

Prediction of Genes in Eukaryotic DNA

Diploma Thesis

Dipl. El. Ing. FH

HSR Hochschule für Technik Rapperswil

January 2, 2006

Authors:

Cassian Strässle and Markus Boos

Advisors:

Prof Dr. Guido M. Schuster¹

Assoc Prof Dr. Tom Ryen²

Prof Dr. Sven Ole Aase²

Prof Dr. Peter Ruoff²

¹HSR Hochschule für Technik Rapperswil

²UiS University of Stavanger, Norway

Contents

Abstract	iii
Problem	v
1. Introduction	1
2. Biological Background	3
2.1. DNA	3
2.2. Genes	4
2.3. From Genes to Proteins	4
2.3.1. RNA	4
2.3.2. Transcription	5
2.3.3. Splicing	5
2.3.4. The Genetic Code	6
2.3.5. Translation	6
2.4. Alternative Splicing	7
3. Properties of DNA Signals and Regions	9
3.1. DNA Grammar	9
3.2. DNA Signal Properties	11
3.2.1. Translation Start Site	11
3.2.2. Translation Stop Site	12
3.2.3. Acceptor Splice Site	12
3.2.4. Donor Splice Site	13
3.3. DNA Region Properties	13
3.3.1. Coding Regions	14
3.3.2. Non-Coding Regions	15
3.3.3. Region Length Distributions	15
3.3.4. Number of Exons per Gene	16
4. Statistical Gene Finder	19
4.1. Fixed Order Markov Model	19
4.2. Hidden Markov Model	19
4.3. Generalized Hidden Markov Model	21
4.3.1. Optimal Parse Problem	22
4.3.2. Transition Probabilities	23
4.3.3. Length Probability Distributions	23

Contents

4.3.4. Signal Sensors	23
4.3.5. Content Sensors	24
4.3.6. Sensor Scoring	24
4.4. Interpolated Markov Model	25
4.5. Maximal Dependence Decomposition	27
4.6. Weight Array Model	29
4.7. Training of the GHMM	29
4.8. Implementation of the GHMM	31
5. Fourier Analysis Gene Finder	35
5.1. DNA Fourier Analysis	35
5.2. Coding Measures	36
5.3. Optimal Discrimination between Coding and Non-Coding Regions . .	38
5.4. Application	41
6. Results	43
6.1. Fourier Analysis Gene Finder	43
6.2. Statistical Gene Finder	45
6.3. Measure of Prediction Accuracy	49
7. Conclusion	53
A. Matlab Program Description	55
B. Matlab Source Code Listings	63
C. Storage and Visualization of Prediction Results	67
C.1. General Feature Format Specification	67
C.2. gff2ps Program	68
Project Process	71
Bibliography	73

Abstract

We investigate the problem of computational gene prediction in eukaryotic DNA. The presence of different statistical features in and around DNA regions that contain genes allow gene prediction using methods known from digital signal processing. We focus on two different methods – one employing the discrete Fourier transform (DFT) and the other employing a generalized hidden Markov model (GHMM).

The DFT is used to reveal the periodicity of three which is present in the essential subregions of a gene. We introduce a novel method that allows to predict the position of genes in an optimal way (in the sense of minimal error probability) based on the complex DFT values at the frequency $1/3$.

The application of a GHMM for computation gene prediction is a common solution to this problem. We focus on the implementation of the model and its decoding algorithm. Furthermore we show how the model parameters are derived from training data.

The methods are used in combination. First of all the DFT based method allows a fast search for possible genes in unknown DNA sequences. Second of all the precise position and structure of a located gene is predicted using the method based on the GHMM.

Problem

Identification of protein coding genes in DNA sequences by signal processing techniques

A DNA sequence for a living organism can be separated into two types of regions: genes and intergenic spaces. Genes contain the information for generation of proteins. Next, a gene can be divided into two types of regions: exons and introns. The introns are spliced out, and the remaining exons are decoded (3 and 3) to amino acids. An amino acid sequence defines a protein.

The problem of identifying exons is not trivial. But, some previous work including Fourier analysis [1] show good results for a set of DNA sequences from different species. Digital filtering [2] is another method.

In this project we will focus on exon identification, both in known and unknown DNA sequences, where digital signal processing techniques are essential.

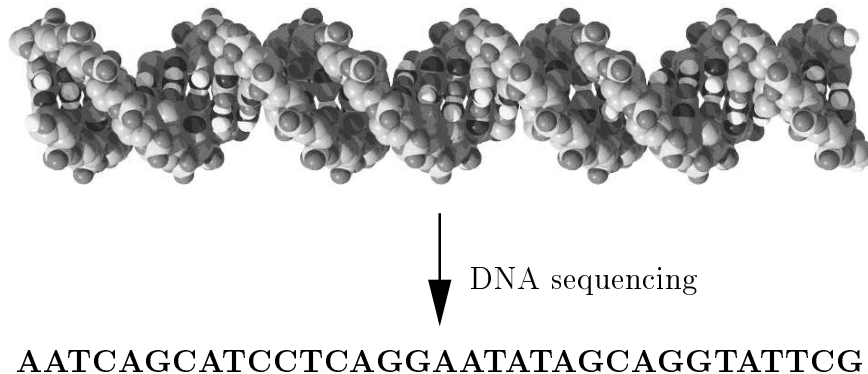
The work with this project will include literature studies, mathematical calculations, Matlab programming. Java/ C++ programming is an option.

[1] S. Tiwari et al, "Prediction of probable genes by Fourier analysis of genomic sequences", CABIOS, volume 13, no. 3, pp.263-270, 1997.

[2] P. P. Vaidyanathan, "Genomics and Proteinomics: A Signal Processor's Tour", IEEE Circuits and Systems magazine, volume 4, no. 4, pp.6-28, 2004.

1. Introduction

Today an immense amount of DNA sequence data is available thanks to numerous genome sequencing projects. The sequencing of DNA molecules, i.e. the determination of the precise sequence of nucleotides, which are the building blocks of the DNA molecule, has become a fast and automated process in the past years. The resulting DNA sequence data is publicly available from various online databases like the GenBank¹. The annotation of the DNA sequence data by biological means can by far not keep pace with the speed with which the data is accumulated. Therefore, computational methods are needed. So the challenge to organize, classify and parse the available data is still an unsolved problem and has triggered a new field of research called computational biology.



The focus of this thesis lies on the prediction of genes in genomic DNA sequences by computational methods. Therefore, we apply two well known signal processing techniques – Fourier analysis and statistical modeling – to the problem of gene prediction.

In molecular biology, a gene is considered to be the region of DNA that determines the structure of a protein, together with the region of DNA that controls when and where the protein will be produced. Through the proteins they encode, genes govern the cells in which they reside. In multicellular organisms they control the development of the individual from the fertilized egg and the day-to-day functions of the cells that make up tissues and organs. The roles of their protein products range from mechanical support of the cell structure to the transportation and manufacture of other molecules to the regulation of other proteins' activities. Therefore, the identification of all the genes is of critical importance when investigating the functioning of an organism.

¹NIH genetic sequence database, <http://www.ncbi.nlm.nih.gov/Genbank/index.html>

Chapter Overview

A certain knowledge of cell molecular biology is essential for the understanding of this thesis. In Chapter 2 we introduce the biological terms and describe the biological processes used in the further chapters.

In Chapter 3 we point out different statistical and structural properties present in eukaryotic DNA – especially in our model organism *Arabidopsis thaliana*. These properties build the basis of our gene prediction attempts.

The present statistical and structural properties are modeled in a statistical model called a generalized hidden Markov model. The mathematical background of this statistical gene finder and its implementation is described in Chapter 4.

The second approach to gene prediction based on Fourier analysis is described in Chapter 5. After a general description of the application of Fourier analysis on DNA sequences we present a novel approach to the gene prediction problem.

The results of both gene finders are presented in Chapter 6 by means of a typical usage example. The prediction quality of the statistical gene finder is furthermore specified in a more general way. Finally, the results are discussed in Chapter 7.

2. Biological Background

This chapter is thought as a brief introduction into the molecular biology of the cell. Refer to the books [Watson et al., 2004] and [Alberts et al., 1994] for further detailed information on cell biology.

2.1. DNA

The genetic information of a living organism is stored in its DNA (deoxyribonucleic acid). DNA is a macro molecule in form of a double helix. Between the two strands of the backbone there are pairs of bases. There are four bases called adenine, cytosine, guanine, and thymine abbreviated with the letters A, C, G, and T respectively. The backbone is a very regular structure made of sugar-phosphates. The molecule consisting of one base, one sugar and one phosphate is called a nucleotide. Each strand of the DNA is a chain of nucleotides. The two strands are held together by weak hydrogen bonds between the bases which sum up to a relatively strong bond between the two strands. Due to their chemical structure A only pairs with T and C only pairs with G. See Figure 2.1 for a schematic view of the DNA molecule.

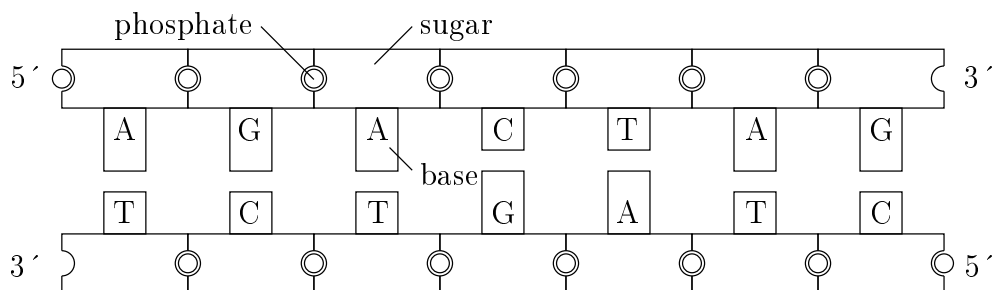


Figure 2.1.: Schematic view of a DNA molecule.

Each DNA strand has a direction which arises from the way the sugar and phosphate build up the backbone. The beginning of a strand, called the 5' end, is defined as the end where the cell machinery begins to process the strand. The ending is called the 3' end. Even if the two strands of the DNA are complementary, each of them contains genetic information independent from the other strand.

In simple organisms called prokaryotes (mostly bacteria), who have no nucleus, the DNA just resides in the cell. In higher organisms called eukaryotes (worms, insects, plants, mammals, ...) the DNA is divided among chromosomes which reside in the nucleus of the cell.

2.2. Genes

A DNA sequence can be separated into two types of regions: The genes and the intergenic spaces. Genes contain the information for the generation of proteins which are the building blocks of every organism. Each gene is responsible for the production of a different protein. Even though all cells in an organism contain identical DNA and therefore have identical genes, only a subset is expressed in any particular family of cells.

In eukaryotes the genes consist of two subregions called exons and introns. Prokaryotes do not have introns. The exons contain the information for the generation of proteins whereas the introns do not. The junctions between exons and introns are called splice sites. See Figure 2.2 for a schematic view.

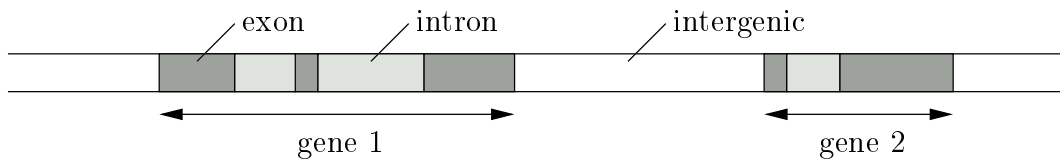


Figure 2.2.: Eukaryotic genes in DNA.

2.3. From Genes to Proteins

The gene is first copied into a single chain of nucleotides called the preliminary messenger RNA (ribonucleic acid), short pre-mRNA. This first step is called transcription. The introns are then removed from the pre-mRNA by a process called splicing to form the mature messenger RNA, short mRNA. In a final step the mRNA is translated to a chain of amino acids according to the genetic code, which will then be folded into a protein.

Figure 2.3 shows a schematic of the whole process and the details of each step are given in the following sections.

2.3.1. RNA

The RNA molecule is closely related to the DNA molecule but consists only of one nucleotide strand. It is also made of four bases but instead of thymine the base uracil (abbreviated as U) is used, which is able to pair with A just like T does¹. The sugar-phosphate backbone is also slightly different. RNA molecules are typically short and short-lived single stranded molecules which are used by the cell as temporary copies of portions of DNA.

¹We will always write T to represent U or T.

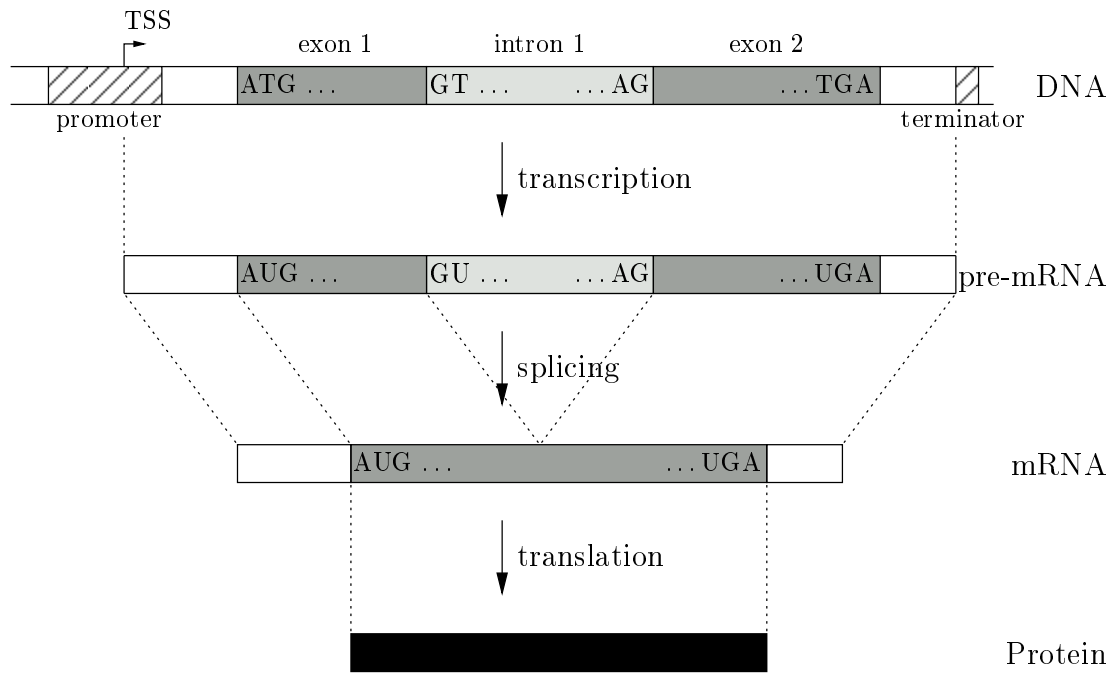


Figure 2.3.: Central dogma of molecular biology.

2.3.2. Transcription

During transcription an enzyme called RNA polymerase synthesizes the pre-mRNA according to one of the two DNA strands that serves as template. For the RNA polymerase to be able to attach to the DNA, certain nucleotide sequences called promoters have to be present around the actual transcription start site (TSS). In eukaryotes there are additional regulatory sequences present around the promoters which influence the start of transcription. The sequence of nucleotides that form a promoter or an additional regulatory sequence depend on the organism and can even vary in the DNA of one organism. At the end of the DNA portion to be transcribed a terminator sequence triggers the polymerase to detach from the DNA and to release the RNA product.

2.3.3. Splicing

During splicing a molecular machine called the spliceosome removes the introns from the pre-mRNA and connects the remaining exons together to form the mRNA. This process must occur with great precision to avoid the loss or addition of even a single nucleotide in the mRNA since this would lead to an incorrect protein (refer to Section 2.3.5). Some pre-mRNAs can be spliced in more than one way, generating alternative mRNAs and therefore different proteins. It is estimated that 60% of the genes in the human genome are spliced in alternative ways. See section 2.4 for more information.

The transition sites from exons to introns are called 5' splice sites or donor splice

2. Biological Background

sites and the transition sites from introns to exons are called 3' splice sites or acceptor splice sites. Donor splice sites have the nucleotide pair GT present at the first two positions of the intron. Acceptor splice sites have the nucleotide pair AG present at the last two positions of the intron.

2.3.4. The Genetic Code

The genetic code which is common to all organisms describes the mapping from mRNA to the corresponding sequence of amino acids. The mRNA is interpreted as a sequence of nucleotide triplets called codons. Evidently there are 64 possible codons. Each codon instructs the cell machinery to synthesize a certain amino acid with the exception of the codon ATG which in addition indicates the beginning of the protein coding part of the gene and the codons TAA, TAG, and TGA which only indicate the end of the coding part. Since there are 61 codons and only 20 different amino acids, the mapping is many-to-one according to Table 2.1.

AAA	Lys	GAA	Glu	TAA	Stop	CAA	Gln
AAG	Lys	GAG	Glu	TAG	Stop	CAG	Gln
AAT	Asn	GAT	Asp	TAT	Tyr	CAT	His
AAC	Asn	GAC	Asp	TAC	Tyr	CAC	His
AGA	Arg	GGA	Gly	TGA	Stop	CGA	Arg
AGG	Arg	GGG	Gly	TGG	Trp	CGG	Arg
AGT	Ser	GGT	Gly	TGT	Cys	CGT	Arg
AGC	Ser	GGC	Gly	TGC	Cys	CGC	Arg
ATA	Ile	GTA	Val	TTA	Leu	CTA	Leu
ATG	Met	GTG	Val	TTG	Leu	CTG	Leu
ATT	Ile	GTT	Val	TTT	Phe	CTT	Leu
ATC	Ile	GTC	Val	TTC	Phe	CTC	Leu
ACA	Thr	GCA	Ala	TCA	Ser	CCA	Pro
ACG	Thr	GCG	Ala	TCG	Ser	CCG	Pro
ACT	Thr	GCT	Ala	TCT	Ser	CCT	Pro
ACC	Thr	GCC	Ala	TCC	Ser	CCC	Pro

Table 2.1.: The genetic code (e.g. AAA=codon, Lys=amino acid Lysine).

2.3.5. Translation

The translation of the mRNA starts at the first ATG codon in the mRNA called the start codon. After the start codon each subsequent triplet of nucleotides is interpreted as a codon and translated into an amino acid according to the genetic code. The start codon does not only mark the starting point of translation, it also defines in which of the three possible ways the mRNA is divided into triplets – it

defines the so called reading frame. Translation stops when one of the three stop codons is reached.

In a cell the translation is done by a complex molecular machine called the ribosome. Together with helper molecules called transfer RNAs (short tRNA) it synthesizes a chain of amino acids for the given mRNA template. The tRNA molecules work as adapters between a specific codon and its corresponding amino acids. There is at least one tRNA molecule per amino acid present in an organism.

2.4. Alternative Splicing

Alternative Splicing [Cartegni et al., 2002] occurs in eukaryotes where the splicing process of pre-mRNA can lead to different mRNA molecules and therefore to different proteins. One extreme example is the fruit fly (*drosophila melanogaster*): The entire genome consist of only approximately 14000 genes but these genes can be processed to generate potentially 38016 different proteins. In the past decades biologists found out that the old idea of one gene coding for one protein is no longer correct.

Figure 2.4 shows the four different modes for alternative splicing. Where starting with a different initial exon is called *alternative selection of initial exon*. Ending with a different terminal exon is called the *alternative selection of terminal exon*. If an intron is retained in the mRNA transcript we call this mode the *intron retaining mode*. The intron must be properly encoding for amino acids. Certain exons are spliced out in the *exon cassette mode* to alter the sequence of amino acids in the expressed protein.

The features or motifs (splice sites) that distinguish alternatively spliced exons are in the DNA itself. The presence of a splice site is not sufficient to know that splicing will occur. The understanding of the whole mechanism is making progress but the researcher don't know enough until today.

2. Biological Background

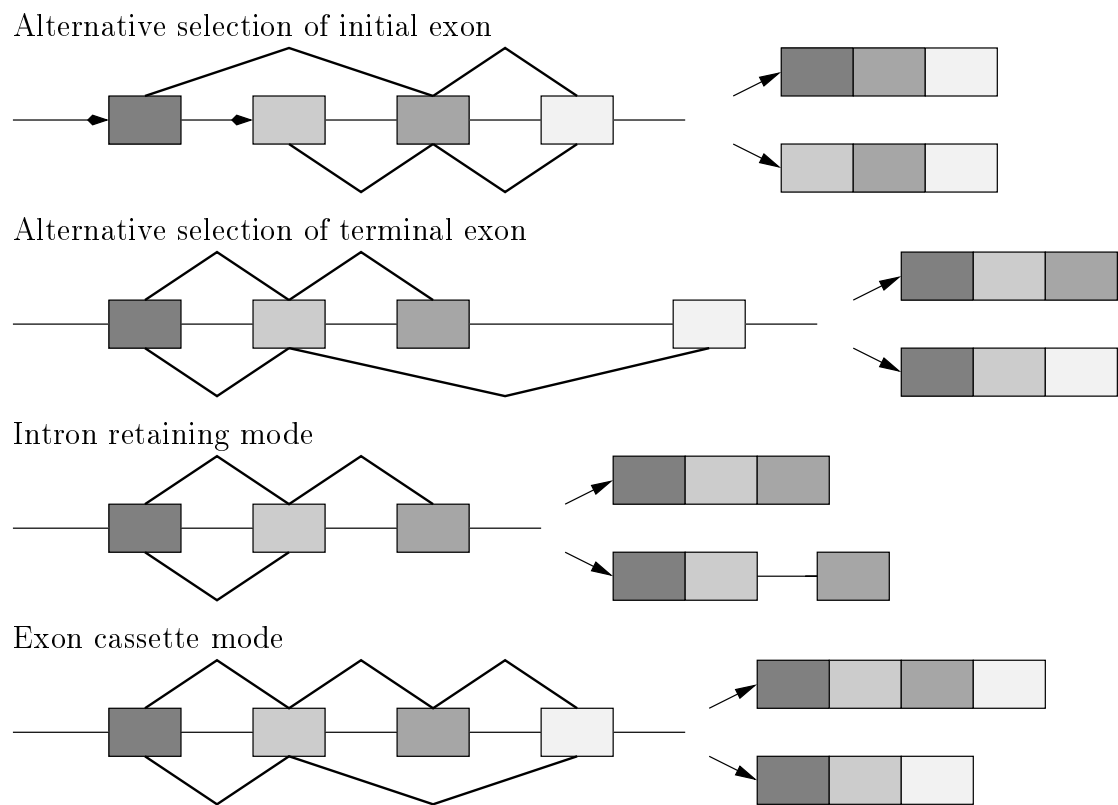


Figure 2.4.: Four different modes for alternative splicing.

3. Properties of DNA Signals and Regions

A DNA signal is a short fixed length nucleotide sequence that marks the junction of two DNA regions. The four basic DNA signals are the translation start site, the translation stop site, the donor splice site, and the acceptor splice site.

A DNA region is an arbitrary length nucleotide sequence between two DNA signals. The three basic DNA regions are the exon, the intron, and the intergenic region. The exons belong to the protein coding regions of the DNA and the introns and intergenic regions belong to the non-coding regions. The properties of the three regions are discussed in Section 3.3.

A DNA signal is a special sequence of nucleotides that triggers the cell machinery to process the following nucleotides in the desired way. In case of the four basic signals the special sequence includes nucleotides that have to be present at certain positions inside the sequence and nucleotides that are more likely than others at certain positions. The profile of a signal is obtained by aligning a large number of known signals of the same type. The profiles of the four basic signals are presented in Section 3.2.

The order of the DNA signals and regions obeys certain rules which are discussed in Section 3.1.

If not stated otherwise, all the presented results are based on the genome data of the plant *Arabidopsis thaliana* (chromosome I, accession: NC_003070.5, date: 25-JAN-2005) available from GenBank.

3.1. DNA Grammar

Given is a DNA sequence that contains genes which are separated by intergenic regions. In a simplified approach a gene begins at the translation start site marked by the start codon which is the first codon of the initial exon. In case the gene contains three or more exons the initial exon is followed by an alternating series of introns and internal exons. The last exon of the gene is called terminal exon and ends at the translation stop site marked by one of the stop codons which also indicates the end of the gene. In case the gene only contains one exon it is called a single exon gene.

The transition point from an exon to an intron is indicated by the donor (splice) site and transition point from an intron to an exon is indicated by the acceptor (splice) site. Figure 3.1 shown an example for a gene with five exons.

3. Properties of DNA Signals and Regions

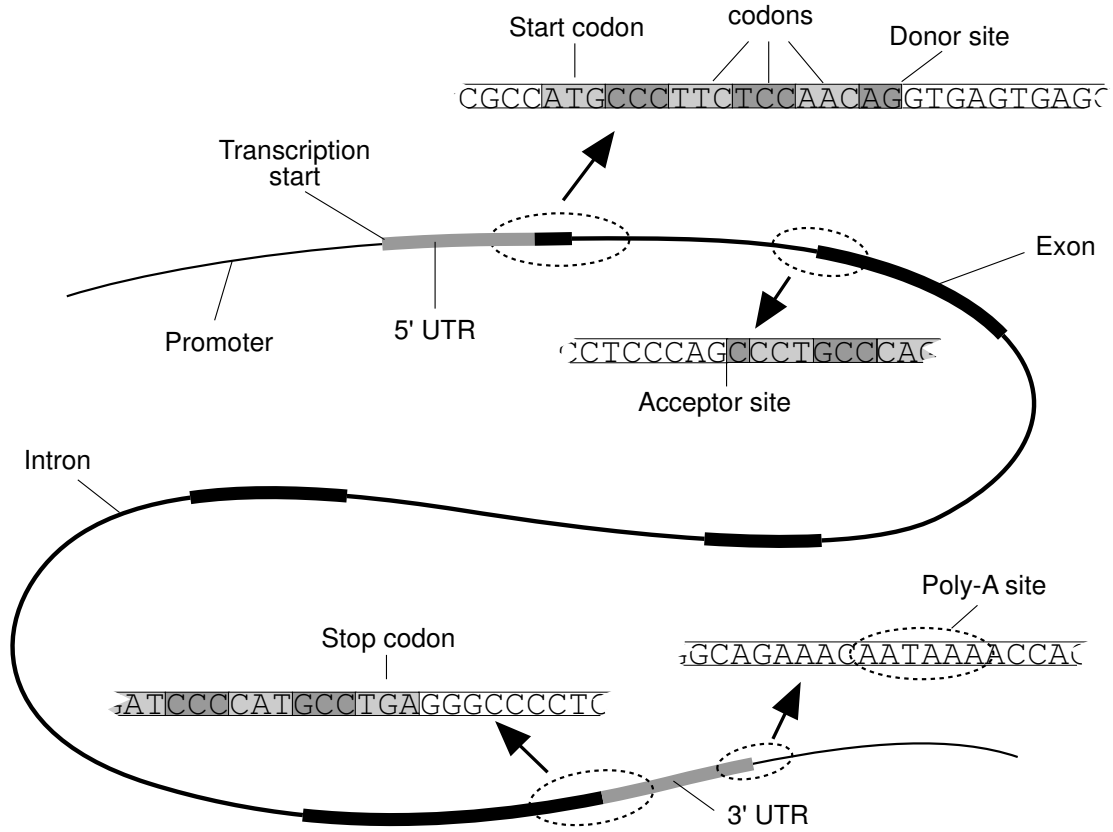


Figure 3.1.: Example gene structure (figure taken from [Salzberg et al., 1998b]).

An additional constraint not explicitly shown in the example is that the sum of the length of all exons in the gene has to be a multiple of three due to the fact that protein coding DNA is made up of codons. Codons have a length of three nucleotides. Furthermore none of the codons inside the exons must be an in phase stop codon other than the stop codon which terminates the gene. The nucleotide triplet TAA (*) is not recognized as a stop codon because it is not in phase with the codons of the exon. Start codons are allowed inside the exons and will be translated into the amino acid Methionine (Met). Figure 3.2 illustrates the mentioned constraints.

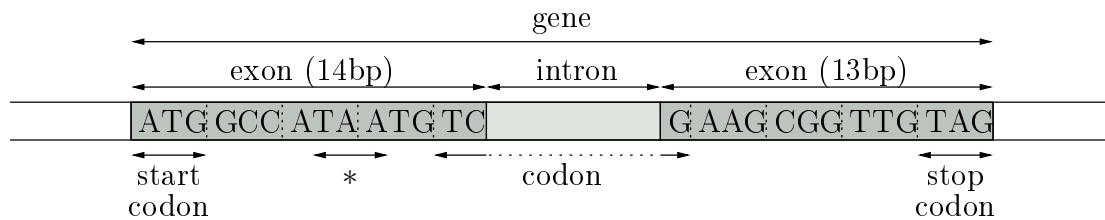


Figure 3.2.: Gene with allowed codons and correct total exon length of 27 base pairs (bp).

3.2. DNA Signal Properties

A signal profile is a table that shows the relative frequency of the nucleotides at certain positions inside a DNA signal. An example is shown in Table 3.1. Nucleotides which have to be present are printed in bold font and nucleotides which have a frequency more than 10% above their average frequency are printed in italic font. The average frequencies of A, C, G, and T in the whole training sequence are 32.08%, 17.96%, 17.91%, and 32.04%.

Signal profiles are not able to visualize statistical dependencies between different nucleotide positions inside the signal sequence. For that a χ^2 (“chi-square”) test of independence is applied as proposed in [Burge and Karlin, 1997]. The test reveals dependencies between the consensus indicator variable C_i (1 if nucleotide at position i matches the consensus at i , 0 otherwise) and the nucleotide indicator X_j identifying the nucleotide at position j . Values exceeding 16.3 (corresponding to $P < 0.001$ and 3 degrees of freedom) indicate significant dependence between the indicator variables. The dependencies with nucleotides that have to be present in the signal are not shown in the dependency matrix because they do not provide any new information. The same is true for the dependency between consensus and nucleotide indicators at the same position ($i = j$).

3.2.1. Translation Start Site

The translation start site marks the beginning of the initial exon or the single exon of a gene which begins at position 0. At the translation start site the nucleotide triplet ATG (start codon) has to be present at positions 0 to 2. In addition [Kozak, 1991] showed that the presence of certain nucleotides close to ATG increases the probability of a given ATG belonging to a real translation start site. Table 3.1 shows the signal profile of all translation start sites of *Arabidopsis thaliana* chromosome I which are recorded in the GenBank and which include the nucleotide triplet ATG at the correct position.

A	35.5	32.8	<i>44.7</i>	<i>49.5</i>	<i>43.0</i>	<i>43.7</i>	100.0	0.0	0.0	23.9	30.6	21.8
C	15.3	23.7	14.3	10.7	28.5	20.0	0.0	0.0	0.0	6.8	<i>36.7</i>	11.3
G	22.8	16.9	22.1	24.9	9.0	23.2	0.0	0.0	100.0	<i>56.1</i>	16.9	<i>35.2</i>
T	26.5	26.7	18.9	15.0	19.4	13.1	0.0	100.0	0.0	13.3	15.8	31.7
	-6	-5	-4	-3	-2	-1	0	1	2	3	4	5

Table 3.1.: Profile of translation start site.

The signal profile shows that certain nucleotides are preferred around the translation start site which allows to formulate the consensus sequence “AAAAATGGCG” for positions -4 to 5.

The dependency matrix (Table 3.2) shows the highest dependency values on the secondary diagonals of the matrix. This means that the main dependencies in the signal are among adjacent positions. Or in other words, the translation start signal features primarily first order dependencies.

3. Properties of DNA Signals and Regions

C_{-4}	–	139.7	57.9	43.6	15.1	7.2	17.2
C_{-3}	274.2	–	191.2	59.7	39.4	8.3	11.1
C_{-2}	78.3	272.3	–	272.8	26.0	42.2	16.6
C_{-1}	32.3	84.9	74.5	–	51.7	61.0	39.4
C_3	45.5	51.4	22.4	67.5	–	155.0	74.3
C_4	21.6	22.1	66.4	92.9	328.2	–	59.8
C_5	33.0	11.0	4.4	62.1	29.1	124.0	–
	X_{-4}	X_{-3}	X_{-2}	X_{-1}	X_3	X_4	X_5

Table 3.2.: Dependency matrix of translation start site.

3.2.2. Translation Stop Site

The translation stop signal marks the end of the terminal exon or single exon of a gene. The following intergenic region begins at position 0 of the signal. At the translation stop site one of the nucleotide triplets TAA, TAG, or TGA (stop codons) has to be present at positions -3 to -1. The signal profile (Table 3.3) does not show any nucleotide preferences besides the nucleotides that have to be present. Therefore no consensus can be formulated and the dependency analysis cannot be performed.

A	29.5	29.9	21.6	0.0	56.4	78.5	37.0	32.0	34.5	33.6	32.0	31.8
C	20.5	23.0	22.2	0.0	0.0	0.0	11.6	18.5	18.7	17.4	16.6	16.8
G	26.1	17.7	21.2	0.0	43.6	21.5	21.5	19.5	17.4	17.5	18.9	17.7
T	23.9	29.4	35.0	100.0	0.0	0.0	29.9	30.1	29.4	31.5	32.6	33.7
	-6	-5	-4	-3	-2	-1	0	1	2	3	4	5

Table 3.3.: Profile of translation stop site.

3.2.3. Acceptor Splice Site

The acceptor splice site marks the beginning of a new internal exon or terminal exon which begins at position 0. At positions -2 and -1 the nucleotide pair AG has to be present. The region before the splice site shows a strong preference for the nucleotide T. According to the signal profile (Table 3.4) the consensus “TTTGCAGGT” can be formulated for positions -7 to 1.

A	24.5	23.7	...	19.0	16.2	26.1	6.3	100.0	0.0	24.5	23.3	29.9
C	14.8	14.6	...	13.9	11.1	8.5	<i>65.2</i>	0.0	0.0	10.4	14.6	14.0
G	15.5	15.8	...	14.9	10.3	<i>37.7</i>	0.8	0.0	100.0	<i>52.7</i>	18.7	22.2
T	<i>45.2</i>	<i>45.9</i>	...	<i>52.2</i>	<i>62.4</i>	27.6	27.7	0.0	0.0	12.4	<i>43.3</i>	33.8
	-20	-19		-6	-5	-4	-3	-2	-1	0	1	2

Table 3.4.: Profile of acceptor splice site.

The dependency matrix (Table 3.5) reveals that the primary dependencies in the acceptor splice site are first order dependencies.

C_{-7}	–	1512.4	37.4	52.5	54.4	1.9	6.9
C_{-6}	93.3	–	1637.1	259.5	71.0	1.8	1.6
C_{-5}	49.8	90.2	–	3553.8	35.7	79.0	4.7
C_{-4}	11.4	55.1	3316.9	–	109.6	455.2	23.5
C_{-3}	41.4	125.4	39.8	192.8	–	345.3	77.4
C_0	40.5	5.8	69.9	537.9	365.0	–	178.4
C_1	27.6	10.4	11.2	38.2	118.1	84.8	–
	X_{-7}	X_{-6}	X_{-5}	X_{-4}	X_{-3}	X_0	X_1

Table 3.5.: Dependency matrix of acceptor splice site.

3.2.4. Donor Splice Site

The donor splice site marks the end of an initial or internal exon. The following intron begins at position 0. At the donor splice site the nucleotide pair GT has to be present at positions 0 and 1. According to the signal profile (Table 3.6) the consensus “CAGGTAAGT” can be formulated for positions -3 to 5.

A	37.0	64.0	9.6	0.0	0.0	65.6	53.6	21.1	23.2
C	32.0	10.6	3.4	0.0	0.0	5.0	13.7	9.5	15.3
G	18.4	8.1	77.0	100.0	0.0	12.1	5.6	49.2	11.1
T	12.6	17.3	10.0	0.0	100.0	17.2	27.1	20.2	50.4
	-3	-2	-1	0	1	2	3	4	5

Table 3.6.: Profile of donor splice site.

The dependency matrix (Table 3.7) reveals strong dependencies among non-adjacent positions inside the donor splice site. To capture these higher order dependencies [Burge and Karlin, 1997] proposed a method called maximal dependency decomposition. See Section 4.5 for more information.

C_{-3}	–	391.1	94.9	84.4	68.5	198.8	19.2
C_{-2}	1550.6	–	808.9	577.0	480.8	1079.6	141.3
C_{-1}	105.0	1110.9	–	623.5	911.5	2027.2	350.8
C_2	65.3	365.4	256.3	–	91.0	535.3	309.0
C_3	92.0	489.1	936.9	1018.9	–	213.8	22.1
C_4	300.5	1109.7	2066.3	532.6	430.2	–	634.5
C_5	40.6	135.6	352.9	300.6	12.5	569.7	–
	X_{-3}	X_{-2}	X_{-1}	X_2	X_3	X_4	X_5

Table 3.7.: Dependency matrix of donor splice site.

3.3. DNA Region Properties

The most interesting DNA region is the exon or coding region because it features short-range correlations in the nucleotide arrangement which are not present in the two noncoding regions.

A	29.03	31.42	24.85
C	18.96	22.91	19.75
G	31.92	17.62	22.64
T	20.10	28.06	32.77
	1	2	3

Table 3.9.: Probability of nucleotides at the three triplet positions (in percent, *Arabidopsis thaliana*).

The uneven usage of the nucleotides at the three triplet positions is the ultimate cause for the three periodicity present in coding regions or exons. The three periodicity can be revealed using autocorrelation or Fourier analysis. This will be shown in Section 5.1.

3.3.2. Non-Coding Regions

Non-coding regions, i.e. introns and intergenic regions, do not feature any known short-range correlations or other significant statistical properties except the general preference for nucleotides A and T as shown in Table 3.10.

A	32.57
C	17.54
G	17.00
T	32.89

Table 3.10.: Nucleotide probabilities in non-coding regions (in percent).

3.3.3. Region Length Distributions

The length distributions of the DNA regions are captured in histograms. The coding regions single exon, initial exon, internal exon and terminal exon are treated separately due to their significant difference in length distribution. The sizes of the histogram bins are chosen according to the number of available samples for a certain region. A low number of samples requires a larger bin size to make sure that the distribution represents the length distribution of the region in general and not only for the given training data. The 1% of the samples which are the longest are unaccounted for the histograms and it is assumed that all exons longer than the cutoff length have the same probability as the bin that contains the cutoff length. To make sure that no bin has zero probability, every empty bin was set to one before calculating the probabilities for every bin. See Figure 3.4 for the length distributions of the six DNA regions.

3. Properties of DNA Signals and Regions

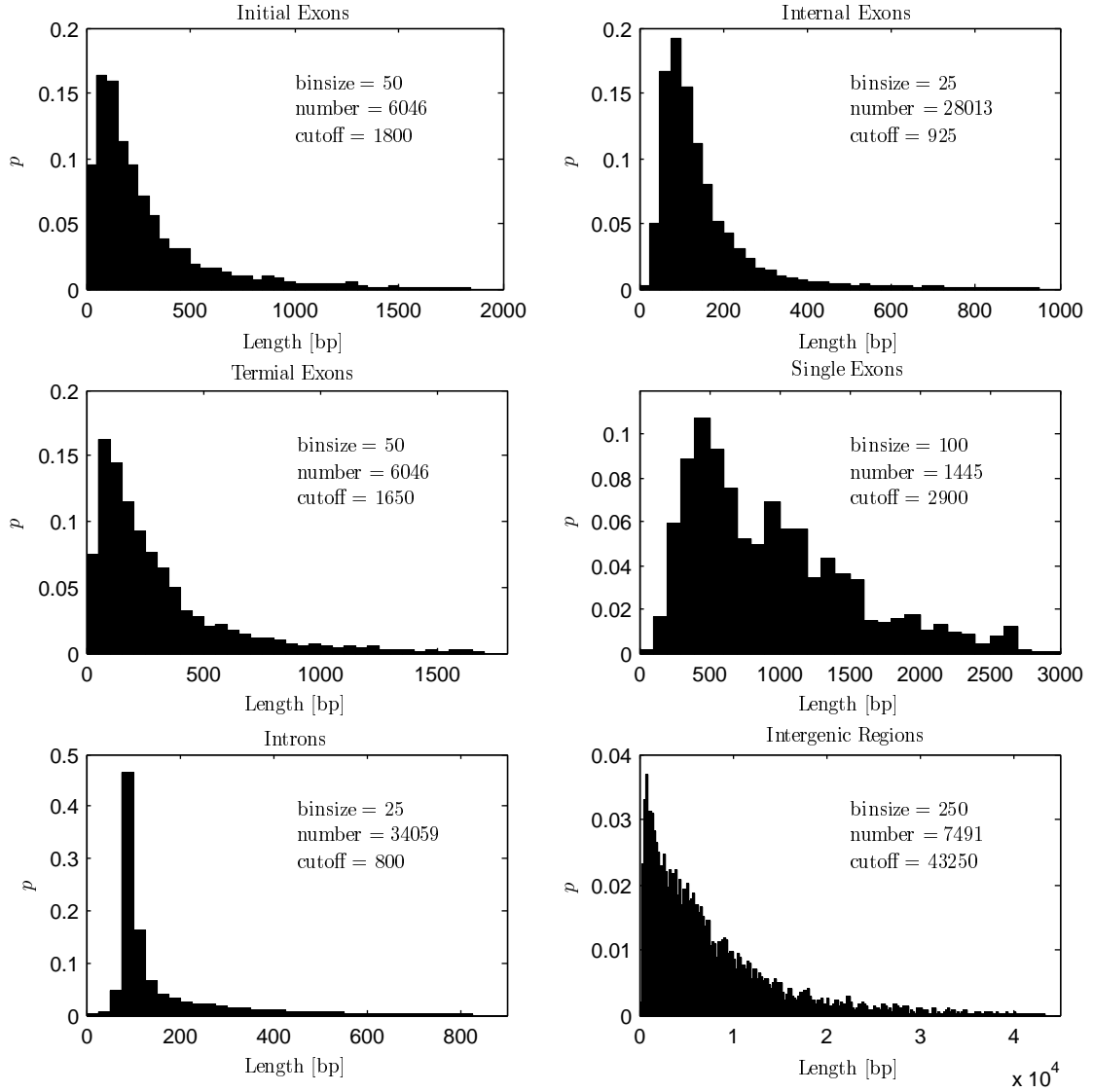


Figure 3.4.: Length distributions of DNA regions (measured in base pairs (bp)).

3.3.4. Number of Exons per Gene

The distribution of the number of exons per gene is captured in a histogram and approximated by the two probabilities P_{s1} which is the probability that the gene ends after one exon (single exon gene) and P_{sn} which is the probability that the gene ends after two or more exons. The probability that a gene has a single exon can then be written as

$$P(1) = P_{s1} \quad (3.1)$$

and the probability that a gene has n exons can be written as

$$P(n) = (1 - P_{s1}) (1 - P_{sn})^{n-2} P_{sn} \quad (n > 1) \quad (3.2)$$

The values of P_{s1} and P_{sn} are calculated using a minimum mean-square error algorithm on the histogram captured from training data. See Figure 3.5 for the histogram and the approximation.

$$P_{s1} = 0.1874$$

$$P_{sn} = 0.1750$$

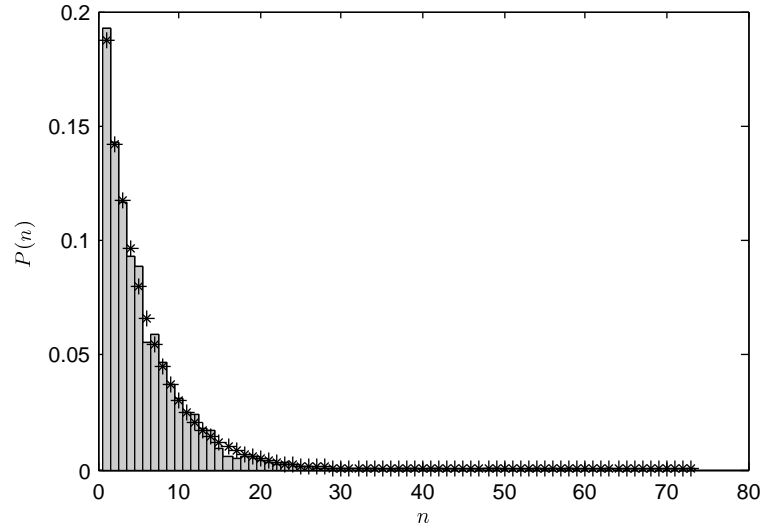


Figure 3.5.: Exons per gene histogram and approximation (*).

4. Statistical Gene Finder

The DNA properties (see Section 3.2) and the grammar of a gene to predict exons in one gene (see Section 3.1) are used to assemble a generalized hidden Markov model (GHMM) [Majoros et al., 2005] which fits best to the signal and region properties of the real genes. A GHMM is based on a hidden Markov model (HMM). A fundamental introduction on the HMM is given in [Rabiner, 1989].

The idea of all Markov models is to use a finite number of calculated values from the past to estimate the current value. Since all of the models used in a GHMM are derivations of a fixed order Markov model (or a Markov process or Markov chain), we explain the fixed order Markov model first. Second we give an introduction to the HMM as a base to understand the GHMM.

4.1. Fixed Order Markov Model

A fixed order Markov model uses the k previous bases to predict the base at the current position x (see Figure 4.1). To learn such a model in an accurate way, it needs enough training data to estimate the probability of each base occurring after every possible combination of k preceding bases. Since a DNA base can have four different states (A, C, G and T), a k^{th} -order Markov model for DNA sequences requires 4^{k+1} probabilities to be estimated from the training data. E.g. for a 5^{th} -order Markov model 4096 probabilities have to be found. Many occurrences of all possible k mers¹ must be present in the data in order to give a valid estimate of these probabilities. This becomes a problem in high-order ($k \geq 5$) Markov models.

4.2. Hidden Markov Model

Figure 4.2 shows an example of a hidden Markov model for donor splice site recognition. Even though we are not using an HMM for this purpose in our work (see Table 4.2), this example gives a good illustration of the use of HMM in general. A given DNA sequence is assumed to start with an exon, to contain one donor splice site, and to end up in an intron. The problem is now to identify the true donor splice site, since Section 3.2 shows that the sequence of exons, splice sites and introns have different statistical properties.

Starting with these information, an HMM is drawn in Figure 4.2. The HMM contains four states, one for each label assigned to a nucleotide: E (exon), G and T

¹The k previous bases (*Oligomers*).

4. Statistical Gene Finder

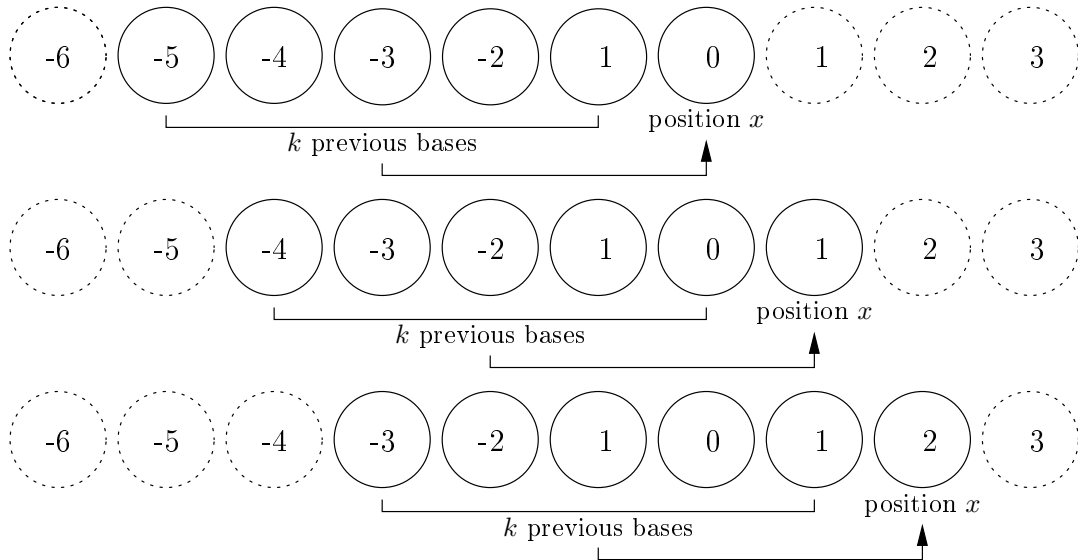


Figure 4.1.: A fixed 5th-order Markov model.

(donor splice site), and I (intron). Above each state the emission probabilities of the symbols for the state are given. In addition every state have transition probabilities (arrows). These are the probabilities of moving from the current state to a new state. The transition probabilities describe the linear order, which one expect the states to occur: One or more exon nucleotides, one donor splice site and one or more intron nucleotides.

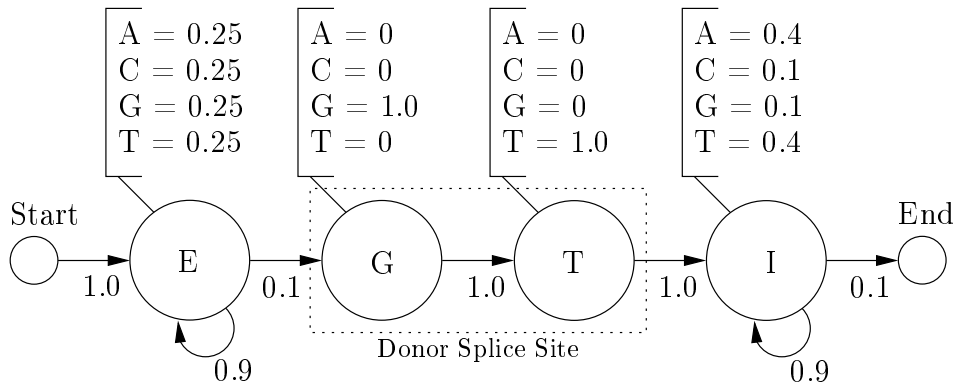


Figure 4.2.: An example of an HMM for donor splice site recognition.

Looking at the HMM as generator, each state emits a symbol (according to the state's emission probabilities) by visiting the state. The model generates two strings of information. One is the underlying state path (all labels: E, G, T and I), as moving from state to state. The other is the observed sequence (the DNA sequence) as the output from the emission probabilities in each state. The state path is a Markov chain. Since only the observed sequence is given, the underlying state path is hidden (the hidden Markov model).

Sequence:	T	G	T	A	A	G	A	C	G	T	A	A	G	T	C	A
State path:	E	E	E	E	E	E	E	E	G	T	I	I	I	I	I	I
Parse I :	●	◇	◇	○	○	○	○	○	○	○	○	○	○	○	○	○
Parse II :	●	●	●	●	●	●	●	●	◇	◇	○	○	○	○	○	○
Parse III:	●	●	●	●	●	●	●	●	●	●	●	●	◇	◇	○	○

Table 4.1.: Outcome of the example HMM.

Table 4.1 shows three possible outcomes (Parse I-III) of the HMM in Figure 4.2. It can be shown that parse II has the highest probability of these three examples. A parse ϕ is the mapping of the DNA sequence S to a possible state sequence of the HMM. The filled dots (●) stands for a possible exon, the diamonds (◇) for the donor splice site, and the empty dots (○) for a possible intron. Every parse has the probability $P(S, \phi | HMM, \theta)$ that an *HMM* with parameters θ generates a state path ϕ (parse) and an observed sequence S . The probability $P(S, \phi | HMM, \theta)$ is the product of all the emission probabilities and transition probabilities that were used. The challenge is now to find the most probable parse $\phi_{optimal}$ given the sequence S and the parses ϕ .

$$\phi_{optimal} = \underset{\phi}{\operatorname{argmax}} P(\phi | S) \quad (4.1)$$

This optimal parsing problem is solved with a dynamic programming algorithm called the Viterbi algorithm [Viterbi, 1967].

4.3. Generalized Hidden Markov Model

To reflect the biological grammar or syntax of genes (see Section 3.1) a model which follows the biological grammar is recommended. A generalized hidden Markov model (GHMM) [Majoros et al., 2005] is able to fit the biological grammar.

GHMM (or *semi-Markov* model) states are optional sub-models emitting variable length sequences rather than single letters, as in a standard HMM. Since the states are able to emit variable length symbols, a modified version of the Viterbi algorithm has to be used in order to solve the optimal parse problem.

In the GHMM, viewed as a generator, each state can emit one or more symbols entering a state. The output of the symbols depends on the probability of the path from state to state.

Figure 4.3 shows a graph representing the flow of the GHMM reflecting the biological syntax of a gene. Starting point in this GHMM is the intergenic region state. One path through the model according to the biological syntax also ends in the intergenic state.

Every state is a sensor, detecting the DNA properties according to this state. The signal states (◇) represents the fixed length DNA signals. The region states (○)

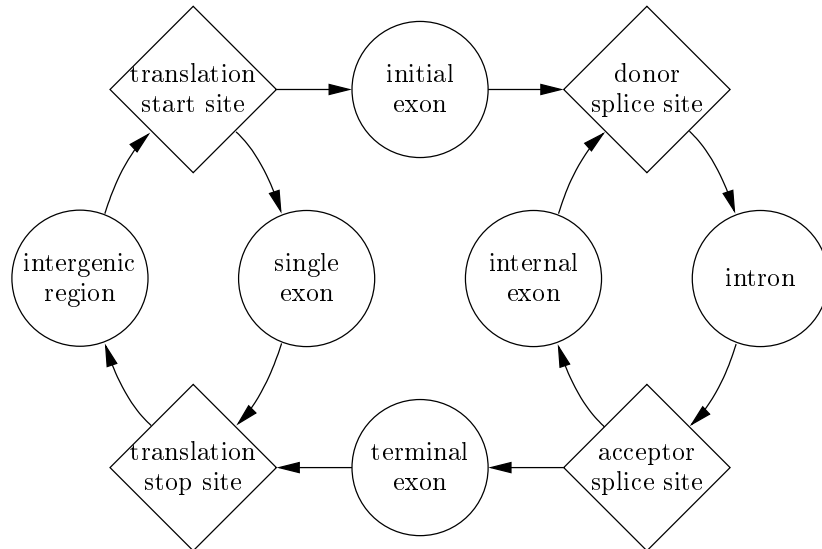


Figure 4.3.: A simple GHMM that models the eukaryotic gene structure.

represent the arbitrary length DNA regions whereas the initial-, internal-, terminal- and single-exons are separated due to their different statistical properties.

4.3.1. Optimal Parse Problem

Since the states of the GHMM are able to emit variable length symbols, the different lengths (see Section 3.3.3) need to be included in the optimal parse problem of the GHMM. The optimal parse problem (see Equation 4.1) comes to the following problem in the case of the GHMM:

$$\begin{aligned}
 \phi_{optimal} &= P(\phi|S) \\
 &= P(S|\phi)P(\phi) \\
 &= \underset{\phi}{\operatorname{argmax}} \prod_{i=1}^n P_e(S_i|q_i, d_i)P_t(q_i|q_{i-1})P_d(d_i|q_i)
 \end{aligned} \tag{4.2}$$

where a parse ϕ is the series of states q_i and state length d_i .

$P_e(S_i|q_i, d_i)$ is the probability that state q_i emits the sequence S_i , given the length d_i . The length d_i is the indicator for the variable length of the DNA regions (see Section 3.3.3). $P_t(q_i|q_{i-1})$ is the probability that the GHMM transitions from state q_{i-1} to state q_i . The probability that state q_i has the length d_i is $P_d(d_i|q_i)$. Note that the states q_0 and q_n will not produce any output. These states are *silent states*. The problem is now to find the parse ϕ out of all possible parses, which maximizes the product of equation 4.2. This problem is solved with a modified version of the Viterbi algorithm [Majoros et al., 2005].

4.3.2. Transition Probabilities

Since the probabilities from Section 3.3.4 (P_{s1} and P_{sn}) describe the numbers of exons per gene, we can use them in our GHMM as transition probabilities.

P_{s1} is the probability of the occurrence of a single exon in the gene. Thus, $1 - P_{s1}$ will be the probability that two or more exons follows the translation start site. Figure 4.4 shows the two translation probabilities after the translation start site.

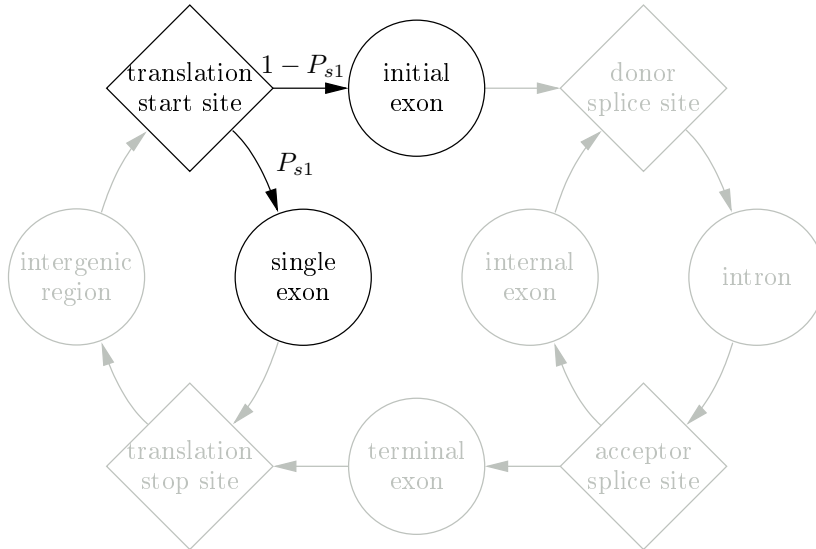


Figure 4.4.: Translation probabilities from the translation start site state to the initial exon state and the single exon state.

Figure 4.5 shows the two probabilities after the acceptor splice site. P_{sn} is the probability that a gene ends after two or more exons. Consequently, $1 - P_{sn}$ is the probability for a new internal exon.

All other transitions have the probability 1.0. Because the state where the transitions comes from has only one transition leaving the state.

4.3.3. Length Probability Distributions

Section 3.3.3 shows that every region state (o) has a different length distribution. $P_d(d_i|q_i)$ reflects these different length distributions in the optimal parsing problem (see Equation 4.2). Where d_i is the duration, or length, of state q_i .

4.3.4. Signal Sensors

Each signal sensor represents a DNA signal property as described in Section 3.2. The signal properties are fixed-length features of the gene grammar. Every biological property is represented with a specific arithmetical model containing the features of this signal (see Table 4.2).

4. Statistical Gene Finder

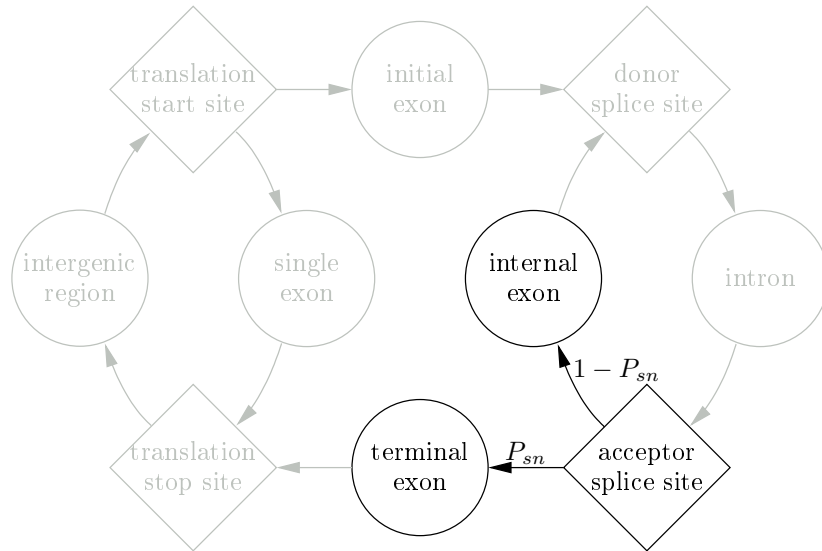


Figure 4.5.: Translation probabilities from the acceptor splice to the internal exon state and the terminal exon state.

Signal State	Model
Translation Start Site	Weight Array Model
Donor Splice Site	Maximal Dependence Decomposition
Acceptor Splice Site	Weight Array Model
Translation Stop Site	Weight Array Model

Table 4.2.: Mapping the biological signal property to an arithmetical model.

Other models to describe signal states are window weight array models (WWAM [Burge and Karlin, 1997]), hidden Markov models, decision tree methods [Salzberg et al., 1996], and multilayer neural networks [Reese et al., 1997].

4.3.5. Content Sensors

Content sensors are used to describe the regions between each signal state. Even for this sensors a model is presented which reflects the features of the region in an optimal way. In our work, the models used for the content sensors are shown in Table 4.3

4.3.6. Sensor Scoring

Every nucleotide in the DNA sequence belongs to a group of nucleotides, which is scored with a single sensor from the GHMM (see Figure 4.6). The sensors do not overlap each other and there is no gap of one or more nucleotides between two sensors.

Content State	Model
Intergenic Region	Fixed Order Markov Model
Initial Exon	Interpolated Markov Model
Single Exon	Interpolated Markov Model
Internal Exon	Interpolated Markov Model
Final Exon	Interpolated Markov Model
Intron	Fixed Order Markov Model

Table 4.3.: Mapping the biological region property to an arithmetical model.

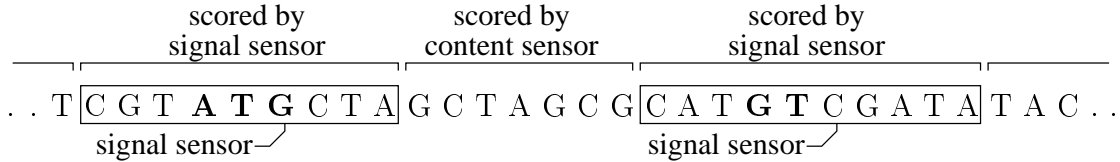


Figure 4.6.: Sensor scoring.

4.4. Interpolated Markov Model

[Salzberg et al., 1998a] describe the interpolated Markov model (IMM) we use. The IMM is able to make a better prediction for the current position than a fixed order Markov model.

An interpolated Markov model overcomes the problem of the fixed order Markov model (the frequency of occurrences of all possible oligomers) by combining probabilities from previous bases (context) of varying lengths to make predictions (see Figure 4.7), and by using only those contexts for which sufficient data is available. Longer oligomers (the k previous bases) are preferred, but only if sufficient data is available to produce good probability estimates. The IMM uses a linear combination of probabilities obtained from several lengths of oligomers to make predictions, giving high weights to oligomers that occur frequently and low weights to those that do not.

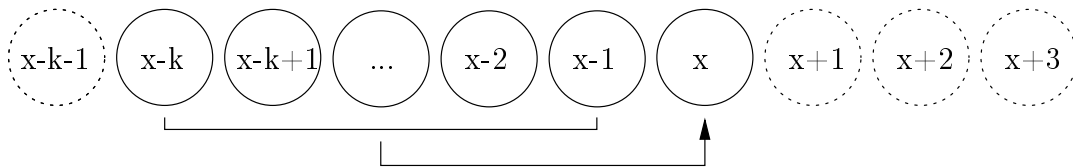


Figure 4.7.: Idea of the interpolated Markov model.

Mathematical Notation

Every content state using an IMM emits a sequence with the probability that model M generates the sequence S . $P(S|M)$, is computed as

$$P(S|M) = \sum_{x=1}^n IMM_k(S_x), \quad (4.3)$$

4. Statistical Gene Finder

where S_x is the oligomer ending at position x , and n is the length of the sequence. The score for the IMM is computed as

$$IMM_k(S_x) = \lambda_k * P_k(S_x) + [1 - \lambda_k] * IMM_{k-1}(S_x), \quad (4.4)$$

where λ_k is the numeric weight related to IMM with the order k . $P_k(S_x)$ is the estimated probability of the base located at position x in the k^{th} -order model. The estimate is obtained from the training data.

In case of this thesis a 8th-order IMM is used. Thus, the IMM score of an oligomer is a linear combination of the predictions made by the 8th and lesser-order models all the way down to the 0th-order model, which is just the simple probabilities of the occurrence of A, C, G and T, respectively.

Setting IMM parameters

First a set of known coding sequences is assembled into a training set. From the training set of genes, the frequencies of occurrence of all possible substring patterns of length 1 (IMM with 0-order) to $k + 1$ (IMM with k -order) are saved in an array.

e.g. Exon Training Set

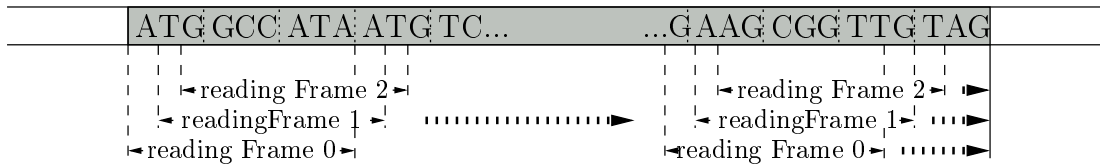


Figure 4.8.: The three reading frames in the training set for the IMM with order 8 ($k=8$).

For each pattern in all of the three reading frames (see Figure 4.8) of the training set an index I is calculated to save the frequency of occurrence of this pattern in an optimal way. The three reading frames are useful since an intron can split up an exon codon in three different positions.

The index I is computed as

$$I = \sum_{i=0}^k \nu * 4^i \quad (4.5)$$

where ν is a value representing one of the four nucleotides (see Table 4.4) and k is the order of the IMM.

From the frequency of each pattern one gets initial estimates of the probability of base s_x occurring given the context string $s_{x-k}, s_{x-k+1}, \dots, s_{x-1}$ denoted by $S_{x,k}$ (i.e., the k bases previous to position x). The probability of base s_x given the k previous bases is computed as

$$P_k(S_x) = P(s_x | S_{x,k}) = \frac{(s_{x-k} s_{x-k+1} \dots s_{x-1} s_x)}{\sum_{b \in (A,C,G,T)} (s_{x-k} s_{x-k+1} \dots s_{x-1} b)} \quad (4.6)$$

Base	Value, ν
A	1
C	2
G	3
T	4

Table 4.4.: Values for the Bases A, C, G and T.

The value of λ that is associated with $P_k(S_x)$ can be regarded as a measure of confidence in the accuracy of this value as an estimate of the true probability. Two criteria used to determine λ . If the number of occurrence exceeds a specific threshold, then λ is set to 1.0. The default value for this threshold is 400 according to the evaluation in [Salzberg et al., 1998a].

When there are insufficiently occurrences of a context string to estimate the probability of the next base with confidence, an additional criterion to assign the λ value is applied.

For a given context string $S_{x,k}$ the observed frequencies of the following base with the previously calculated IMM_{k-1} probabilities using the next shorter context are compared. With the χ^2 "Chi-Square" goodness of fit test (described in [Lipschutz, 1998]) and three degrees of freedom, one determine how likely it is that the four observed frequencies are consistent with the IMM values from the next shorter context

$$p = \frac{(S_{x,k}X - \sum_{b \in (A,C,G,T)} S_{x,k}b * IMM_{k-1})^2}{\sum_{b \in (A,C,G,T)} S_{x,k}b * IMM_{k-1}} \quad (4.7)$$

where X is one of the four nucleotides (A, C, G and T).

Now one calculate the χ^2 confidence c that the frequencies are not consistent with the IMM probabilities

$$c = 1 - \chi^2(p) \quad (4.8)$$

and set

$$\lambda_k = \begin{cases} 0.0 & \text{if } c < 0.50 \\ \frac{c}{400} \sum_{b \in (A,C,G,T)} (s_{x-k} s_{x-k+1} \dots s_{x-1} b) & \text{if } c \geq 0.50 \end{cases} \quad (4.9)$$

If the frequencies differ significantly from the IMM values, the frequencies get a higher λ_k value. In the case the frequencies are consistent with the IMM values, the frequencies get a lower λ_k value.

4.5. Maximal Dependence Decomposition

The maximal dependence decomposition (MDD) model was introduced by [Burge and Karlin, 1997] to capture dependencies between positions (both adjacent and non-adjacent) in the donor splice site signal (see Section 3.2.4). The biological

4. Statistical Gene Finder

explanation of these dependencies as well as a detailed description of the MDD model can be found in the cited article.

The central tool in the MDD is the statistical test called χ^2 *test of independence* which is used to reveal the position dependencies. Because the application details of this statistical test are left out in the cited article we will provide the necessary details on the following lines. DNA signals in general and the donor splice site signal in special are described in Section 3.2.

χ^2 Test of Independence

The test is used to reveal the dependencies between the consensus indicator variable C_i (1 if nucleotide at position i matches the consensus at i , 0 otherwise) and the nucleotide indicator X_j identifying the nucleotide at position j in the signal. The consensus indicator C_i can take the values 0 or 1 and the nucleotide indicator X_j can take the values A, C, G or T which gives eight possible indicator combinations.

In the following example 2500 known signals are analyzed. For every position pair (i, j) the number of occurrence of the eight indicator combinations is counted. For one specific position pair (i, j) this results in the following table of observed frequencies.

C_i	X_j				Total
	A	C	G	T	
0	430	136	108	161	835
1	1095	173	112	285	1665
Total	1525	309	220	446	2500

Next, the expected frequencies of the eight indicator combinations are calculated assuming independence among them.

$$\text{expected frequency} = \frac{(\text{row total})(\text{column total})}{\text{grand total}}$$

Using the totals from the table above this results in the following table of expected frequencies.

C_i	X_j			
	A	C	G	T
0	509.4	103.2	73.5	149.0
1	1015.7	205.8	146.5	297.0

Finally the χ^2 value as the following sum over all eight entries of the tables.

$$\begin{aligned} \chi^2 &= \sum_{C_i, X_j} \frac{(\text{observed} - \text{expected})^2}{\text{expected}} \\ &= 60.0 \end{aligned}$$

This calculation has to be done for every position pair (i, j) to get the complete dependency matrix for the signal.

4.6. Weight Array Model

The n -th order weight array model (WAM) [Zhang and Marr, 1993] is a DNA signal model capable to capture dependencies between positions with a distance up to n nucleotides. For every position i in the signal the conditional probability $P_i(x_i|x_{i-n}, \dots, x_{i-1})$ ($x_i = \{A, C, G, T\}$) of a nucleotide given its n preceding nucleotides is recorded. The exceptions are the first n positions in the signal where the conditional probabilities can only be recorded starting from the first position in the signal.

In case of a 1st order WAM the model would consist of 4 unconditional probabilities $P_1(x_1)$ for the first signal position and 16 1st order conditional probabilities $P_i(x_i|x_{i-1})$ ($i > 1$) for every following position as shown in Figure 4.9.

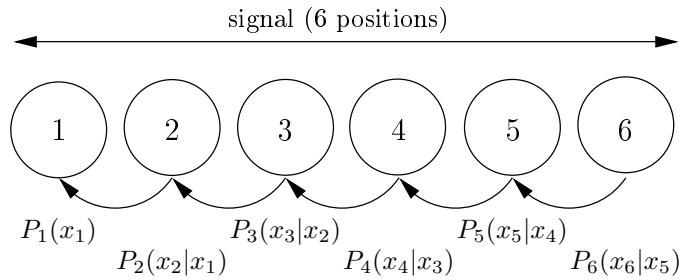


Figure 4.9.: Example weight array model.

4.7. Training of the GHMM

The genetic diversity of all species makes it hard to train a model covering every case. Different organisms have different gene structures. To train a model one need to set clear measurement methods to improve the model with the training. And one need to think how to assemble a proper training set from a database of well known genes (see Section 6.3).

The training of the GHMM is divided into two sections. Training each sensor model and training the global transition probabilities of the GHMM.

Sensor Model Training

Training the sensor models of the GHMM (see Figure 4.3) hardly depends on the accuracy of the training set. Because the probabilities for each model is calculated out from the training set. The training set to train the sensor models is assembled with well known genes from the GenBank using one chromosome from the organism we want to analyze (In our case: chromosome I from the *Arabidopsis thaliana*).

To train the signal models (\diamond), we align the genes to the required signal and count the occurrence of each nucleotide before and after the signal. The range of the nucleotides before and after the signal is a finite number. Section 3.2 shows the number of the additional bases to train each signal sensor. Now, we calculate

4. Statistical Gene Finder

the probabilities for each signal out of the properties of the nucleotides around each signal.

For the training of the region models (○) we assemble each region from the gene training set together. In the end we have three different training sequences. One to train the single-, initial-, internal- and the terminal-exon states. One training sequence for the intron state containing all the introns from the gene training set. And one training set to train the intergenic region sensor. To calculate the probability for each region sensor we count all the patterns, which occur in the training sequence according to the models described in Table 4.3.

Global Training

After the separate training of each sensor model, the GHMM is trained on the training set starting with initial transition probabilities (see Section 3.3.4). In the training process every probability is called a parameter of the model. According to our model (see Section 4.3.2) we have only two global transition probabilities. Every other probability can be calculated from these two or has the probability 1.0 (see Table 4.5).

Transition/Parameter	Probability
start - single exon	P_{s1}
start - initial exon	$1 - P_{s1}$
acceptor - terminal exon	P_{sn}
acceptor - internal exon	$1 - P_{sn}$
others	1.0

Table 4.5.: Transition Probabilities in the GHMM.

For the training of the global parameters we use a complete analyzing GHMM and apply a gradient ascent method to find the best transition probabilities from one sensor to an other (adopted from [Majoros and Salzberg, 2004]). The training set is divided into several chunks. Now the GHMM is used in a gradient ascent method to find the best parameters for this chunk. Once the parameters for each chunk is calculated, the average of parameters over all chunks are used for the GHMM to analyze an unknown DNA sequence. Figure 4.10 shows the process of the global parameter training.

Gradient Ascent Method

All parameters from the GHMM builds a n -dimensional parameter space Θ_n where n is the number of parameters. The gradient ascent algorithm choose one after the other parameter θ from Θ_n and calculate a new value for the parameter. The new parameter value is used in the GHMM to evaluate a better prediction accuracy. Listing B.1 shows the implemented algorithm.

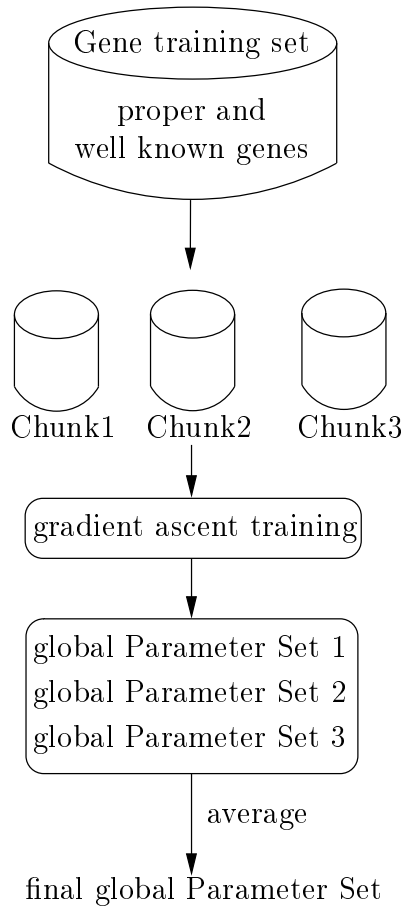


Figure 4.10.: Process flow of the global training.

4.8. Implementation of the GHMM

The implementation of the GHMM decoding algorithm follows closely the implementation of the *PSA decoding algorithm* as it is described in [Majoros et al., 2005]. It is recommended to read this article before continuing with reading this section as this section only provides additional detail information and information specific to our implementation.

Exon Prefix Sum Arrays

The IMM that models exons is a 3-periodic IMM in order to capture the 3-periodic nature of coding regions. Therefore three prefix sum arrays (PSA) have to be initialized, each assuming that the 3-periodic codons are in a different phase relative to the start of the analyzed DNA sequence. Figure 4.11 shows the assumed codon position relative to the start of the sequence for the three PSA phases γ .

The Exon Phase Challenge

During the trace forward part of the decoding algorithm the final structure of the gene that is to be predicted is unknown. This implies that it is unknown at which

4. Statistical Gene Finder

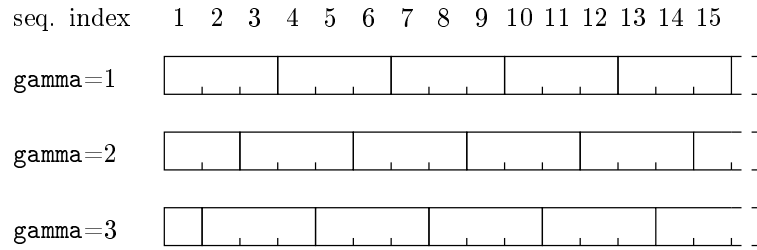


Figure 4.11.: Assumed codon positions for the three phases γ .

triplet position (see Figure 3.3) a possible donor splice site terminates its preceding exon. This furthermore implies that it is unknown which exon PSA to use to score the preceding exon. The only solution is to keep track of the three possible exon phases ω for the donor and the acceptor splice sites. The translation start and stop sites are defined to be in phase $\omega=1$ because they always start or stop an exon at triplet position 1.

Figure 4.12 shows an example of the score calculation of a donor site for a possible preceding acceptor site. Given the donor site is in phase ω then its score would be the sum of the inductive score of the preceding acceptor site in phase $\omega_{\text{pre}} = \text{mod}((\omega-1)-L, 3)+1$; and the score from exon PSA in phase $\gamma = \text{mod}((\omega-1)-(\text{pos}-1), 3)+1$; (pos is the one based position of the donor site in the DNA sequence). The transition-, length- and signal scores that have to be added to the new donor site score too are omitted in this example to keep it as simple as possible.

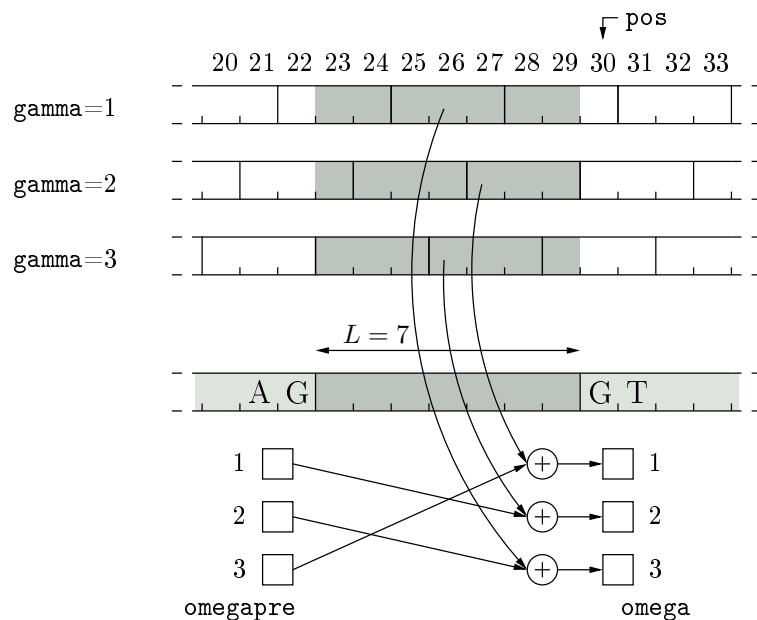


Figure 4.12.: Example of the score calculation of a donor site.

Start- and Stop States

The current implementation assumes that the analyzed sequence begins and ends in intergenic regions. That is the decoding algorithm inserts a pseudo translation stop site at the first position of the sequence and a pseudo translation start site at the last position of the sequence. The length of the first and last intergenic region content state are not scored because their real length are not known.

5. Fourier Analysis Gene Finder

Based on the periodicity with period 3 present in coding regions of DNA (refer to Section 3.3), different methods to discriminate between coding and non-coding regions using statistical methods [Staden and McLachlan, 1982], autocorrelation [Fickett, 1982], and finally Fourier analysis [Tiwari et al., 1997] were developed.

We will focus on the Fourier analysis and present a new method to discriminate between coding and non-coding regions using Fourier analysis.

5.1. DNA Fourier Analysis

The discrete Fourier transform (DFT) $X[k]$ of a given numeric sequence $x[n]$ of length N is defined by

$$X[k] = \sum_{n=0}^{N-1} x[n] e^{-j\frac{2\pi}{N}nk} \quad 0 \leq k \leq N-1 \quad (5.1)$$

where n is the sequence index and k corresponds to the discrete frequency of k/N . Because a DNA sequence D is a string of the four characters A, C, G and T which represent the four nucleotides, numerical values have to be assigned to each character. The common way is to assign a binary indicator sequence to each of the four bases [Voss, 1992]. The binary indicator sequence $x_b[n]$ of base b ($b = \{A, C, G, T\}$) will take a value of 1 or 0 at position n depending on the existence of base b at position n in the DNA sequence.

D	G	G	A	T	A	T	C	A	C	T	T	T
$x_A[n]$	0	0	1	0	1	0	0	1	0	0	0	0
$x_C[n]$	0	0	0	0	0	0	1	0	1	0	0	0
$x_G[n]$	1	1	0	0	0	0	0	0	0	0	0	0
$x_T[n]$	0	0	0	1	0	1	0	0	0	1	1	1

Table 5.1.: Example of binary indicator sequences.

The total estimated power spectrum $S[k]$ of the DNA sequence is defined as the sum of the four individual estimated power spectra.

$$S[k] = \sum_b |X_b[k]|^2 \quad f = \frac{k}{N} \quad (5.2)$$

5. Fourier Analysis Gene Finder

Figures 5.1 and 5.2 show the total power density spectrum of a coding and a non-coding region respectively. The periodicity with period 3 of a coding region can be seen as a distinct peak at frequency $f = 1/3$ in the power density spectrum.

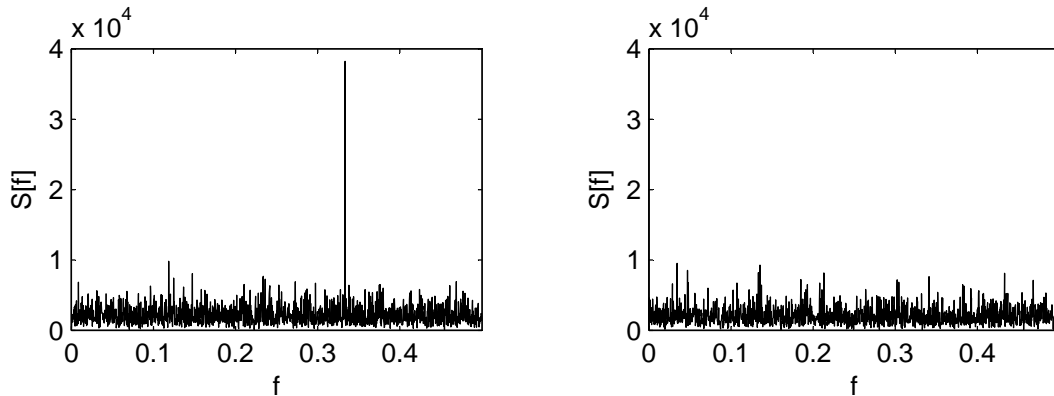


Figure 5.1.: Power density spectrum of coding section.

Figure 5.2.: Power density spectrum of non-coding section.

Since the only significant difference between the spectra of coding and non-coding regions is at $f = 1/3$ or $k = N/3$ it is sufficient to evaluate the DFT $X_b[k]$ at $k = N/3$.

$$X_b \left[\frac{N}{3} \right] = \sum_{n=0}^{N-1} x_b[n] e^{-j \frac{2\pi}{3} n} \quad (5.3)$$

Note that for N being a multiple of 3 the sum over the four values $X_b[N/3]$ will always be zero.

$$X_A \left[\frac{N}{3} \right] + X_C \left[\frac{N}{3} \right] + X_G \left[\frac{N}{3} \right] + X_T \left[\frac{N}{3} \right] = 0 \quad (5.4)$$

5.2. Coding Measures

Based on Fourier analysis different measures to discriminate between coding and non-coding regions were developed. There exist the measures know as *spectral content measure* [Tiwari et al., 1997], *optimized spectral content measure* [Anastassiou, 2000], and *spectral rotation measure* [Kotlar and Lavner, 2003].

In order to have a measure that tells us if the nucleotides at a certain position in the analyzed DNA sequence D belong to a coding region or not, a sliding analysis window of length N (with N being an odd multiple of three) is introduced. The sliding window is moved by the step size Δ and has its center at position i relative to the beginning of D . The sliding window positions are numbered using index variable $m = \{0, 1, 2, \dots\}$.

$$i = \frac{N-1}{2}, \frac{N-1}{2} + \Delta, \frac{N-1}{2} + 2\Delta, \dots, \frac{N-1}{2} + m\Delta, \dots \quad (5.5)$$

Equation (5.3) is rewritten to include the sliding window position i as follows.

$$X_{b,i} \left[\frac{N}{3} \right] = \sum_{n=0}^{N-1} x_b \left[n + i - \frac{N-1}{2} \right] e^{-j \frac{2\pi}{3} n} \quad (5.6)$$

Then the *spectral content measure* is the direct application of equation (5.2) on a sliding window of length N with its center at position i (where N must be an odd multiple of 3) which is evaluated along a given DNA sequence D for $k = N/3$.

$$M_{sc}[N, i] = \left| X_{A,i} \left[\frac{N}{3} \right] \right|^2 + \left| X_{C,i} \left[\frac{N}{3} \right] \right|^2 + \left| X_{G,i} \left[\frac{N}{3} \right] \right|^2 + \left| X_{T,i} \left[\frac{N}{3} \right] \right|^2 \quad (5.7)$$

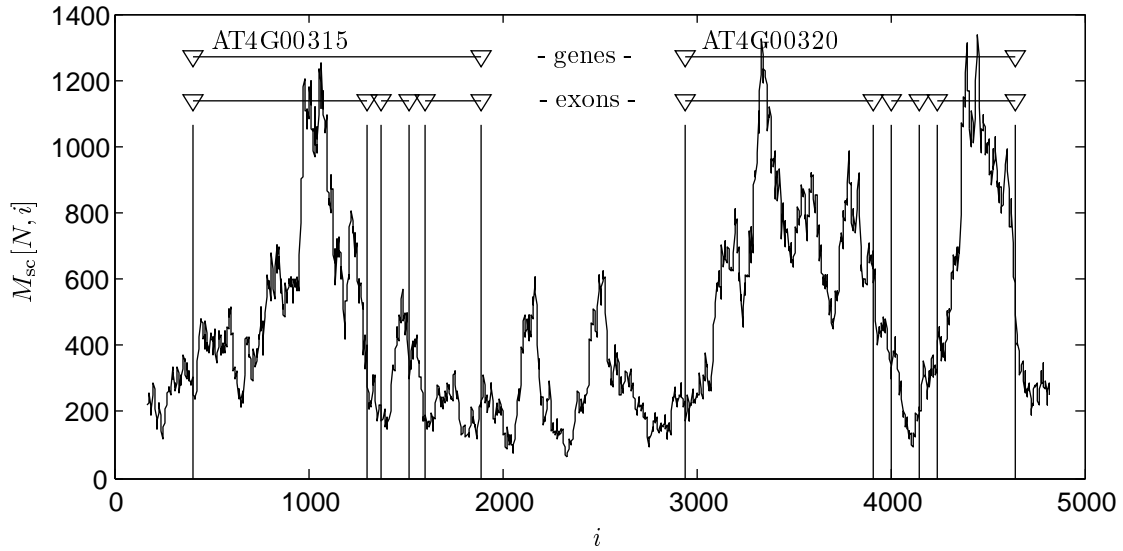


Figure 5.3.: Spectral content measure of *A. thaliana* chromosome IV; Two genes with three exons each; Positions relative to 137000; $N = 351$.

The further two measures introduce a set of parameters p_b which are calculated from known training data using different methods. Both measures are of the following form.

$$M_{osc, sr}[N, i] = \left| p_A X_{A,i} \left[\frac{N}{3} \right] + p_C X_{C,i} \left[\frac{N}{3} \right] + p_G X_{G,i} \left[\frac{N}{3} \right] + p_T X_{T,i} \left[\frac{N}{3} \right] \right|^2 \quad (5.8)$$

The decision if the DNA sequence inside the sliding window belongs to a coding or non-coding region is made by comparing the calculated measure to a threshold value.

5.3. Optimal Discrimination between Coding and Non-Coding Regions

The presented coding measures employ a rather ad-hoc way of deciding if the DNA sequence in the current analysis window belongs to a coding region or not. Together with Prof. Sven Ole Aase we found the following optimal approach.

Taking the discrete random variables $Z_b = X_b[N/3]$ ($b = \{A, C, G, T\}$) as a basis for the optimal decision according to the Bayesian decision theory [Schürmann, 1996], the current analysis window would be classified as coding when the following inequality is true.

$$P(\text{coding}) P_c(\mathbf{Z} = \mathbf{z}) > P(\text{noncoding}) P_{nc}(\mathbf{Z} = \mathbf{z}) \quad (5.9)$$

$$\begin{aligned} P_c(\mathbf{Z} = \mathbf{z}) &= P(Z_C = z_C, Z_G = z_G, Z_T = z_T | \text{coding}) \\ P_{nc}(\mathbf{Z} = \mathbf{z}) &= P(Z_C = z_C, Z_G = z_G, Z_T = z_T | \text{noncoding}) \end{aligned}$$

The probabilities $P(\text{coding})$ and $P(\text{noncoding}) = 1 - P(\text{coding})$ reflect the ratio between the total length of coding regions and non-coding regions in the genome of a given organism and are calculated from training data. Z_A is omitted from the joint probabilities because it is implicitly given by equation (5.4) when Z_C , Z_G , and Z_T are known.

The challenge of this approach is to calculate the joint probability distributions $P_c(\mathbf{Z} = \mathbf{z})$ and $P_{nc}(\mathbf{Z} = \mathbf{z})$. The first step is to check if the random variables Z_C , Z_G , and Z_T are independent because then the joint probability distribution could be calculated as follows from the individual probability distributions.

$$P(\mathbf{Z} = \mathbf{z}) = P(Z_C = z_C) P(Z_G = z_G) P(Z_T = z_T) \quad (5.10)$$

To analyze the dependencies we set $N = 3$ without loss of generality. In that case the random variables Z_b can take seven different values (shown in Figure 5.4) calculated from the eight possible binary indicator triplets $x_b = \{000, 001, 010, \dots, 111\}$.

$$\begin{aligned} Z_b &= X_b[1] \\ &= x_b[0] + x_b[1]e^{-j\frac{2\pi}{3}} + x_b[2]e^{j\frac{2\pi}{3}} \end{aligned} \quad (5.11)$$

As a representative example the dependency of the events $Z_C = 1$ ($x_C = \{100\}$) and $Z_G = e^{j\frac{2\pi}{3}}$ ($x_G = \{001\}$) is analyzed. In addition it is assumed that the 64 possible DNA triplets occur with uniform probability. The following table lists the DNA triplets which lead to the given events.

5.3. Optimal Discrimination between Coding and Non-Coding Regions

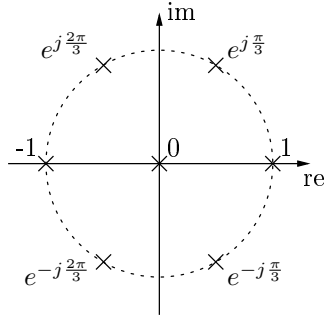


Figure 5.4.: Possible values of Z_b for $N = 3$.

$Z_C = 1$	CAA	$Z_G = e^{j\frac{2\pi}{3}}$	AAG
	CAG		ACG
	CAT		ATG
	CGA		CAG
	CGG		CCG
	CGT		CTG
	CTA		TAG
	CTG		TCG
	CTT		TTG

$$P((Z_C = 1) \cap (Z_G = e^{j\frac{2\pi}{3}})) = \frac{2}{64} \quad (5.12)$$

$$\approx 0.0313$$

$$P(Z_C = 1) P(Z_G = e^{j\frac{2\pi}{3}}) = \frac{9}{64} \frac{9}{64} \quad (5.13)$$

$$\approx 0.0198$$

The random variables Z_C and Z_G are dependent because equations (5.12) and (5.13) do not give the same result. From this it follows that the joint probability distribution $P(\mathbf{Z} = \mathbf{z})$ cannot be calculated according to equation (5.10).

Exact Solution

Since the joint probability distribution $P(\mathbf{Z} = \mathbf{z})$ is discrete it would be possible to calculate it exactly using a computer. But $P(\mathbf{Z} = \mathbf{z})$ is the distribution of three complex variables which means six real dimensions. The amount of memory needed to store it is a function of N and the size of the used data type d in byte is

$$\text{mem} = \left(\frac{N}{3} + 1\right)^3 \left(\frac{4N}{3} + 1\right)^3 d \quad (5.14)$$

For a minimal value of N which is used in practice, e.g. $N = 150$ and a 32bit data type the amount of memory needed would be approximately 4 Tbyte (2^{42} byte). This is far beyond the possibilities of modern 32bit computers.

Approximate Solution

Since it is not possible to find the exact discrete distribution $P(\mathbf{Z} = \mathbf{z})$ for practical values of N due to the given reason, the next step is to find a suitable approximation. The following approximation approach was found in [Schürmann, 1996] Chapter 8.

A large amount of training vectors $\mathbf{z}_0, \mathbf{z}_1, \mathbf{z}_2, \dots$ is calculated from known training data (either coding or non-coding). Each training vector holds the results of the Fourier analysis of one sliding window position in the training data and is of the form

$$\mathbf{z}_m = \begin{bmatrix} X_{C,i}[N/3] \\ X_{G,i}[N/3] \\ X_{T,i}[N/3] \end{bmatrix} \quad \text{with} \quad i = \frac{N-1}{2} + m \Delta \quad . \quad (5.15)$$

For the training vectors to be independent of each other the sliding windows are chosen to be non-overlapping ($\Delta = N$). These vectors are then clustered into L clusters C_l ($l = 1, \dots, L$) using the following iterative vector quantization algorithm.

The algorithm starts with one cluster containing all training vectors. A cluster is represented by its centroid μ_l which is the mean of all vectors belonging to cluster C_l and its covariance matrix Σ_l . A cluster C_l is split into two new clusters if it satisfies the splitting criterion

$$\text{VAR}[\mathbf{z}_m | C_l] = \text{trace}[\Sigma_l] > t_{\text{cluster}} \quad (5.16)$$

where t_{cluster} is the cluster threshold parameter. Cluster C_l is split by adding and subtracting a short vector \mathbf{r} to its centroid μ_l to generate the new cluster seed points \mathbf{s}_l and \mathbf{s}_h .

$$\begin{aligned} \mathbf{s}_l &= \mu_l + \mathbf{r} \\ \mathbf{s}_h &= \mu_l - \mathbf{r} \end{aligned} \quad (5.17)$$

The direction of \mathbf{r} is chosen to be the same as the direction of the eigenvector corresponding to the largest eigenvalue of Σ_l . After all clusters that satisfy the splitting criterion are split, all training vectors \mathbf{z}_m are reassigned to the nearest cluster. The distance between a training vector and a cluster is defined as the Euclidean distance between the training vector and the cluster seed point. Now the centroids and covariance matrices of all clusters are updated and the algorithm enters the next iteration. The clustering is finished when there is no cluster left that fulfills the splitting criterion. All the details of the algorithm can be found in the source code listing B.2.

Every cluster C_l is now defined by its centroid μ_l , its covariance matrix Σ_l and P_l which is equal to the number of training vectors belonging to the cluster divided by the total number of training vectors.

The statistical model that represents each cluster is defined to be a multivariate Gaussian distribution $p(\mathbf{Z} = \mathbf{z} | C_l)$ (D is the number of dimensions).

$$p(\mathbf{Z} = \mathbf{z} | C_l) = \frac{1}{(2\pi)^{D/2} |\Sigma_l|^{1/2}} e^{-\frac{1}{2}(\mathbf{z} - \mu_l)^\top \Sigma_l^{-1} (\mathbf{z} - \mu_l)} \quad (5.18)$$

The total continuous distribution $p(\mathbf{Z} = \mathbf{z})$ is the linear combination of the L cluster distributions.

$$p(\mathbf{Z} = \mathbf{z}) = \sum_{l=1}^L P_l p(\mathbf{Z} = \mathbf{z} | C_l) \quad (5.19)$$

This approximation process has to be performed twice – once on coding and once on non-coding training data – to find the approximations to $P_c(\mathbf{Z} = \mathbf{z})$ and $P_{nc}(\mathbf{Z} = \mathbf{z})$ which are needed to evaluate equation (5.9).

5.4. Application

During analysis of an unknown DNA sequence the values $X_C[N/3]$, $X_G[N/3]$ and $X_T[N/3]$ are calculated for a sliding window of length N at position i . Based on equation (5.9) and using the distributions $P_c(\mathbf{Z} = \mathbf{z})$ and $P_{nc}(\mathbf{Z} = \mathbf{z})$ calculated from training data, the coding ability of the current sliding window is then expressed using the measure

$$M_{\text{bay}}[N, i] = \frac{P_c(Z_C = X_{C,i}[\frac{N}{3}], Z_G = X_{G,i}[\frac{N}{3}], Z_T = X_{T,i}[\frac{N}{3}])}{P_{nc}(Z_C = X_{C,i}[\frac{N}{3}], Z_G = X_{G,i}[\frac{N}{3}], Z_T = X_{T,i}[\frac{N}{3}])} \quad (5.20)$$

and the decision threshold t_{bay} value is

$$t_{\text{bay}} = \frac{P(\text{noncoding})}{P(\text{coding})} \quad (5.21)$$

There is still one problem that has to be solved. During the training of the distribution $P_c(\mathbf{Z} = \mathbf{z})$ it was assured that the training sequence was a sequence of adjacent codons and that the sliding window was moved by a multiple of three (e.g. by N) as shown in Figure 5.5. Therefore the values $X_b[N/3]$ used to calculate the distribution are all in the same phase relative to each other.

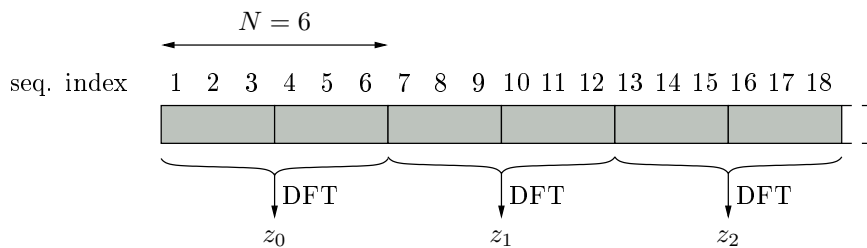


Figure 5.5.: Generation of training vectors from training sequence.

Now, when analyzing an unknown sequence the codons inside the exons can be at any position relative to the sliding window as shown in Figure 5.6. Therefore the measure M_{bay} has to be calculated three times, once for every possible position of

5. Fourier Analysis Gene Finder

the codons. Finally the measure is extended by the phase parameter $\varphi = \{0, \frac{2\pi}{3}, \frac{4\pi}{3}\}$ and is defined as

$$M_{\text{bay}}[N, i, \varphi] = \frac{P_{\text{c}}(Z_{\text{C}} = X_{\text{C},i}[\frac{N}{3}]e^{j\varphi}, Z_{\text{G}} = X_{\text{G},i}[\frac{N}{3}]e^{j\varphi}, Z_{\text{T}} = X_{\text{T},i}[\frac{N}{3}]e^{j\varphi})}{P_{\text{nc}}(Z_{\text{C}} = X_{\text{C},i}[\frac{N}{3}]e^{j\varphi}, Z_{\text{G}} = X_{\text{G},i}[\frac{N}{3}]e^{j\varphi}, Z_{\text{T}} = X_{\text{T},i}[\frac{N}{3}]e^{j\varphi})} \quad (5.22)$$

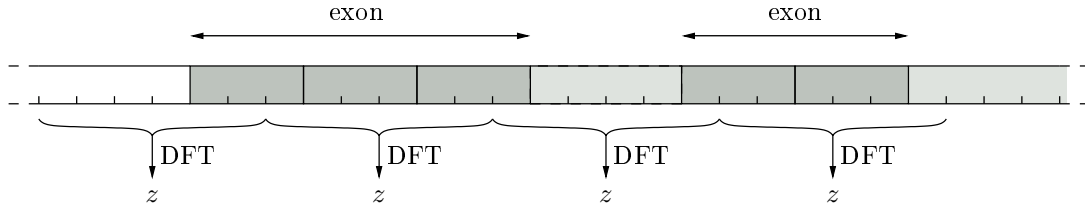


Figure 5.6.: Arbitrary position of codons in analysis.

6. Results

The Results of the Fourier analysis gene finder and the statistical gene finder are shown by means of a typical application example. The Fourier analysis gene finder is used to predict the approximate position of possible genes in a given DNA sequence. The located regions are then analyzed using the statistical gene finder to determine the exact position of the genes.

Furthermore, the performance of the statistical gene finder is examined in a more general way and expressed with common performance measures presented in Section 6.3.

6.1. Fourier Analysis Gene Finder

Training

The Fourier analysis gene finder is trained on artificial DNA sequences. The coding sequence which serves for the calculation of the coding distribution $P_c(\mathbf{Z} = \mathbf{z})$ is a weighted random sequence of codons (nucleotide triplets). The codon probabilities are taken from the codon usage table of *Arabidopsis thaliana* (Table 3.8). The non-coding sequence which serves for the calculation of the coding distribution $P_{nc}(\mathbf{Z} = \mathbf{z})$ is a weighted random sequence of nucleotides. The nucleotide probabilities are taken from Table 3.10.

The training vectors \mathbf{z}_m are calculated from 20'000 non-overlapping sliding windows of length N ($\Delta = N$) for $N=351$ – which is a common window length in DNA Fourier analysis [Tiwari et al., 1997]. The training vectors are clustered using a cluster threshold of $t_{cluster}=9.6$ for $N=351$. This results in a reasonable number of 16 clusters for each distribution approximation.

The decision threshold t_{bay} is equal to the total length of all non-coding regions divided by the total length of all coding regions of *Arabidopsis thaliana* chromosome I.

$$t_{bay} = 5.367$$

Analysis

The Fourier analysis gene finder is suitable for a coarse search of potential coding regions in long DNA sequences. As an example, the DNA in *Arabidopsis thaliana* chromosome IV from position 160'000 to position 172'000 is analyzed.

To improve the prediction quality the result plots of the Fourier analysis are complemented with the position of in phase stop codons (refer to Section 3.1). A

6. Results

high coding probability combined with the absence of in phase stop codons is a strong indicator for the existence of a real exon.

Figure 6.1 shows the developing of the coding measure $M_{\text{bay}}[N, i, \varphi]$ (solid line) along the sequence for the three phases $\varphi = \{0, \frac{2\pi}{3}, \frac{4\pi}{3}\}$ and $N=351$. The dashed horizontal line indicates the decision threshold level and the dots indicate the positions of in phase stop codons.

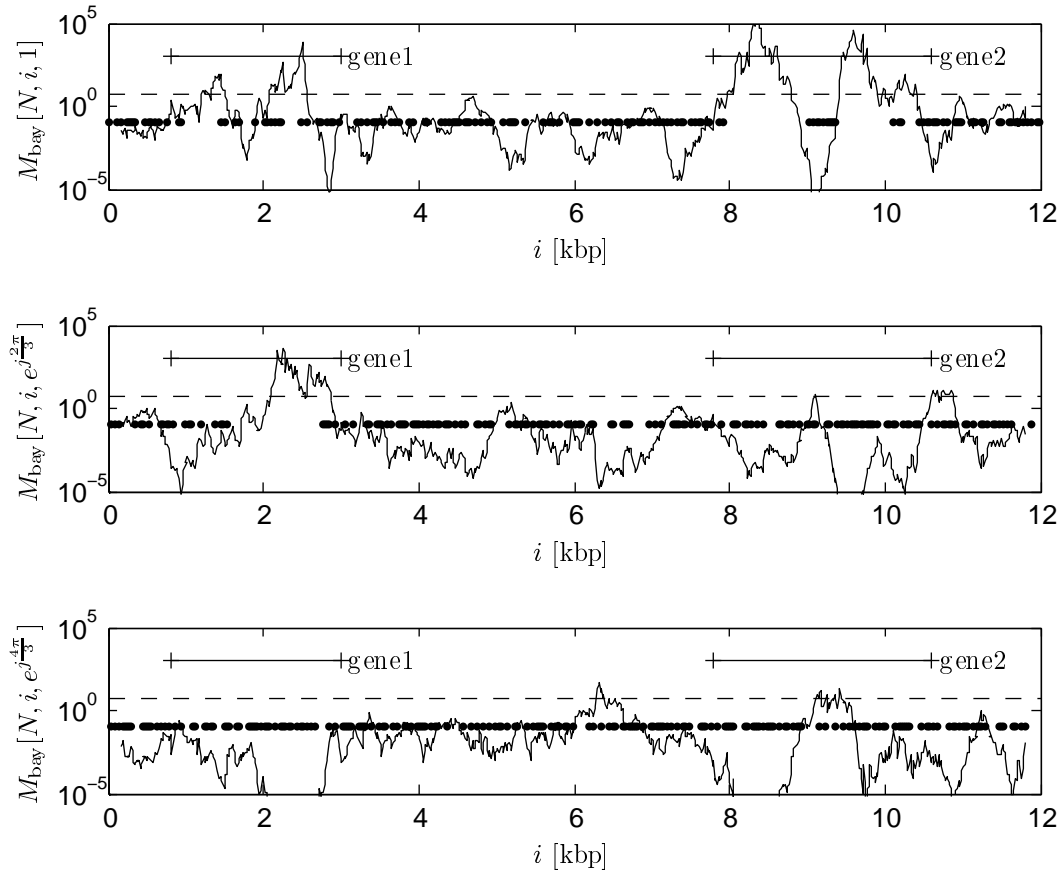


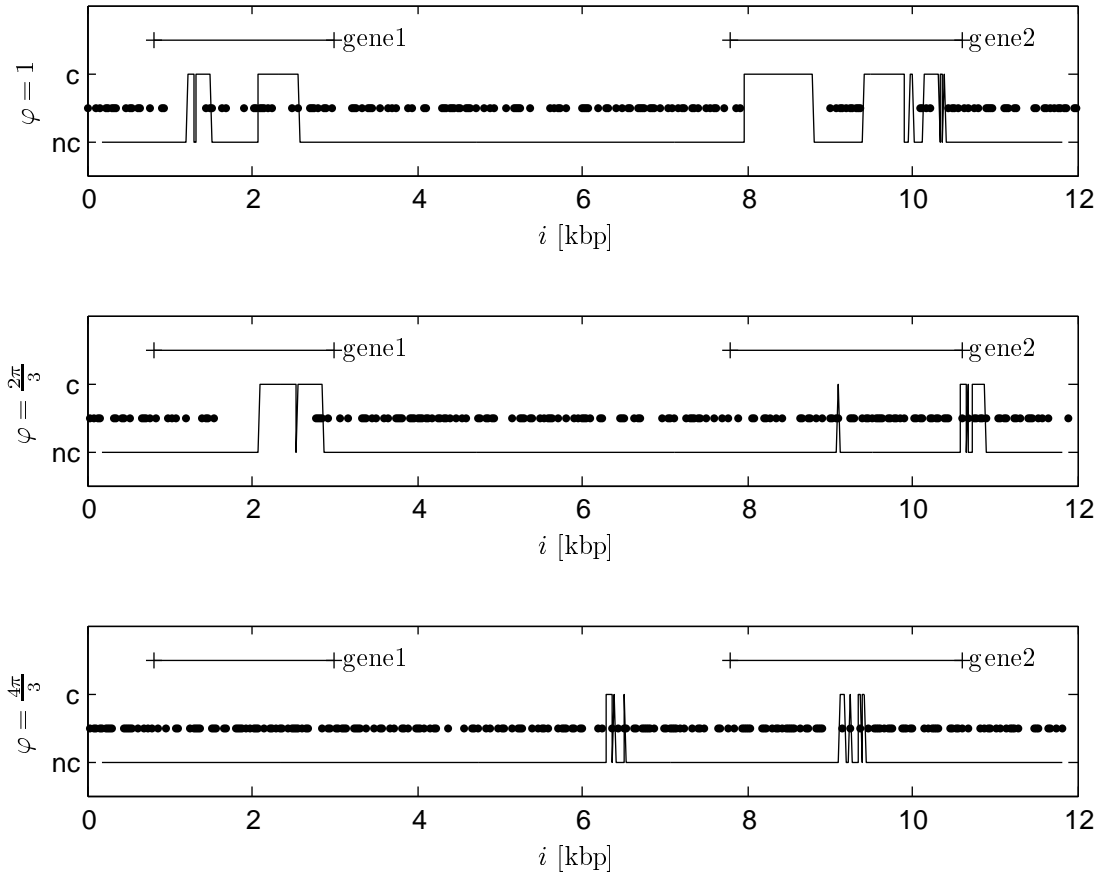
Figure 6.1.: Measure plot of Fourier analysis for $N=351$.

Figure 6.2 shows the result of the optimal Bayesian decision along the sequence. The nucleotides at position i are considered to belong to a coding region if

$$M_{\text{bay}}[N, i, \varphi] > t_{\text{bay}} \quad .$$

According to the result plots one is able to predict the approximate position of two genes marked as “gene1” and “gene2”. The high coding probability around positions 6500 in phase $4\pi/3$ and 10800 in phase $2\pi/3$ can be considered as false positives due to the high density of in phase stop codons around these positions.

To figure out the exact positions of the potential genes and their exons the two marked DNA sections are further analyzed using the developed statistical gene finder.

Figure 6.2.: Decision plot of Fourier analysis for $N=351$.

6.2. Statistical Gene Finder

We show the accuracy of the statistical gene finder on two applications for the GHMM. The first application is to use the statistical gene finder in addition to the Fourier analysis gene finder. The Fourier analysis gene finder predicts a coding region in a DNA sequence and then the statistical gene finder is used on the coding region to predicted genes.

The second application is the measurement of accuracy of the statistical gene finder. The accuracy is measured on well-known gene sequences from the GenBank. This is a performance measurement to compare the statistical gene finder with other statistical gene finders or to compare our gene finder with the GenBank.

Both of the following results are produced using the same statistical gene finder. The statistical gene finder is trained on the Chromosome I of the plant *Arabidopsis thaliana* (NC_003070, date: 25-JAN-2005). The transition probabilities are the same as described in Table 4.5 ($P_{s1} = 0.1874$ and $P_{sn} = 0.1750$).

Gene Prediction based on the Fourier Analysis

We take the start and the end position of the two marked DNA sections (“gene1” and “gene2”) from the Fourier Analysis to predict exons in the marked DNA sections (refer to Figure 6.2).

The decision plot from the Fourier analysis predicts a first potential gene (“gene1”) in the DNA section between position 160800 and the position 163000 (absolute positions in the chromosome). Those two positions define a *search window* for the statistical gene finder. The upper gene in the Figure 6.3 shows the prediction of our statistical gene finder in the potential DNA section. The lower gene is the true gene out of the GenBank as a comparison to the prediction of the statistical gene finder.

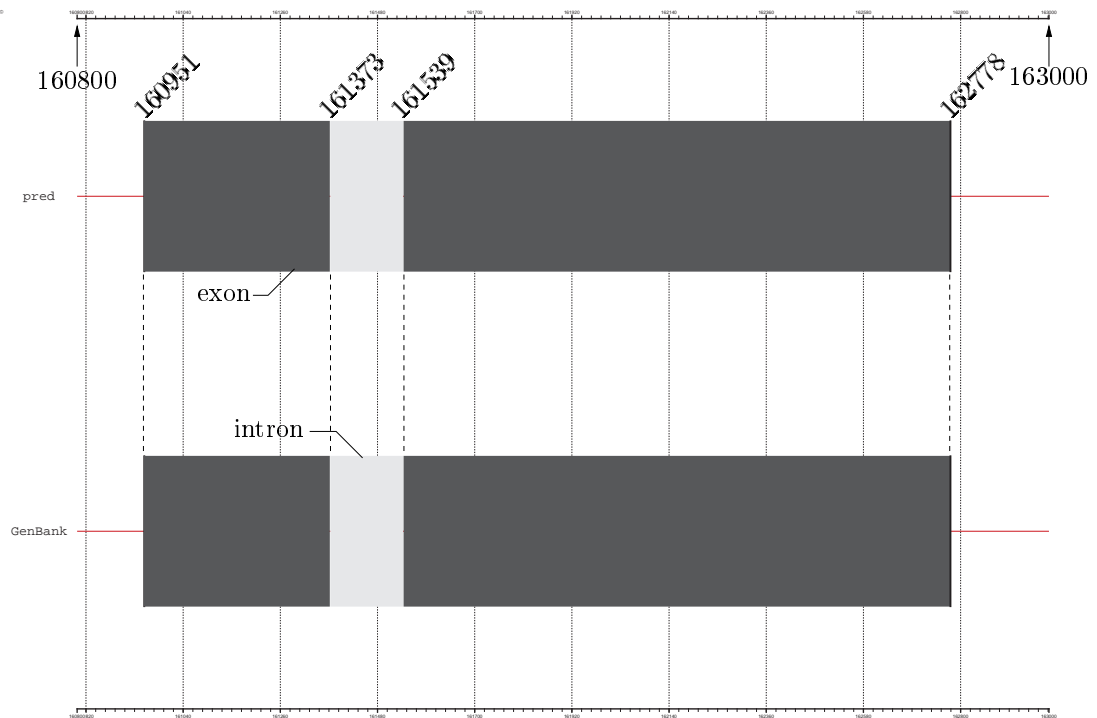


Figure 6.3.: Gene in marked the DNA section “gene1”.

The second potential gene (“gene2”) is between position 167800 and position 170600. We use the sequence between the positions as a new search window for our statistical gene finder. Figure 6.4 shows the prediction of our statistical gene finder for the region with start at position 167800 and end at position 170600. Again, the upper sequence is the predicted gene and the lower sequence is the gene from the GenBank for this region.

Figure 6.3 and Figure 6.4 show a perfect match (horizontal lines) between the prediction from the statistical gene finder and the true gene from the GenBank.

The prediction for a coding region from the Fourier analysis gene finder is an important first step for the usage of our statistical gene finder. Figure 6.5 shows the

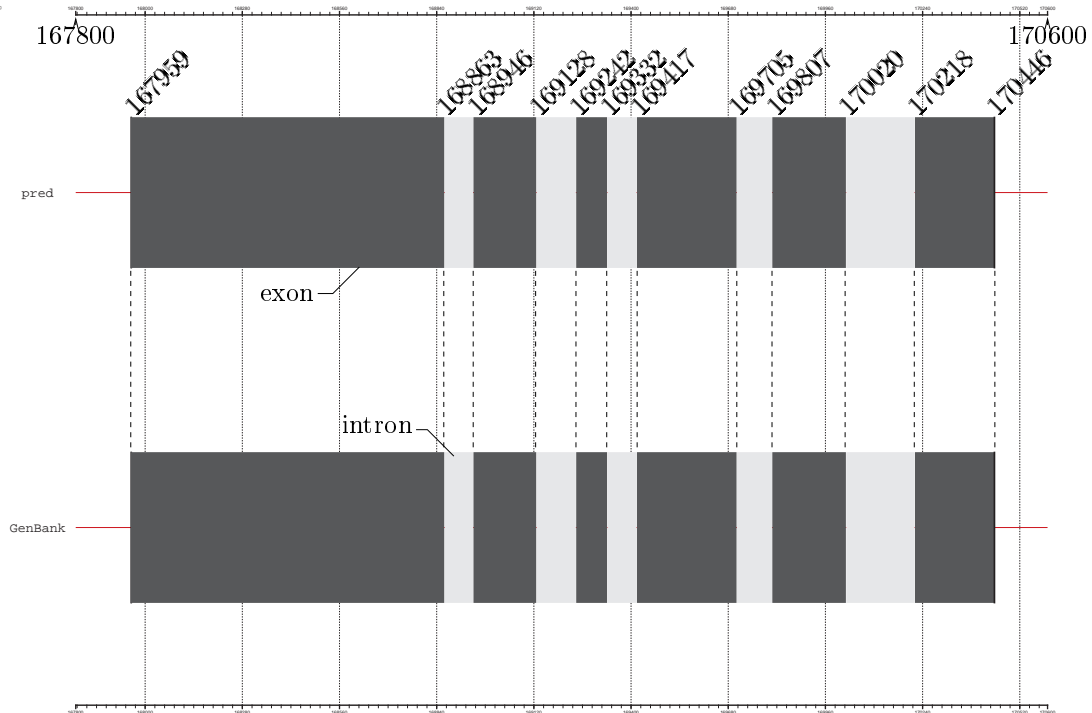


Figure 6.4.: Gene in marked DNA section “gene2”.

prediction of the “gene2” marked DNA section with 1000 bases before and after the search window. Thus the new search window start position is 166800 and the end position is 171600.

The statistical gene finder now predicts more exons as are in the GenBank for this region. Thus, Figure 6.5 shows that the prediction of our statistical gene finder hardly depends on the start and stop positions of the searching window.

Performance

We made a performance measurement of the statistical gene finder on the Chromosome IV of the *Arabidopsis thaliana* (NC_003075, date: 4-NOV-2005). The measurement methods we use for our statistical gene finder are described in Section 6.3. This performance measurement only compares our statistical gene finder with the entries in the GenBank. To compare our gene finder with other gene finders, our gene finder should be trained and tested on the same sequences as the other gene finders.

We built test sequences out of all the 4655 genes from the Chromosome. A search window was made over every known gene with 100 bases before and after the gene.

Now we are looking if there is a missing or an unknown base in the search window and discard those search windows for the prediction. In the end we had 4655 proper search windows for the statistical gene finder (all search windows were proper).

6. Results

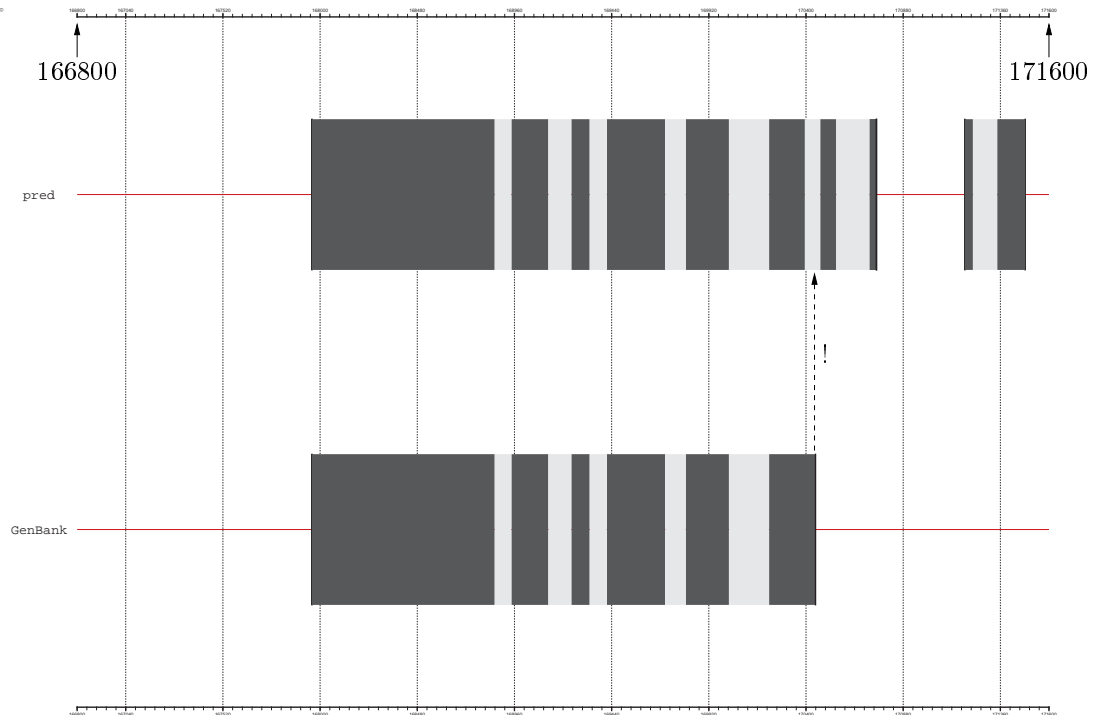


Figure 6.5.: Statistical gene finder prediction on window 166800-171600, Chromosome IV.

The statistical gene finder tries to predicted the genes in the search windows. We counted all True Positives, True Negatives, False Positives and False Negatives for the nucleotide and exon level in each search window.

Table 6.1 shows the results for the nucleotide level. The statistical gene finder has a sensitivity of 0.66 ($Sn = 0.66$) and a specificity of 0.68 ($Sp = 0.68$) on the nucleotide level. The simple matching coefficient is 0.68 ($SMC = 0.68$). This reveals that **68 percent** of the nucleotides have the same coding value in reality and prediction.

Measurement	Value
<i>Sensitivity</i>	0.66
<i>Specificity</i>	0.68
<i>SMC</i>	0.68

Table 6.1.: Measurement values for the nucleotide level.

Table 6.2 shows the result for the exon level. On this level we only calculate the simple matching coefficient, which has a value of 0.41 ($SMC = 0.41$). This testifies that **41 percent** of the exon boundaries have the same coding value in reality and

prediction.

Measurement	Value
<i>SMC</i>	0.41

Table 6.2.: Measurement value for the exon level.

6.3. Measure of Prediction Accuracy

To measure the accuracy of a gene predictor, a wide accepted method should be used to compare different gene finders. The measurement introduced by [Burset and Guigó, 1996] has become a standard for gene finders. The accuracy of the prediction can be measured on three different levels: coding nucleotide sequence, exonic structure and protein product. According to the gene finder described in this thesis, we only focus on the nucleotide and the exon level.

Nucleotide Level

At the nucleotide level the accuracy of a prediction is measured by comparing the predicted value (coding and non coding) with the true coding value for each nucleotide. Each predicted nucleotide belongs to one of the four groups.

- True Positives (TP). The number of coding nucleotides correctly predicted as coding.
- True Negatives (TN). The number of non coding nucleotides predicted as non coding.
- False Negatives (FN). The number of coding nucleotides predicted as non coding.
- False Positives (FP). The number of non coding nucleotides predicted as coding.

Figure 6.6 shows the classification on the nucleotide level.

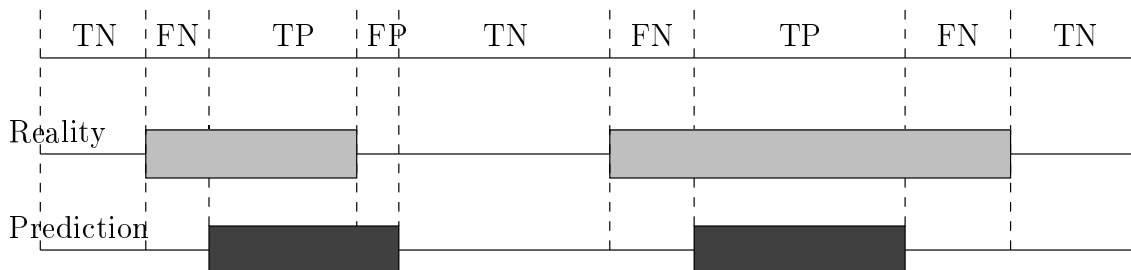


Figure 6.6.: Prediction classification on the nucleotide level.

6. Results

The four groups should be shown in context to each other. So a Sensitivity Sn and a Specificity Sp is calculated from the four groups.

$$Sn = \frac{TP}{TP + FN} \quad Sp = \frac{TN}{TN + FP} \quad (6.1)$$

Sensitivity is the proportion of coding nucleotides that have been correctly predicted as coding, and Specificity is the proportion of non coding nucleotides that have been predicted as non coding.

Since the frequency of non coding nucleotides in genomic DNA sequences is much greater than the frequencies of coding nucleotides (both in reality and in the prediction), True Negatives tends to be much larger than False Positives, and thus Specificity, as computed above, systematically produces very large non informative values. Thus the Specificity has been computed instead as

$$Sp = \frac{TP}{TP + FP} \quad (6.2)$$

that is, in the context of gene structure prediction programs, Specificity is the proportion of predicted coding nucleotides that are actually coding.

Neither Sn nor Sp alone constitute good measures of global accuracy, and it appears desirable to use single scalar value summarizing both of them as a measure of global accuracy. We use the simple matching coefficient, SMC , which is defined as

$$SMC = \frac{TP + TN}{TP + FN + FP + TN} \quad (6.3)$$

and is the probability of correct prediction (that is, of a nucleotide having the same coding value in reality and prediction).

Exon Level

There is no unique criterion to consider an exon as correctly predicted [Burset and Guigó, 1996]. Therefore, we choose our own measurement method.

Correctly predicted exon start boundaries (start codon and acceptor splice site) are True Positives (TP). Correctly predicted exon stop boundaries (donor splice site and stop codon) are True Negatives (TN). If there is a true exon start boundary but no predicted exon start boundary, we count this as a False Positive (FP). A False Negative (FN) is counted if there is a true exon stop boundary but no predicted exon stop boundary (see Figure 6.7). We only measure the simple matching coefficient SMC on the exon level.

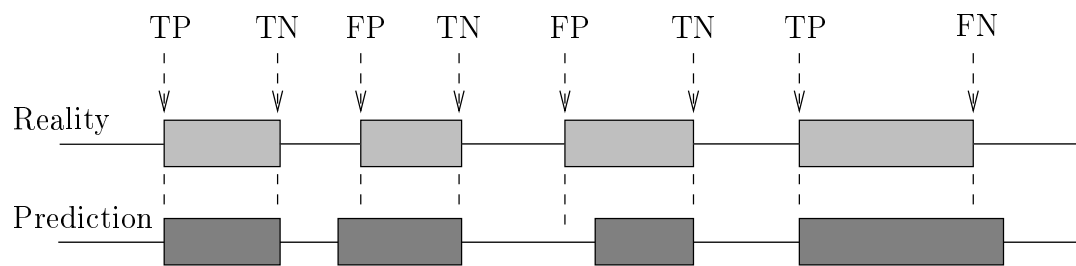


Figure 6.7.: Prediction classification on the exon level.

7. Conclusion

Fourier Analysis Gene Finder

We found an optimal solution (in the sense of minimal error probability) to the problem of discriminating between coding and non-coding regions in DNA sequences using Fourier Analysis. However, an exact implementation of this solution is not possible due to its enormous demand for computational power. The current implementation employs an approximation and can hence not be considered optimal. The evaluation of the actual prediction accuracy and comparisons with other Fourier analysis based coding measures are subject of future research.

The results of the current implementation are intended to be interpreted by a human operator. That is the search for coding regions that make up possible genes is a manual process. The automation of this process is subject of future research.

Statistical Gene Finder

The prediction accuracy of the statistical gene finder is heavily dependent on the size of the chosen search window. This weakness can be partially compensated with help of the Fourier analysis gene finder which allows to predict the approximate position of a gene and therefore allows to use a narrow search window. However, the better solution is to improve the used generalized hidden Markov model. A possible improvement would be the addition of two signal states to model the transcription promoter and terminator sequences present close to the beginning and ending of a gene.

The current implementation employs the prefix sum array (PSA) decoding algorithm to find the most probable parse of a given DNA sequence. Possible improvements can be found in [Majoros et al., 2005].

The overall performance of our statistical gene finder can be considered satisfactory for the time and manpower that was available to develop it. But it is not as good as other currently available gene finder solutions. Unfortunately, a direct comparison of our and other gene finders is not possible because we did not train and run our gene finder on a standardized set of DNA sequences commonly used for comparison.

A. Matlab Program Description

This chapter describes the file and directory structure to use our files. The description of the directories follows the logical way to use our programs.

Data Directory ../data

The data directory contains all GenBank¹ files from the plant *Arabidopsis Thaliana*. The .gb format is readable with any editor understanding unix-formatted files.

Every file contains the complete sequence of the chromosome. The structure of the information stored in the .gb-files is described here².

File	Content	Date
NC_003070.gb	Chromosome I	25-JAN-2005
NC_003071.gb	Chromosome II	04-NOV-2005
NC_003074.gb	Chromosome III	04-NOV-2005
NC_003075.gb	Chromosome IV	04-NOV-2005
NC_003076.gb	Chromosome V	04-NOV-2005

The filename is also the accession number for the GenBank. Chromosome I has an older date because, we had problems to read out the information from the newest release with the `genbankread` function from the Matlab Bioinformatics Toolbox.

get_gene.m	
Paramter	GenBank Accession Number
internal Parameter	none
return Value	gene: structure with all genes in the gbid file; nucseq: the nucleotide sequence over the whole chromosome; totlen: the total length of the nucleotide sequence
Description	GET_GENE extract the information stored in the GenBank file and save the information in a matlab structure. Because reading the matlab structure is faster then reading the GenBank file. Uses the <code>genbankread</code> function.

¹GenBank Database: <http://www.ncbi.nlm.nih.gov/Genbank/index.html>

²GenBank Sample Record: <http://www.ncbi.nlm.nih.gov/Sitemap/samplerecord.html>

A. Matlab Program Description

genbankread.m	
Parameter:	GenBank Accession Number
internal Parameter:	none
return Value:	data: A structure containing the information from the GenBank file.
Description:	This function differs with the function from Matlab since the Matlab function has a bug. Uses the <code>matchstart</code> and the <code>referenceparse</code> function.

Fourier Training Directory ../fouriertraining

main.m	
Parameter	none
internal Parameter	<code>gbid</code>
return Value	none
Description	Calls all training functions of the Fourier analysis gene finder.

train_coding.m	
Parameter	none
internal Parameter	<code>N</code> , <code>n_trainvec</code> , <code>threshold</code>
return Value	<code>cluster_coding.mat</code>
Description	Calculates the approximated probability distribution for coding regions.

train_noncoding.m	
Parameter	none
internal Parameter	<code>N</code> , <code>n_trainvec</code> , <code>threshold</code>
return Value	<code>cluster_noncoding.mat</code>
Description	Calculates the approximated probability distribution for noncoding regions.

train_threshold.m	
Parameter	<code>gene</code> , <code>totlen</code>
internal Parameter	none
return Value	<code>threshold.mat</code>
Description	Calculates the optimal decision threshold for given training data.

Fourier Analysis Directory ../fourieranalysis

main.m	
Parameter	none
internal Parameter	gbid, start, stop
return Value	none
Description	Runs the Fourier analysis gene finder on DNA from file gbid in region start index to stop index and plots results.

fourier_analysis.m	
Parameter	charseq
internal Parameter	N, step, spread
return Value	pos, phase, thresh
Description	Fourier analysis subroutine. Analyzes charseq in all three phases, locates in-phase stop codons and loads the threshold value.

Sensor Training Directory ../sensortraining

main.m	
Parameter	none
internal Parameter	none
return Value	none
Description	Call the functions to train all sensors and the transitions of the GHMM.

weightarray_acceptor_training.m	
Parameter	gene, nucseq
internal Parameter	reg5, reg3, offset
return Value	wam_acceptor.mat
Description	Train the acceptor splice site sensor with a weight array model (WAM).

dependency_donor_training.m	
Parameter	gene, nucseq
internal Parameter	consensus structure
return Value	dependency_donor.mat
Description	Train the sensor of the donor splice site. Uses the dependency_donor_recursion function.

length_distribution_training.m	
Parameter	gene, totlen
internal Parameter	none
return Value	length_distribution.mat
Description	Train the intron, exon and noncoding length distributions.

A. Matlab Program Description

mm_intron_training.m	
Parameter	gene, nucseq
internal Parameter	order
return Value	mm_intron.mat
Description	Train the intron sensor with a fixed 5 order Markov model.

trainingIMM.m	
Parameter	gene, nucseq
internal Parameter	IMMORDER
return Value	trainingIMM.mat
Description	Train the exon sensor with an interpolated Markov model.

mm_intergenic_training.m	
Parameter	gene, nucseq
internal Parameter	order
return Value	mm_intergenic.mat
Description	Train the intergenic sensor with a fixed 5 order Markov model.

weightarray_start_training.m	
Parameter	gene, nucseq
internal Parameter	offset, len, gboffset
return Value	wam_start.mat
Description	Train the translation start site sensor with a weight array model (WAM).

weightarray_stop_training.m	
Parameter	gene, nucseq
internal Parameter	offset, len, gboffset
return Value	wam_stop.mat
Description	Train the translation stop site sensor with a weight array model (WAM).

transprob_training.m	
Parameter	gene
internal Parameter	maxcount
return Value	transprob.mat, exoncount_dist.mat
Description	Train the transition probabilities of the GHMM.

Simple Analysis Directory ../simpleanalysis

main.m	
Parameter	none
internal Parameter	none
return Value	none
Description	The main file to predict one gene with the GHMM and compare the prediction with the true gene from the GenBank.

ghmm.m	
Parameter	seq
internal Parameter	none
return Value	result, queue
Description	The implementation of the generalized hidden Markov model.

mm_intron.m	
Parameter	seq
internal Parameter	none
return Value	score
Description	Score a sequence with intron properties.

IMM.m	
Parameter	sequence, initphase
internal Parameter	none
return Value	score
Description	Score a sequence with exon properties.

dependency_donor.m	
Parameter	seq
internal Parameter	none
return Value	pos, score, sigprop
Description	Score a sequence with donor splice site properties.

weightarray_start.m	
Parameter	seq
internal Parameter	none
return Value	pos, score, sigprop
Description	Score a sequence with start codon properties.

weightarray_stop.m	
Parameter	seq
internal Parameter	none
return Value	pos, score, sigprop
Description	Score a sequence with stop codon properties.

A. Matlab Program Description

weightarray_acceptor.m	
Parameter	seq
internal Parameter	none
return Value	pos, score, sigprop
Description	Score a sequence with acceptor splice site properties.

plotresult.m	
Parameter	result, soll, interval
internal Parameter	none
return Value	none
Description	Plot the prediction from the GHMM with the true gene from the GenBank.

Global Training Directory ../globaltraining

results.m	
Parameter	none
internal Parameter	none
return Value	prednuc.mat, predexon.mat, exoncount_dist.mat
Description	Run the GHMM on 1000 random chooses genes from one chromosome and save the TP, TN, FP, FN for the nucleotide and the exon level. Uses evaluate function.

globalTraining.m	
Parameter	none
internal Parameter	csize, cnum
return Value	chunkX.mat, ParameterChunkX.mat
Description	Run the GHMM to evaluate better transition probabilities over an amount of genes. Uses evaluate().

makeParameters.m	
Parameter	none
internal Parameter	none
return Value	parameter.mat
Description	Transition probabilities for the global training of the GHMM.

gradientascent.m	
Parameter	none
internal Parameter	none
return Value	tmpParameter.mat
Description	Algorithm to find better transition probabilities. Uses the <code>evaluate</code> function.

evaluate.m	
Parameter	none
internal Parameter	none
return Value	score
Description	Evaluate the GHMM with the new transition probabilities and save the TP, TN, FP, FN for the <code>results</code> function or calculate a score out of TP, TN, FP, FN for the <code>gradientascent</code> function.

GFF Analysis Directory ../gffanalysis

main.m	
Parameter	none
internal Parameter	none
return Value	ghmm.mat
Description	Function to store the prediction of the GHMM in a General Feature Format (GFF).

makeArray.m	
Parameter	none
internal Parameter	none
return Value	none
Description	Function to prepare the prediction and the true gene, to store the information in a GFF file.

gff.m	
Parameter	seq, seqname, source
internal Parameter	none
return Value	*.gff file
Description	Make a General Feature Format file of a given sequence.

B. Matlab Source Code Listings

Listing B.1: Gradient Ascent Method.

```
function gradientAscent()
currentScore = evaluate();
changes = true;
while(changes)
    changes = false;
    for dim=1:numDim
        if(~tmpParameters(dim).direction)
            if (rand(1) < 0.5)
                tmpParameters(dim).direction = -1;
            else
                tmpParameters(dim).direction = 1;
            end
        end
        minimum = tmpParameters(dim).minimum;
        maximum = tmpParameters(dim).maximum;
        delta = tmpParameters(dim).direction * tmpParameters(dim)
            .stepSize;
        %Try to step in the current direction
        newProb = tmpParameters(dim).prob + delta;
        if(newProb <= maximum && newProb >= minimum)
            tmpParameters(dim).prob = newProb;
            newScore = evaluate();
            if(newScore > currentScore)
                currentScore = newScore;
                changes = true;
                continue
            end
        end
    end
end
```

B. Matlab Source Code Listings

```
%direction was not good. try the opposite direction
newProb = tmpParameters(dim).prob - delta;
if(newProb <= maximum && newProb >= minimum)
    tmpParameters(dim).prob = newProb;
    newScore = evaluate();

    if(newScore > currentScore)
        currentScore = newScore;
        changes = true;
        tmpParameters(dim).direction = tmpParameters(dim)
            .direction * -1;
        continue
    end
end

%neither direction works -> reduce stepSize
tmpParameters(dim).direction = 0;
if (tmpParameters(dim).stepSize > tmpParameters(dim).
    minStepSize)
    tmpParameters(dim).stepSize = tmpParameters(dim).
        stepSize / 2;
    changes = true;
else
    tmpParameters(dim).stepSize = tmpParameters(dim).
        minStepSize
end
end
end
```

Listing B.2: Vector quantization clustering.

```
% Vector quantisation clustering of training vectors zm

% initialisation
ncluster = 1;

% initial cluster
cluster(1).seed = zeros(size(zm,1),1);;
cluster(1).z = [real(zm); imag(zm)];
cluster(1).centroid = mean(cluster(1).z,2);
cluster(1).n = size(zm,2);
for i=1:size(cluster(1).centroid,1)
    tmpz(i,:) = cluster(1).z(i,:) - cluster(1).centroid(i,1);
end
cluster(1).covar = tmpz * tmpz' / cluster(1).n;
cluster(1).sigma = sqrt(trace(cluster(1).covar));
```

```

while(true)
    % do splitting
    nosplit = true;
    for i=1:ncluster
        if (cluster(i).sigma > threshold)
            nosplit = false;
            [V,D] = eig(cluster(i).covar);
            [dummy,idx] = max(diag(D));
            splitdirvec = V(:,idx);
            cluster(i).seed = cluster(i).centroid + 1e-3*
                splitdirvec;
            ncluster = ncluster+1;
            cluster(ncluster).seed = cluster(i).centroid - 1e-3*
                splitdirvec;
        end
    end

    % finished?
    if nosplit
        break;
    end

    for i=1:ncluster
        cluster(i).z = [];
    end

    % reassign training vectors
    for iz=1:length(zm)
        for icl=1:ncluster
            dist(icl) = sum((cluster(icl).seed - [real(zm(:,iz));
                imag(zm(:,iz))]).^2);
        end
        [dummy,idx] = min(dist);
        cluster(idx).z = [cluster(idx).z [real(zm(:,iz)); imag(zm
            (:,iz))]];
    end

    % update clusters
    for i=1:ncluster
        cluster(i).centroid = mean(cluster(i).z,2);
        cluster(i).n = size(cluster(i).z,2);
        tmpz = [];
        for k=1:size(cluster(i).centroid,1)
            tmpz(k,:) = cluster(i).z(k,:) - cluster(i).centroid(k
                ,1);
        end
    end

```

B. Matlab Source Code Listings

```
        end
        cluster(i).covar = tmpz * tmpz' / cluster(i).n;
        cluster(i).sigma = sqrt(trace(cluster(i).covar));
    end
end
```

C. Storage and Visualization of Prediction Results

To exchange the prediction results from a gene finder, it is recommended to use a wide accepted format to store the predictions. To store the information from the predicted genes, we use the General Feature Format (GFF). To visualize the predictions the `gff2ps` program is used.

C.1. General Feature Format Specification

This chapter gives is a short introduction the the General Feature Format (GFF) Specification. More information is available from here¹. The GFF is simple readable format where every row describes a feature from the prediction of the gene finder. Here is a example record from our thesis (Prediction of gene AT3G49850 from the chromosome III of *Arabidopsis thaliana*, file `resultAT3G49850.gff`):

```
##date 2005-12-15 10:51:53
##Type DNA result
##produced with matlab
result pred start 18500436 18500438 . . . startcodon
result pred exon 18500436 18500585 . . . exon0
result pred intron 18500586 18500662 . . . intron0
result pred exon 18500663 18500877 . . . exon1
result pred intron 18500878 18500981 . . . intron1
result pred exon 18500982 18501066 . . . exon2
result pred intron 18501067 18501178 . . . intron2
result pred exon 18501179 18501262 . . . exon3
result pred intron 18501263 18501362 . . . intron3
result pred exon 18501363 18501716 . . . exon4
result pred stop 18501714 18501716 . . . stopcodon
```

Comments

The `##` indicates comments for additional information in the gff file. The comment should be at the first position in a row to make the whole line a comment or after all the required fields.

¹http://www.sanger.ac.uk/Software/formats/GFF/GFF_Spec.shtml

Fields

The space between every field is a tabulator (ASCII: 0x09). The fields in a line are: <seqname> <source> <feature> <start> <end> <score> <strand> <frame> [attributes] [comments] If a field is left empty, one write a single point “.” in to the field.

<seqname>

The name of the sequence. It is a label to allocate a feature to a certain group. We use the following names: result is the sequence name for the prediction from our gene finder; geneInRangeX is the name for the sequences from the GenBank², the X indicates the number of the genes in the same range, because some genes have more than one entries.

<source>

The source of this feature. The field indicates where does the feature come from. The feature could be from the prediction “pred” or from the GenBank entries.

<feature>

The feature type name. We only use the start, stop, exon and intron features. These are the startcodon, the stopcodon, the exons and the introns for one gene.

<start>, <end>

The absolute start and stop position for the feature in the Chromosome. <start> must be less or equal to <end>.

<score>

We do not give a score to our features. Therefor we write a single point “.” in this field.

<strand>

The strand where the feature occurs is not relevant for us. The field is left empty.

<frame>

We do not care in which frame or phase does the feature occur. So the field is empty.

[attributes]

We write out the name of the feature in this field. And count the numbers of introns and exons between one start- and stopcodon.

[comments]

We do not comment the features we use, since they are self explaining.

C.2. gff2ps Program

The gff2ps (General Feature Format to Postscript) program is used the visualize the information stored in the gff files. The program is available from here³ under the GNU General Public License. Gff2ps is a command-line tool and can be used on linux and windows based workstations. You need to install the cygwin⁴ environment

²<http://www.ncbi.nlm.nih.gov/Genbank/index.html>

³<http://www1.imim.es/software/gfftools/GFF2PS.html>

⁴www.cygwin.com

when you are using a windows pc because gff2ps needs the awk and the ps program. Each Postscript output file uses a specific configuration file to zoom to the relevant region.

Bibliography

- [Alberts et al., 1994] Alberts, B., Bray, D., Lewis, J., Raff, M., Roberts, K., and Watson, J. D., 1994. *Molecular Biology of the Cell*. Garland Publishing, Inc.
- [Anastassiou, 2000] Anastassiou, D., 2000. Frequency-domain analysis of biomolecular sequences. *Bioinformatics*, **16**:1073–1081.
- [Burge and Karlin, 1997] Burge, C. and Karlin, S., 1997. Prediction of complete gene structures in human genomic DNA. *J Mol Biol*, **268**:78–94.
- [Burset and Guigó, 1996] Burset, M. and Guigó, R., 1996. Evaluation of gene structure prediction programs. *Genomics*, **34**:353–367.
- [Cartegni et al., 2002] Cartegni, L., Chew, S. L., and Krainer, A. R., 2002. Listening to silence and understanding nonsense: exonic mutations that affect splicing. *Nature Review, Genetics*, **3**:285–298.
- [Fickett, 1982] Fickett, J. W., 1982. Recognition of protein coding regions in DNA sequences. *Nucleic Acids Research*, **10**:5303–5318.
- [Grantham et al., 1980] Grantham, R., Gautier, C., Gouy, M., Mercier, R., and Pavé, A., 1980. Codon catalog usage and the genome hypothesis. *Nucleic Acids Research*, **8**:r49–r62.
- [Kotlar and Lavner, 2003] Kotlar, D. and Lavner, Y., 2003. Gene prediction by spectral rotation measure: a new method for identifying protein-coding regions. *Genome Research*, **13**:1930–1937.
- [Kozak, 1991] Kozak, M., 1991. Structural features in eukaryotic mRNAs that modulate the initiation of translation. *Journal of Biological Chemistry*, **266**:19867–19870.
- [Lipschutz, 1998] Lipschutz, S., 1998. *Schaum's Outline of Introduction to Probability and Statistics*. McGraw-Hill Professional Book Group.
- [Majoros et al., 2005] Majoros, W. H., Pertea, M., Delcher, A. L., and Salzberg, S. L., 2005. Efficient decoding algorithms for generalized hidden Markov model gene finders. *BMC Bioinformatics*, **6**:16–16.
- [Majoros and Salzberg, 2004] Majoros, W. H. and Salzberg, S. L., 2004. An empirical analysis of training protocols for probabilistic gene finders. *BMC Bioinformatics*, **5**:206–206.

Bibliography

- [Rabiner, 1989] Rabiner, L. R., 1989. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, **77**:257–286.
- [Reese et al., 1997] Reese, M. G., Eeckman, F. H., Kulp, D., and Haussler, D., 1997. Improved splice site detection in Genie. *J Comput Biol*, **4**:311–323.
- [Salzberg et al., 1996] Salzberg, S., Chen, X., Henderson, J., and Fasman, K., 1996. Finding genes in DNA using decision trees and dynamic programming. *Proc Int Conf Intell Syst Mol Biol*, **4**:201–210.
- [Salzberg et al., 1998a] Salzberg, S. L., Delcher, A. L., Kasif, S., and White, O., 1998a. Microbial gene identification using interpolated markov models. *Nucleic Acids Research*, **26**:544–548.
- [Salzberg et al., 1998b] Salzberg, S. L., Searls, D. B., and Kasif, S., editors, 1998b. *Computational Methods in Molecular Biology*. Elsevier Science B. V.
- [Schürmann, 1996] Schürmann, J., 1996. *Pattern Classification: A Unified View of Statistical and Neural Approaches*. John Wiley & Sons, Inc.
- [Staden and McLachlan, 1982] Staden, R. and McLachlan, A. D., 1982. Codon preference and its use in identifying protein coding regions in long DNA sequences. *Nucleic Acids Res*, **10**:141–156.
- [Tiwari et al., 1997] Tiwari, S., Ramachandran, S., Bhattacharya, A., Bhattacharya, S., and Ramaswamy, R., 1997. Prediction of probable genes by Fourier analysis of genomic sequences. *Comput Appl Biosci*, **13**:263–270.
- [Viterbi, 1967] Viterbi, A. J., 1967. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transaction on Information Theory*, **13**:260–267.
- [Voss, 1992] Voss, R., 1992. Evolution of long-range fractal correlations and 1/f noise in DNA base sequences. *Physical Review Letters*, **68**:3805–3808.
- [Watson et al., 2004] Watson, J. D., Baker, T. A., Bell, S. P., Gann, A., Levine, M., and Losick, R., 2004. *Molecular Biology of the Gene*. Pearson Education, Inc.
- [Zhang and Marr, 1993] Zhang, M. Q. and Marr, T. G., 1993. A weight array method for splicing signal analysis. *Computer Applications in the Biosciences*, **9**:499–509.