span style="font-variant:small-caps">c</span>P<span style="font-variant:small-caps">romoter</span> **on a set of exactly annotated vertebrate start sites.** The accuracy of different sequence model densities and classification approaches are compared. (IMC: interpolated Markov chains; SSM: promoter segment model; NN: neural network classifier instead of Bayes)

model	threshold	TP	sn	FP	FP rate	sp	exact TSS	close hits
IMC	0	103	50.9	556	1/825	15.6	12 (60 %)	4
	0.01	95	47.0	450	1/1,020	17.4		
	0.03	45	22.2	147	1/3,122	23.4		
SSM	-0.01	103	50.9	440	1/1,043	19.0	14 (70 %)	8
	0	92	45.5	303	1/1,514	23.3		
	0.01	57	28.2	189	1/2,428	23.2		
NN	0.9	91	45.0	373	1/1,230	19.6	12 (60 %)	7
	0.92	86	42.5	320	1/1,434	21.2		

Table 8.8: **Results of** MCPROMOTER **on the human cytomegalovirus sequence.** The accuracy of different sequence model densities and classification approaches are compared. The last two columns give the number of total and close hits referring to 20 exactly mapped start sites.

because it provides an explicit model state of the initiator. Surprisingly, using the neural network instead of the simple Bayesian classifier leads to worse results than the SSM approach, even though the cross-validation evaluation of the classifier suggested a different outcome (see figure 8.19).

Next, the *Drosophila* promoter sequence models are evaluated on the 3 megabase long *Adh* region. The background models are trained on a subset of the non-promoter sequences described

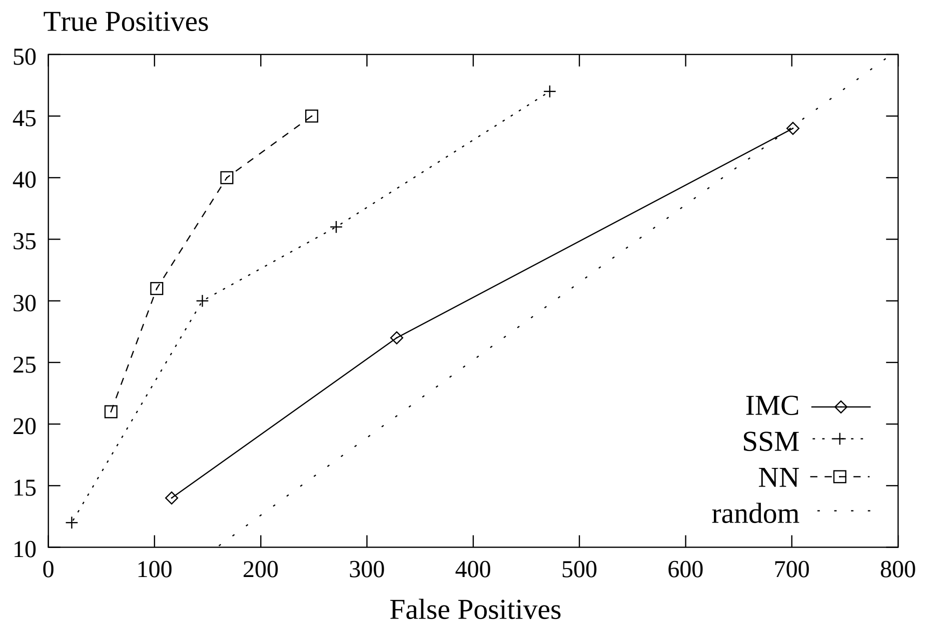


Figure 8.20: **Results of sequence-based promoter models on the *Drosophila Adh* region.** The dotted line denotes the bottom line, based on the assumption of equally distant random predictions.

in chapter 4, including a total of 240 non-coding and 711 coding sequences. This was the set used in the *Drosophila* genome annotation assessment project (Reese et al., 2000a; Ohler, 2000) and ensures that the results can be directly compared. As true positive regions, the region from -500 to +50 of 92 full-length-cDNA confirmed transcription start sites is used. The negative set consists of the regions downstream of these start sites until the end of the gene annotations. Only the sense strand is included because in some cases, genes are located within introns on the anti-sense strand (Ashburner et al., 1999). Figure 8.20 compares the different approaches by plotting the number of false positives against the true positives for different thresholds of the classifiers. It also shows the worst case scenario of random predictions, whose slope is given by the fraction of the size of the positive against the negative region. In agreement with the above results on human data, the segment model is considerably better than the simple Markov chain. A noteworthy distinction is immediately apparent: The neural network improves significantly on the results obtained by the Bayesian classifier, suggesting that non-linear dependencies among the promoter segments play an important role in *Drosophila* promoters. Above a certain true positive rate, the additional improvement leads us closer to the random line again. Thus, the threshold which leads to the largest Euclidean distance from the random line is a good compromise of sensitivity versus specificity. For the best performing neural network classifier, the sensitivity at this threshold (0.9)

Model	true pos.	avg. dist. (bases)	close hits
IMC	44	210	13
SSM	47	157	23
NN	45	100	21

Table 8.9: **Accuracy of promoter localization on the *Drosophila* data set.** For thresholds delivering a comparable number of true positive predictions, the average distance of the predictions from the annotated 5' start of the corresponding genes is given. The right column shows the number of close hits among the total number.

is 43.4 % with a specificity of 19.2 %.

The average distance of the predictions from the annotated 5' ends is shown in table 8.9: An improvement of the localization accuracy is obtained along with the improved classification. The quite large average distance is partly caused by the fact that the start sites were derived from alignments of cDNA sequences and not detailed mapping experiments. Therefore, a considerable portion of the 5' UTR might be missing, as the selected cDNAs were only known to contain the annotated start codon. The table also contains the number of close hits, in this case hits within +/- 50 bases of the possible start site.

Finally, the much larger set of 339 known genes in human chromosome 22 is studied. In agreement with previously published literature (Scherf et al., 2001; Hannenhalli and Levy, 2001), a likely region of -2000 to +500 is considered, and only the best maximum within 2,000 bases is retained. This makes the hysteresis filtering superfluous, and we therefore use a 3-median as output filter for all the models. As with the *Drosophila Adh* region, the regions downstream of the likely promoter regions until the end of the corresponding gene annotations are taken as the negative set. For efficiency reasons, we use the variable length Markov chain models instead of the interpolated Markov chains within the segment model states (see table 8.6). Figure 8.21 shows the number of false versus true positives for different threshold values. Very surprisingly and not in agreement with the smaller human data sets, the simple IMC models perform best, and the SSM model follows slightly behind. This can be due to the much larger likely region for the chromosome 22 promoter set; as soon as more reliable 5' annotations are available, additional experiments will hopefully help to clarify this. As it was observed on smaller data sets, the neural network classifier is definitely less successful than the Bayes classifier with the stochastic segment models. It can therefore be assumed that there are no non-linear dependencies between the promoter sub-states of the human model. The question remains open why the cross-validated classification results are not in agreement with these results — whether the extracted training set

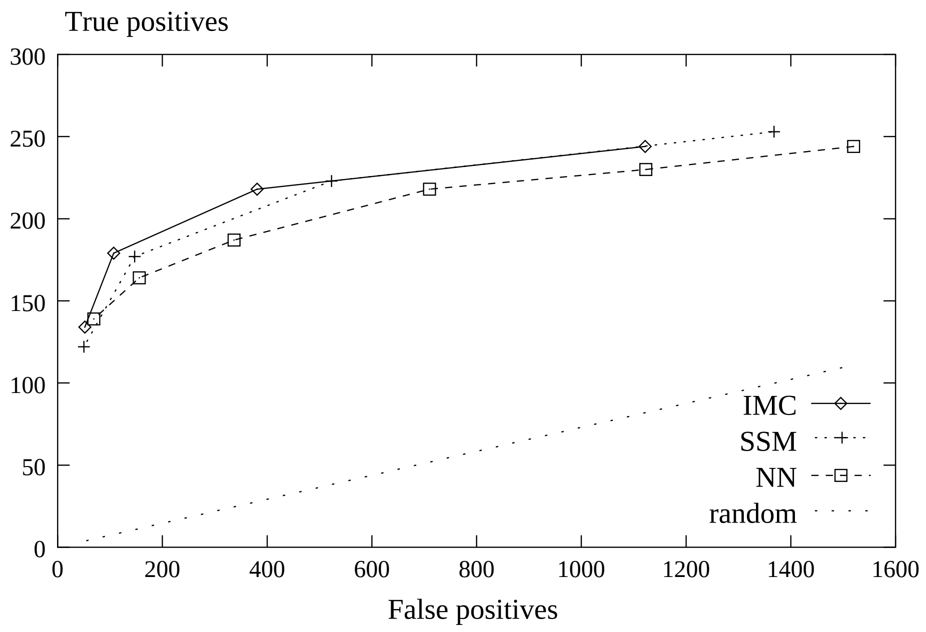


Figure 8.21: **Results of sequence-based promoter models on human chromosome 22.** The dotted line denotes the bottom line, based on the assumption of equally distant random predictions.

is considerably different from the other promoters, or whether the localization problem is considerably different from the classification problem. For the best performing interpolated Markov chain models at the threshold with the largest distance from random predictions, the sensitivity is 64.3 %, with a specificity of 36.4 %.

As with the *Drosophila* evaluation set, there are hardly any exactly mapped transcription start sites available for the genes in chromosome 22, and we look at the average distance of predictions from the annotated gene starts instead (table 8.10). The overall average distances are larger than for the *Drosophila* set, due to the larger UTRs in human. Unlike in the *Drosophila* case (cf. table8.9), the simpler models also perform better on the exact localization than the non-linear neural network classifier. The SSM predictions have the same average distance, but a higher fraction is close to the annotated 5' end.

60 % of the promoters on chromosome 22 are located within one of the 540 CpG islands that are annotated for release 2.3. Table 8.10 also shows the number of predictions which are located within or in a distance less than 150 bp from one of these islands. Stunningly, the predictions of all different approaches are highly correlated with the CpG islands; instead of the expected 60 %, the number is roughly 80 %. This means that CpG island correlated promoters are the ones that are easy to recognize, and the others are much harder to predict. Chapter 9 discusses this

Model	**true pos.**	**avg. dist. (bases)**	**close hits**	**CpG island hits**
IMC	218	254	85	179 (82.1%)
SSM	223	257	103	180 (80.7%)
NN	218	324	85	173 (79.3%)

Table 8.10: **Accuracy of promoter localization on human chromosome 22.** For thresholds delivering a comparable number of true positive predictions, the average distance of true predictions from the annotated 5' start of the corresponding genes, the number of close hits within a radius of 100 bases, and finally the true predictions correlated with CpG islands are given.

phenomenon which was also observed by other groups studying promoter prediction in vertebrate sequences.

8.3 Property-profile-based models of promoters

After the evaluation of the sequence-based promoter predictors, the following study deals with DNA property-based classification of promoters. Starting point is a DNA property profile, calculated and smoothed as described in figure 6.1.

To see how useful each of the possible 14 properties is, the first experiments examine how well a classification based on every single property can be carried out. Thereby, I followed the modeling approach of the sequence properties: A promoter is modeled by several densities for the promoter segments, and the background by one density each for profiles of coding and non-coding sequences. A promoter sequence segment model as above calculates the optimal segment boundaries. Because the parameter sets are sequence symmetric, all profiles look the same no matter whether they are calculated from the sense or the anti-sense strand, and we therefore need to look at only one side of a sequence.

I study two sets of features: the first consists of the mean profile values within a segment or subsequence (equation 6.4), the second one of the mean values plus the linear regression line slopes (equation 6.5). For half of the training data, these features are calculated, using either the whole profile for the background classes, or the six promoter segments obtained by the Viterbi segmentation of the promoter sequences using the segment model of figure 8.14. These features are then used to train eight Gaussian distributions (equations 6.10 and 6.11, six for the promoter segments plus two for the backgrounds), either with one dimension if only the mean values are used as features, or with two dimensions if the slope is used as well.

For the other half of the data, the features are calculated in the same way and passed through

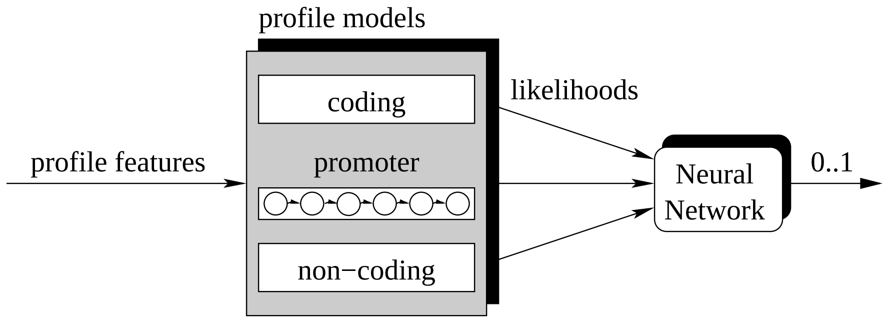


Figure 8.22: **Property profile based promoter classification.** The features extracted from individual profiles are judged by Gaussian distributions for promoter segments and background classes, and the likelihoods are fed into a multi-layer perceptron.

the pre-trained Gaussians. The resulting likelihoods are taken as independent data set to train a neural network that combines the promoter segment and background likelihoods. Thus, the approach for classification is exactly as with the sequence models above: use densities to compute likelihoods for promoter segments and backgrounds, and feed them into a neural network to take non-linear dependencies into account (see figure 8.22). The neural networks are multi-layer perceptrons with eight input nodes, six (*Drosophila*) or eight (human) hidden nodes and one output node — corresponding to the network topology of figure 8.18 without a total promoter likelihood input node — and is again trained with online back-propagation.

The performance of the individual features can be seen from table 8.11. Profiles were calculated using the parameter sets of appendix C. The two columns on the left show the results for the mean value parameter sets, modeled by one-dimensional Gaussians, the right two columns for the “mean+slope” parameter sets, modeled by two-dimensional Gaussians with full covariance matrices. We use the equal recognition rate (ERR) for classification into promoter/non-promoter and the integral over the receiver operating characteristics (ROC) as measures (cf. section 7.3).

A classification based on the profile means of the six promoter segments already results in a surprisingly high classification performance for many of the parameter sets. The physical property leading to the highest classification rate is protein-DNA-twist (70.4% ERR). Only B-DNA twist leads to a classification that is just slightly above chance (51.0% ERR). Taking both slope and mean as features is helpful in some cases, for example for DNA bendability or protein-DNA-twist, but also makes the results worse in others, such as GC content or propeller twist. A modeling by Gaussian mixture distributions (equation 6.12) with two components leads to highly similar results in the case of one-dimensional Gaussians; apart from an increase for the DNA bendability ROC value to 6824.0 and a decrease for nucleosome positioning to 7313.7, the

physical property	mean		mean+slope	
	ERR	ROC	ERR	ROC
tri-nucleotide GC content	69.7	7487.5	67.7	7330.1
DNA bendability	62.7	6710.8	66.8	7092.5
A-philicity	65.8	7037.8	65.3	7073.7
protein induced deformability	65.1	7115.8	62.6	6846.1
B-DNA twist	51.0	5224.7	49.7	4952.6
protein-DNA twist	70.4	7512.0	71.2	7722.2
Z-DNA stabilizing energy	70.2	7493.1	70.0	7406.6
nucleosome positioning	69.2	7458.6	66.3	7459.3
stacking energy	67.0	7443.0	64.9	7111.7
propeller twist	68.0	7434.1	63.8	7005.6
duplex stability (disrupt energy)	64.9	6912.6	59.8	6649.6
DNA denaturation	68.0	7344.5	66.2	7199.4
DNA bending stiffness	70.1	7567.4	69.0	7320.9
duplex stability (free energy)	67.9	7295.6	66.9	7314.0

Table 8.11: **Classification of fruit fly promoters based on physical properties of DNA.**

results are almost the same. For the two-dimensional Gaussians, a mixture never improves on the ROC values obtained by single one- or two-dimensional densities.

A somewhat similar picture is obtained for human promoters, as can be seen in table 8.12, but the classification results are worse throughout, and the ranking of the individual features are very different from *Drosophila*. The best result is obtained by using the mean feature values of stacking energy; surprisingly, B-DNA twist is almost equally good — for *Drosophila*, B-DNA twist derived features resulted in random classification. Protein-DNA-twist features, which lead to the best results for the fruit fly, are clearly worse than many others.

In a last experiment, the features were combined by principal component analysis. The mean respectively "mean+slope" values for all 14 properties were used as input, which resulted in 14- respectively 28-dimensional vectors. The following observations were made for the resulting eigenvalues of all PCAs of the segment and background feature vectors:

1. For the mean value feature vectors, the first eigenvalue ranged from 10.2–12, the second largest from 0.9–1.6, and at least eight eigenvalues were smaller than 0.1. Apparently, the 14-dimensional space can be reduced to a one-dimensional space without losing much information.

physical property	mean		mean+slope	
	ERR	ROC	ERR	ROC
tri-nucleotide GC content	63.8	6995.3	59.4	6427.3
DNA bendability	55.2	5755.8	58.9	6200.0
A-philicity	60.5	6517.5	62.0	6705.4
protein induced deformability	55.1	5840.3	58.4	6298.9
B-DNA twist	63.1	6819.5	64.3	7022.8
protein-DNA twist	56.0	5948.4	55.9	5835.5
Z-DNA stabilizing energy	63.1	6840.8	58.7	6356.5
nucleosome positioning	60.1	6448.0	60.8	6499.4
stacking energy	64.5	7023.1	62.3	6860.5
propeller twist	60.0	6507.3	59.9	6432.5
duplex stability (disrupt energy)	60.6	6632.9	56.2	6150.6
DNA denaturation	61.1	6516.9	59.8	6415.7
DNA bending stiffness	63.5	6825.6	57.7	6250.8
duplex stability (free energy)	61.9	6890.5	59.4	6342.6

Table 8.12: **Classification of human promoters based on physical properties of DNA.**

2. For the feature vectors with 28 components, the first two eigenvectors are considerably larger than the rest (10.2–13.8 and 7.8–10.3). All others were smaller than 2, and again more than half below 0.1. Thus, an information preserving reduction to a two-dimensional sub-space is possible.

Table 8.13 shows the results for the one- and two-dimensional modeling, using the PCA transformed feature vectors. A clear improvement over the single-feature classification is visible for the *Drosophila* data set, but not on the human data. The 28-dimensional input reflects the ambiguity already observed for the mean+slope feature set of individual profiles. As suspected from the small eigenvalues, using additional vectors corresponding to smaller eigenvalues does not improve on the classification for both feature sets. Compared with the classification based on DNA sequences, the profile based approach thus performs worse than the simplest approach studied, the Markov chains — there, an ERR of 83.0 % for *Drosophila* and 82.6 % for human was reported in section 8.2.1. The difference is even more pronounced for the similar sequence-based classification approach using a multi-layer perceptron, with an ERR of 88.3 % on human data.

PCA		***Drosophila***		**Human**	
input	output	ERR	ROC	ERR	ROC
14	1	74.0	7979.0	61.7	6714.3
28	1	72.9	7762.4	59.8	6493.5
28	2	68.7	7641.9	60.8	6589.5

Table 8.13: **Principal component analysis of physical property features.**

8.4 Sequence/profile joint models

8.4.1 Integration of sequence and profile models

The final section of the McPromoter system design and evaluation describes how the sequence and profile models are put together. In principle, the segment model formalism could easily be extended to include densities for profile features; instead of calculating the segment probabilities based on the sequence alone, we replace equation 5.31 by the joint probability on sequence and profile:

$$P_j(\boldsymbol{w}_i, \boldsymbol{p}_i) = d_j(\tau_i) \cdot b_j(\boldsymbol{w}_i|\tau_i) \cdot c_j(\boldsymbol{p}_i[\boldsymbol{w}_i, \tau_i]) \tag{8.1}$$

Now, each state comprises a probabilistic sub-model c_j that describes the likelihood of a profile $\boldsymbol{p}_i$, given the sequence and its length. No other changes are necessary, all algorithms that were applicable to the sequence model can also be used for the joint sequence/profile model. However, the underlying assumption that profile and sequence likelihood are independent, is apparently not true; the profiles are calculated from the sequence, and a correlation therefore surely exists. In practice, we therefore have to pursue a different way.

Because the neural network was clearly successful, at least for the classification task, the same approach to use the segment densities as input variables is kept. In principle, the sequence/structure product probability of equation 8.1 can replace the simple sequence probability. A more promising way, though, is to split up each likelihood input node by a sequence/profile double node and connect them solely with each other in the first hidden layer. This automatically results in a linear weighting for the relative importance of sequence and physical property, and accounts for the dependence of the sequence and profile likelihood. Figure 8.23 shows the resulting network topology.

The best segmentation remains solely based on the sequence probabilities instead of running the Viterbi algorithm using the sequence/profile product likelihoods. The profile features are thus calculated based on the best path delivered by the application of the sequence model. This

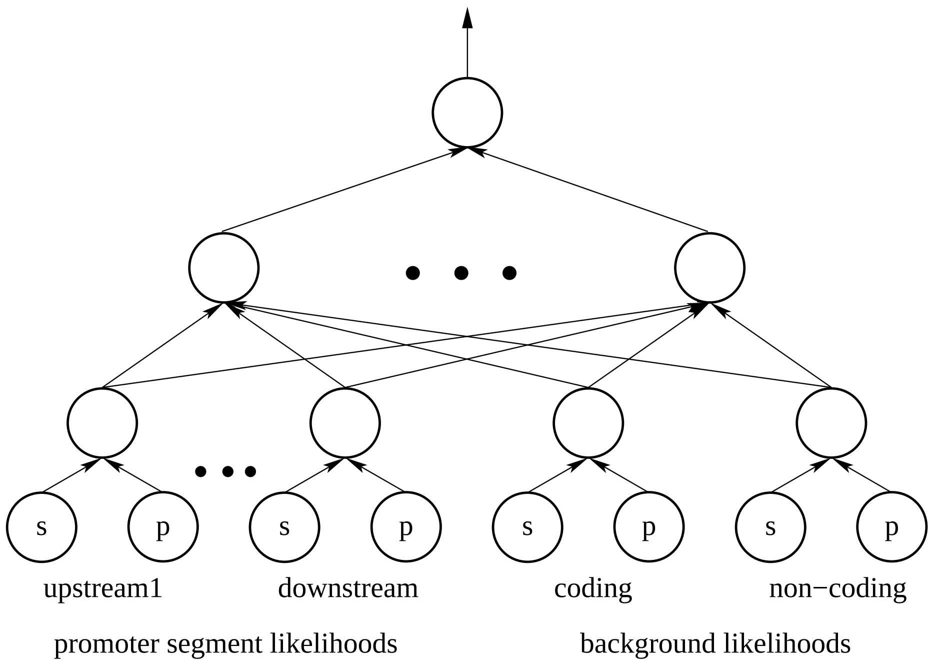


Figure 8.23: **The neural network classifier for sequence and profile model densities.** The multi-layer perceptron takes the sequence and profile likelihoods of each promoter state as well as the likelihoods of the background models as input (16 nodes). The connections between input and second node layer are restricted; they only combine the sequence and profile likelihoods of each state, and the layer therefore has eight nodes. Second and third layer are fully connected; the third layer has eight nodes in the human and six in the *Drosophila* model. An additional 17th input node represents the total promoter likelihood (cf. figure 8.17) and is directly and fully connected to the third layer.

approximation makes little difference for the mean value and slope features that we extract from the profiles because they are calculated on whole segments and should not change very much when the segment position is slightly different.

To summarize, we therefore have the following setup for feature calculation when we include a principal component analysis:

1. Compute the promoter segment and the background likelihoods of the actual 300 bp window with the sequence models; send the segmentation to the profile module.
2. Compute profiles of all physical property over the sequence and smooth them with a mean filter of width 21.
3. Calculate the profile feature values for the six segments and the background for all profiles.

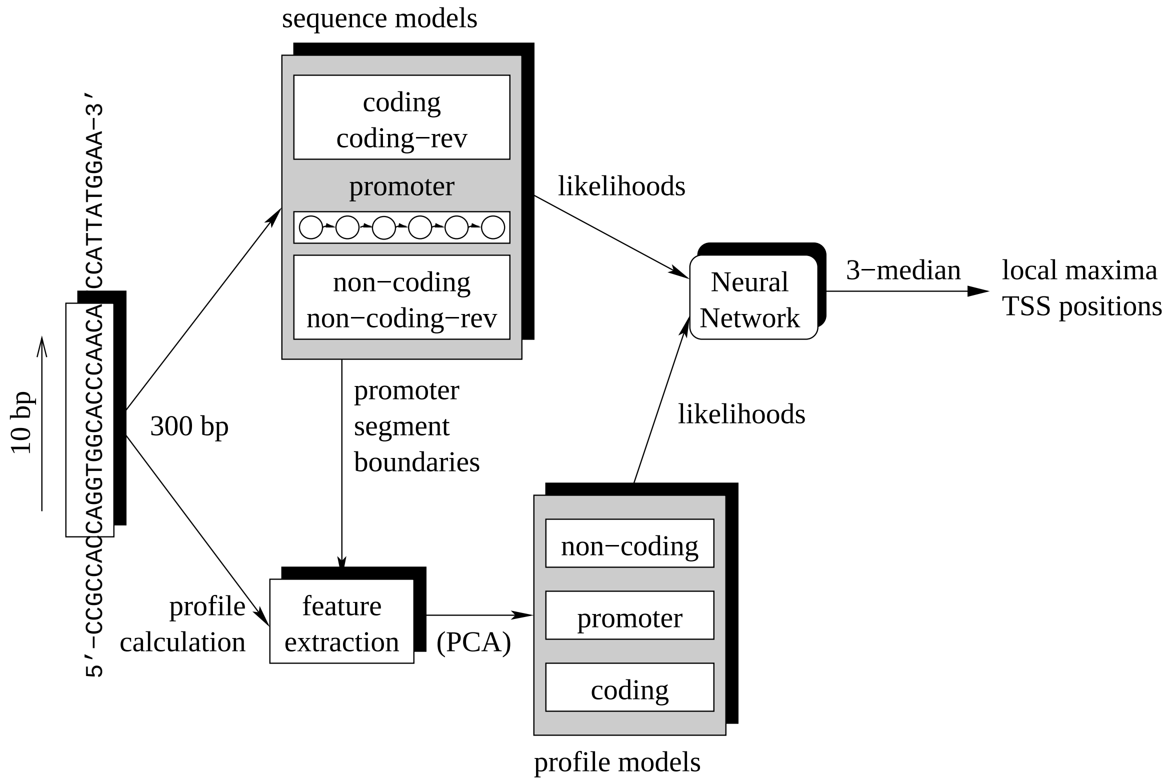


Figure 8.24: **The promoter finding system including structural features.**

4. Perform a principal component analysis of the features separately for each promoter state and each background model.
5. Judge the transformed features with corresponding Gaussian distributions for each segment and background.

Without PCA, the features of one property profile are directly passed through to the Gaussians. The sequence and profile likelihoods are then fed into the neural network input layer, as can be seen in figure 8.23. There is no need for reverse background profile models because profile parameters are symmetric and lead to the same profiles on both strands. The complete system including structural features is depicted in figure 8.24.

8.4.2 Evaluation of the joint models

The last section of this chapter finally turns to the evaluation of promoter prediction using the joint models. To decide which structural features are best suited for the human model, different neural networks were trained that took the sequence likelihoods as well as the likelihoods of

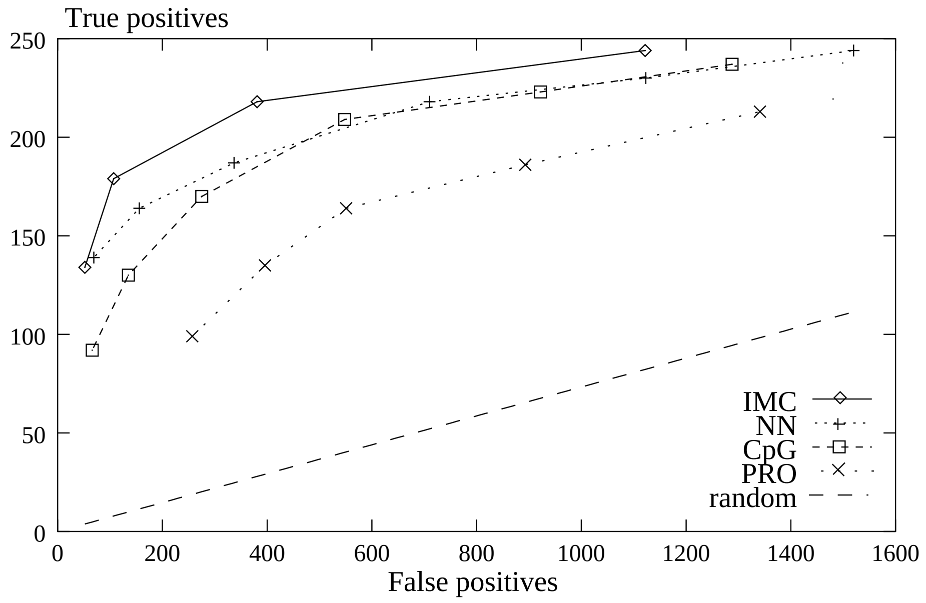


Figure 8.25: **Results of sequence/profile-based promoter models on human chromosome 22.** The dotted line denotes the base line, based on the assumption of uniformly distributed random predictions. Compare also with the results in figure 8.21. (CpG: neural network with additional CpG island features, PRO: neural network with additional profile features)

single profile features and the PCA transformed features as input. The network which leads to the best ROC integral value in the five-fold cross-validation experiments was finally chosen; it was the network using the mean value set of stacking energy. Figure 8.25 compares the results of this system with both the best IMC sequence models from figure 8.21 and the neural network using only sequence features. It shows that for the case of human data, additional profile features lead to no improvement over the neural network taking only sequence likelihoods as input; on the contrary, the results are distinctly worse at the first look. However, when we examine the correlation with CpG islands as above, we see that predictions that coincide with CpG islands account for only 61 % (see table 8.14). This means that a highly distinct subset of promoters is recognized when we include structural features in the model, one that is different from the CpG island associated one that is easily captured by sequence models.

Figure 8.25 also depicts results obtained with a neural network using the sequence likelihoods as in figure 8.18, but with two additional input features, the GC content and CpG di-nucleotide ratio (see section 6.1, equations 6.1 and 6.2). In comparison to the complex profile modeling, these simple features related to DNA structure perform better, but still decrease the performance of the system. The fraction of the apparently "easier" CpG island correlated predictions is smaller

Model	true pos.	avg. dist. (bases)	close hits	CpG island hits
CpG	209	379	72	158 (75.5%)
PRO	213	456	31	130 (61.0%)

Table 8.14: **Accuracy of promoter localization with sequence/profile models on human chromosome 22.** For thresholds delivering a comparable number of true positive predictions, the average distance of true predictions from the annotated 5’ start of the corresponding genes, the number of close hits within 100 bases, and finally the true predictions correlated with CpG islands are given.

than without CpG island features (cf. table 8.10), but larger than with the sequence/profile joint model. The exact numbers, along with the localization accuracy, are given in table 8.14.

As there is not enough data for a cross-validation in the case of *Drosophila*, the most suitable property for the combined sequence/structure model is selected according to the best ROC integral value on the neural network *validation* set. Even though the PCA derived features performed better than the features derived from any single profile, combining them with sequence features does not lead to better ROC values on the validation set; the best value is obtained by using the mean value set of the nucleosome positioning preference profile. Indeed, this model leads to an improvement of the system and to the overall best results for *Drosophila*. This can be seen in figure 8.26, which compares the neural network classifier with and without structural features.

For the results in figure 8.26, the sensitivity at the largest distance from random predictions is 52.1 % at a specificity of 17.6 %. The average distance of the predictions at this threshold is 117 bases, including 25 close predictions out of a total of 48 correct ones, which is on the same scale as the best model without structural features.

Figure 8.27 shows an example system output of the different models applied on the promoter sequence of the *beat-B* gene. It has a 5’ UTR of more than 21,500 bp, and the annotated transcription start site is at position 2,411,679 on the reverse strand. We can see that all the models give a high score close to the annotated site, but that the neural network classifier output, with or without structural features, looks more pronounced than the Bayes classifier approach. The large difference in the likelihood of IMC and SSM on the left side of the picture is due to the application of the Viterbi algorithm to compute the SSM likelihood. This corresponds to the likelihood for only the best path through the promoter model and not to the total likelihood as computed by the Markov chains. The “fuzzy” appearance of the IMC prediction is caused by hysteresis post-processing.

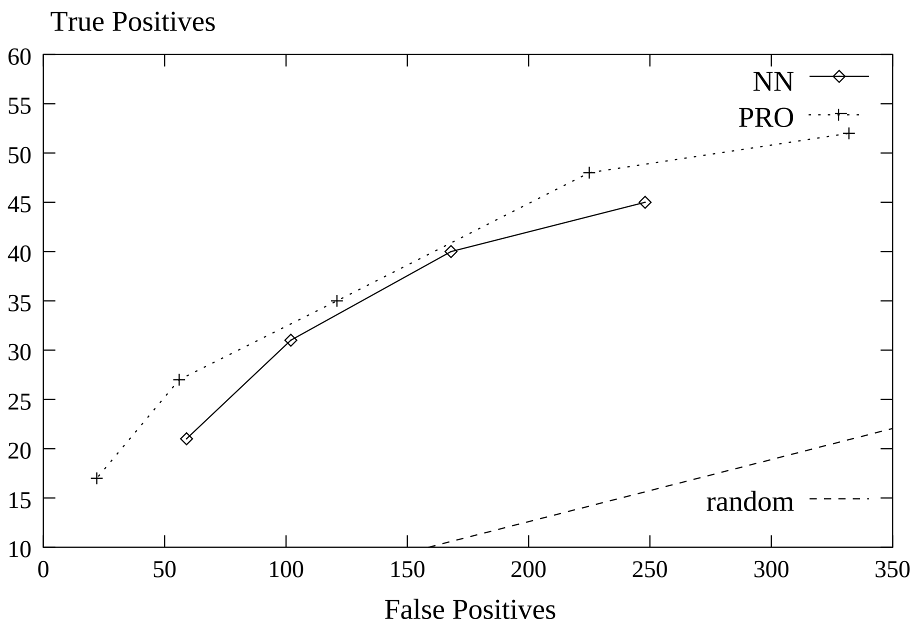


Figure 8.26: **Results of sequence/profile promoter models on the *Drosophila Adh* region.** The dotted line denotes the bottom line, based on the assumption of equally distant random predictions. Compare with the results in figure 8.20. (PRO: model with nucleosome positioning profile features)

The "best" promoter predictors. The goal of this work is not only to come up with good promoter recognition rates, but also to draw conclusions from the difference in performance of different models. Nevertheless, a short summary about the models for each *Drosophila* and human that lead to the best recognition rates in this work is provided in the following.

The best performing *Drosophila* system follows the outline of figure 8.24:

- *Sequence background models.* 5th order Markov chains with rational interpolation (equation 5.14) are taken as models of coding and non-coding sequences. They are applied on both strands of the 300 bp sequence window, and the two background likelihoods are uniform mixtures of forward and backward strand likelihoods.
- *Promoter sequence model.* This is a six-state model as specified in figure 8.14, with rationally interpolated Markov chain submodels as specified in table 8.4. The Viterbi algorithm (figure 5.9) is used to compute the promoter likelihood.
- *Profile features.* A nucleosome positioning preference profile is calculated from the sequence (see appendix C). As features, mean profile values are computed (equation 6.4), once for the whole 300 bp window to be judged by the background profile models, and

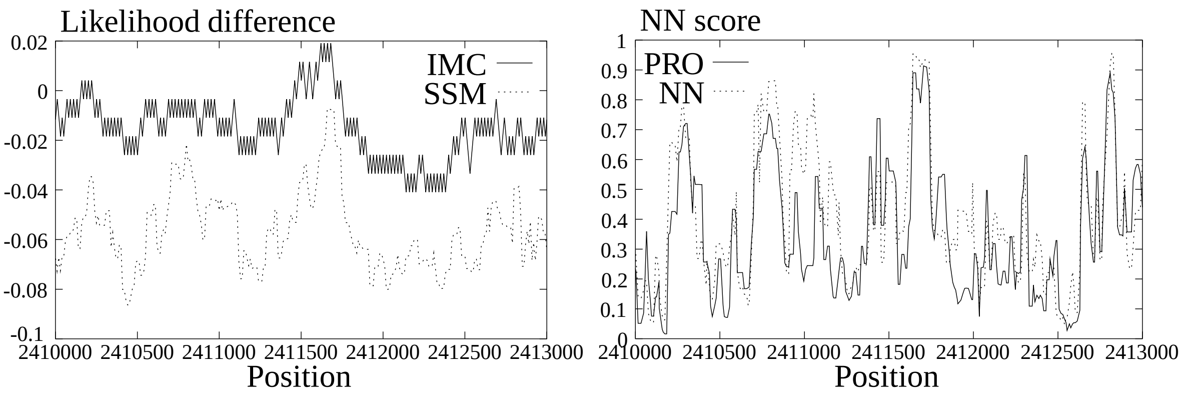


Figure 8.27: **Example prediction of the *Drosophila beat-B* promoter.** The left panel shows the output of the interpolated Markov chain (IMC) and stochastic segment (SSM) modeling approach, the right panel of the neural network classifier approach without (NN) and with (PRO) structural features.

once for the segments obtained by the Viterbi algorithm and the promoter segment model.

- *Profile models.* One-dimensional Gaussian distributions (equation 6.9) are used to represent the profiles of the six promoter states and the two background classes.
- *Neural network.* A multi-layer perceptron with the topology of figure 8.23 classifies the features. The output is smoothed with a median of width 3. Local maxima above a user-specified threshold are then reported as promoter predictions.
- *Training.* Half of the data is used to train the rational background IMCs (Schukat-Talamazzini et al., 1997), the promoter segment model (figure 5.10), and the Gaussian distributions (equations 6.10 and 6.11), the other half to train the neural network with online back-propagation (equation 7.12 with learning rate $\eta = 0.005$ and threshold $\nu = 0.05$), stopping when the best conditional entropy value (equation 7.14) is achieved on an independent validation set (a third of the whole neural network training set).

For human, no easy “best model” can be given: On small well-mapped data sets, the stochastic segment model is clearly better than the IMC (cf. tables 8.7 and 8.8). On the large chromosome 22 set though, the IMC is generally better, but for higher sensitivity, the SSM performs equally good, and a larger fraction of SSM predictions is close to the annotated gene start sites. For the general case, it appears best to therefore pursue the stochastic segment model approach of figure 8.15.

- *Sequence background models.* 7th order Markov chains with rational interpolation (equation 5.14) are taken as models of coding and non-coding sequences. They are applied on both strands of the 300 bp sequence window, and the two background likelihoods are uniform mixtures of forward and backward strand likelihoods.

- *Promoter sequence model.* This is a six-state model as specified in figure 8.14, with rationally interpolated Markov chain submodels as specified in table 8.4. The Viterbi algorithm (figure 5.9) is used to compute the promoter likelihood.
- *Bayes classifier.* The modified Bayesian classifier of equation 7.6 is used. For contiguous sequences, the score is smoothed with a median of width 3, and local maxima above a user-specified threshold are reported as promoter predictions.
- *Training.* All the data is used to train the background IMCs (Schukat-Talamazzini et al., 1997) and the segment model (figure 5.10).

The following chapter will now put these models and results in relation to other approaches.

Chapter 9

Discussion and Outlook

At the end of this work, the following pages provide a short wrap-up of the results, a comparison with related work, and the conclusions we can draw. This will lead to a number of possible directions for future research in promoter finding, and also to the related topic of promoter analysis: Once one has identified the promoter sequences of an organism, a natural interest lies in the identification of the regulatory elements hidden in them. The final section therefore provides a short account of different methods for pattern discovery in promoter sequences.

9.1 Promoter recognition

Overall conclusions and comparison to related work. A very general — but maybe also the most important — conclusion immediately comes to mind when comparing the results of fruit fly and human promoter finding: *There is no such thing as eukaryotic promoter recognition in general*. The results clearly indicate that vertebrate promoters have a structural organization that is very different from *Drosophila*. In vertebrates, the outstanding sequence pattern in promoters is their frequent localization within CpG islands, and thus, very simple models based on overall sequence composition are more successful regarding the overall recognition rate as more complicated ones. For *Drosophila*, a more detailed modeling of the proximal promoter region increases the recognition rate substantially.

Another promoter predictor by Reese (2000), NNPP, was applied on the same set of the *Drosophila Adh* region. It is based on a neural network that combines models of the TATA box and the initiator. In comparison to the models developed in this work, it performs better than the simple IMC model; starting with the SSM, the approaches in this work are more successful. Table 9.1 compares the results given by Reese (2000) with the best McPromoter model. For higher recognition rates, a reduction of false positives by a factor of about 2.5–3 is observed; at a more

System	**sn**	**FP rate**
NNPP	21.7	1/6,227
	38.0	1/2,416
	53.2	1/1,096
McPromoter	18.4	1/38,780
	38.0	1/7,051
	52.1	1/3,791

Table 9.1: **Comparison of McPromoter with the NNPP predictors on fruit fly data.** Shown are the true and false positive rate on the *Drosophila Adh* promoter set for different thresholds delivering comparable numbers of true positives.

restrictive threshold, this reduction can go up to a factor of 6. The localization accuracy is also higher — the average distance of predictions from the annotated start sites is 149 bases for NNPP, but only 117 for the McPromoter models with the best classification results. Consequently, McPromoter has therefore been used by annotators of the *Drosophila* Genome Project to assist in the semi-automatic annotation of the whole *Drosophila* genome.

Turning to vertebrate promoter prediction, the results on human chromosome 22 (figure 8.21 and 8.25) show that the performance of the Markov chain sequence models is most distant from random predictions at a sensitivity of about 65 %. At this point, only 18 % of the true positives are not correlated to CpG islands. This trend is even stronger when we increase the threshold of the Markov chain models: At a sensitivity of 52.8 %, the specificity is 62.6 %, with a CpG island correlation of 90.5 %; at a sensitivity of 39.5 %, the specificity has gone up to 72.0 %, with 94 % located within CpG islands. This was also observed by another recent publication on promoter finding in chromosome 22 (Scherf et al., 2001): Out of 111 true positive predictions, only 5 were not associated with a CpG island — an even more severe correlation than observed in this work. Hannenhalli and Levy (2001) also note that models using simple CpG island features are as successful as more complicated models. It is therefore notable that despite of their weaker performance, sequence/profile joint models do not show a recognition that is biased towards promoters within CpG islands.

Table 9.2 compares the results of other predictors, as described by Scherf et al. (2001) (PromoterInspector) and Hannenhalli and Levy (2001), with McPromoter. Furthermore, we compare McPromoter to the program Dragon 1.2 which had not yet been published at the time this evaluation was done (see http://sdmc.krdl.org.sg/promoter/). A general caveat of this comparison is that the evaluation for Hannenhalli/Levy and PromoterInspector was performed on an earlier

System	**sn**	**sp**	**FP rate**
Hannenhalli/Levy	42.9	—	1/216,440
PromoterInspector	44.9	33.6	—
Dragon 1.2	22.1	57.1	—
	30.7	26.3	—
	60.2	21.9	—
McPromoter	39.5	72.0	1/237,475
	52.8	62.6	1/115,408
	64.3	36.4	1/32,411

Table 9.2: **Comparison of McPromoter with other predictors on human data.** Shown are the numbers of sensitivity and specificity as well as false prediction rate, where available. This table should be read with caution; a number of differences in the evaluation of the methods is described in the text. A dash denotes that the information was not available.

chromosome 22 annotation release with 247 known genes, instead of the set of 339 genes used to evaluate McPromoter and Dragon 1.2.

All sensitivity numbers are given for the respective subset of known genes. PromoterInspector does not make strand-specific predictions and does not attempt to exactly predict the TSS: It delivers regions of varying size as result, from 193 to 2433 bp, with an average of 555 bp, and a true positive is counted if this region overlaps the likely region to any extent. Both Dragon and Hannenhalli/Levy therefore extend the likely -2000/+500 region by the average size of a PromoterInspector prediction, i. e. to -2555/+1055. This is not done in this evaluation, and the sensitivity of McPromoter is therefore expected to be somewhat under-estimated compared to the sensitivity of the other predictors. The specificity numbers of PromoterInspector and Dragon 1.2 are calculated from results given at http://sdmc.krdl.org.sg/promoter/ which use the regions spanned by the known genes on *both* strands as negative set. The McPromoter results in this thesis are calculated using only the sense strand of the known genes as negative set, to exclude genes possibly contained within introns on the anti-sense strand. To better compare the values, we can assume that McPromoter makes the same number of false predictions on the reverse strand as measured on the forward strand. At 39.5 % sensitivity, its specificity is then 56.3 %, at 52.8 % it is 45.4 %, and at the 64.3 % sensitivity level it is 22.2 %. These results are clearly better than the ones reported for PromoterInspector and Dragon 1.2.

The detailed results of Hannenhalli/Levy are not publicly available, and only the total number of predictions along the whole chromosome is exactly known, so all predictions outside the known genes are here counted as “false” predictions. Some of these hits will certainly concern genes not in the set of known genes, and the real FP rate is most definitely smaller. Despite all the differences, it is nevertheless clear that the overall performance is on the same scale for all the methods, even though McPromoter appears to be more successful than Dragon 1.2 and PromoterInspector. A thorough comparison would require exactly equal test and also equal training set conditions.

The human models of McPromoter were used by different groups in virus research, such as at the Institute for Clinical and Molecular Virology at the University of Erlangen-Nuremberg, to guide experimental verification of promoter activity (see also Koelle et al., 2001; Rimessi et al., 2001). McPromoter is accessible to the scientific community at http://promoter.informatik.uni-erlangen.de, and had been used about 1,500 times between April and September 2001.

Promoter sub-classes and physical properties. Concerning future work, a separate modeling of two promoter subclasses might elucidate whether a recognition of non-CpG-correlated promoters is feasible at all, or if the sequence information in the proximal promoter regions is simply not enough to define the functionality of a vertebrate promoter. Such a separate modeling can also help to find out whether non-linear dependencies of the promoter sub-states are not observed for all vertebrate promoters, or whether this phenomenon is also related to the strong pattern of the CpG islands. For this, a comparison of multi-layer perceptrons with simpler discriminatory functions such as linear or quadratic discriminant analysis will be useful.

The results in this work suggest that DNA structure can be exploited to successfully classify promoters. A more detailed evaluation of the correlation among the sequence and structure likelihoods can now help, first to elucidate what can actually be gained by the combination of both, and second to determine an optimal subset of features that indeed carries information going beyond the sequence. The difference in promoter classification between human and *Drosophila* is very notable on the level of physical properties, especially when using the principal components, and also inspires further study. Furthermore, a completely different approach to extract structural features can be pursued: Profiles can be discretized and mapped to a small number, such as four to sixteen, values. By doing so, they can be treated in exactly the same way as the DNA sequence itself, and possibly represented by the same discrete stochastic models as presented in this thesis.

Modeling aspects. Future work can also deal with the current promoter model. Submodels other than Markov chains may be more appropriate for the core promoter parts, ones that use position specific probability distributions such as hidden Markov models. As long as the required algorithms for these submodels are given, the existing segment model framework provides the integration. Different promoter sub-classes can also be represented by an SSM using several parallel paths through the model.

Instead of the two-stage modeling with probabilistic sequence/structure models and the neural network classifier, the framework of Bayesian networks (Jensen, 1996) allows one to integrate both in one model: Thinking of the neural network in figure 8.23 as a Bayesian network, one could plug in sub-networks representing the Markov chain models and Gaussian distributions. Two possible obstacles here are the treatment of the duration distributions, and the high complexity of the Bayesian network training and propagation algorithms, especially for the highly connected part representing the neural network.

Algorithmic aspects. As successful as the Markov chain models turned out to be for the promoter recognition problem, there are some points that can lead to further improvement. One is the learning of the variable length Markov chain models. The optimization according to the maximum mutual information criterion was carried out with constant background models and could be replaced by a combinatorial optimization. Also, a combination of VLMCs and interpolated Markov chains might represent the true underlying distribution more accurately. A different objective function is based on the minimum description length principle and was recently used by Seldin et al. (2001) for VLMC structure learning in a context very similar to segment models. As described in section 8.2.1, applying the MMI principle for segment models can also be revisited to derive detailed estimation equations for the case of the entire segment model instead of deferring it to the sub-models.

Exploiting orthologous sequences. Now that the DNA of a large number of organisms has been sequenced, algorithms can exploit the sequence information of more than one organism at once. One way to do so is to use the similarity of promoters of orthologous genes, i. e. of genes that are derived from the same ancestral gene. For these genes, it can be assumed that their regulatory sequence has not diverged too much. One caveat is that the sequence outside of the transcription factor binding sites may mutate without affecting the functionality, and promoters can therefore be expected to be more degenerate than coding regions. It is thus not clear whether additional features about sequence similarity can help to find the proximal promoter and the transcription start site, or if similarity can only help to narrow down the search region for pattern identification (see section 9.2 below). This will depend strongly on the two organisms for which the similarity is computed.

An integration of similarity information into the existing system can happen on the level of the promoter sub-models, thus taking possibly different degrees of conservation of e. g. TATA box state (high) and spacer state (low) into account.

Computational promoter recognition has always been notorious for its difficulty. Nevertheless, this work has shown how far we can push the recognition rate when using state-of-the-art probabilistic modeling. Now that the genomes of both man and fruit fly are available, large-scale sequencing projects of cDNAs including the 5’ mRNA cap structure will lead to larger high-quality data sets for both training and evaluation of current models (Suzuki et al., 2001). In this way, experimental and computational approaches will help each other to obtain genome-wide sets of proximal promoters, and to finally understand the complex phenomenon of eukaryotic gene regulation.

Recent developments. After the thesis was submitted in November 2001, the models described above were retrained using new data obtained from exactly these large-scale cap-selected full-length cDNA sequencing efforts at the Berkeley Drosophila Genome Project. A much larger training set of 1,864 promoters, 2,635 coding sequences and 1,755 intron sequences in the same form as described in chapter 4 could thus be extracted. The new results are largely in accordance with what is described above and shall therefore be only sketched; the details will be published elsewhere (Ohler et al., 2002).

Whereas the old data set was too small to perform a cross-validation evaluation using a neural network classifier, it is now possible to perform a five-fold cross-validation classification of this data, using the system with sequence features only (cf. figure 8.17), profile derived features only (figure 8.22), and the combination of both feature types (figure 8.24). Three parts of the data were used for density estimation, one part for neural network training, and the fifth part for testing. Again, principal component analysis did not lead to the best combined performance, which was achieved by the mean value feature set of stacking energy, as observed earlier for the human set. Most of the improvement is already gained on the classification performance using sequence features alone, which is not only due to the larger data set, but also to the much better quality compared to the old set (see Ohler et al. (2002)).

The models were finally applied another time on the genomic *Drosophila* test set, after being re-trained on the full data set excluding 22 of the 1,864 promoters that matched entries of the Adh test set. Table 9.3 shows the new results on a genomic scale; depending on the threshold used, a reduction of false positives by about three to four times is achieved (cf. table 9.1).

9.2 Analysis of promoter sequences

This chapter closes with a look at the analysis of promoter sequences, assuming that we know about their location. The interest in this field (Ohler and Niemann, 2001) received a great boost

sn	**FP rate**	**sp**
19.5	1/106,647	69.2
36.9	1/25,853	50.7
52.1	1/12,016	40.3

Table 9.3: McPromoter **results on fruit fly data after re-training** with a larger data set of promoter and non-promoter sequences. Shown are the true and false positive rate as well as the corresponding specificity value on the *Drosophila Adh* promoter set for different thresholds delivering comparable numbers of true positives as in table 9.1.

with the arrival of microarray gene expression data: Once a group of genes with a similar expression profile is determined (e.g., that are activated at the same time in the cell cycle (Spellman et al., 1998)), a natural assumption is that the similar profile is (partly) caused by and reflected in a similar structure of the regulatory regions involved in transcription. The ultimate goal is the automated construction of specific promoter models containing a combination of several regulatory elements (cf. section 3.3.3). Research so far has focused on the detection of *single* motifs (representing transcription factor binding sites) common to the promoter sequences of putatively co-regulated genes. Although this problem might seem simple at first, it is very complex and requires that one finds

- a pattern of unknown size that might not be well conserved between promoters
- in a set of sequences that do not necessarily represent the complete promoters, and
- that was in many cases grouped together by a clustering algorithm that itself can be error-prone and include genes that are not co-expressed *in vivo*.

Therefore, studies have mainly concentrated on the rather "simple" genome of the yeast *S. cerevisiae* — it was the first fully sequenced eukaryotic organism, and the first one for which a comprehensive amount of expression data became publicly available. Statistics on mapped transcription start sites (Zhu and Zhang, 1999) show that its 5' UTR sequences are rather short (a mean of 89 bp), and most of the known regulatory elements are close to the translated part of the genes, the majority being found between 10 and 700 bp upstream from the translation start codon. This means that for yeast, the region upstream of the start codon can be used as a good approximation of a promoter region, in contrast to the higher eukaryotes that are in the focus of this work. Most algorithms searching for conserved patterns in yeast promoters thus take 500–1000 bp upstream of the start codons of supposedly co-regulated genes as data set.

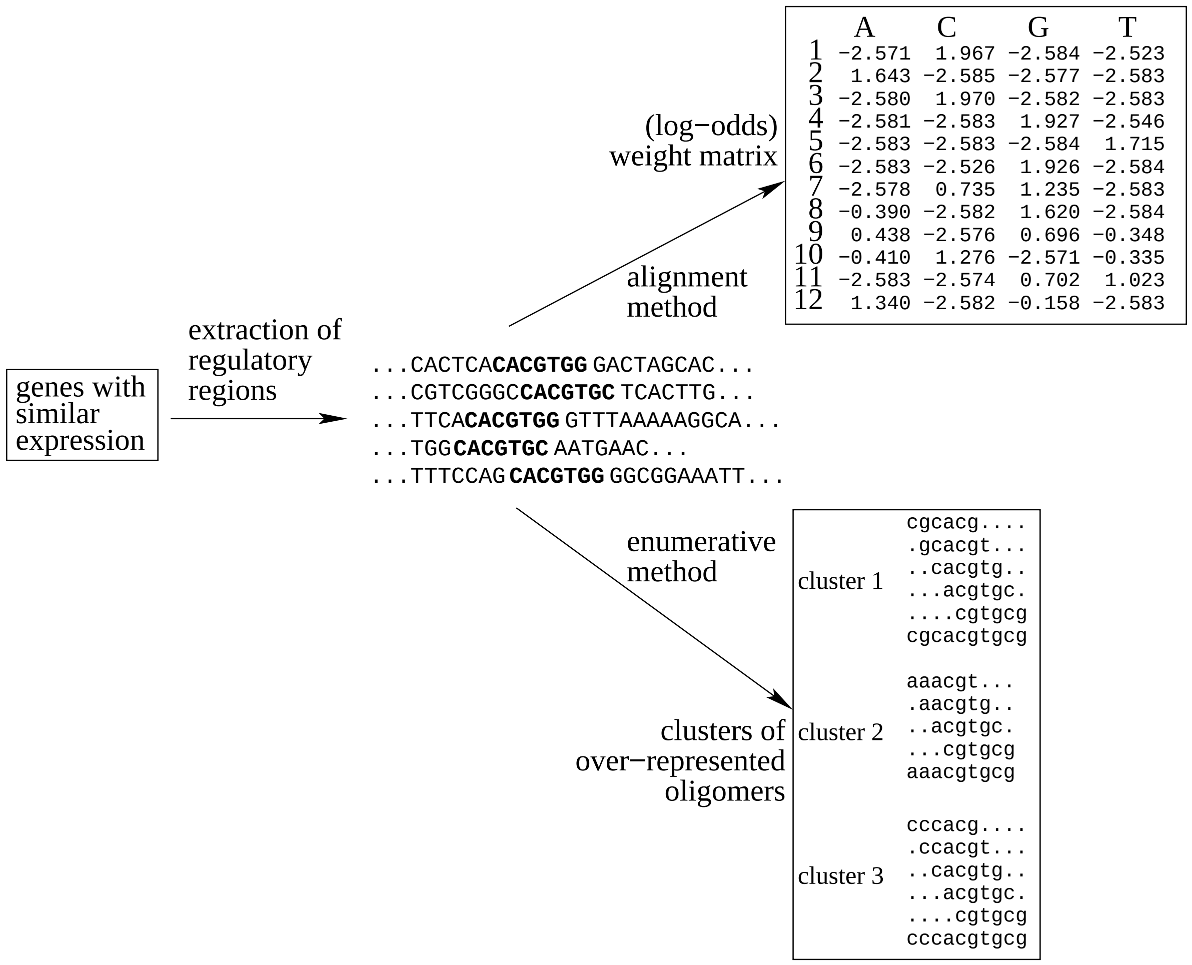
There are two fundamentally different approaches to tackle the problem:

- *Alignment* methods aim at the identification of unknown signals by a significant local multiple alignment of all sequences. As a direct multiple alignment would be computationally very expensive, the methods go a different way. For example, the CONSENSUS algorithm approximates a multiple alignment by aligning sequences one by one (Hertz and Stormo, 1999) and optimizing the information content of the weight matrix constructed from the alignment. Other algorithms use a probabilistic approach; they consider the *start positions* of the motifs in the sequences to be unknown and perform a local optimization over the sequence to determine which positions deliver the most conserved motif. Two important methods are Gibbs sampling (Lawrence et al., 1993) and Expectation Maximization in the case of the MEME system (Bailey and Elkan, 1995).
- *Enumerative* or *exhaustive* methods examine all oligomers of a certain length and report those that occur more often than expected from the overall promoter sequence composition(van Helden et al., 1998; Brazma et al., 1998). This approach has gained in popularity since the arrival of complete genomes, and it is far from trivial — for example, a normalization regarding self-overlapping or palindromic patterns has to be carried out. Köstler (2001) provides an extensive treatment with an application for motif identification in viral regulatory regions.

An orthologous approach is to identify elements not by analyzing different promoters from the same organism, but promoters of the same gene from different related species (Blanchette et al., 2000). An optimal alignment of a small region of specified size is constructed that takes the phylogenetic distance into account.

From a practical point of view, the most eye-catching difference between exhaustive and alignment methods is maybe the shape of the result: The alignment approaches deliver a model of the motifs (usually a weight matrix) built from the alignment, the enumerative methods a list of over-represented oligomers, possibly already grouped to form consensus sequences. Figure 9.1 shows an exemplified flowchart to illustrate this.

Background models. One important aspect of pattern discovery approaches concerns the *background* model. Without a reliable background model, the results are biased towards generally over-represented patterns. Therefore, patterns such as mainly GC-rich motifs in organisms whose promoters have a high GC content, or the TATA box, will be found, and patterns that are specific to a subset of interest will be lost. A good background model is constructed from the set of all promoters and takes their specific sequence composition into account. This means that a specific model, at least for each organism, has to be trained, and the necessary information is not always available. Detailed studies have examined that approaches become more prone to fail


genes with similar expression
extraction of regulatory regions
...CACTCA**CACGTGG**GACTAGCAC...
...CGTCGGGC**CACGTGC**TCACTTG...
...TTCA**CACGTGG**GTTTAAAAAGGCA...
...TGG**CACGTGC**AATGAAC...
...TTTCCAG**CACGTGG**GGCGGAAATT...
alignment method
(log−odds) weight matrix
enumerative method
clusters of over−represented oligomers
cluster 1
cgcacg....
.gcacgt...
..cacgtg..
...acgtgc.
....cgtgcg
cgcacgtgcg
cluster 2
aaacgt...
.aacgtg..
..acgtgc.
...cgtgcg
aaacgtgcg
cluster 3
cccacg....
.ccacgt...
..cacgtg..
...acgtgc.
....cgtgcg
cccacgtgcg

	A	C	G	T
1	−2.571	1.967	−2.584	−2.523
2	1.643	−2.585	−2.577	−2.583
3	−2.580	1.970	−2.582	−2.583
4	−2.581	−2.583	1.927	−2.546
5	−2.583	−2.583	−2.584	1.715
6	−2.583	−2.526	1.926	−2.584
7	−2.578	0.735	1.235	−2.583
8	−0.390	−2.582	1.620	−2.584
9	0.438	−2.576	0.696	−0.348
10	−0.410	1.276	−2.571	−0.335
11	−2.583	−2.574	0.702	1.023
12	1.340	−2.582	−0.158	−2.583

Figure 9.1: **An exemplified flowchart to illustrate the two different approaches for motif identification.** I analyzed 800 bp upstream from the translation start sites of the 5 genes from the yeast gene family PHO by the publicly available systems MEME (alignment, http://meme.sdsc.edu) and RSA (exhaustive search, http://www.ucmb.ulb.ac.be/bioinformatics/rsa-tools). MEME was run on both strands, one occurrence per sequence mode, and found the known motif ranked as second best. RSA tools was run with oligo size 6 and non-coding regions as background.

if the motif is not very well conserved among the sequences and the sequences to be examined become too large, and also showed that good background models can help to avoid this problem to some extent (Pevzner and Sze, 2000; Thijs et al., 2001).

New directions. Recent research concentrated on the detection of co-occurring motifs separated by a fixed (van Helden et al., 2000) or variable spacer length (Sinha and Tompa, 2000), or a variable motif length in general (Bussemaker et al., 2000). To allow for mismatches, ambiguous nucleotide letters (see appendix A) are included in the oligomer alphabet.

Thus it seems as if the enumerative approach is the method of choice: It exhaustively searches over all possible oligomers and provides a global solution. In practice, though, alignment methods are more flexible: They can find long motifs the detection of which is simply not feasible by an exhaustive approach. Also, they deliver a probabilistic model for a motif, such as a linear HMM, which can be used more flexible than a string pattern for searching purposes. An ideal approach would therefore contain two steps: First apply an enumerative approach, and use the results to initialize or guide the model for an alignment method.

A new way to look at the data is to cluster genes based on *both* expression levels and common motifs at the same time (Holmes and Bruno, 2000; Bussemaker et al., 2001). This can help to separate gene groups that are active under the same conditions but belong to separate regulatory pathways.

The question remains how we can use all these methods when we move on to the analysis of higher eukaryotes with their highly complex genomes. The euchromatin of *D. melanogaster* has a gene density of roughly one gene every 9 kilobases and an average predicted transcript size of 3,058 bp (Adams et al., 2000), leaving a huge portion of the genome as potential locations of regulatory elements. In some cases, the alignment of non-coding sequences from two related species, known as phylogenetic footprinting, can help to narrow the search region and reveal conserved and potentially regulatory regions (Duret and Bucher, 1997, cf. section 3.3.2). A publication by Wasserman et al. (2000) closes the gap between this approach and motif identification: 28 orthologous co-regulated gene pairs from human and rat were automatically aligned to identify conserved un-gapped sequence blocks, and the subsequent analysis of the conserved parts with a Gibbs sampling approach revealed the known motifs that were missed otherwise. But in general, we will also need reliable *ab initio* promoter prediction, such as described in this work, to find the true regions in which the regulatory patterns are hidden.

Chapter 10

Summary

The massive data generation efforts that have taken place in molecular biology, such as the determination of the complete DNA sequence of a multitude of organisms, has lead to the new field of bioinformatics. It deals with the organization, classification, and interpretation of these data, and finally helps to give insight into the underlying phenomena. One topic of interest is the computer-assisted annotation of primary DNA sequence, which includes the localization of all genes and their regulatory regions. This thesis describes the automatic identification of eukaryotic promoters, regulatory DNA regions in higher organisms that largely control the expression of genes. Two different probabilistic promoter models for the fruit fly *Drosophila melanogaster* and man are trained and applied on a genome wide scale. These models take features of the DNA sequence as well as the DNA structure into account.

Hereditary information is stored within DNA, a double stranded molecule which consists of a string of basic units, the nucleotides. In eukaryotic organisms, the DNA is divided into a number of chromosomes to enable the necessary tight packing. The information stored on the DNA is organized in discrete stretches called genes. The largest class of protein-encoding genes is expressed in a two-step process: First, a polymerase enzyme transcribes the gene and synthesizes a complementary messenger RNA copy of it, then this copy is translated into a protein with a specific functional or structural role. Genes of other classes are not translated and lead to functional molecules different from proteins. Computer-assisted genome annotation can be divided into two stages. At first, structural annotation deals with the identification of sequence patterns such as genes and promoters, either by machine learning methods or by similarity to other sequences. Then, functional annotation derives information about proteins by matches to similar sequences with known function, or by prediction of certain properties of subunits and sites. Gene function can also be related to the organization of its regulatory region.

Gene regulation is observed on a number of levels, and perhaps the most important one is the

transcription by the polymerase. To a large extent, it is controlled by the promoter, a region close to the transcription start site. Of particular importance is the core promoter region, in which general transcription factors, i. e. proteins which guide the polymerase to the correct start site, bind to specific DNA sequence patterns. Subunits of only one general transcription factor undoubtedly interact with the DNA in a direct way. In the neighborhood of the core promoter, less frequent patterns interact with specific transcription factors. The combination of several of these patterns leads to the highly specific regulation of gene expression required for complex organisms, which must control the precise time and place of the activation of every gene in a genome. Apart from the DNA sequence, DNA structure in promoter regions plays a key role in regulation: A necessary pre-requisite to transcription is the accessibility of DNA to transcription factors and the polymerase. Therefore, potentially active regulatory regions can be found in an open structure of a chromosome and in less flexible parts of the DNA double helix. Enhancer regions up to several thousand nucleotides away have also been found to affect the transcriptional activity of a promoter. On a higher level, locus control regions and matrix attachment regions influence the activity of several genes at once.

Computational promoter recognition for eukaryotes has so far been based on models of the promoter sequence only. Signal approaches identify transcription factor binding sites, and make a decision based on the co-occurrence of several patterns within a certain window. Neural networks or positional weight matrices are the most common models. Content approaches use oligonucleotide statistics of long promoter and non-promoter sequences, for example in the form of Markov chains. Hybrid approaches combine both ideas. The DNA structure in promoter regions has been extensively studied, but only one approach to recognize promoters using structural features has been described, and it focused on prokaryotes. For a few well-studied cases, models for the recognition of a small promoter sub-class have been proposed, but not enough knowledge has been gained to cover a large fraction of all promoters.

Representative human and *Drosophila* training sequences were obtained: The promoters were taken from the Eukaryotic Promoter Database and one additional collection, and the non-promoter sequences from data sets collected for the GENIE gene finder. For an independent evaluation, 92 putative promoter regions were extracted from the *Drosophila Adh* region. In addition, the start points of 339 known genes of human chromosome 22 (release 2.3) were used.

Probabilistic models that are trained on these data sets include stationary Markov chains and stochastic segment models. Markov chains calculate the probability of each symbol in a sequence conditioned on a limited number of predecessors, the context. The size of the context denotes the order of the model. If the probability distributions are independent of the sequence position, the Markov chain is called stationary. The parameters then consist of the set of nucleotide probability distributions conditioned on all possible contexts. A common objective function to estimate the parameters is maximum likelihood (ML). Discriminative objective functions such as maximum mutual information (MMI) take both negative and positive samples into account and can lead to better classification results than ML. For MMI estimation, a modified gradient descent technique suited for rational objective functions is devised. The complexity can be tackled with a corrective training scheme, where only mis-classified samples are considered in each iteration.

The number of Markov chain parameters grows exponentially with the order, and models of higher order easily over-adapt to the training data. Two approaches are described to take long-range dependencies into account despite this: Interpolated Markov chains use weighted Markov chains of different orders, and can be additionally adjusted to the frequency of each context. In contrast, variable length Markov chains contain conditional distributions of differing order, represented as a context tree. A node is part of this tree if it provides additional information when compared to its parent and can be estimated reliably. Two algorithms for building context trees are described, and a cross-validation scheme for the construction of optimal trees is proposed.

Stochastic segment models (SSMs) provide a more complex framework for sequence modeling. They consist of a number of states and transitions between them. Each time a state is visited, a subsequence of a certain length is emitted, according to a distribution on the length of the emitted sequence and a distribution to generate a particular sequence of that length. SSMs are therefore generalized hidden Markov models (HMMs): In contrast to HMMs which emit only one symbol per state visit, the SSM uses an explicit length modeling, and the output distributions are allowed to take arbitrary dependencies among the symbols into account. Algorithms to compute sequence likelihoods and to train the parameters of the model are derived from simpler versions for HMMs, and special care is taken to ensure a practical runtime complexity.

With a different approach, promoters can be classified using features related to DNA structural properties. The first group of features is related to CpG islands, regions with a high GC content and high CG di-nucleotide frequency. These regions are correlated with an open chromosomal structure, therefore hinting at accessible DNA regions. The other feature group is extracted from DNA property profiles. A profile is a transformed representation of a sequence, where overlapping di- or tri-nucleotides are replaced by experimentally defined values related to properties such as DNA bendability. The average value and the slope of the regression line of several profile sub-regions are used as features and modeled with Gaussian (mixture) distributions. To combine the features of several properties, they are evaluated by principal component analysis.

Representatives of two different groups of decision functions are used to distinguish between promoters and non-promoters. As an example of statistical classifiers, a modified Bayesian classifier is described. Bayes' classification is based on the highest *a posteriori* probability, which minimizes the risk of mis-classification under certain assumptions. Neural networks are an example of distribution-free classifiers. Only the most popular type of multi-layer perceptrons is considered, in which a number of computing nodes is arranged in several layers. Each node is only connected to the layers immediately preceding and following it, and the training algorithm of error back-propagation explicitly exploits this restricted topology.

All these techniques are finally put together in the McPromoter system for promoter recognition in eukaryotic genomic DNA. A 300 bp moving window is analyzed by probabilistic models of promoter and non-promoter-sequences, and then used to classify the sequence window. To suppress the effect of multiple adjacent hits, the classifier output is smoothed, and only the best local maxima within a certain range and above a certain threshold are retained.

To choose among the presented models and classifiers, detailed cross-validation classification experiments are performed on sequence sets of positive and negative samples. At first, different Markov chain models with Bayes' classification rule are compared. MMI parameter estimation turns out to be equal or slightly better than ML estimation. Variable-length Markov chains are less prone to over-adaptation with increasing model order, but a greater improvement is gained by interpolated Markov chains, where no over-adaptation is observed for practical model orders. Equal recognition rates of 82.6 % (human) and 83.0 % (*Drosophila*) are obtained. Promoter sequences are screened for positionally conserved patterns, and different linear stochastic segment models are trained accordingly. The strength of the patterns represented in the states is examined. The classification results are an improvement when compared to Markov chain promoter models: 86.9 % for human, and 85.6 % for *Drosophila*. To account for non-linear dependencies among the promoter states, a multi-layer perceptron replaces the Bayes classifier. It takes the normalized likelihoods of promoter states and background models as input. To avoid over-adaptation, the sequence models and the network are trained on independent parts of the data.

A multi-layer perceptron is also used to classify promoters using features derived from structural profiles. With features derived from single profiles, better classification results are obtained for *Drosophila* than for human. The combination of all profiles with principal component analysis leads to better results for *Drosophila*: Using the first component of 14 features leads to an equal recognition rate of 74%. For human, the overall best rate is 64.3%.

To localize promoters in genomic DNA, the sequence and profile features are finally joined in a larger neural network and compared to the other models under consideration. For *Drosophila*, the results indeed improve when the model takes more dependencies or features into account, and the sequence/profile joint models perform best. The output threshold causing the largest distance from random predictions leads to a sensitivity of 52.1% and a false prediction every 3,791 bases. This is significantly better than other published approaches. Due to the strong correlation of many human promoters to CpG islands, interpolated Markov chain models are as good as the sequence/profile joint model, and at a sensitivity of 64.3%, a false prediction is made every 32,411 bases. This is competitive with all other systems evaluated on the same data set so far.

Appendix A

Ambiguous Nucleotide Letters

To deal with incomplete specification of bases in nucleic acid sequences, the Nomenclature Committee of the International Union of Biochemistry (NC-IUB) issued a nomenclature where single letter symbols are assigned to groups of nucleotides. This is useful in cases where two or more bases are permitted at a particular position, or where uncertainty exists as to extent and/or identity. These ambiguous codes are often used to describe consensus sequences, i. e. the common denominator of several instances of a binding site. They are given in table A.1.

G	G	Guanine
A	A	Adenine
T	T	Thymine
C	C	Cytosine
R	G or A	puRine
Y	T or C	pYrimidine
M	A or C	aMino
K	G or T	Keto
S	G or C	Strong interaction (3 H bonds)
W	A or T	Weak interaction (2 H bonds)
H	A or C or T	not-G, H follows G in the alphabet
B	G or T or C	not-A, B follows A
V	G or C or A	not-T (not-U), V follows U
D	G or A or T	not-C, D follows C
N	G or A or T or C	aNy

Table A.1: Ambiguous nucleotide letter code

Appendix B

Conversion of a context tree into an automaton

It was mentioned in section 5.2.3 that it is advantageous to convert a context tree representing a variable length Markov chain into a probabilistic finite automaton (PFA) that generates symbols according to the same probability distribution. The reason for that is that one has to descend up to N steps to reach a leaf in the context tree of Nth order. A PFA, on the other side, only has to look up the transition probability in a matrix, which means a speed gain of up to factor N. In the following we will use the same notation as in chapter 5 and figures 5.2 and 5.3. For $\hat{\boldsymbol{v}} = \hat{v}_1 \dots \hat{v}_l$, $\text{prefix}(\hat{\boldsymbol{v}}) = \hat{v}_1 \dots \hat{v}_{l-1}$.

A PFA consists of a set of states $Q = \{q^0, \dots, q^{n-1}\}$. For each symbol $v \in V$, there exists a transition to another state, as defined by the transition function $\sigma : Q \times V \longrightarrow Q$. A symbol is generated (and therefore, a transition is followed) with a certain probability, given by the probability distribution over the next symbol $\gamma : Q \times V \longrightarrow [0, 1]$. A start vector $\boldsymbol{\pi} : Q \longrightarrow [0, 1]$ completes the definition of a PFA, which is given by the 5-tuple $\{Q, V, \sigma, \gamma, \boldsymbol{\pi}\}$. The probability to generate a sequence of symbols $\boldsymbol{w} = w_1 \dots w_T$ is therefore calculated like this:

$$P(\boldsymbol{w}) = \sum_{q_0 \in Q} \pi_{q_0} \cdot \prod_{i=0}^{T-1} \gamma(q_i, w_{i+1}), \tag{B.1}$$

with $q_{i+1} = \sigma(q_i, w_{i+1})$.

The conversion of a context tree into a PFA is straightforward:

1. Extend the tree to guarantee a well defined transition function.
2. Build up the transition function σ.
3. Identify each node of the extended tree with a state of the PFA; set the corresponding entries in γ to the probability distribution of the node.

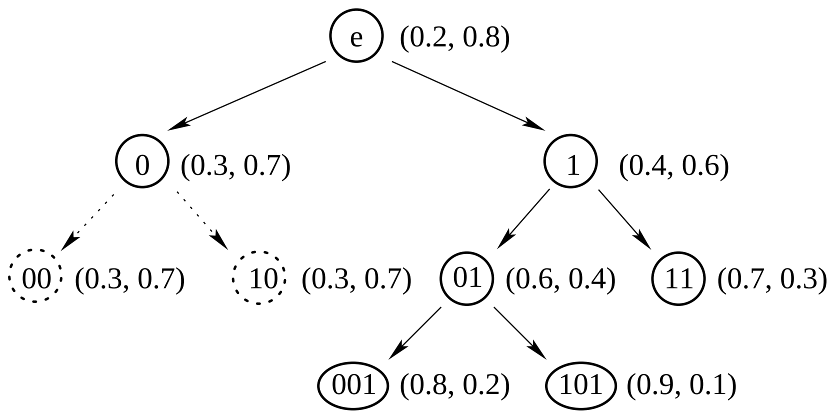


Figure B.1: **A context tree**, from Kulicke (2000). The tree was extended by the leaves 01 and 11, depicted with dotted lines.

4. Set $\boldsymbol{\pi}$ to $(1, 0, \ldots, 0)$, i. e. let the state representing the root node be the only possible start state.

Naturally, one would start the conversion by taking the *leaves* of the tree as states of the automaton. But the next node in the tree might depend on symbols that are further back in history and therefore not contained in the current node label. Figure B.1 shows such a context tree, where the transition function would not be well defined without the dotted extra leaves: From the leaf with label 0, transitions to both 001 and 101 are possible when observing a 1, depending on whether a 0 or a 1 was observed before the label 0. In general, this happens when there is a leaf with label $\hat{\boldsymbol{v}}$ and a symbol v so that no leaf can be found whose label $\hat{\boldsymbol{v}'}$ is a suffix of $\hat{\boldsymbol{v}}v$. That means that for every leaf in the tree, the longest prefix must be either a leaf or an internal node. If this is not the case, the leaf must be extended, and the new children inherit the probability distribution from the parent node. For sparse trees, this might have the consequence of a much larger automaton.

The function σ is then build by searching for *every* node $\hat{\boldsymbol{v}}$ of the tree and every symbol v which is the node with the label $\hat{\boldsymbol{v}'} = \text{suffix}(\hat{\boldsymbol{v}}v)$. The resulting PFA for the example tree in figure B.1 is depicted in figure B.2. One can see that once the states corresponding to the inner nodes of the tree have been left, the automaton remains in states of former leaf nodes. This part corresponds to a probabilistic suffix automaton which has the property that the set of its state labels SL is suffix free:

$$\forall sl \in SL : \text{suffix}^*(sl) \cap SL = \{sl\} \tag{B.2}$$

with $\text{suffix}^*(\boldsymbol{x}) = \{x_i \ldots x_l | 1 \leq i \leq l\} \cup \{e\}$. The part outside the suffix automaton can be re-

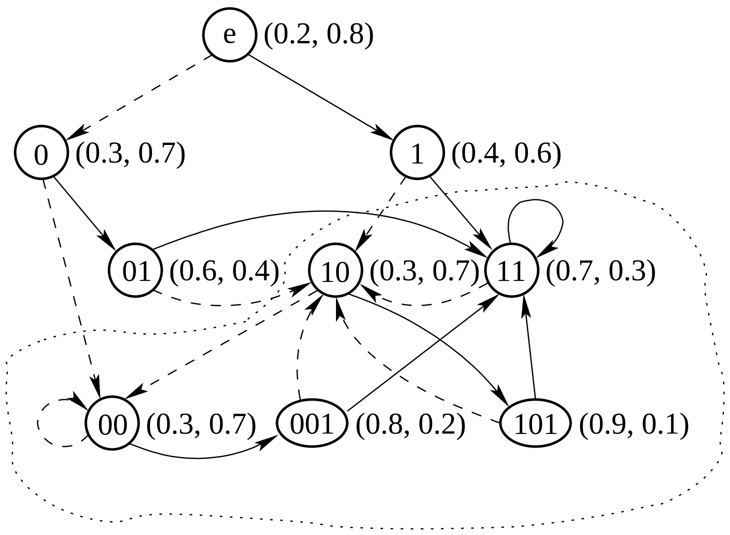


Figure B.2: **Result of the conversion** of the extended context tree in picture B.1 to a PFA (from Kulicke (2000)). The circled part corresponds to a probabilistic suffix automaton, the part outside to the inner nodes of the original context tree.

placed by a more elaborate start distribution π if an additional property holds; interested readers are referred to (Ron et al., 1996).

Appendix C

Parameters of physico-chemical properties

This appendix contains the complete list of parameter sets used to compute physico-chemical profiles (see chapter 6).

First	Second base			
base	A	C	G	T
A	0.97	0.13	0.33	0.58
C	1.04	0.19	0.52	0.33
G	0.98	0.73	0.19	0.13
T	0.73	0.98	1.04	0.97

Table C.1: Parameters for A-philicity

First	Second base			
base	A	C	G	T
A	35.5	33.1	30.6	43.2
C	37.7	35.3	31.3	30.6
G	39.6	38.4	35.3	33.1
T	31.6	39.6	37.7	35.5

Table C.2: Parameters for B-DNA twist

First	Second	Third base			
base	base	A	C	G	T
A	A	-0.274	-0.205	-0.081	-0.280
	C	-0.006	-0.032	-0.033	-0.183
	G	0.027	0.017	-0.057	-0.183
	T	0.182	-0.110	0.134	-0.280
C	A	0.015	0.040	0.175	0.134
	C	-0.246	-0.012	-0.136	-0.057
	G	-0.003	-0.077	-0.136	-0.033
	T	0.090	0.031	0.175	-0.081
G	A	-0.037	-0.013	0.031	-0.110
	C	0.076	0.107	-0.077	0.017
	G	0.013	0.107	-0.012	-0.032
	T	0.025	-0.013	0.040	-0.205
T	A	0.068	0.025	0.090	0.182
	C	0.194	0.013	-0.003	0.027
	G	0.194	0.076	-0.246	-0.006
	T	0.068	-0.037	0.015	-0.274

Table C.3: Parameters for DNA bendability

First	Second base			
base	A	C	G	T
A	35	60	60	20
C	60	130	85	60
G	60	85	130	60
T	20	60	60	35

Table C.4: Parameters for DNA bending stiffness

First	Second base			
base	A	C	G	T
A	66.51	108.80	85.12	72.29
C	64.92	99.31	88.84	85.12
G	80.03	135.83	99.31	108.80
T	50.11	80.03	64.92	66.51

Table C.5: Parameters for DNA denaturation

First	Second base			
base	A	C	G	T
A	1.9	1.3	1.6	0.9
C	1.9	3.1	3.6	1.6
G	1.6	3.1	3.1	1.3
T	1.5	1.6	1.9	1.9

Table C.6: Parameters for duplex stability — disrupt energy

First	Second base			
base	A	C	G	T
A	-1.2	-1.5	-1.5	-0.9
C	-1.7	-2.3	-2.8	-1.5
G	-1.5	-2.3	-2.3	-1.5
T	-0.9	-1.5	-1.7	-1.2

Table C.7: Parameters for duplex stability — free energy

First base	Second base	Third base A	C	G	T
A	A	0	1	1	0
	C	1	2	2	1
	G	1	2	2	1
	T	0	1	1	0
C	A	1	2	2	1
	C	2	3	3	2
	G	2	3	3	2
	T	1	2	2	1
G	A	1	2	2	1
	C	2	3	3	2
	G	2	3	3	2
	T	1	2	2	1
T	A	0	1	1	0
	C	1	2	2	1
	G	1	2	2	1
	T	0	1	1	0

Table C.8: Parameters for GC trinucleotide content

First base	Second base	Third base			
		A	C	G	T
A	A	-36	-6	6	-30
	C	6	8	8	11
	G	-9	25	8	11
	T	-13	7	18	-30
C	A	-9	17	-2	18
	C	8	13	2	8
	G	31	25	2	8
	T	-18	8	-2	6
G	A	-12	8	8	7
	C	13	45	25	25
	G	-5	45	13	8
	T	-6	8	17	-6
T	A	-20	-6	-18	-13
	C	8	-5	31	-9
	G	8	13	8	6
	T	-20	-12	-9	-36

Table C.9: Parameters for nucleosome positioning

First base	Second base			
	A	C	G	T
A	-18.66	-13.10	-14.00	-15.01
C	-9.45	-8.11	-10.03	-14.00
G	-13.48	-11.08	-8.11	-13.10
T	-11.85	-13.48	-9.45	-18.66

Table C.10: Parameters for propeller twist

First	Second base			
base	A	C	G	T
A	35.1	31.5	31.9	29.3
C	37.3	32.9	36.1	31.9
G	36.3	33.6	32.9	31.5
T	37.8	36.3	37.3	35.1

Table C.11: Parameters for protein-DNA-twist

First	Second base			
base	A	C	G	T
A	2.9	2.3	2.1	1.6
C	9.8	6.1	12.1	2.1
G	4.5	4.0	6.1	2.3
T	6.3	4.5	9.8	2.9

Table C.12: Parameters for protein-induced deformability

First	Second base			
base	A	C	G	T
A	-5.37	-10.51	-6.78	-6.57
C	-6.57	-8.26	-9.69	-6.78
G	-9.81	-14.59	-8.26	-10.51
T	-3.82	-9.81	-6.57	-5.37

Table C.13: Parameters for stacking energy

First	Second base			
base	A	C	G	T
A	3.9	4.6	3.4	5.9
C	1.3	2.4	0.7	3.4
G	3.4	4.0	2.4	4.6
T	2.5	3.4	1.3	3.9

Table C.14: Parameters for Z-DNA stabilizing energy

Appendix D

Design and Implementation Details

In this appendix, some remarks about the design and implementation of the McPromoter system are discussed, and references to programs developed by other researchers are given.

D.1 Densities for Sequences

In figure D.1, the main C++ classes for DNA sequence modeling and their relations are depicted (see Stroustrup (1997) for details on the C++ programming language). All classes are derived from the abstract class Density and are integrated into the C++ programming environment PUMA (Paulus and Hornegger, 2001). The abstract class specifies that all the derived classes have to provide certain functions, such as the calculation of probabilities.

One branch of the hierarchy models Markov chain densities (section 5.2), either as interpolated Markov chains with encapsulated C code by Schukat-Talamazzini et al. (1997), or as variable length Markov chains, described in more detail by Kulicke (2000).

A segment model (section 5.3) is also derived from the Density class and has to contain instances of SegmentDensity, which are the states of a segment model. General algorithms such as Viterbi training and the forward algorithm are implemented on this level, as we know that all possible densities are derived from the same base class Density and therefore provide the necessary functions. More specific versions of the segment model algorithms can be implemented in derived classes, as is the case for the SegmentMarkov class in which the states are restricted to be segment densities that are based on a Markov chain model. SegmentDensity also contains an instance of a duration distribution, in our case of a discrete histogram DurHisto, which is again derived from the Density base class.

Continuous Gaussian densities which are used to model DNA property profile features (see chapter 6) are also derived from the Density base class; this is described in more detail in

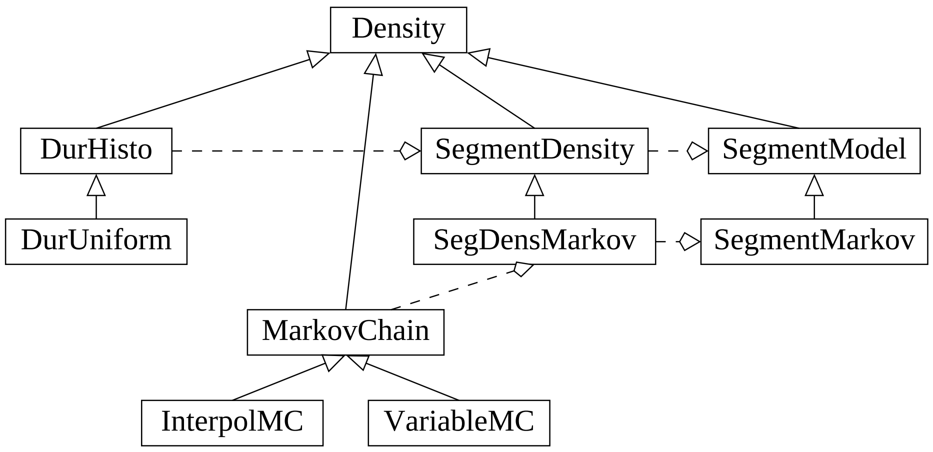


Figure D.1: **The class hierarchy for sequence modeling.**

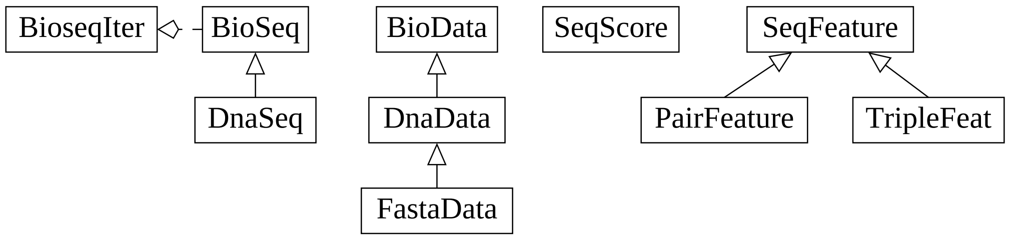


Figure D.2: **The class hierarchy for DNA sequence handling.**

(Hornegger, 1996).

D.2 DNA sequence handling classes

A number of C++ classes were designed to enable the processing and manipulation of DNA sequences (figure D.2). The class DnaSeq is derived from a more general class BioSeq and contains methods to substitute ambiguous symbols, give the reverse complement of a sequence, give a sub-sequence and so forth. A class BioseqIter serves as an iterator class which moves a specified window along a DNA sequence object. For input and output, BioData and derived classes are provided, which contain methods to read in or store files with sets of DNA sequences in a certain format and convert them to DnaSeq objects.

Continuous values referring to positions in sequences are handled with SeqScore. In the case of the McPromoter system, this is both the output of the system as well as the property profiles computed along sequences (see section 6.2). This class contains various filters as well as methods to compute the features derived from property profiles.

The property profiles themselves are calculated by SeqFeature objects, which are instantiated with a di- or tri-nucleotide parameter set (PairFeature respectively TripleFeature, see appendix C).

D.3 Further references

The neural networks used in the McPromoter system were trained with the Stuttgart Neural Network Simulator SNNS version 4.2 (Zell et al., 1999) and converted to C-code. The principal component analysis was implemented in C by Georg Stemmer at the Chair for Pattern Recognition, University of Erlangen. The output of the system is given in the general feature format (http://www.sanger.ac.uk/Software/formats/GFF/).

Appendix E

DVD with Data Sets

A DVD with all data sets described in chapter 4 are available upon request. These include the representative sequence sets for model training as well as the contiguous DNA sequences used to evaluate the system, along with the lists of transcription start sites in them.

Bibliography

M. D. Adams et al. The genome sequence of *Drosophila melanogaster*. *Science*, 287:2185–2195, 2000.

B. Alberts, D. Bray, J. Lewis, M. Raff, K. Roberts, and J. D. Watson, editors. *Molecular Biology of the Cell*. Garland Pub, 3rd edition, 1994.

S. F. Altschul, W. Gish, W. Miller, E.W. Myers, and D. J. Lipman. Basic local alignment search tool. *J Mol Biol.*, 215:403–410, 1990.

F. Antequera and A. Bird. Number of CpG islands and genes in human and mouse. *Proc. Natl Acad. Sci. U.S.A.*, 90:11995–11999, 1993.

I. Arkhipova. Promoter elements in *Drosophila melanogaster* revealed by sequence analysis. *Genetics*, 139:1359–1369, 1995.

M. Ashburner et al. An exploration of the sequence of a 2.9-megabase region of the genome of *Drosophila melanogaster* — The 'Adh' region. *Genetics*, 153(1):179–219, 1999.

S. Audic and J.-M. Claverie. Detection of eukaryotic promoters using Markov transition matrices. *Comput Chem.*, 21:223–227, 1997.

S. Audic and J.-M. Claverie. Visualizing the competitive recognition of TATA-boxes in vertebrate promoters. *Trends Genet.*, 14:10–11, 1998.

V. N. Babenko, P. S. Kosarev, O. V. Vishnevsky, V. G. Levitsky, V. V. Basin, and A. S. Frolov. Investigating extended regulatory regions of genomic DNA sequence. *Bioinformatics*, 15:644–653, 1999.

V. Bafna and D. H. Huson. The conserved exon method for gene finding. In *Proc Int Conf Intell Syst Mol Biol.*, volume 8, pages 3–12, 2000.

L. Bahl, P. Brown, P. de Souza, and R. Mercer. Maximum Mutual Information estimation of hidden Markov model parameters for speech recognition. In *Proc. Int. Conf. on Acoustics, Speech and Signal Processing*, pages 49–52, Tokyo, 1986.

T. L. Bailey and C. Elkan. Unsupervised learning of multiple motifs in biolpolymers using expectation maximization. *Machine Learning*, 21:51–83, 1995.

A. Bairoch and R. Apweiler. The SWISS-PROT protein sequence database and its supplement TrEMBL in 2000. *Nucleic Acids Res.*, 28:45–48, 2000.

P. Baldi and S. Brunak. *Bioinformatics: The Machine Learning Approach*. MIT Press, Cambridge, MA, 1998.

P. Baldi, S. Brunak, Y. Chauvin, C. A. Andersen, and H. Nielsen. Assessing the accuracy of prediction algorithms for classification: an overview. *Bioinformatics*, 16:412–424, 2000.

P. Baldi, Y. Chauvin, T. Hunkapiller, and M. McClure. Hidden Markov models of biological primary sequence information. *Proc. Natl Acad. Sci. U.S.A.*, 91:1059–1063, 1994.

A. Bateman, E. Birney, R. Durbin, S. R. Eddy, K. L. Howe, and E. L. Sonnhammer. The Pfam protein families database. *Nucleic Acids Res.*, 28:263–266, 2000.

S. Batzoglou, L. Pachter, J. P. Mesirov, B. Berger, and E. S. Lander. Human and mouse gene structure: comparative analysis and application to exon prediction. *Genome Res.*, 10(7):950–958, 2000.

G. Bejerano and G. Yona. Variations on probabilistic suffix trees: statistical modeling and prediction of protein families. *Bioinformatics*, 17:23–43, 2001.

E. Birney, A. Bateman, M. E. Clamp, and T. J. Hubbard. Mining the draft human genome. *Nature*, 409: 827–828, 2001.

E. Birney and R. Durbin. Using GeneWise in the *Drosophila* annotation experiment. *Genome Res.*, 10(4): 547–548, 2000.

C. M. Bishop. *Neural Networks for Pattern Recognition*. Clarendon Press, Oxford, 1995.

M. Blanchette, B. Schwikowski, and M. Tompa. An exact algorithm to identify motifs in orthologous sequences from multiple species. In *Proc Int Conf Intell Syst Mol Biol.*, volume 8, pages 37–45, 2000.

M. Borodovsky, K. E. Rudd, and E. V. Koonin. Intrinsic and extrinsic approaches for detecting genes in a bacterial genome. *Nucleic Acids Res.*, 22:4756–4767, 1994.

J. M. Bower and H. Bolouri, editors. *Computational Modeling of Genetic and Biochemical Networks*. MIT Press, Cambridge, MA, 2001.

A. Brazma, I. Jonassen, J. Vilo, and E. Ukkonen. Predicting gene regulatory elements in silico on a genomic scale. *Genome Res.*, 8(11):1202–1215, 1998.

R. J. Britten and E. H. Davidson. Gene regulation for higher cells: a theory. *Science*, 165:349–358, 1969.

P. Bucher. Weight matrix description of four eukaryotic RNA polymerase II promotor elements derived from 502 unrelated promotor sequences. *J Mol Biol.*, 212:563–578, 1990.

P. Bühlmann. Model selection for variable length Markov chains and tuning the context algorithm. In *Annals of the Institute of Statistical Mathematics*, volume 52, pages 287–315. ETH Zürich, 2000.

P. Bühlmann and A. J. Wyner. Variable length Markov chains. *Annals of Statistics*, 27:480–513, 1999.

C. Burge. *Identification of Genes in Human Genomic DNA.* PhD thesis, Stanford University, 1997.

C. Burge and S. Karlin. Prediction of complete gene structures in human genomic DNA. *J Mol Biol.*, 268: 78–94, 1997.

C. Burge and S. Karlin. Finding the genes in genomic DNA. *Current Opinion in Structural Biology*, 8: 346–354, 1998.

T. W. Burke and J. T. Kadonaga. The downstream core promoter element, DPE, is conserved from Drosophila to humans and is recognized by TAFII60 of Drosophila. *Genes Dev*, 11:3020–3031, 1997.

M. Burset and R. Guigo. Evaluation of gene structure prediction programs. *Genomics*, 34(3):353–367, 1996.

H. Bussemaker, H. Li, and E. D. Siggia. Building a dictionary for genomes: Identification of presumptive regulatory sites by statistical analysis. *Proc. Natl Acad. Sci. U.S.A.*, 97:10096–10100, 2000.

H. Bussemaker, H. Li, and E.D. Siggia. Regulatory element detection using correlation with expression. *Nat. Genet.*, 27:167–171, 2001.

G. E. Chalkley and C. P. Verrijzer. DNA binding site selection by RNA polymerase II TAFs: a TAF$_{II}250$–TAF$_{II}150$ complex recognizes the initiator. *EMBO J*, 18:4835–4845, 1999.

Q. K. Chen, G. Z. Hertz, and G. D. Stormo. PromFD 1.0: a computer program that predicts eukaryotic pol II promoters using strings and IMD matrices. *Bioinformatics*, 13:29–35, 1997.

J.-M. Claverie and I. Sauvaget. Assessing the biological significance of primary structure consensus patterns using sequence databanks. I. Heat-shock and glucocorticoid control elements in eukaryotic promoters. *Comp Appl Biosc.*, 2:95–104, 1985.

P. Clote and R. Backofen. *Computational Molecular Biology: An Introduction.* John Wiley & Sons, 2000.

E. M. Crowley, K. Roeder, and M. Bina. A statistical model for locating regulatory regions in genomic DNA. *J Mol Biol.*, 268:8–14, 1997.

J. L. DeRisi, V. R. Iyer, and P. O. Brown. Exploring the metabolic and genetic control of gene expression on a genomic scale. *Science*, 278:680–685, 1997.

R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. John Wiley & Sons, New York, 2nd edition, 2000.

I. Dunham et al. The DNA sequence of human chromosome 22. *Nature*, 402:489–495, 1999.

R. Durbin, S. Eddy, A. Krogh, and G. Mitchison. *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids*. Cambridge University Press, Cambridge, 1998.

L. Duret and P. Bucher. Searching for regulatory elements in human noncoding sequences. *Current Opinion in Structural Biology*, 7:399–406, 1997.

S. Eddy. Multiple alignment using hidden Markov models. In *Proc Int Conf Intell Syst Mol Biol.*, volume 3, pages 114–120, 1995.

S. Eddy, G. Mitchison, and R. Durbin. Maximum discrimination hidden Markov models of sequence consensus. *J Comp Biol.*, 2(1):9–23, 1995.

J. W. Fickett and A. G. Hatzigeorgiou. Eukaryotic promoter recognition. *Genome Res.*, 7:861–878, 1997.

J. W. Fickett and W. W. Wasserman. Discovery and modeling of transcriptional regulatory regions. *Curr Opin Biotechnology*, 11:19–24, 2000.

W. Fleischmann, S. Moller, A. Gateau, and R. Apweiler. A novel method for automatic functional annotation of proteins. *Bioinformatics*, 15:228–233, 1999.

L. Florea, G. Hartzell, Z. Zhang, G. M. Rubin, and W. Miller. A computer program for aligning a cDNA sequence with a genomic sequence. *Genome Res.*, 8:967–974, 1998.

N. Friedman, M. Linial, I. Nachman, and D. Pe'er. Using Bayesian networks to analyze expression data. *J Comp Biol.*, 7:601–620, 2000.

M. Gardiner-Garden and M. Frommer. CpG islands in vertebrate genomes. *J Mol Biol.*, 196:261–282, 1987.

W. Gish and D. J. States. Identification of protein encoding regions by database similarity search. *Nature Genet*, 3:266–272, 1993.

P. S. Gopalakrishnan, D. Kanevsky, A. Nadas, and D. Nahamoo. A generalization of the Baum algorithm to rational objective functions. In *Proc. Int. Conf. on Acoustics, Speech, and Signal Processing*, 1989.

B. R. Graveley. Alternative splicing: increasing diversity in the proteomic world. *Trends Genet.*, 17: 100–107, 2001.

R. Guigo, P. Agarwal, J. F. Abril, M. Burset, and J. W Fickett. An assessment of gene prediction accuracy in large DNA sequences. *Genome Res.*, 10:1631–1642, 2000.

B. B. Haab, M. J. Dunham, and P. O. Brown. Protein microarrays for highly parallel detection and quantitation of specific proteins and antibodies in complex solutions. *Genome Biology*, 2:research0004.1–0004.13, 2001.

S. Hannenhalli and S. Levy. Promoter prediction in the human genome. *Bioinformatics*, 17:S90–S96, 2001.

R. C. Hardison. Conserved noncoding sequences are reliable guides to regulatory elements. *Trends Genet.*, 16:369–372, 2000.

D. Haussler. Computational gene finding. *Trends supplement*, pages 12–15, 1998.

R. Hendrych. Permutierte Polygramme zur grammatischen Sprachmodellierung. Diploma thesis, University of Erlangen-Nuremberg, 1995.

G. Z. Hertz and G. D. Stormo. Identifying DNA and protein patterns with statistically significant alignments of multiple sequences. *Bioinformatics*, 15:563–577, 1999.

I. Holmes and W. J. Bruno. Finding regulatory elements using joint likelihoods for sequence and expression profile data. In *Proc Int Conf Intell Syst Mol Biol.*, volume 8, pages 202–210, 2000.

J. Hornegger. *Statistische Modellierung, Klassifikation und Lokalisation von Objekten*. Shaker, Aachen, 1996.

G. B. Hutchinson. The prediction of vertebrate promoter regions using differential hexamer frequency analysis. *Comp Appl Biosc.*, 12(5):391–398, 1996.

I. P. Ioshikhes and M. Q. Zhang. Large-scale human promoter mapping using CpG islands. *Nat. Genet.*, 26:61–63, 2000.

F. Jelinek. Self–organized Language Modeling for Speech Recognition. In A. Waibel and K.-F. Lee, editors, *Readings in Speech Recognition*, pages 450–506. Morgan Kaufmann, San Mateo, 1990.

F. V. Jensen. *An Introduction to Bayesian Networks*. UCL Press, London, 1996.

N. Kingsbury and P. Rayner. Digital filtering using logarithmic arithmetic. *Electronical Letters*, 7:56–58, 1971.

R. Knippers. *Molekulare Genetik*. Thieme, Stuttgart, 7th edition, 1997.

S. Knudsen. Promoter2.0: for the recognition of PolII promoter sequences. *Bioinformatics*, 15:356–361, 1999.

D. M. Koelle, H. B. Chen, M. A. Gavin, A. Wald, W. W. Kwok, and L. Corey. CD8 CTL from genital Herpes Simplex lesions: Recognition of viral tegument and immediate early proteins and lysis of infected cutaneous cells. *J Immunol*, 166:4049–4058, 2001.

H. Köstler. Analyse von eukaryontischen Promotorregionen mit exhaustiver Suche. Student's thesis, University of Erlangen-Nuremberg, 2001.

U. Krengel. *Einführung in die Wahrscheinlichkeitstheorie und Statistik*. Vieweg, Wiesbaden, 5th edition, 2000.

A. Krogh. Two methods for improving performance of an HMM and their application for gene finding. In *Proc Int Conf Intell Syst Mol Biol.*, volume 5, pages 179–186. AAAI Press, 1997.

A. Krogh. Using database matches with HMMGene for automated gene detection in Drosophila. *Genome Res.*, 10(4):523–528, 2000.

A. Krogh, M. Brown, I. S. Mian, K. Sjölander, and D. Haussler. Hidden Markov models in computational biology. *J Mol Biol.*, 235:1501–1531, 1994a.

A. Krogh, B. Larsson, G. von Heijne, and E. L. Sonnhammer. Predicting transmembrane protein topology with a hidden Markov model: application to complete genomes. *J Mol Biol.*, 305:567–580, 2001.

A. Krogh, I. S. Mian, and D. Haussler. A hidden Markov model that finds genes in E. coli DNA. *Nucleic Acids Res.*, 22:4768–4778, 1994b.

A. Kulicke. Variable Length Markov Chains. Student's thesis, University of Erlangen-Nuremberg, 2000.

D. Kulp, D. Haussler, M. G. Reese, and F. H. Eeckman. A generalized hidden Markov model for the recognition of human genes in DNA. In *Proc Int Conf Intell Syst Mol Biol.*, volume 4, pages 134–142, 1996.

A. K. Kutach and J. T. Kadonaga. The downstream promoter element DPE appears to be as widely used as the TATA box in *Drosophila* core promoters. *Mol Cell Biol.*, 20:4754–4764, 2000.

T. Lagrange, A. N. Kapanidis, H. Tang, D. Reinberg, and R. H. Ebright. New core promoter element in RNA polymerase II-dependent transcription: sequence-specific DNA binding by transcription factor IIB. *Genes Dev.*, 12:34–44, 1998.

D. S. Latchman. *Gene Regulation — A Eukaryotic Perspective*. Stanley Thornes Ltd, 3rd edition, 1998.

C. E. Lawrence, S. F. Altschul, M. S. Boguski, J. S. Liu, A. F. Neuwald, and J. C. Wootton. Detecting subtle sequence signals: a Gibbs sampling strategy for multiple alignment. *Science*, 262(5131):208–214, 1993.

B. Lewin. *Genes VII*. Oxford Univ Press, 1999.

S. E. Lewis, M. Ashburner, and M. G. Reese. Annotating eukaryote genomes. *Curr Opin Struct Biol*, 10: 349–354, 2000.

G.-C. Liao, E. J. Rehm, and G. M. Rubin. Insertion site preferences of the P transposable element in *Drosophila melanogaster*. *Proc. Natl Acad. Sci. U.S.A.*, 97:3347–3351, 2000.

S. Lisser and H. Margalit. Determination of common structural features in *Escherichia coli* promoters by computer analysis. *Eur. J. Biochem.*, 223:823–830, 1994.

F. Lyko. DNA methylation learns to fly. *Trends Genet.*, 17:169–172, 2001.

S. Matis, Y. Xu, M. B. Shah, D. Buley, X. Guan, J. R. Einstein, R. J. Mural, and E. C. Uberbacher. Detection of RNA polymerase II promoters and polyadenylation sites in human DNA sequence. *Comput Chem.*, 20:135–140, 1996.

B. Merialdo. Phonetic recognition using hidden Markov models and Maximum Mutual Information training. In *Proc. Int. Conf. on Acoustics, Speech, and Signal Processing*, pages 111–114, 1988.

T. M. Mitchell. *Machine Learning*. McGraw Hill, New York, 1997.

A. Nadas, D. Nahamoo, and M. A. Picheny. On a model-robust training method for speech recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 36:1432–1435, 1988.

H. Niemann. *Klassifikation von Mustern*. Springer, Berlin, 1983.

H. Niemann. *Pattern Analysis and Understanding*. Springer, Berlin, 1990.

D. B. Nikolov and S. K. Burley. RNA polymerase II transcription initiation: A structural view. *Proc. Natl Acad. Sci. U.S.A.*, 94:15–22, 1997.

Y. Normandin and S. D. Morgera. An improved MMIE training algorithm for speaker-independent, small vocabulary, continuous speech recognition. In *Proc. Int. Conf. on Acoustics, Speech, and Signal Processing*, pages 537–540, 1991.

U. Ohler. Polygramme und Hidden Markov Modelle zur DNA–Sequenzanalyse. Student's thesis, University of Erlangen-Nuremberg, 1995.

U. Ohler. Promoter prediction on a genomic scale — the Adh experience. *Genome Res.*, 10:539–542, 2000.

U. Ohler, S. Harbeck, and H. Niemann. Discriminative estimation of language model classifiers. In G. Olaszy, G. Nemeth, and K. Erdöhegyi, editors, *Proc. European Conf. on Speech Communication and Technology*, volume 4, pages 1607–1610, Budapest, 1999a.

U. Ohler, S. Harbeck, H. Niemann, E. Nöth, and M. G. Reese. Interpolated Markov chains for eukaryotic promoter recognition. *Bioinformatics*, 15(5):362–369, 1999b.

U. Ohler, S. Harbeck, G. Stemmer, and H. Niemann. Stochastic segment models of eukaryotic promoter regions. In *Pac Symp Biocomput.*, volume 5, pages 380–391, 2000.

U. Ohler, G.-C. Liao, G. Stemmer, H. Niemann, and G. M. Rubin. Annotation of core promoters in the complete *Drosophila* genome. In preparation, 2002.

U. Ohler and H. Niemann. Identification and analysis of eukaryotic promoters: recent computational approaches. *Trends Genet.*, 17:56–60, 2001.

U. Ohler, H. Niemann, G. Liao, and G. M. Rubin. Joint modeling of DNA sequence and physical properties to improve eukaryotic promoter recognition. *Bioinformatics*, 17:S199–S206, 2001.

M. Ostendorf, V. Digalakis, and O. A. Kimball. From HMMs to Segment Models: a unified view of stochastic modeling for speech recognition. *IEEE Transactions on Speech and Audio Processing*, 4: 360–378, 1996.

J. Park, K. Karplus, C. Barrett, R. Hughey, D. Haussler, T. Hubbard, and C. Chothia. Sequence comparison using multiple sequences detect twice as many remote homologues as pairwise methods. *J Mol Biol.*, 284:1201–1210, 1998.

D. Paulus and J. Hornegger. *Applied Pattern Recognition: A Practical Introduction to Image and Speech Processing in C++*. Vieweg, Wiesbaden, 3rd edition, 2001.

P. Pavlidis, T. S. Furey, M. Liberto, D. Haussler, and W. N. Grundy. Promoter region-based classification of genes. In *Pac Symp Biocomput.*, volume 6, pages 151–164, 2001.

A. G. Pedersen, P. Baldi, S. Brunak, and Y. Chauvin. Characterization of prokaryotic and eukaryotic promoters using hidden Markov models. In *Proc Int Conf Intell Syst Mol Biol.*, volume 4, pages 182–191, 1996.

A. G. Pedersen, P. Baldi, Y. Chauvin, and S. Brunak. DNA structure in human RNA polymerase II promoters. *J Mol Biol.*, 281:663–673, 1998.

A. G. Pedersen, L. J. Jensen, S. Brunak, H.-H.Staerfeldt, and D. W. Ussery. A DNA structural atlas for *Escherichia coli*. *J Mol Biol.*, 299:907–930, 2000.

L. A. Pennacchio and E. M. Rubin. Genomic strategies to identify mammalian regulatory sequences. *Nature Reviews Genetics*, 2:100–109, 2001.

R. Cavin Perier, V. Praz, T. Junier, C. Bonnard, and P. Bucher. The Eukaryotic Promoter Database (EPD). *Nucleic Acids Res.*, 28:302–303, 2000.

P. Pevzner and S.-H. Sze. Combinatorial approaches to finding subtle signals in DNA sequences. In *Proc Int Conf Intell Syst Mol Biol.*, volume 8, pages 269–278, 2000.

W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, 2nd edition, 1993.

D. S. Prestridge. Predicting Pol II promoter sequences using transcription factor binding sites. *J Mol Biol.*, 249:923–932, 1995.

D. S. Prestridge and C. Burks. The density of transcriptional elements in promoter and non-promoter sequences. *Hum Mol Genet.*, 2:1449–1453, 1993.

L. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77:257–285, 1989.

L. Rabiner and B. H. Juang. *Fundamentals of Speech Recognition*. Prentice Hall, Englewood Cliffs, 1993.

M. G. Reese. *Computational Prediction of Gene Structure and Regulation in the Genome of* D. melanogaster. PhD thesis, University of Hohenheim, 2000.

M. G. Reese. Application of a time-delay neural network to the annotation of the *Drosophila melanogaster* genome. *Comput Chem.*, 2001. To appear.

M. G. Reese, G. Hartzell, N. L. Harris, U. Ohler, J. F. Abril, and S. E. Lewis. Genome annotation assessment in *Drosophila melanogaster*. *Genome Res.*, 10:483–501, 2000a.

M. G. Reese, D. Kulp, H. Tammana, and D. Haussler. Genie — gene finding in *Drosophila melanogaster*. *Genome Res.*, 10:529–538, 2000b.

P. Rimessi et al. Transcription pattern of human herpesvirus 8 open reading frame K3 in primary effusion lymphoma and Kaposi's sarcoma. *J Virol*, 75:7161–7174, 2001.

R. G. Roeder. The role of general initiaton factors in transcription by RNA polymerase II. *Trends Biochem. Sci.*, 21:327–334, 1996.

D. Ron, Y. Singer, and N. Tishby. The power of amnesia: Learning probabilistic automata with variable memory length. *Machine Learning*, 25:117–149, 1996.

P. Rouzé, N. Pavy, and S. Rombauts. Genome annotation: which tools do we have for it? *Curr Opin Plant Biol*, 2:90–95, 1999.

S. L. Salzberg, A. L. Delcher, S. Kasif, and O. White. Microbial gene identification using interpolated Markov models. *Nucleic Acids Res.*, 26:544–548, 1998a.

S. L. Salzberg, D. B. Searls, and S. Kasif, editors. *Computational Methods in Molecular Biology*. Elsevier Science, 1998b.

M. Scherf et al. First pass annotation of promoters on human chromosome 22. *Genome Res.*, 11:333–340, 2001.

M. Scherf, A. Klingenhoff, and T. Werner. Highly specific localization of promoter regions in large genomic sequences by PromoterInspector: a novel context analysis approach. *J Mol Biol.*, 297:599–606, 2000.

E. G. Schukat-Talamazzini. *Automatische Spracherkennung*. Vieweg, Braunschweig, 1995.

E. G. Schukat-Talamazzini, F. Gallwitz, S. Harbeck, and V. Warnke. Rational interpolation of Maximum Likelihood predictors in stochastic language modeling. In *Proc. European Conf. on Speech Communication and Technology*, pages 2731–2734, Rhodes, Greece, 1997.

Y. Seldin, G. Bejerano, and N. Tishby. Unsupervised sequence segmentation by a mixture of switching variable memory Markov sources. In *Proc. 18th Intl. Conf. Mach. Learn.*, pages 513–520, San Francisco, 2001.

D. D. Shoemaker et al. Experimental annotation of the human genome using microarray technology. *Nature*, 409:922–927, 2001.

G. B. Singh, J. A. Kramer, and S. A. Krawetz. Mathematical model to predict regions of chromatin attachment to the nuclear matrix. *Nucleic Acids Res.*, 25:1419–1425, 1997.

S. Sinha and M. Tompa. A statistical method for finding transcription factor binding sites. In *Proc Int Conf Intell Syst Mol Biol.*, volume 8, pages 344–354, 2000.

V. Solovyev and A. Salamov. The Gene-Finder computer tools for analysis of human and model organisms genome sequences. In *Proc Int Conf Intell Syst Mol Biol.*, volume 5, pages 294–302, 1997.

P. Spellman, G. Sherlock, M. Zhang, V. Iyer, K. Anders, M. Eisen, P. Brown, D. Botstein, and B. Futcher. Comprehensive identification of cell-cyle regulated genes of the yeast *Saccharomyces cerevisiae* by microarray hybridization. *Mol Biol Cell*, 9:3273–3297, 1998.

G. Stemmer. Segmentmodelle zur Promotorendetektion und Landessprachenidentifikation. Diploma thesis, University of Erlangen-Nuremberg, 1999.

G. Stoesser et al. The EMBL nucleotide sequence database. *Nucleic Acids Res.*, 29:17–21, 2001.

G. D. Stormo. DNA binding sites: representation and discovery. *Bioinformatics*, 16:16–23, 2000a.

G. D. Stormo. Gene-finding approaches for eukaryotes. *Genome Res.*, 10(4):394–397, 2000b.

B. Stroustrup. *The C++ Programming Language*. Addison-Wesley, Reading, Mass., 3rd edition, 1997.

Y. Suzuki et al. Identification and characterization of the potential promoter regions of 1031 kinds of human genes. *Genome Res.*, 11:677–684, 2001.

Y. Tateno, S. Miyazaki, M. Ota, H. Sugawara, and T. Gojobori. DNA data bank of Japan (DDBJ) in collaboration with mass sequencing teams. *Nucleic Acids Res.*, 28:24–26, 2000.

S. A. Teichmann, C. Chothia, and M. Gerstein. Advances in structural genomics. *Curr Opin Struct Biol*, 9:390–399, 1999.

The Arabidopsis Genome Initiative. Analysis of the genome sequence of the flowering plant arabidopsis thaliana. *Nature*, 406:796–815, 2000.

The Gene Ontology Consortium. Gene Ontology: tool for the unification of biology. *Nat. Genet.*, 25: 25–29, 2000.

The Genome International Sequencing Consortium. Initial sequencing and analysis of the human genome. *Nature*, 409:860–921, 2001.

The National Human Genome Research Institute. Glossary of genetic terms. http://www.nhgri.nih.gov/DIR/VIP/Glossary/, 2002.

G. Thijs, M. Lescot, K. Marchal, S. Rombauts, B. De Moor, P. Rouzé, and Y. Moreau. A higher-order background model improves the detection of promoter regulatory elements by Gibbs sampling. *Bioinformatics*, 2001. To appear.

R. Tupler, G. Perini, and M. R. Green. Expressing the human genome. *Nature*, 409:832–833, 2001.

J. van Helden, B. Andre, and J. Collado-Vides. Extracting regulatory sites from the upstream region of yeast by computational analysis of oligonucleotide frequencies. *J Mol Biol.*, 281:827–842, 1998.

J. van Helden, A. F. Rios, and J. Collado-Vides. Discovering regulatory elements in non-coding sequences by analysis of spaced dyads. *Nucleic Acids Res.*, 28(8):1808–1818, 2000.

J. C. Venter et al. The sequence of the human genome. *Science*, 291:1304–1351, 2001.

C. P. Verrijzer and R. Tjian. TAFs mediate transcriptional activation and promoter selectivity. *Trends Biochem. Sci.*, 21:338–342, 1996.

A. Wagner. Genes regulated cooperatively by one or more transcription factors and their identification in whole eukaryote genomes. *Bioinformatics*, 15:776–784, 1999.

V. Warnke, S. Harbeck, E. Nöth, H. Niemann, and M. Levit. Discriminative estimation of interpolation parameters for language model classifiers. In *Proc ICASSP*, pages 525–528, Phoenix, 1999.

W. W. Wasserman and J. W. Fickett. Identification of regulatory regions which confer muscle-specific gene expression. *J Mol Biol.*, 278:167–181, 1998.

W. W. Wasserman, M. Palumbo, W. Thompson, J. W. Fickett, and C. E. Lawrence. Human-mouse genome comparisons to locate regulatory sites. *Nature Gen.*, 26:225–228, 2000.

T. Werner. Models for prediction and recognition of eukaryotic promoters. *Mammalian Genome*, 10: 168–175, 1999.

D. L. Wheeler et al. Database resources of the National Center for Biotechnology Information. *Nucleic Acids Res.*, 29:11–16, 2001.

E. Wingender et al. The TRANSFAC system on gene expression regulation. *Nucleic Acids Res.*, 29: 281–283, 2001.

R.-F. Yeh, L. P. Lim, and C. B. Burge. Computational inference of homologous gene structures in the human genome. *Genome Res.*, 11:803–816, 2001.

A. Zell et al. Stuttgart neural network simulator. http://www-ra.informatik.uni-tuebingen.de/SNNS/, 1999.

M. Q. Zhang. Identifcation of human gene core promoters in silico. *Genome Res.*, 8:319–326, 1998.

J. Zhu and M. Q. Zhang. SCPD: a promoter database of the yeast *Saccharomyces cerevisiae*. *Bioinformatics*, 15(7/8):607–611, 1999.