

Gene prediction using a combination of GHMM and SVM

DIPLOMA THESIS

DIPL. EL. ING. FH

HOCHSCHULE RAPPERSWIL
DEPARTMENT ELEKTROTECHNIK

Authors:

Philipp Beck and Pascal Frei

Advisors:

Prof Dr. Guido M. Schuster¹

Assoc Prof Dr. Tom Ryen²

Prof Dr. Trygve Eftestol²

¹HSR Hochschule für Technik Rapperswil

²UiS University of Stavanger, Norway

Rapperswil, 30.12.2006

Abstract

The progress of biologist during the last years has required the help of computerised methods. These days, one of the main issues is identifying genes in a DNA. Mainly two different techniques have been developed to identify genes in DNAs, the coding region identifying methods and the *Generalized Hidden Markov Model*.

The aim of this thesis is to present a combination of a *Generalized Hidden Markov Model* and coding region identifying methods. Since there are different coding region identifying methods available, different *classifiers* such as k -nearest neighbor, Parzen window and SVM have been applied by using these methods as input. It is shown that the accuracy of the *classifiers* is more precise than the accuracy of the individual methods.

A way to add more information to the content sensors of the *GHMM* by using the probabilistic output of a *classifier* has been developed.

Extensive measures based on a well known and widely used test set have been executed to prove the accuracy of this combination. The obtained results of the combination show a noteworthy improvement in respect to the *GHMM* itself. However, the combination can not keep up with other gene prediction programs available these days.

Contents

1	Introduction	1
2	Genetic Fundamentals	3
2.1	History of the DNA	3
2.2	Structure of the DNA	3
2.3	From the DNA to the genes	4
2.3.1	Intergenic region	4
2.3.2	Translation Initiation Sites (TIS)	4
2.3.3	Exon	4
2.3.4	Intron	5
2.3.5	Translation stop site TSS	5
2.4	Protein machine	5
2.4.1	Transcription	5
2.4.2	Splicing	6
2.4.3	Translation	6
3	Identifying Coding Regions	9
3.1	Introduction	9
3.2	The Sliding Window	9
3.3	Fourier Analysis	10
3.4	Detrended Fluctuation Analysis	12
3.5	The Z-Curve	15
3.6	Hexacount Analysis	18
3.7	Spectral Rotation Measure	19
3.8	Entropy Deficiency	21
3.9	Interpolated Markov Model	22
4	Classifier	25
4.1	Introduction	25
4.2	Non-parametric density estimation	25
4.2.1	k -Nearest Neighbors	25
4.2.2	Parzen Window	26
4.3	Support Vector Machine	27
4.3.1	Introduction	27
4.3.2	Linear Classifier	27
4.3.3	Nonlinear Classifier	29
4.3.4	Decision Function	30
4.3.5	Mapping geometrical distances to probabilities	30

4.4	Applying classifiers on coding region identifying methods	31
4.5	Receiver Operation Characteristic Curve	32
4.5.1	Introduction	32
4.5.2	Computing the ROC-Curve	32
5	GHMM	37
5.1	Introduction	37
5.2	Differences between GHMM and HMM	37
5.3	Overview of the implemented GHMM	38
5.4	Signal Sensors	41
5.5	Content Sensor	43
5.6	Length Distribution	44
5.7	The Algorithm itself	46
5.7.1	Pre-analyse	46
5.7.2	Analyse	46
5.7.3	Backtrack	47
6	Combination of the GHMM and the classifier	49
6.1	Introduction	49
6.2	The classifier output as content sensor	49
6.3	A combination of IMM and classifier	51
7	Results	53
7.1	Used data sets	53
7.2	Performance of the different classifiers	54
7.2.1	Training	54
7.2.2	Discussion	55
7.3	Performance of the combination of GHMM and SVM	56
7.3.1	Introduction	56
7.3.2	Training	58
7.3.3	Discussion	59
8	Conclusion	61
8.1	Review	61
8.2	Outlook	61
9	Software	63
9.1	Introduction	63
9.2	Server	63
9.3	Servers configuration file	64
9.3.1	[ALL]	64
9.3.2	[FEATURE]	64
9.3.3	[CLASSIFIER]	65
9.3.4	[GHMM]	65
9.3.5	[SERVER]	66

9.3.6	[MAKEFASTA]	66
9.4	Client	66
9.5	Communication between Clients and Server	68
9.6	Gene program parser	70
A	Class diagrams of the Server application	71
B	Class diagrams of the GUI application	77
	Bibliography	82

List of Figures

1.1	Exponential growth of identified nucleotides.	1
2.1	Structural view of the DNA	4
2.2	Processes from the DNA to the proteins	5
2.3	Alternative Splicing: Some exons are skipped to build a protein	6
3.1	Sliding Window	9
3.2	Fouriertransformation of a DNA region	10
3.3	Fourier prediction	11
3.4	DNA walk	12
3.5	DFA on a window with length $N = 512$ and segment length $l = 64$	13
3.6	Detrended Fluctuation Analysis of a DNA region	14
3.7	Period-3 fractals deviations of the three phases	14
3.8	MAXFD codon index	15
3.9	YZ-Score for a sequence of 10000 nucleotides.	17
3.10	Hexacount analysis	18
3.11	The distributions of $\arg\left[X_b\left(\frac{N}{3}\right)\right]$ of coding regions	19
3.12	The distributions of $\arg\left[X_b\left(\frac{N}{3}\right)\right]$ of non coding regions	20
3.13	SR Measure over 10000 nucleotides	20
3.14	Entropy Deficiency over 10000 nucleotides	22
3.15	Summed up logarithm probabilities of the four IMMs.	23
3.16	IMM analysis of 10000 nucleotides	23
4.1	k -Nearest Neighbors with $k = 3$ and $k = 5$	26
4.2	Parzen windows with two different h parameters. a) $h = 0.1$, b) $h = 0.3$	27
4.3	Concept of a hyperplane as a linear classifier	28
4.4	Maximizing the margin γ	29
4.5	Cross-entropy error	31
4.6	From a given DNA sequence to a binary prediction output	32
4.7	Measures of prediction accuracy	33
4.8	The ROC-Curve	34
5.1	Implemented GHMM	40
5.2	Signal lengths and offsets	42
5.3	The length distribution of different contents	44
5.4	A DSSF signal example	47
6.1	GHMM with SVM for test set number 7	50
6.2	GHMM with SVM for testset number 21	50

6.3	GHMM content sensor parameter evaluation	52
7.1	Structure of the sequences in AraSet	53
7.2	ROC-Curve of the different classifiers	55
7.3	GHMM used by <i>GenScan</i>	57
7.4	58
9.1	GUI of the client	67
9.2	The handling of a request from a client to the server.	69
A.1	Class diagram of the feature part	72
A.2	Class diagram	73
A.3	Class diagram of the GHMM part	74
A.4	Class diagram of the Server part	75
B.1	Class diagram of the view part	78
B.2	Class diagram of the selectors part	79
B.3	Class diagram of the ROC part	80
B.4	Class diagram of the Server part	81

List of Tables

2.1	Translation from codons to amino acids	7
2.2	Signaling codons	7
3.1	Binary mapping	10
5.1	χ^2 dependency matrix for the translation start site	41
5.2	χ^2 dependency matrix for the donor splice site	42
5.3	Signal types their models and lengths	42
5.4	IMMs used	43
5.5	Bin size and cut off length of the different genomes	45
6.1	Changes to the SVM output for the GHMM	50
6.2	Changes to the SVM output for the GHMM	52
7.1	Sizes of the classifier training sets	54
7.2	Parameters for each classifier	54
7.3	Comparison of the different coding identifying methods	56
7.4	Sizes of the training sets used to train the SVM.	59
7.5	Prediction of the <i>AraSet</i>	59
7.6	Prediction of the first million nucleotides of chromosome IV	59

Chapter 1

Introduction

During the last 15 years biologists have made huge progress in extracting the DNA of different organisms. The amount of extracted DNA sequences has even been growing exponential in the last recent years. Figure 1.1 shows the gene bank size of the European Bioinformatics Institute. This gene bank is open to scientist, who want to publish their DNA sequences. At the moment there are DNA sequences with a total length of 150'178'605'649 nucleotides¹. A nucleotide is the logical unit a DNA is made of.

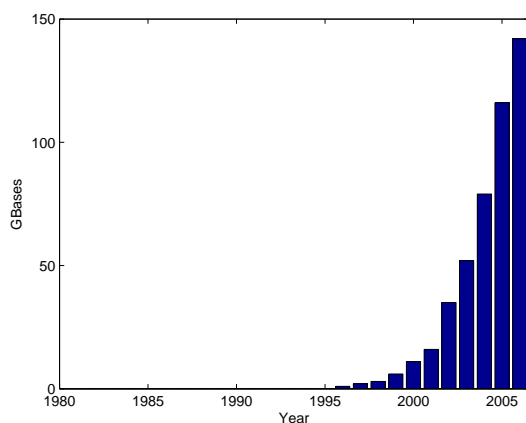


Figure 1.1: Exponential growth of identified nucleotides.
Source:European Bioinformatics Institute

The enormous amount of this data has created a new scientific field, called Bioinformatic. Bioinformatic, as its name implies, deals with applying computer science methods on biological problems. It is cut into different subareas, whereas the area concerning this thesis is *Gene prediction*.

The main part of a DNA sequence does not contain any relevant information, but only a small part does contain the genes. The field of *Gene prediction* deals with the problem of finding this genes, which still is not solved satisfyingly.

The goal of this thesis is to present a combination of a General Hidden Markov Model, well known gene identifying methods and classifiers. The classifiers are used to predict, with help of the known identifying methods, where in a DNA sequence a gene most likely

¹22. December 2006

could be. Then, this additional information is used for the content sensors of a General Hidden Markov Model to identify the exact position of the genes.

Chapter 2

Genetic Fundamentals

2.1 History of the DNA

Every organism is made out of billions and trillions of cells. Two different types of cells exist, namely the procaryotic and the eukaryotic cell. The main difference between this to cell types are the size of the cell and the location of the genome. The size of the procaryotic cell is about a tenth the size of the eukaryotic cell. In contrast to the procaryotic cell, the genome in the eukaryotic cell is located in the nucleus. In this thesis we only consider the eukaryotic cell, which is part of humans, animals and plants.

In 1869 Friedrich Miescher discovered that the nucleus is different to the rest of the cell. With experiments he showed the similarity of the nucleus to acids. Miescher also wrote in his paper that if there is any information about heredity in a cell, it must be stored in the nucleus.

After the dead of Miescher his discovery became of interest again in the forties of the last century because of an evidence that showed the appearance of the genetic information in the nucleus.

In 1953 James Watson and Francis Crick could explain the structure of the DNA and its function. Their work is deemed to be one of the most important discovery in the biological field and was prized with an Nobel prize in 1962.

2.2 Structure of the DNA

The DNA(Deoxyribonucleic acid) is formed of two anti parallel strands, which are complementary and wrapped around an imaginary axis in form of a double helix structure. These two strands consist of the sugar deoxyribose and a phosphate, referred to as backbone and a base. The combination of these three forms the structural unit of the DNA, the so called nucleotide. Four types of nucleotides with the abbreviation A, C, G and T exist, standing for adenine, cytosine, guanine and thymine. Since each nucleotide forms hydrogen bonds readily to only one other, namely A with T and C with G, each of the two strands is dependent on the other. Figure 2.1 shows a planar structural view of the DNA. Interestingly, although the two strands are dependent, each one contains separate genetic information. This makes it necessary to differ between these two strands. This is done by the so called direction. A strand begins always with a deoxyribose, referred to as 5' end, or a phosphate, referred to as 3' end. The possible direction are now 5'→3' or 3'→5'. Since the two strands are complementary to each other, the direction of one

strand is always the opposite of the other.

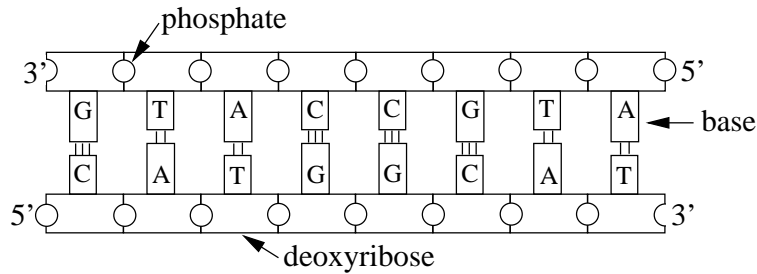


Figure 2.1: Structural view of the DNA

2.3 From the DNA to the genes

Up to 98.5% of the human DNA does not contain any information or more specifically, is not known yet to have a function. Humans for example have about 3.2 billion nucleotides and approximately about 15000 genes, although the true number is not known yet.

The DNA is divided into several regions which are shown in Figure 2.2. Following is an overview of the function of the several regions.

2.3.1 Intergenic region

The nucleotides between to genes are called intergenic region. Although this region contains so called promoters (see section 2.4.3), which can announce the possible start of gene, it does not contain any hereditary information.

2.3.2 Translation Initiation Sites (TIS)

The TIS indicates the protein machine to start the translation process which is described later. The TIS contains only one codon¹, namely AGT.

2.3.3 Exon

The exon is an integral part of the gen. It contains the codons which are used to form the proteins. The initial exon begins with the TIS. A gene is made of several exons. For example the human DNA contains genes with up to 178 exons. This exons are separated by the introns, which are explained in the next section.

¹The term “codon” will be explained in the next sections.

2.3.4 Intron

The introns are similar to the intergenic region, with the only difference that introns are located between the start and the end of a gene. The beginning of an intron is indicated by the sequence GT, the so called donor splice site (5' splice site), while the end of an intron is indicated by the sequence AG, the acceptor splice site (3' splice site).

2.3.5 Translation stop site TSS

The TSS marks the end of gene. This end is always indicated by the stop codon TGA. All nucleotides between a start codon and a stop codon are in the so called open reading frame ORF.

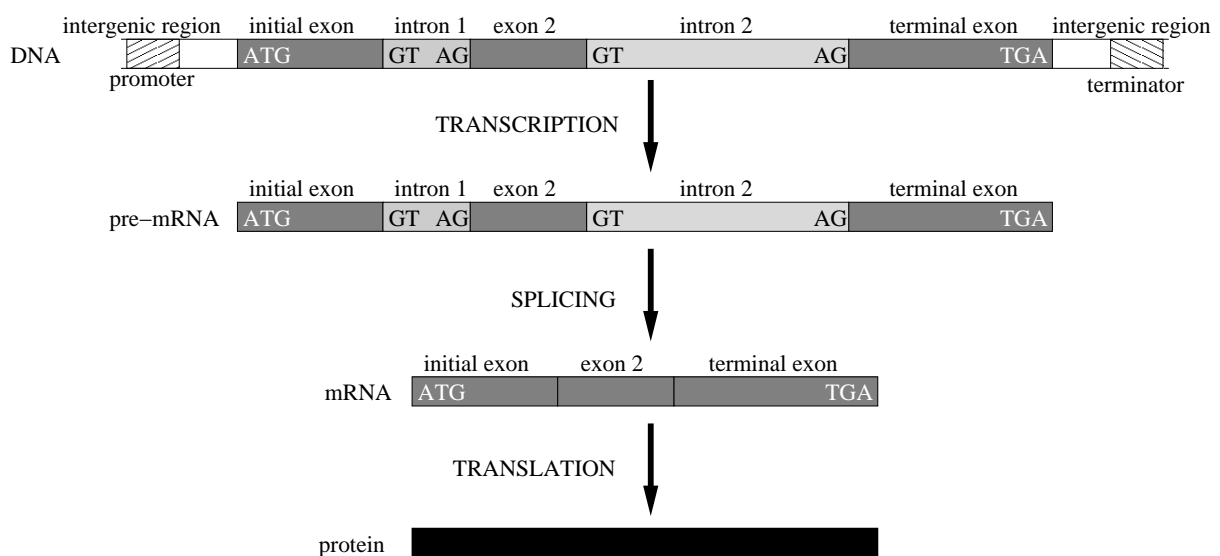


Figure 2.2: Processes from the DNA to the proteins

2.4 Protein machine

With Protein Machine the entire group of processes is meant, which are needed to translate a DNA sequence to a protein. In the following section these processes are explained.

2.4.1 Transcription

During the process called transcription, all the nucleotides between the promoter and the terminator are copied, leading to the pre-mRNA (pre messenger ribonucleic acid). The pre-mRNA is almost a perfect copy of a selected DNA sequence with the difference that the nucleotide thymine is replaced by the nucleotide uracil (U). Since this nucleotide forms exactly like thymine only hydrogen bonds with adenine, every T can just be replaced by an U. For the matter of simplicity, in this thesis U is always represented by T.

2.4.2 Splicing

The step after the transcription is called splicing. Through splicing removes all introns from the pre-mRNA, leading to the mRNA. The mRNA contains all exons in a row, without a single intron between them. The mRNA is now ready for the next step, the translation.

Alternative splicing

It has been shown that not always all exons between a start and a stop codon are used to form a gene. It is possible that some exons are skipped and other ones are not. Figure 2.3 shows how different genes can be formed out of 4 exons. These days it is very difficult for biologist to decide which exons are involved to form a protein and which are not. Alternative splicing is a process which needs to be researched in the future.

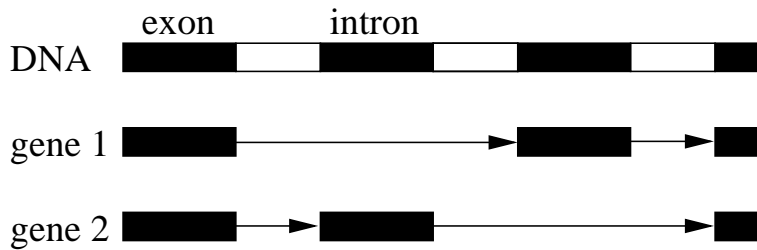


Figure 2.3: Alternative Splicing: Some exons are skipped to build a protein

2.4.3 Translation

Translation is the last step on the way to the protein. During the translation always three nucleotides are taken together to form a codon. This codons are translated in a second step to amino acids. Three out of four nucleotides can form 64 codons. But only 61 codons are translated to amino acids. The remaining three codons represent the stop codons. As exception, the sequence ATG is an start codon and represents also the amino acid Methionine. Table 2.2 shows all possible signaling codons. According to the table 2.1 the codons are translated to amino acids. Since there are only 20 amino acids, some codons are translated to the same amino acid. All the translated amino acids represent now the protein.

Amino Acid			Codon
A	Ala	Alanine	GCA GCC GCG GCT
C	Cys	Cysteine	TGC TGT
D	Asp	Aspartic acid	GAC GAT
E	Glu	Glutamic acid	GAA GAG
F	Phe	Phenylalanine	TTC TTT
G	Gly	Glycine	GGA GGC GGG GGT
H	His	Histidine	CAC CAT
I	Ile	Isoleucine	ATA ATC ATT
K	Lys	Lysine	AAA AAG
L	Leu	Leucine	TTA TTG CTA CTC CTG CTT
M	Met	Methionine	ATG
N	Asn	Asparagine	AAC AAT
P	Pro	Proline	CCA CCC CCG CCT
Q	Gln	Glutamine	CAA CAG
R	Arg	Arginine	AGA AGG CGA CGC CGG CGT
S	Ser	Serine	AGC AGT TCA TCC TCG TCT
T	Thr	Threonine	ACA ACC ACG ACT
V	Val	Valine	GTA GTC GTG GTT
W	Trp	Tryptophan	TGG
Y	Tyr	Tyrosine	TAC TAT

Table 2.1: Translation from codons to amino acids

Codon	Type
ATG	Start codon
TAA	Stop codon
TAG	Stop codon
TGA	Stop codon

Table 2.2: Signaling codons

Chapter 3

Identifying Coding Regions

3.1 Introduction

A nucleotide is considered as being coding if it belongs to an exon and non coding if it belongs to an intron or a intergenic region. In the last 20 years a lot of different methods have been developed to locate these coding regions in a DNA strand. A overview of these methods can be found in [13].

In this chapter, a closer look is given on seven methods, all of them are used later on as features for the classifiers.

3.2 The Sliding Window

Since most of the identifying methods described here can only predict whether a given sequence, limited to a specific length N , belongs to a coding region or not, a technique is needed to rate a whole DNA strand.

This is done by using sliding windows. The length N of the window is depending on which coding identifying method is used. The nucleotides inside this window are then rated. After that, the window is shifted L nucleotides downstream as shown in Figure 3.1 and the whole process starts over. The output of every rated window belongs to L nucleotides at a specific position i . i is defined separately by every method, but often the half of the window length N is added to the start position of the sliding window (x in Figure 3.1).



Figure 3.1: Sliding Window

3.3 Fourier Analysis

The Fourier Analysis was introduced by Tiwari *et al* [23]. It uses the fact that coding regions are more likely to have higher energy in the third spectral period.

To transform a given DNA sequence, the first thing needed is some kind of mapping nucleotides to numbers. Several mappings are possible, for example a conjugate complex mapping like $A \rightarrow 1 + j$, $C \rightarrow -1 + j$, $G \rightarrow -1 - j$ and $T \rightarrow 1 - j$. Here a binary mapping introduced by [25] is used. As shown in Table 3.1, every base $b \in \{A, C, G, T\}$ is stored in a separate array U_b , with the symbols $\{0, 1\}$, indicating the appearance at a specific position i in a DNA sequence.

x_j	A	C	A	T	C	G
U_A	1	0	1	0	0	0
U_C	0	1	0	0	1	0
U_G	0	0	0	0	0	1
U_T	0	0	0	1	0	0

Table 3.1: Binary mapping

Now the full fourier spectrum is simply the sum of the four transformed binary sequences

$$S\left(\frac{k}{N}\right) = \sum_b S_a\left(\frac{k}{N}\right) = \sum_b \frac{1}{N^2} \left| \sum_{i=1}^N U_b(x_i) e^{j2\pi \frac{k}{N} i} \right|^2 \quad (3.1)$$

In Figure 3.2(b)¹ a significant peak is clearly visible at position $S(\frac{N}{3})$ in the coding region, while in the non coding region the spectrum is mainly flat.

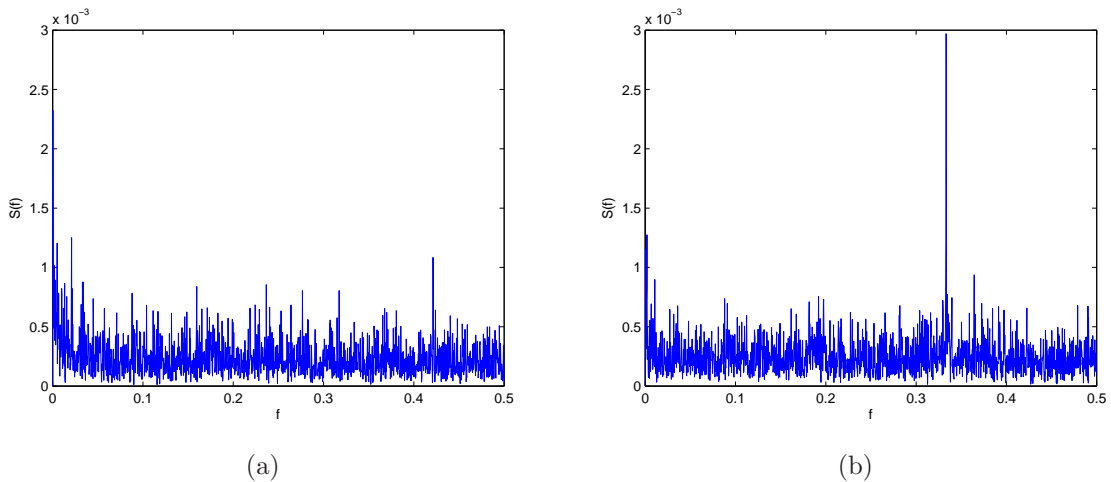


Figure 3.2: Fourier transformation of a DNA region a) non coding , b) coding

¹Since a binary mapping results always in a high DC energy, the mean of every sequence U_b was removed before transformation for visual purposes.

The energy of this peak at $\frac{N}{3}$ is compared to the average energy

$$\bar{S} = \frac{2}{N} \sum_{k=1}^{N/2} S\left(\frac{k}{N}\right) = \frac{1}{N} \left(1 + \frac{1}{N} - \sum_b \rho_b^2 \right) \quad (3.2)$$

in the entire DNA sequence and is taken as an indicator P for coding regions. ρ_b denotes the frequency of occurrence of every base in the sequence of the length N .

$$P = \frac{S\left(\frac{N}{3}\right)}{\bar{S}} \quad (3.3)$$

The Fourier analysis is done for a window with the length N , beginning at position x . The resulting P is then saved to the corresponding index $j = x + N/2$.

As an example in Figure 3.3 a window with length $N = 351$ and the step size $L = 3$ was used to analyze the Arabidopsis Thaliana Chromosome IV from position 167000 to 177000. As one can see, long exons are clearly indicated while it is more difficult to identify short exons.

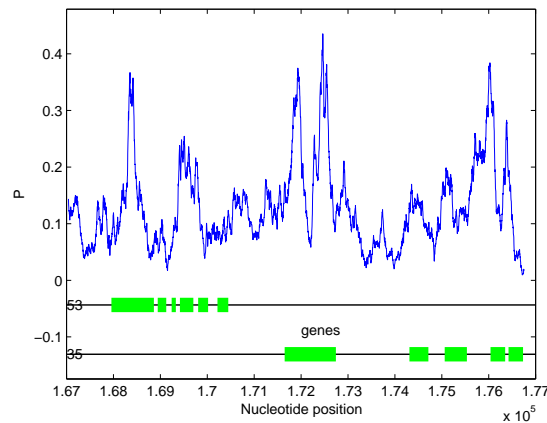


Figure 3.3: Fourier prediction

3.4 Detrended Fluctuation Analysis

The application of the Detrended Fluctuation Analysis (DFA) on genetic problems was introduced by Gao *et al* [12] in 2004. A short overview of that publication follows.

First of all, a mapping of nucleotides to numbers is needed. Azbel [3] suggested to code the bases with strong hydrogen bonds with the same symbol and the bases with weak hydrogen bonds with an other one. The mapping used here is A or T $\rightarrow u(i) = -1$ and G or C $\rightarrow u(i) = 1$.

Then a random walk (also called DNA walk) is done. A random walk is simply done by integrating over the sequence $u(i)$, resulting in $y(n)$.

$$y(n) = \sum_{i=1}^n u(i), \quad n = 1, 2, 3 \dots \quad (3.4)$$

Figure 3.4 shows a DNA walk over chromosome IV of Arabidopsis Thaliana from the position 167000 to 177000.

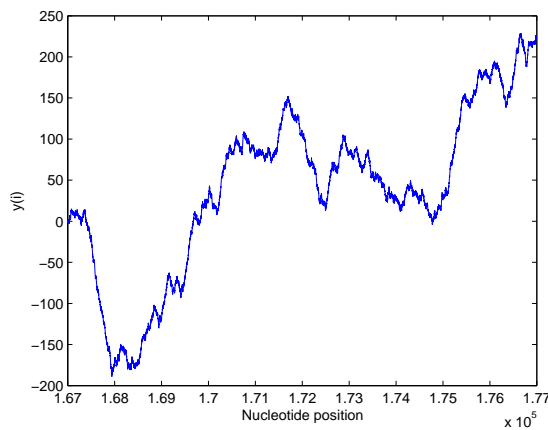


Figure 3.4: DNA walk

Now a DFA can be applied to this DNA walk $y(n)$. The DFA works as follows. A window over the DNA walk with the length N is divided into $a = \lfloor \frac{N}{l} \rfloor$ non overlapping segments of the length l . The curves in this segments are then approximated by straight lines $y_a(n)$ with the least squared error. As an example in Figure 3.5 a window is shown with the length $N = 512$ and a segment length l of 64. Then the fluctuation $F(l)$ is computed by taking the root-mean-squared error of the straight lines $y_a(n)$ compared to the DNA walk $y(n)$.

$$F(l) = \sqrt{\frac{1}{N} \sum_{n=1}^N (y(n) - y_a(n))^2} \propto l^H \quad (3.5)$$

H is the so called Hurst parameter. By computing F for different segment lengths l , H can be estimated as the slope of the best fitting straight line in a log-log plot. When

$H > 0.5$, the DNA walk contains long-range correlations, which are very common in non coding sequences. When $H = 0.5$, the DNA walk is completely at random.

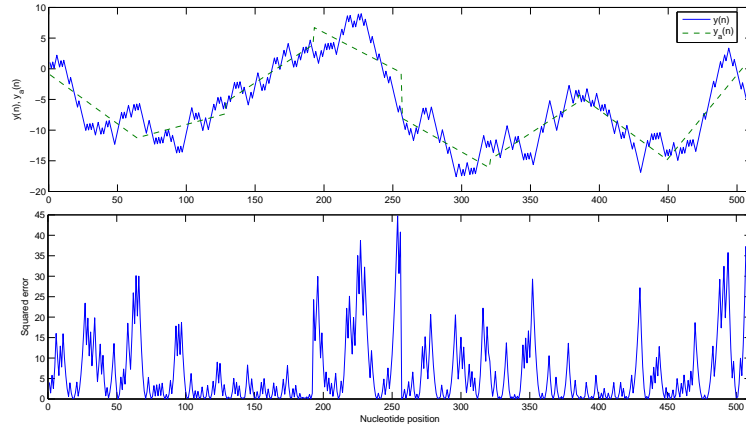


Figure 3.5: DFA on a window with length $N = 512$ and segment length $l = 64$

The exponential conduct of $F(l)$ is called the fractal scaling law. However, this law does not hold if there is a periodicity in sequence. As seen before, coding regions often have a periodicity of 3. This circumstance can now be used as an indicator for coding regions in the following way:

1. $F(l)$ is calculated for different $l > 3$.
2. Now the best fitting straight line $\log \hat{F} = H \log l + b$ needs to be found.
3. Then the period-3 fractal deviation is found by $PFD = |\log \hat{F}(3) - \log F(3)|^2$.

Figure 6.6 shows the difference between the coding and non coding region. The used window length is $N = 512$ and l is stepped from 2^2 to 2^6 .

Since the deviation in a coding region is only measurable when the reading frame is in phase with the start of a coding region, three reading frames, one for each phase, is needed. Figure 3.7 shows the values of this three reading frames for the position 167000 to 177000 of the chromosome IV of Arabidopsis Thaliana. It can be seen that long exons are indicated with a peak in one of the reading frames.

To decide which phase indicates the coding region best, simply the maximum is taken. Furthermore to reduce the distortion the average of M values is taken, which leads to the following codon index:

$$\text{MAXFD} = \frac{1}{\lfloor M/3 \rfloor} \sum_{m=1}^{\lfloor M/3 \rfloor} \max(\text{PFD1}(m), \text{PFD2}(m), \text{PFD3}(m)) \quad (3.6)$$

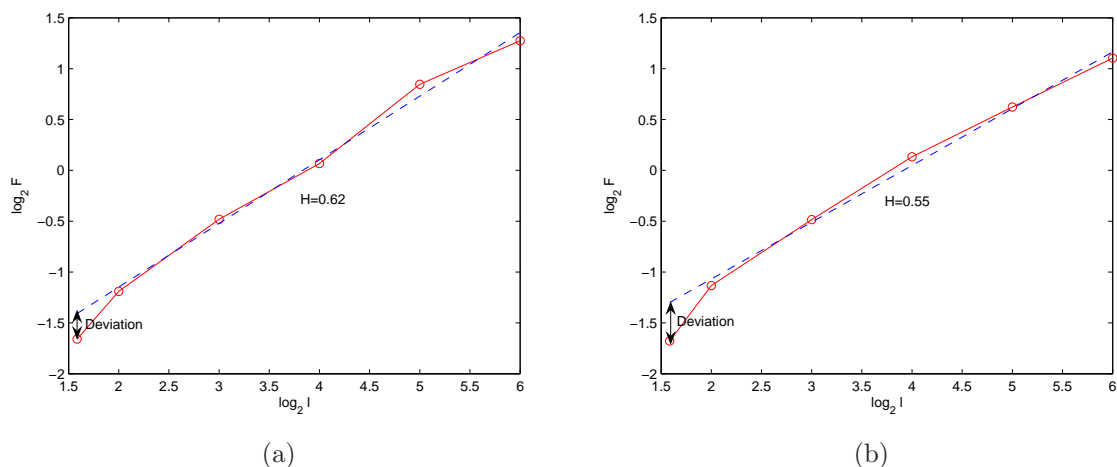


Figure 3.6: Detrended Fluctuation Analysis of a DNA region a) non coding , b) coding

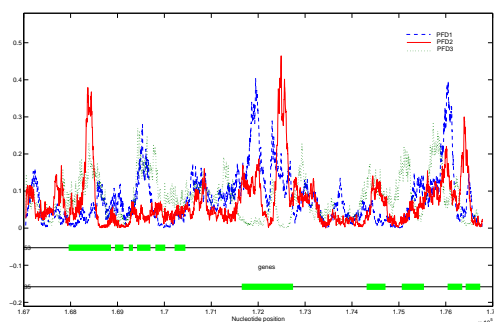


Figure 3.7: Period-3 fractals deviations of the three phases. The curves have been filtered with a fifth order moving average filter. The window size N is 256 and the scaling range l is $[2^2 \ 2^5]$.

MAXFD is shown in Figure 3.8 for the same nucleotides positions as above. Again, long exons are better indicated than short ones. There is also a problem of selectivity since very small introns are not clearly visible.

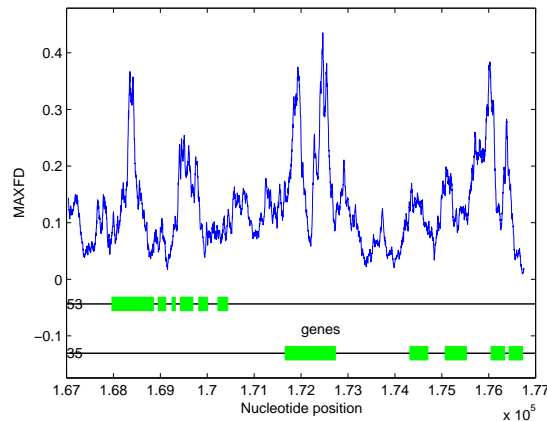


Figure 3.8: MAXFD codon index with a window length $N = 256$ and a average factor $M = 9$.

3.5 The Z-Curve

The Z-Curve was introduced by Chun-Ting Zhang and Ju Wang [9] in 2000. It has been proved that the Z-Curve is a very powerful tool for identifying coding regions. The Z-Curve represents a DNA sequence of the length N in a three-dimensional space. Furthermore, from a given Z-Curve the DNA sequence can be reconstructed easily and uniquely. The Z-Curve is constructed as follows:

$$Z_n = \begin{cases} x_n &= (A_n + G_n) - (C_n + T_n) \\ y_n &= (A_n + C_n) - (G_n + T_n) \\ z_n &= (A_n + T_n) - (C_n + G_n) \end{cases} \quad n = 0, 1, 2, \dots, N \quad (3.7)$$

X_n denotes the number of bases in the DNA sequence of the length n . Interestingly, the components of Z represent the biological behavior of the DNA. The component x_n represents the distribution of bases of the types purine/pyrimidine, the component y_n represents the distribution of bases of the types amino/keto and finally, the component z_n represents the distribution of bases of the weak and strong H-bond.

As seen above, the periodicity of three in the coding region is a strong indicator. So, the Z-Curve takes advantage of this property too by splitting the Z-Curve into three phase curves $Z_n(1)$ with the nucleotide positions $1, 4, 7, \dots$, $Z_n(2)$ with the nucleotide positions $2, 5, 8, \dots$ and $Z_n(3)$ with the nucleotide positions $3, 6, 9, \dots$. To simplify later calculations the components are approximated by straight lines

$$Z_n(1) = \begin{cases} x_n(1) &\approx k_x(1) \cdot n \\ y_n(1) &\approx k_y(1) \cdot n \\ z_n(1) &\approx k_z(1) \cdot n \end{cases}$$

$$Z_n(2) = \begin{cases} x_n(2) & \approx k_x(2) \cdot n \\ y_n(2) & \approx k_y(2) \cdot n \\ z_n(2) & \approx k_z(2) \cdot n \end{cases} \quad (3.8)$$

$$Z_n(3) = \begin{cases} x_n(3) & \approx k_x(3) \cdot n \\ y_n(3) & \approx k_y(3) \cdot n \\ z_n(3) & \approx k_z(3) \cdot n \end{cases}$$

with

$$\begin{aligned} k_x(1) &= 3 \cdot x_{N/3}(1)/N \\ k_y(1) &= 3 \cdot y_{N/3}(1)/N \\ k_z(1) &= 3 \cdot z_{N/3}(1)/N \\ k_x(2) &= 3 \cdot x_{N/3}(2)/N \\ k_y(2) &= 3 \cdot y_{N/3}(2)/N \\ k_z(2) &= 3 \cdot z_{N/3}(2)/N \\ k_x(3) &= 3 \cdot x_{N/3}(3)/N \\ k_y(3) &= 3 \cdot y_{N/3}(3)/N \\ k_z(3) &= 3 \cdot z_{N/3}(3)/N \end{aligned} \quad (3.9)$$

Now every DNA sequence can be represented as a point in the following 10-dimensional space:

$$\begin{aligned} u_1 &= k_x(1) \\ u_2 &= k_y(1) \\ u_3 &= k_z(1) \\ u_4 &= k_x(2) \\ u_5 &= k_y(2) \\ u_6 &= k_z(2) \\ u_7 &= k_x(3) \\ u_8 &= k_y(3) \\ u_9 &= k_z(3) \\ u_{10} &= \bar{a}^2 + \bar{c}^2 + \bar{g}^2 + \bar{t}^2 \end{aligned} \quad (3.10)$$

where \bar{a} , \bar{c} , \bar{g} and \bar{t} are the average occurrence frequencies of the different bases.

In [9] is a Fisher algorithm presented, which computes the Fisher coefficients \mathbf{c} from a trainings set for the following decision function:

$$F(\mathbf{u}) = \mathbf{u} \bullet \mathbf{c} - c_0 = \begin{cases} > 0 & \text{for coding regions} \\ < 0 & \text{for non coding regions} \end{cases} \quad (3.11)$$

The threshold c_0 is found at the point where the true positive rate is the same as the true negative rate. The YZ-Score can then be computed as follow:

$$YZ = \frac{F(\mathbf{u}) - F_{min}^-}{F_{max}^+ - F_{min}^-}, YZ \in [0, 1] \quad (3.12)$$

with $F_{min}^- = -0.3$ and $F_{max}^+ = 0.3$. The decision if a region is coding or non coding becomes simply $YZ > 0.5$ / $YZ < 0.5$.

Since the Z-Curve analysis in [9] is used for open reading frames which is not suitable for later use with a classifier, a sliding window is taken to analyze a DNA sequence. Interestingly, there is no dependence on the window phase. This leads to a simple window with the length N , which is shifted one by one nucleotide downstream. Figure 3.9 shows the YZ-Score of chromosome IV of Arabidopsis Thaliana from position 167000 to 177000. By taking 0.5 as decision value, exons are very good indicated.

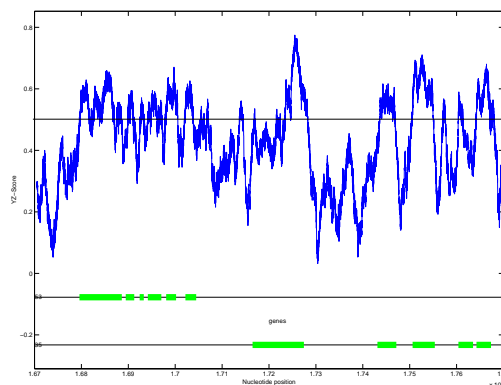


Figure 3.9: YZ-Score for a sequence of 10000 nucleotides. The used window length is $N = 102$.

Since the vector \mathbf{u} contains probably more information than the scalar YZ it makes sense to us it as input for the classifiers (see 4.4).

3.6 Hexacount Analysis

The Hexacount analysis for identification of coding regions was introduced by JM. Claverie and L. Bougueleret [14] in 1986. The Hexacount analysis is based on the idea that probabilities of tuples are different in coding alternatively non coding regions. A k -tuple contains k nucleotides. For example the sequence “ATGCGATCGATT” can be divided into four 3-tuples “ATG”, “CGA”, “TCG” and “ATT”. Since a DNA is made out of four different bases, the number of possible k -tuples is 4^k . Now the frequency of occurrence of every single k -tuple is counted in a non coding region and in a coding region respectively. For this thesis this has been done for $k = 5$. The two regions were made with exons and introns from the chromosome I of Arabidopsis Thaliana, which were joined together to have two sequences, one with only coding nucleotides and an other one with only non coding nucleotides.

With the frequencies of occurrence of the different tuples in these two sequences, the probabilities $P_{coding}(k\text{-tuple})$ and $P_{noncoding}(k\text{-tuple})$ can be estimated.

To analyze a DNA sequence, the codon index with a discriminator threshold of 0.5

$$\frac{P_{coding}(k\text{-tuple})}{P_{coding}(k\text{-tuple}) + P_{noncoding}(k\text{-tuple})} \quad (3.13)$$

is evaluated. This is done by shifting a window with the length $N = k = 5$ one by one nucleotide downstream. Since this approach leads to a very noisy codon index, a simple moving average filter with the length $M = 31$ is applied. In Figure 3.10 the sequence 167000 to 177000 of the chromosome IV is analyzed. By taking 0.5 as threshold, some of the exons are clearly indicated.

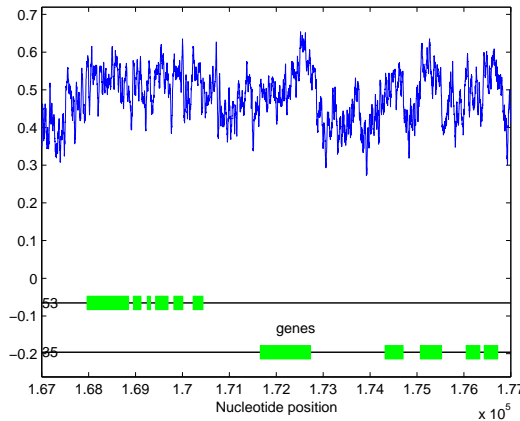


Figure 3.10: Hexacount analysis with 5-tuples from the positions 167000 to 177000 of the chromosome IV. The data has been filtered with a moving average filter with the length $M = 31$.

3.7 Spectral Rotation Measure

The spectral rotation (SR) measure, introduced in 2003 by Kotlar [10], is based on the Fourier analysis. But unlike the fourier analysis described before (see 3.3), it does not take advantage of the magnitude. Instead of the magnitude the phase is used to predict coding regions.

The same binary mapping as for the Fourier analysis is used, leading to the four arrays U_A, U_C, U_G and U_T . Now the Fourier transformation is taken of these four arrays. Since only the value at the position $\frac{N}{3}$ is of interest, the discrete Fourier transformation can be written as

$$X_b \left(\frac{N}{3} \right) = \sum_{n=0}^{N-1} u_b(n) e^{-j \frac{2\pi}{3} n}. \quad (3.14)$$

Since there appear only three different terms in equation 3.14 another simplification can be done:

$$X_b \left(\frac{N}{3} \right) = f(b, 1)1 + f(b, 2)e^{-j \frac{2\pi}{3}} + f(b, 3)e^{j \frac{2\pi}{3}} \quad (3.15)$$

$f(b, i)$ is the number of nucleotides of each base b at position n with n modulo 3 = i of the sequence with length N .

The main idea of the SR measure is now based on the fact that the distribution of $\arg[X_b(\frac{N}{3})]$ is different for the two regions coding and non coding. Figure 3.11 shows the distributions for $\arg[X_b(\frac{N}{3})]$ of 30000 sequences of chromosome I of Arabidopsis Thaliana. As one can see, most of the distributions are clearly gaussian shaped. In Figure 3.12 the distribution for $\arg[X_b(\frac{N}{3})]$ is shown. Here, the distribution is almost uniform.

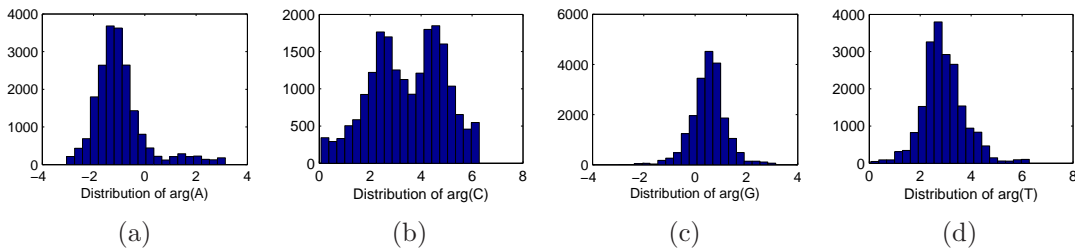


Figure 3.11: The distributions of $\arg[X_b(\frac{N}{3})]$ of coding regions

By taking the means $\bar{u}_A, \bar{u}_C, \bar{u}_G$ and \bar{u}_T of the coding distributions, every new sequence can be compared to them. If every new vector $X_b(\frac{N}{3})$ is multiplied with $e^{-\bar{u}_b}$ it should point direction 0 if the sequence is coding and to any angle if the sequence is non coding. Now all the bases are summed up to build one measure, the SR measure V^2 :

$$V^2 = \left| U_A \left(\frac{N}{3} \right) \frac{e^{-\bar{u}_A}}{\sigma_A} + U_C \left(\frac{N}{3} \right) \frac{e^{-\bar{u}_C}}{\sigma_C} + U_G \left(\frac{N}{3} \right) \frac{e^{-\bar{u}_G}}{\sigma_G} + U_T \left(\frac{N}{3} \right) \frac{e^{-\bar{u}_T}}{\sigma_T} \right|^2 \quad (3.16)$$

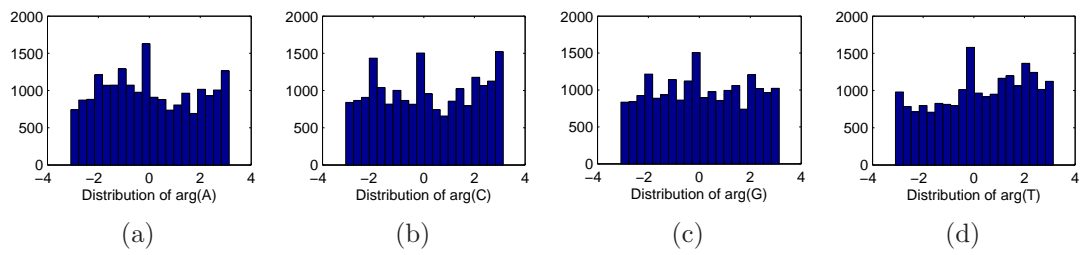


Figure 3.12: The distributions of $\arg[X_b(\frac{N}{3})]$ of non coding regions

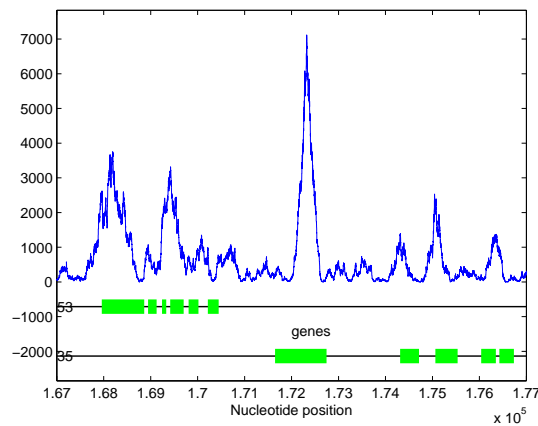


Figure 3.13: SR Measure over 10000 nucleotides. The used window length N is 351

If the sequence is coding all summands are pointing in the same direction and thereby V gets large. In the non coding case, all summands point to a different direction which leads to a small V . The summands are weighted with their corresponding standard deviation, whereby a summand with a small standard deviation has more weight.

Figure 3.13 shows as an example the SR measure of chromosome IV of Arabidopsis Thaliana from position 167000 to 177000.

3.8 Entropy Deficiency

The Entropy Deficiency (DS) was introduced by Almagor [2] in 1985. It is therefore one of the first developed methods for identifying coding regions. The method itself is quite simple and is based, as its name implies, on the entropy known from the information theory.

Every input sequence with the length N is divided into three in phase sequences of the length $\frac{N}{3}$. Then for every phase i the probabilities that the base b appears in the respective phase is computed. Next the entropy S_i for every phase is computed by using the base probabilities:

$$S_i = - \sum_{b=A,C,G,T} P_i(b) \ln[P_i(b)]. \quad (3.17)$$

The maximal value S_{\max} of an entropy is achieved for a uniform distribution

$$S_{\max} = - \sum_{b=A,C,G,T} \frac{3}{N} \ln \left[\frac{3}{N} \right] = \ln \left[\frac{N}{3} \right]. \quad (3.18)$$

Now the Entropy Deficiency DS_i is defined as

$$DS_i = S_{\max} - S_i = \ln \left[\frac{N}{3} \right] - \sum_{b=A,C,G,T} \frac{3}{N} \ln \left[\frac{3}{N} \right] = \sum_{b=A,C,G,T} P_i(b) \ln \left[P_i(b) \frac{N}{3} \right]. \quad (3.19)$$

Since it is assumed that a non coding region has almost a uniform distribution of the for bases b , DS_i should be close to zero. On the other hand in coding regions, DS_i should have a higher value. To decide which phase to choose, simply $\max\{DS_i\}_{i=1}^3$ is taken.

Figure 3.14 shows the well known sequence of chromosome IV of Arabidopsis Thaliana from position 167000 to 177000. As one can see, it is quite hard to reliably identify coding regions.

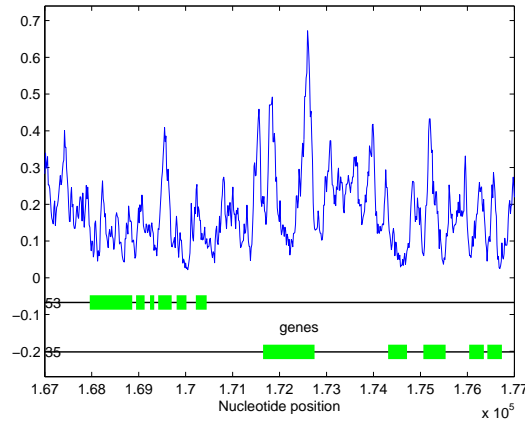


Figure 3.14: Entropy Deficiency over 10000 nucleotides. The used window length N is 150 and the stepsize L is 15

3.9 Interpolated Markov Model

The Interpolated Markov Model (IMM) was introduced by Burge [6] in 1997 to use it as content sensor in the gene prediction program *GenScan*. To use different IMMs for identifying coding regions an approach as described here has been used. Four IMMs are used to define the probabilities $P_j(\text{coding}|i)$ with $j = 1, 2, 3$ and $P(\text{noncoding}|i)$. $P_j(\text{coding}|i)$ is the probability that a nucleotide at position i belongs to a coding region in phase j whereas $P(\text{noncoding}|i)$ is the probability that a nucleotide at position i belongs to a non coding region.

Now the probabilities for every position i of a DNA sequence with the length N are computed and the logarithm of them is summed up:

$$u(n) = \sum_{i=1}^n \log[P(\text{noncoding}|i)] \quad (3.20)$$

$$y_j(n) = \sum_{i=1}^n \log[P_j(\text{coding}|i)] \quad (3.21)$$

Figure 3.15 shows as an example the summed up logarithm probabilities of chromosome IV of *Arabidopsis Thaliana* from position 167000 to 169000. The curves have almost the same slope as long as the region is non coding. In a coding region the slope of $u(n)$ starts to differ from the slope of $y_j(n)$.

To decide whether a sequence of the length N is coding or not the following differences are used.

$$\Delta u = u(1) - u(N) \quad (3.22)$$

$$\Delta y_j = y_j(1) - y_j(N) \quad (3.23)$$

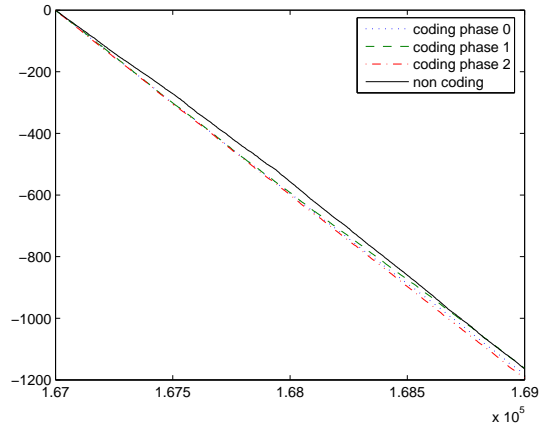


Figure 3.15: Summed up logarithm probabilities of the four IMM models.

The IMM coding index I is then defined as

$$I = \max\{\Delta y_j - \Delta u\}_{j=1}^3. \quad (3.24)$$

The larger I is the more likely a sequence of the length N is coding. To analyze a whole DNA sequence a sliding window is used again. Figure 3.16 shows the IMM coding index I from chromosome IV of *Arabidopsis Thaliana* from position 167000 to 177000. Both, long and short exons are almost on the whole length clearly indicated.

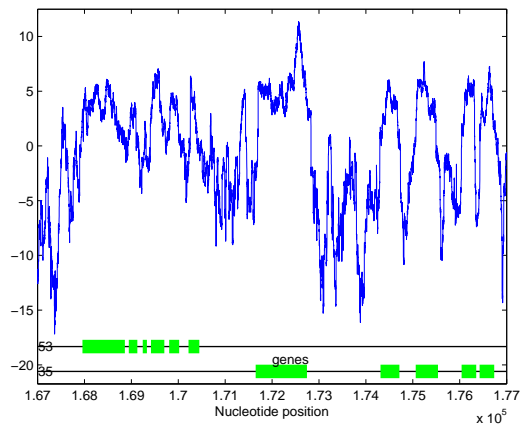


Figure 3.16: IMM analysis of 10000 nucleotides, the used window length N is 100 and the stepsize L is 1.

Chapter 4

Classifier

4.1 Introduction

A classifier is, spoken in mathematical terms, a mapping of a feature space $X \subseteq \mathbb{R}^n$ to a set of discrete labels $Y \in \mathbb{R}$. In the feature space X , a vector describes the characteristic attributes of a single observation. For example these attributes could be the size and weight of different persons. The classifier could then map these vectors to labels, for example “male” or “female”. In the following sections, different classifiers are explained in detail.

4.2 Non-parametric density estimation

The goal of the non-parametric density estimation is to find a density function $p(\mathbf{x})$ from a given training set.

The probability that a vector \mathbf{x} belongs to a certain region R is

$$P = \int_R p(\mathbf{z}) d\mathbf{z}. \quad (4.1)$$

If n denotes the size of the parametric set and n is large, $p(\mathbf{x})$ can be approximated by $\frac{k}{n \cdot V}$, whereas V is the volume of the region R and k is the number of vectors of the training set in this volume. The probability that a vector \mathbf{x} belongs to a certain region R simplifies to

$$P = \int_R p(\mathbf{z}) d\mathbf{z} \approx p(\mathbf{x})V = \frac{k}{n}. \quad (4.2)$$

The following two methods are used to define continuous density functions and to use them as classifiers.

4.2.1 k -Nearest Neighbors

The k -Nearest Neighbors method uses a sphere as volume V . The size of this sphere is increased until it contains k vectors of the training set in it. Figure 4.1 shows for example two different spheres defined over a parametric set with two classes ω_1 and ω_2 . The sphere on the left contains $k = 3$ vectors while the sphere on the right contains $k = 5$ vectors. As one can see, the euclidean distance from the center of the sphere to the

nearest neighbors defines the size of the sphere. Other distances are also thinkable, like the squared euclidean distance, manhattan distance or weighted euclidean distance.

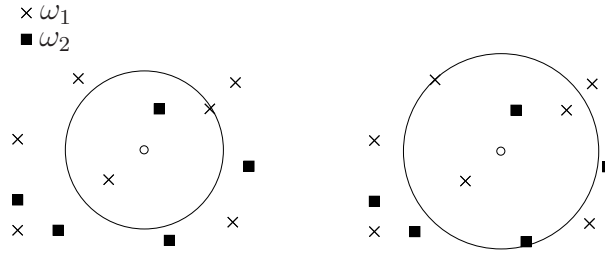


Figure 4.1: k -Nearest Neighbors with $k = 3$ and $k = 5$

If once the sphere is defined, the relative probability densities $p(\mathbf{x}|\omega_i)$ inside this sphere can be computed by simple counting all the vectors N_i belonging to a class ω_i and dividing it by the number of all vectors k inside this sphere

$$p(\mathbf{x}|\omega_i) = \frac{N_i}{k}. \quad (4.3)$$

Since the number of vectors of different classes in the parameter set are known a priori, the probability $P(\omega_i)$ is given. With the Bayes rule

$$P(\omega_i|\mathbf{x}) = \frac{p(\mathbf{x}|\omega_i)P(\omega_i)}{\sum_j p(\mathbf{x}|\omega_j)P(\omega_j)} \quad (4.4)$$

can be calculated.

4.2.2 Parzen Window

The Parzen window method is used to get a continuous probability density function out of a parameter set. This is done by placing a normalized kernel function around every sample. If N_i is the number of samples of every class in the parameter set, a probability density function for each class can be computed by

$$p(\mathbf{x}|\omega_i) = \frac{1}{N_i} \sum_{j=1}^{N_i} \varphi\left(\frac{\mathbf{x} - \mathbf{x}_{ij}}{h}\right). \quad (4.5)$$

In order to be a valid kernel function $\varphi(\cdot)$ has to meet the following conditions:

$$\varphi(\mathbf{z}) \geq 0 \quad \text{and} \quad \int_{-\infty}^{\infty} \varphi(\mathbf{z})d\mathbf{z} = 1 \quad (4.6)$$

Often a Gaussian function is taken as kernel function, leading to

$$p(\mathbf{x}|\omega_i) = \frac{1}{N_i} \sum_{j=1}^{N_i} \frac{1}{\sqrt{2\pi}h} \exp\left(-\frac{(\mathbf{x} - \mathbf{x}_i)(\mathbf{x} - \mathbf{x}_i)}{h^2}\right). \quad (4.7)$$

The parameter h is in case of a Gaussian function the standard deviation. If h is large, the single density functions around every sample are very wide and the sum of all these density functions becomes more and more to one single Gaussian density function. If $h \rightarrow 0$ the density function around each sample becomes to a Dirac delta function. Figure 4.2 shows the density function with two different h parameter.

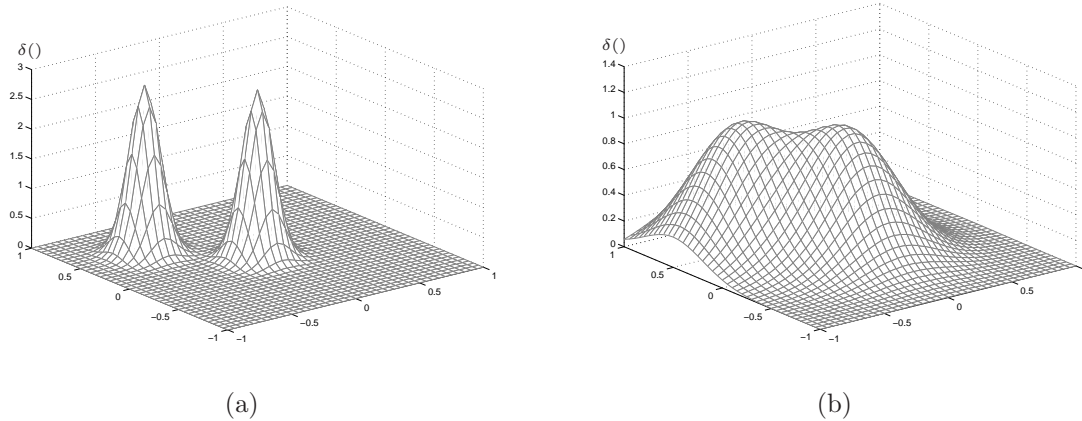


Figure 4.2: Parzen windows with two different h parameters. a) $h = 0.1$, b) $h = 0.3$

By using the Bayes' rule of equation 4.4 again, $P(\omega_i|\mathbf{x})$ can be computed for every class.

4.3 Support Vector Machine

4.3.1 Introduction

The Support Vector Machine (SVM) is a binary classifier and was introduced by Vapnik in 1979 but only became popular the last decade. Following a short introduction about the basic functionality of the SVM is given. For a introduction in more detail see the semester thesis [20] and the explanations of Burges [5], Christianini and Taylor [18] and Eitrich [11].

4.3.2 Linear Classifier

The main idea of the SVM is to separate two different classes by an hyperplane. Every vector $X \subseteq \mathbb{R}^n$ of a training set with the size l is labeled with a label $Y \in \{1, -1\}$ depending on which class the vector belongs to. Figure 4.3 illustrates this general idea.

The question is now, which of all possible hyperplanes is the best. It is obvious that the hyperplane with the largest margin to the closest vectors is the best. A hyperplane in a n dimensional space is represented by

$$\langle \mathbf{w}, \mathbf{x} \rangle + b = 0. \quad (4.8)$$

Once a hyperplane is known, a new vector is simply classified by taking the sign of $f(\mathbf{x})$ which is defined as

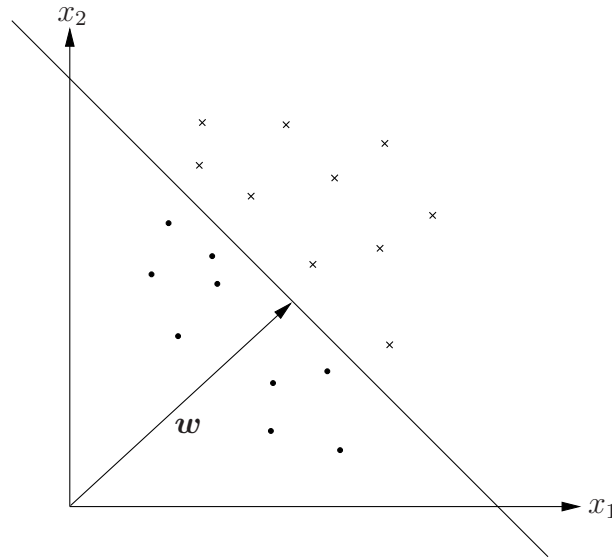


Figure 4.3: Concept of a hyperplane as a linear classifier

$$f(\mathbf{x}) = \frac{\langle \mathbf{w}, \mathbf{x} \rangle + b}{\|\mathbf{w}\|}. \quad (4.9)$$

According to Figure 4.4 the hyperplane with the largest margin γ needs to be found, leading to the following inequality:

$$\begin{aligned} \gamma &\leq y_i \frac{\langle \mathbf{w}, \mathbf{x}_i \rangle + b}{\|\mathbf{w}\|} \\ \gamma \|\mathbf{w}\| &\leq y_i (\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \quad \forall i. \end{aligned} \quad (4.10)$$

By setting $\|\mathbf{w}\| = 1/\gamma$ the equation 4.10 simplifies to

$$y_i (\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq 1 \quad \forall i. \quad (4.11)$$

Since not every real world problem is linear separable, so called slack variables ξ_i are introduced to allow classifying errors, changing equation 4.12 to

$$y_i (\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq 1 - \xi_i \quad \forall i. \quad (4.12)$$

Now the optimizing problem can be formulated as

$$\min_{\mathbf{w}, b, \xi} \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^l \xi_i \quad (4.13)$$

with the side conditions $y_i (\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq 1 - \xi_i$ and $\xi_i > 0 \quad \forall i$

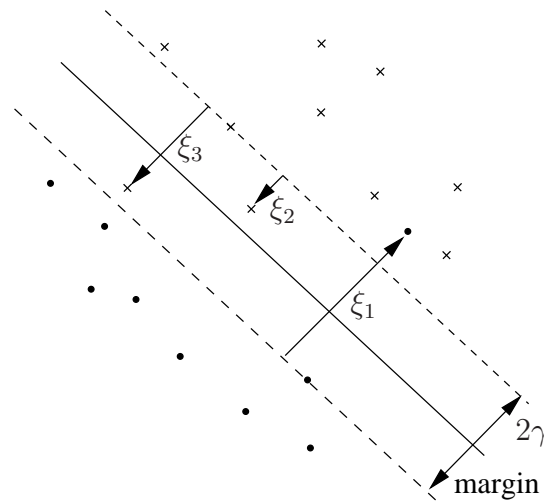


Figure 4.4: Maximizing the margin γ to find the best hyperplane. The slack variables ξ_i allow to define an hyperplane in the non separable case.

The constant C is used to weight the classifying errors and is not part of the optimizing problem. By using Lagrangian multipliers, the optimizing problem can be rewritten as

$$W(\alpha) = \sum_{i=1}^l \alpha_i - \frac{1}{2} \sum_{i=1}^l \sum_{j=1}^l y_i y_j \alpha_i \alpha_j \langle \mathbf{x}_i, \mathbf{x}_j \rangle. \quad (4.14)$$

The optimal α is now found by

$$\max_{\alpha} W(\alpha) \quad (4.15)$$

$$\text{with the side condition } 0 \leq \alpha_i \leq C \text{ and } \sum_{i=1}^l \alpha_i y_i = 0 \quad \forall i$$

This is a quadratic programming problem which can be solved by using numeric algorithms. After solving the problem, all vectors which are wrong classified or defining the margin γ are called support vectors, since only they are relevant for the best hyperplane.

4.3.3 Nonlinear Classifier

The SVM is expanded to a nonlinear classifier by using so called kernels. Kernels define an inner product of two vectors in a feature space F without transforming the single vectors first. In mathematical terms, if Φ is a transformation from a input space X to a feature space F a kernel is defined as

$$K(\mathbf{x}_i, \mathbf{x}_j) = \langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}_j) \rangle. \quad (4.16)$$

Several different kernels exist, the most popular are listed below.

- Polynomial kernel: $K(\mathbf{x}, \mathbf{z}) = (\langle \mathbf{x}, \mathbf{z} \rangle + a)^d$
- Radial Basis Function (RBF) kernel: $K(\mathbf{x}, \mathbf{z}) = \exp\left(-\frac{\|\mathbf{x}-\mathbf{z}\|^2}{2\sigma}\right)$
- Sigmoid kernel: $K(\mathbf{x}, \mathbf{z}) = \tanh(\kappa_1 \langle \mathbf{x}, \mathbf{z} \rangle + \kappa_2)$

To use the SVM as a nonlinear classifier, the inner product in equation 4.14 simply needs to be replaced by an kernel function.

4.3.4 Decision Function

Once the optimizing problem is solved, $f(\mathbf{x})$ in equation 4.9 can be rewritten as

$$f(\mathbf{x}) = \sum_{i=1}^l a_i^* y_i K(\mathbf{x}_i, \mathbf{x}) + b^*, \quad (4.17)$$

whereas the bias b^* is computed from \mathbf{a}^* . Now, a new vector is classified by simply evaluating the sign of $f(\mathbf{x})$.

4.3.5 Mapping geometrical distances to probabilities

Since the output of the SVM is only a signed distance to the hyperplane, a mapping to posterior probabilities is needed for later use with a GHMM and to optimize the prediction with the ROC-Curve (see 4.5). In [22] a way, of which a short overview is given here, is introduced to expand the SVM with a probabilistic output.

The Idea

As described before $f(\mathbf{x}) = \sum_{i=1}^l a_i^* y_i K(\mathbf{x}_i, \mathbf{x}) + b^*$ is the signed distance from the decision border for every vector. These signed distances can now be used to compute the probability density functions $p(f(\mathbf{x}_i)|y_i = \pm 1)$. With the Bayes' rule the posterior probabilities $P(y_i|f(\mathbf{x}_i))$ can be computed again.

The main idea is now based on the assumption that most of these posterior probabilities functions are sigmoid shaped. So, a function needs to be found, fitting best to a given training set $(f(\mathbf{x}_i), y_i)$.

Fitting the Sigmoid

Since it is assumed that the posterior probabilities functions are sigmoid shaped, it makes sense to use a sigmoid function to approximate them. The used function

$$P(y = 1|f(\mathbf{x}_i)) = p_i = \frac{1}{1 + \exp(Af(\mathbf{x}_i) + B)} \quad (4.18)$$

has two parameters A and B , which need to be found from a given training set $(f(\mathbf{x}_i), y_i)$. These two parameters are found by minimizing the cross-entropy function:

$$\min - \sum_i t_i \log(p_i) + (1 - p_i) \log(1 - p_i), \quad (4.19)$$

where t_i is defined as $\frac{y_i+1}{2}$.

Figure 4.5 shows the error curve of one summand of the cross-entropy function with the parameters $A = -1$ and $B = 0$ and the class $y_i = 1$. In this case the prediction is correct as long as the signed distance is positive. As one can see, in Figure 4.5 the error is small for all positive signed distances and big for negative signed distances.

To solve the two parameter minimizing problem, numerical solver can be used.

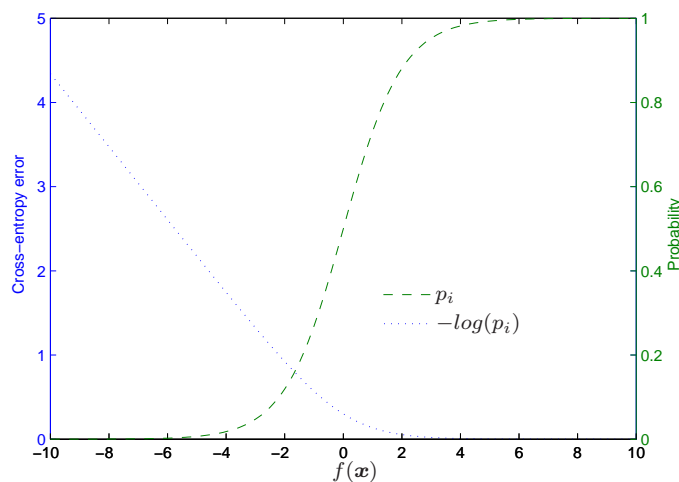


Figure 4.5: Cross-entropy error

To prevent over fitting, the output $(f(\mathbf{x}_i), y_i)$ of the SVM is not used directly to solve the minimizing problem. In fact a new unbiased data set is formed out of this output to find the best parameters. For details see [22].

4.4 Applying classifiers on coding region identifying methods

To predict whether a nucleotide belongs to a coding or non coding region the coding region identifying methods¹ are used as features for the classifier in the following way. A given DNA sequence is first analyzed by every single method and their output is stored. After all methods have analyzed the sequence, feature vectors for every nucleotide position are built by using the precomputed output of these methods, leading to vector of the following form:

¹The used methods are: Entropy Deficiency (Equation 3.19), IMM (Equation 3.24), SRMeasure (Equation 3.16), Fourier (Equation 3.3), Detrended Fluctuation Analysis (Equation 3.60), HexaCount (Equation 3.13) and Z-Curve (Equation 3.11))

[Entropy IMM SRMeasure Fourier DFA HexaCount $\underbrace{u_1 u_2 u_3 u_4 u_5 u_6 u_7 u_8 u_9 u_{10}}_{Z\text{-Curve}}$]

These feature vectors are then used by a classifier which predicts the posterior probability $P(\text{coding}|\text{nucleotide})$ for every nucleotide of the given sequence. At last the probabilistic output of the classifier is post processed by applying a certain threshold, leading to a binary (coding/non coding) output. The whole process is illustrated in Figure 4.6.

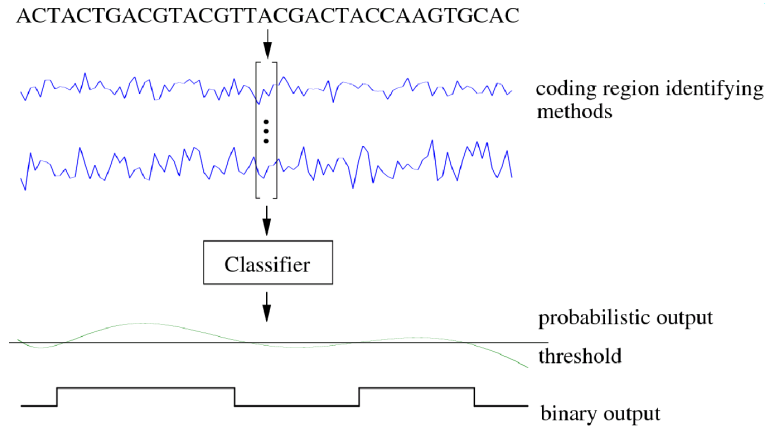


Figure 4.6: From a given DNA sequence to a binary prediction output by using classifiers.

4.5 Receiver Operation Characteristic Curve

4.5.1 Introduction

The Receiver Operation Characteristic Curve (ROC-Curve) is a useful method to find the best trade off between sensitivity and selectivity of a classifier. The ROC-Curve was actually invented during the Second World War to optimize the performance for identifying Japanese aircrafts on US radars. These days the ROC-Curve is very popular in medical research.

4.5.2 Computing the ROC-Curve

The probability output of every classifier can be post processed by defining a threshold. The threshold is for example set at 0.5, so all probabilities higher than 0.5 are interpreted belonging to class ω_1 and all probabilities below 0.5 are interpreted belonging to class ω_2 . This binary output can now be compared to the real data and making it possible to compute the following four statistical values:

- True Positive (TP): Number of values which are positive in prediction as in reality

- False Positive (FP): Number of values which are positive in prediction but not in reality
- True Negative (TN): Number of values which are negative in prediction as in reality
- False Negative (FN): Number of values which are negative in prediction but not in reality

		Reality	
		positive	negative
Prediction	positive	TP	FP
	negative	FN	TN

Figure 4.7: Measures of prediction accuracy

Once, this four values are computed the sensitivity is obtained by

$$S_n = \frac{TP}{TP + FN}, \quad (4.20)$$

which is the proportion of how many positive values are the same in reality and prediction. With $S_n = 1$ the prediction of true values corresponds perfectly to the reality.

The specificity

$$S_p = \frac{TN}{TN + FP} \quad (4.21)$$

denotes the proportion of how many negative values are the same in reality and prediction. However, since in gene prediction TN are much more common, due the amount of intergenic nucleotides, than FP, the specificity gets a noninformative high value. Hence, the specificity has been redefined by Burset and Guigo [16] as

$$S_p = \frac{TP}{TP + FP}. \quad (4.22)$$

Two other common measures can be obtained from the these four statistical values. One is the accuracy

$$Acc = \frac{TP + TN}{TP + FP + TN + FN}, \quad (4.23)$$

which is the number of all correct predicted values in relation to all predictions done. The other one is the correlation coefficient

$$CC = \frac{TP \cdot TN - FN \cdot FP}{\sqrt{(TP + FN)(TN + FP)(TP + FP)(TN + FN)}} \quad (4.24)$$

which is one when reality and prediction match perfect and zero when the prediction is completely random.

However, only the specificity and sensitivity are used to compute the ROC-Curve. By setting the threshold on different levels, the values of the specificity and sensitivity will change. These different values are now used to draw the ROC-Curve in the following way. The sensitivity values are used for the position on the ordinate, while $1 - S_p$ is used for the position on the abscissa. This leads to a point for every single threshold level.

Figure 4.8 shows an example for a ROC-Curve. If $S_n = 1$ and $S_p = 1$ the prediction would match perfectly with the reality. In real world, the ROC-Curve looks usually like a quarter of a ellipse. The best threshold is found by the corresponding point next to the upper left corner. If the prediction is completely at random, a straight line would result as ROC-Curve.

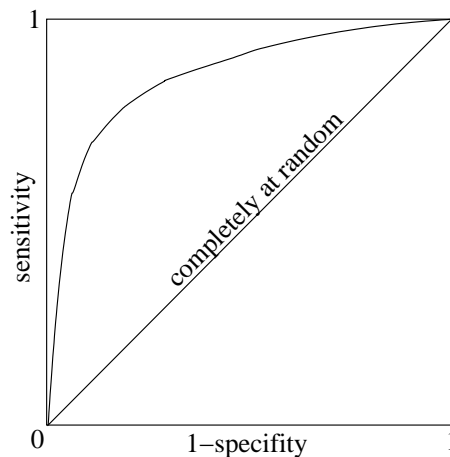


Figure 4.8: The ROC-Curve

If the prediction is exactly the opposite of the reality, the ROC-Curve is lower the straight line.

To compare different classifiers the Area Under the Curve (AUC) is often taken as measurand. If $f(x)$ describes the ROC-Curve, AUC is simply obtained by integrating $f(x)$. Since the ROC-Curve is usually computed depending on discrete threshold values, the trapezoid rule is used to compute the AUC:

$$AUC = \int_0^1 f(x)dx \approx \left[\frac{(x_{i+1} - x_i)}{2} \right] (y_{i+1} + y_i). \quad (4.25)$$

For a useful classifier the AUC has to always be in the interval]0.5, 1].

Chapter 5

GHMM

5.1 Introduction

The generalised hidden Markov model (GHMM) is used in gene prediction since the early 90s. It is used because its ability to model the grammatical information available in the DNA strands. For example that a gene has to start and to stop with defined short sequences of nucleotides called signals. Further it provides the opportunity to force the length distribution of a sequence emitted by a state to a given distribution.

In the next section the differences between the hidden Markov model and the GHMM are discussed. Followed by an overview of the GHMM of our implementation.

5.2 Differences between GHMM and HMM

The hidden Markov model (HMM) is already very well known in speech recognition and therefore it is well documented in text books concerning speech recognition for example in [15]. The GHMM is as the name says a generalisation of the HMM but with this generalisation more information can be used by the model.

In contrast to the HMM the GHMM can not change between the states at every nucleotide position. Which is also the case in genomic data. In genomic data the translation initiation start site, where a gene starts, needs the nucleotide sequence "ATG" to be present. This kind of information can be mapped onto the GHMM but not onto the HMM.

In the plant *Arabidopsis Thaliana* the length distribution of single exon genes has a shape according to Figure 5.3. This shape holds some important information. So it is almost impossible to get a single exon with less than 100 nucleotides. Whereas the HMM does not provide any possibility to map this information onto the model the GHMM does.

All these extensions makes a GHMM better fitting the problem of gene prediction than HMM. Therefore this model is often used for problems like that.

5.3 Overview of the implemented GHMM

The genetic signals represent positions where it is possible to change from one state to another. According to [4] they are:

- Translation Initiation Start Site TIS nucleotide sequence "atg"
- Donor Splice Site DSS nucleotide sequence "gt"
- Acceptor Splice Site ASS nucleotide sequence "ag"
- Translation Stop Site TSS nucleotide sequence "tga","tag" or "taa"

Because both strands have to be analysed simultaneously it is necessary to look for eight different signals. The underlying models are called signal sensors which are represented in Figure 5.1 by a (\diamond).

The sequence in between two signals needs to be rated to find out which sequence of signals and contents is the most likely. That the statistical properties in coding and non coding differ was one of the first discovery in bioinformatics. Different models are available to catch these statistical differences. The interpolated Markov model (IMM) introduced by [24] is a very famous one and also used in our GHMM. Only three different content sensors have been used because they need a lot of data to get well trained. They are:

- intergenic
- intron
- exon

The exon content sensor has been used in all content sensors with coding content. The content sensors are represented in Figure 5.1 by a (\circ).

In Chapter 6 other approaches for the content sensors have been tried.

As mentioned above the length of the different content hold information as well. The length distribution are shown in Figure 5.3. They are also used in the GHMM. As there is no difference between the forward and reverse stand the following length distributions are used:

- inter genic length
- intron length
- single exon length
- initial exon length
- internal exon length
- terminal exon length

If there is more than one exit out of a state, the GHMM needs to know the probability of each exit, so called transition probabilities. The following ones are needed by the model:

- Probability that the gene is single exon P_{s1}
- Probability that the gene has more internal exons P_{sn}
- Probability that after a gene on the forward strand follows a gene on the forward strand $P(f|f)$
- Probability that after a gene on the reverse strand follows a gene on the reverse strand $P(r|r)$

The complete model with all possible transitions is shown in Figure 5.1. Note that there are just a few possible transitions. In a way this leads to a path where the model has to go through. This path comes from the genetic grammar and is very useful in gene prediction.

The dotted lines are not transitions. They show a possible way. The sequence between the two signals is always rated by the intergenic content sensor.

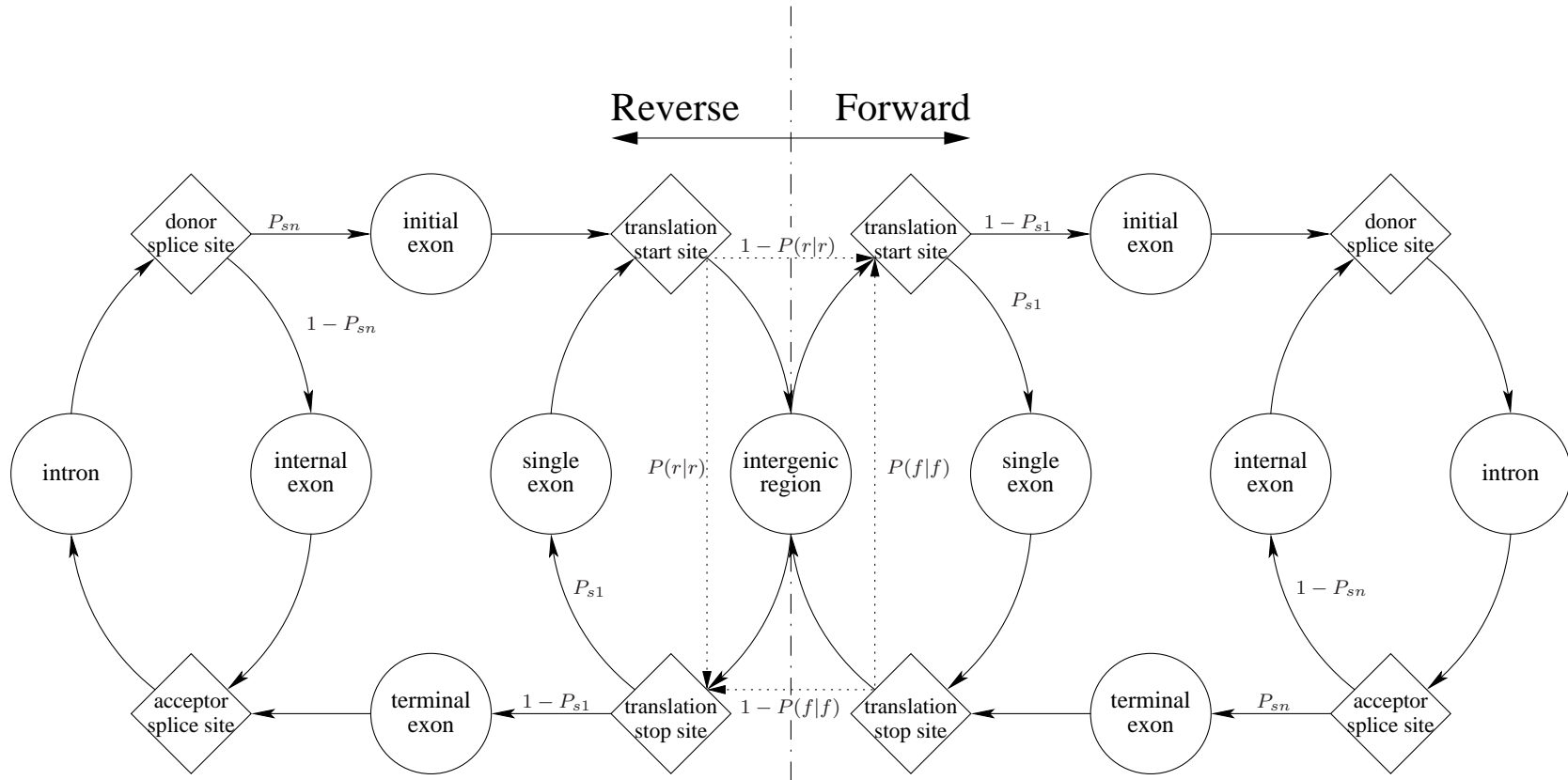


Figure 5.1: The structure of the implemented GHMM (○) represents a content sensor (◇) represents a signal sensor

5.4 Signal Sensors

The genetic grammar says a defined signal needs to be present to change between two states. On the nucleotide sequence from 1 to 100000 on chromosome IV of Arabidopsis Thaliana 3787 start signals can be found. But only 28 genes, correct start signals, are in this sequence. Therefore these 3787 start signals have to be rated in order to find the correct ones.

Studies such as [6] found out that there is information located around the signal which can be used to determine the probability that this signal is a true signal. To rate the signal two different models are available the weight array matrix (WAM) introduced by [17] and the maximum dependency decomposition (MDD) introduced by [6]. The model difference in how they analyse the influence from one nucleotide to one another. The WAM only takes the first order dependency whereas the MDD is able to analyse all the dependencies.

The χ^2 dependency matrix is needed to choose the appropriate model for a specific signal type. The χ^2 matrix calculates the influence from nucleotide at position x on a nucleotide at position y . The tables below show the chi-squared matrix. The columns contain the nucleotide positions where the influence comes from, the rows contain on which nucleotide positions the influence goes to. I is the position of the signal. Because the genomic grammar defines the nucleotides at the position I until $I + 2$ for the translation start site it is not of interest and therefore these positions are masked out. The influence from a nucleotide on itself does not give any information and is masked out as well. Then for better reading the values are scaled by the maximum value.

	$I - 6$	$I - 5$	$I - 4$	$I - 3$	$I - 2$	$I - 1$	$I + 3$	$I + 4$	$I + 5$
$I - 6$	-	0.518	0.294	0.237	0.179	0.032	0.005	0.011	0.010
$I - 5$	0.863	-	0.332	0.285	0.085	0.017	0.053	0.046	0.027
$I - 4$	0.252	0.498	-	0.507	0.267	0.126	0.046	0.021	0.083
$I - 3$	0.361	0.280	1.000	-	0.506	0.108	0.069	0.025	0.064
$I - 2$	0.209	0.192	0.244	0.818	-	0.599	0.044	0.104	0.080
$I - 1$	0.022	0.040	0.105	0.265	0.156	-	0.147	0.173	0.134
$I + 3$	0.009	0.097	0.096	0.093	0.049	0.161	-	0.332	0.178
$I + 4$	0.007	0.077	0.023	0.037	0.177	0.067	0.322	-	0.097
$I + 5$	0.001	0.024	0.043	0.059	0.008	0.071	0.117	0.241	-

Table 5.1: χ^2 dependency matrix for the translation start site scaled by 202.8

The two highest values in both tables have been marked **red**. In Table 5.1 the highest dependencies are located on the secondary diagonals. the WAM analyses exactly this dependencies. But the donor splice site χ^2 matrix does not look like translation start site χ^2 matrix. Here the position $I - 1$ has the highest influence on position $I + 4$ which is far away from the secondary diagonals. The WAM can not analyse that at all whereas the MDD can.

The other signals such as translation stop site have approximately the same χ^2 matrix as the translation start site. Therefore the WAM is used for all signals except the donor splice signals in both direction. There the MDD is used.

	$I - 3$	$I - 2$	$I - 1$	$I + 2$	$I + 3$	$I + 4$	$I + 5$
$I - 3$	-	0.156	0.060	0.089	0.018	0.101	0.011
$I - 2$	0.708	-	0.345	0.284	0.195	0.501	0.087
$I - 1$	0.038	0.546	-	0.320	0.461	0.987	0.164
$I + 2$	0.087	0.186	0.173	-	0.029	0.270	0.126
$I + 3$	0.028	0.198	0.465	0.435	-	0.127	0.015
$I + 4$	0.162	0.506	1.000	0.266	0.250	-	0.454
$I + 5$	0.019	0.082	0.144	0.133	0.008	0.400	-

Table 5.2: χ^2 dependency matrix for the donor splice site scaled by 1664

The other question is how far away from the signal itself the signal dependent nucleotides are because that has a direct influence on the length of the signal sensors. All the numbers are taken from the diploma thesis [8]. The model and the corresponding length are shown in Table 5.3. For a better understanding the signals are shown in Figure 5.2 with their length and offset.

Signal type	Model type	length of the signal
Translation start site	WAM	12
Donor splice site	MDM	9
Acceptor splice site	WAM	23
Translation stop site	WAM	12

Table 5.3: Signal types their models and lengths

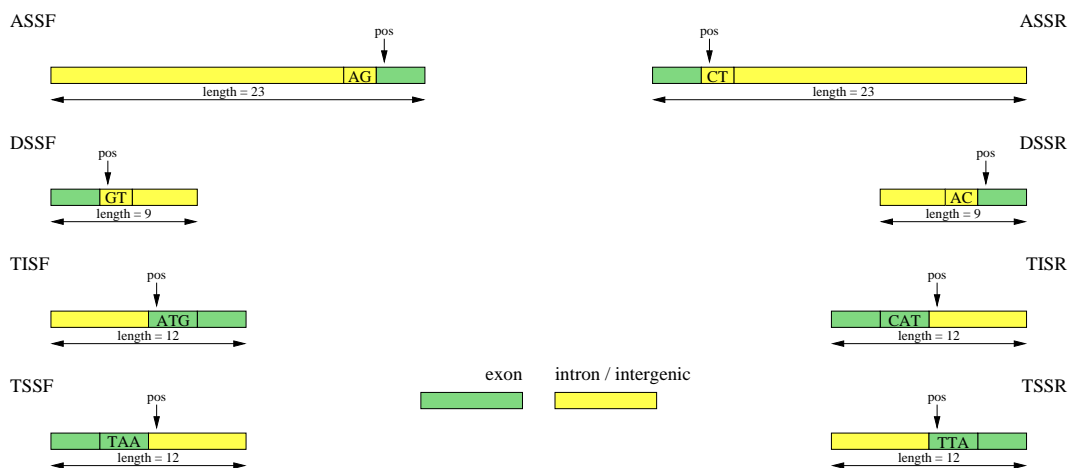


Figure 5.2: Signal lengths and offsets

5.5 Content Sensor

The content sensor rates the nucleotides in between the signals. It should be able to clearly differentiate between the different contents. The implemented GHMM has three different content sensors but the sub models are all the same. The sub model is the interpolated Markov model (IMM), introduced by [6].

Before the IMM was introduced normal Markov models (MM) were been used. But they had the problem that they needed a large amount of training data to provide accurate predictions. The IMM is mainly a compilation of different MM with different orders. Increasing the order gives a higher specialisation but also a higher amount of training data is needed. Which develops a problem with a tenth order MM. To achieve useful data in the MM there should be 100 sequences in the training set would be needed for every possibility. This would lead to a sequence of length 104857600. But there are probably some sequences which occur a lot of times. Thats where the IMM came into play. It collects different MMs. For the prediction it chooses the highest Markov model which has a certain amount of occurrence in the test set. The paper [6] also introduce a kind of similarity check with the χ^2 function. This is not implemented in the GHMM.

The implemented GHMM uses basically the IMM as content sensor. Three different content sensors are taken. The intergenic, intron and exon. The exon content sensor has been more specialised because there is information stored in the phase. The information stored in the phase comes from the genetic grammar. The transcription process always transcribes three nucleotides into one amino acid. The nucleotide probability depends on its position in the amino acid. This position in a amino acid is called phase. For the GHMM it is always important to know in which phase the next nucleotide is. That is the phase problem. It will not be discussed here because it has been already in [20].

There are nine IMMs used in the implemented GHMM. They are used in different content sensors as shown in Table 5.4.

content	number of IMMs	direction dependent	phase dependent
intergenic	1	no	no
intron	2	yes	no
exon	6	yes	yes

Table 5.4: IMMs used

The six IMMs for the exon content are very good and important for the GHMM. If the GHMM takes the wrong phase the prediction results gets worse rapidly. An example: The IMM has been trained well. So the IMM for phase one will never have a "TGA" as the nucleotides at position -3 to -1, because this is a stop codon. The exon is ended if this codon appears in phase. In the training set this will never happen. The IMM goes probably down to the second or even first order Markov model. In an other phase, there is the very well trained and specialized IMM which would use a sixth order Markov model. The sixth order would better fit than the one second or even first order. The result is that the prediction gets worse. Phase shift was one of the major problem in the implementation of the GHMM.

5.6 Length Distribution

The ability of the GHMM to model the length of a content is very interesting because the length highly differ from content to content. In Figure 5.3 the length distributions of the Arabidopsis Thalaian chromosome I for the six different contents are shown. The differences are obvious.

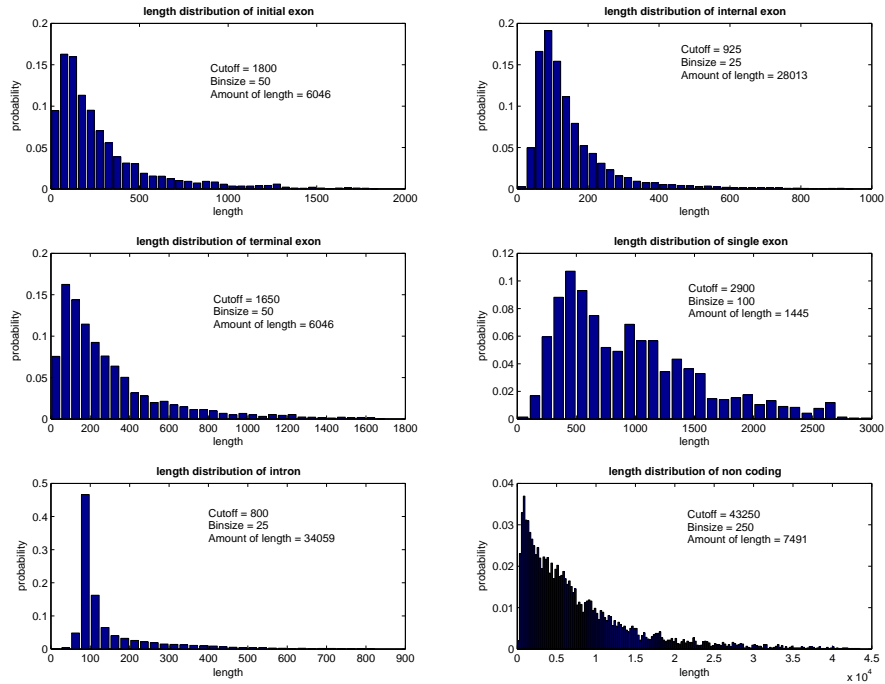


Figure 5.3: The length distribution of different contents

There is almost no difference between the forward and reverse strand. Therefore the same distributions are used for the forward and reverse strand.

But there can be a difference between the types of the genome. Once a vertebrate test set had to be analysed. The used bin size and cut off length are shown in Table 5.5. All the values are the same except for the intergenic region. This comes from cutting out the genes by a human. The normal statistics is vanished. Therefore for test sets the intergenic region length probability has been neglected.

	arabidopsis thaliana		vertebrate test set	
	bin size	cut off length	bin size	cut off length
intergenic	250	43250	25	800
intron	25	800	25	800
single exon	100	2900	100	2900
initial exon	50	1800	50	1800
internal exon	25	925	25	925
terminal exon	50	1650	50	1650

Table 5.5: Bin size and cut off length of the different genomes

The numbers in Table 5.5 are chosen that way that the distribution keeps their distinctive shape but there should not be a zero in a bin. The cut off length is set to the length where just a few are over this length.

5.7 The Algorithm itself

The GHMM algorithm consists of three parts, pre-analyse, analyse and backtrack. In the pre-analyse part all the necessary information are pre-calculated to fasten the GHMM algorithm. In the analyse part the proper GHMM algorithm analyses the sequence. Then at the end the backtrack part chooses the most likely sequence which results to the prediction. Note that at the end of the analyse part there is no prediction available.

5.7.1 Pre-analyse

In the analyse phase it is important to know where the possible signals are. Therefore the signal sensors have to search the signals in the sequence which should be analysed before the analysis starts.

To make the GHMM algorithm fast the whole sequence is rated by the different content sensors before the algorithm starts. This is done because otherwise a lot of nucleotides are rated several times as showed in Figure 5.4. A lot of nucleotides which have to be rated are overlapping. There is the possibility to pre-calculate the probability per nucleotide. In this way the algorithm only has to multiply the nucleotide probabilities together and the result is available.

To make it even faster all probabilities are in logarithmic scale and summed up. Then the algorithm takes the difference between these two positions which saves computation time but on the other hand it also needs more memory.

5.7.2 Analyse

In the analyse part the algorithm moves from signal to signal. At every signal all the possible predecessors are taken into consideration and then the best possible one is chosen which means the one with the highest probability. This is similar to the Viterby algorithm.

The accumulated probability for the actual signal is the accumulated probability of the last signal added the probability (in log space) of the content and the length. Out of this the one with the highest probability is chosen. Then the appropriate signal probability of the actual signal is added and stored as the accumulated probability for the actual signal. This is shown in Equation 5.1. The $P(DSSF|seq)$ represents the accumulated probability. A graphical example is given in Figure 5.4.

$$P(DSSF|seq) = \max(P(TSSFx|seq) * content_prob * length_prob) * P(DSSF) \quad (5.1)$$

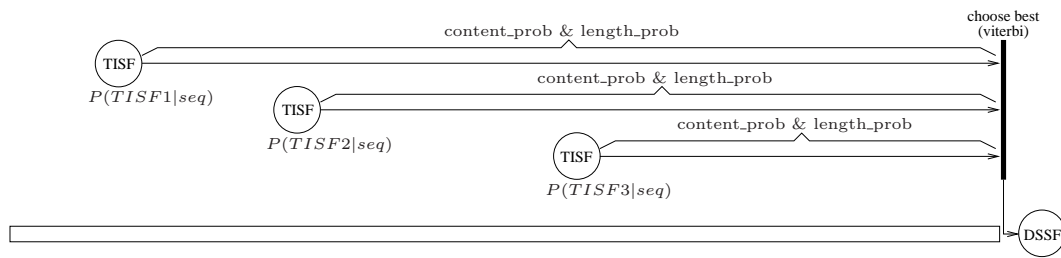


Figure 5.4: A DSSF signal example

5.7.3 Backtrack

Out of the last signal of each type the one with the highest probability is chosen. Then always the best predecessor is taken until the first signal is reached. In this way the most likely sequence for the analysed nucleotide sequence can be found.

Chapter 6

Combination of the GHMM and the classifier

6.1 Introduction

The classifier output almost never hits a correct signal because it predicts nucleotide by nucleotide and therefore does not look at the genetic grammar. The idea of combining the classifier output with the GHMM is to use the genetic grammar information for a classifier. Two approaches have been proved. Firstly using the probability output of the classifier as content sensor and secondly a combination of IMM and classifier as content sensor.

6.2 The classifier output as content sensor

One way of combining is to replace the content sensors of the GHMM with the classifier output. As can be seen in Figure 5.1 there are a lot of different content sensors represented by \circ . But the output of the classifier is only one probability. This probability is used in some way in the content sensors. It is for sure that there is not as much information in the classifier as in the different IMMs.

For prediction only the best classifier the SVM is used. The SVM output can be seen in Figure 6.1 and 6.2. Interestingly the output is sometimes 0 or 1 which means the SVM is for 100% sure that the actual nucleotide belongs to class coding or non coding. That is not good for the GHMM because it is forced to be in a defined state. If there is no signal available during the change, the whole prediction brakes-down. Therefore the output of the SVM needs to be changed in order to give the GHMM more options. The SVM output has therefore been changed according to Equation 6.1. The values have been set according to Table 6.1. Note that these values have not been optimised. But this is not necessary because the results are basically wrong what will be discussed later.

$$cp = cp * (high_border - low_border) + low_border \quad (6.1)$$

The only data available from the classifier is how likely this nucleotide is coding or not. Therefore this probability cp is used for the exon content sensors. And $1 - cp$ is used for the intergenic and intron content sensors.

$$\begin{array}{l|l} \text{high_border} & 0.8 \\ \text{low_border} & 0.2 \end{array}$$

Table 6.1: Changes to the SVM output for the GHMM

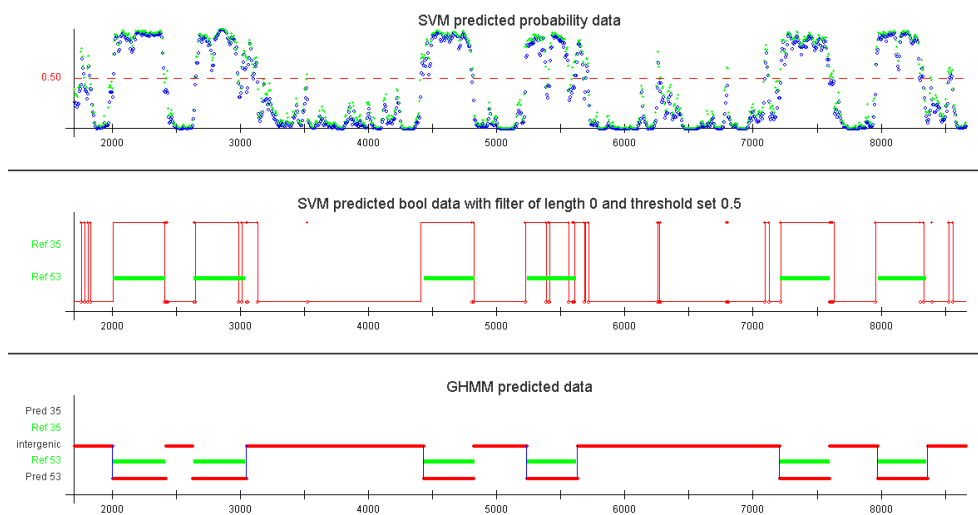


Figure 6.1: The classifier output of the SVM and the prediction of the GHMM with the classifier as content sensors for test set number 7

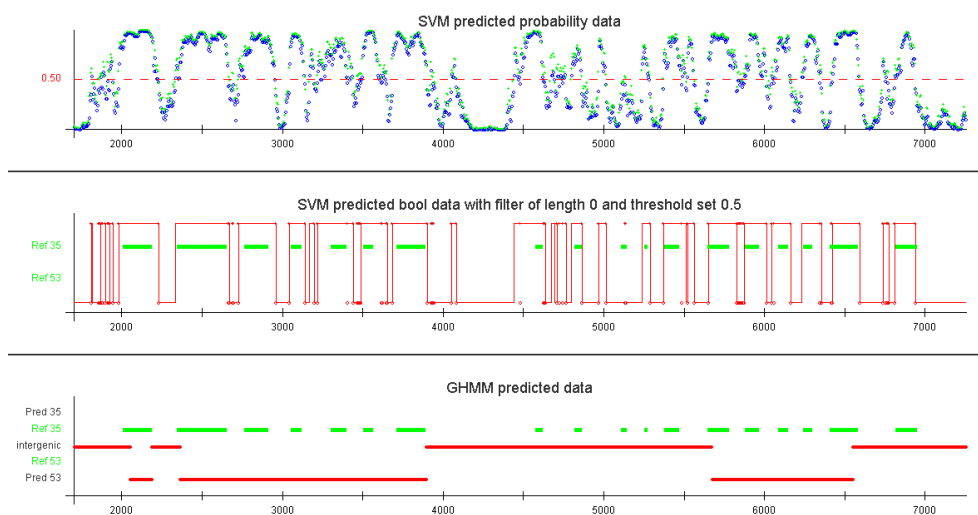


Figure 6.2: The classifier output of the SVM and the prediction of the GHMM with the classifier as content sensors for test set number 21

In Figure 6.1 it can be seen that the GHMM output is better if the classifier output is near to 1 or 0 and the genes and intergenic regions are long. In Figure 6.2 the genes are not really long, the intergenic regions and introns are short and the classifier is not close to 0 or 1. Therefore the prediction is over all wrong. The GHMM is not able to differentiate between the two directions.

What it does in this configuration a signal is searched which is near the changes in the SVM. It does not have any information about the direction. If the algorithm finds a better signal on the wrong direction the algorithm will choose this signal during the state change of the classifier. Therefore the prediction changes to the wrong side and the short gaps are filled up. That is not a good combination and has therefore not been followed up.

6.3 A combination of IMM and classifier

In the section before it can be seen that the classifier output does not hold much of data which means that it is not trained on all the different data as the IMM for example. Another approach is needed. The idea was to combine the classifier data with the data from the IMM which already gives good results as a content sensors. The idea was to combine the IMMs which can differentiate between phases and directions with a classifier output.

The combination has been done by taking the average of the probability of the classifier and the IMM. The classifier probability also have been changed according to Equation 6.1. Where the *low_border* and *high_border* represent a degree of how much the classifier data can be thrust.

The optimal values for the classifier are found by a 2-dimensional grid search over the first 500000 nucleotides of Arabidopsis Thaliana chromosome IV. The results are shown in Figure 6.3. The best results are located by the values shown in Table 6.2. There is another position (0.6 0.4) which leads to slightly more correct exons but also to much more missed exons.

For all the exon content sensors Equation 6.2 and for the others Equation 6.3 has been used. The new nucleotide probability is denoted by np the IMM probability by ip and the changed classifier probability by cp .

$$np = (ip + cp)/2 \quad (6.2)$$

$$np = (ip + (1 - cp))/2 \quad (6.3)$$

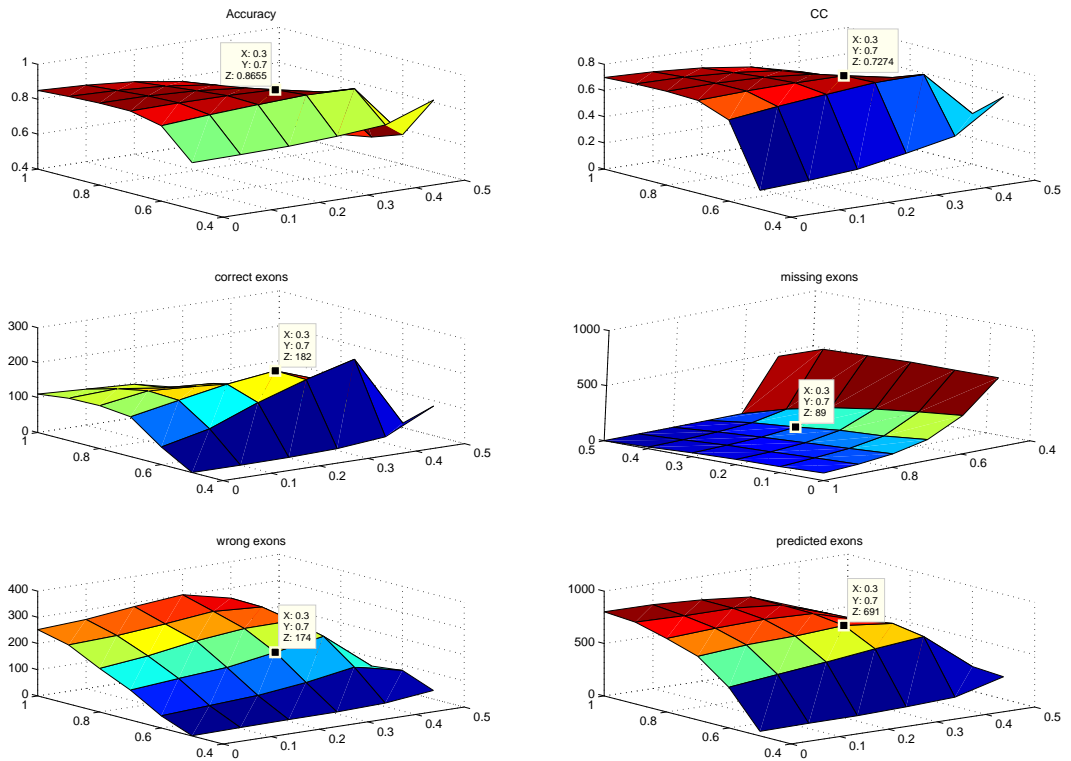


Figure 6.3: Parameter evaluation over the first 500000 nucleotides of Arabidopsis Thaliana chromosome IV

	optimal values	second values
<i>high_border</i>	0.7	0.6
<i>low_border</i>	0.3	0.4
Correct exons	182	217
Missing exons	57	118

Table 6.2: Changes to the SVM output for the GHMM

The results of this combination are documented in Section 7.3.

Chapter 7

Results

7.1 Used data sets

To evaluate the accuracy of different gene prediction programs a common test set is needed. In this thesis the test set of Pavy *et al.* [19] from 1999 is used. The test set contains 74 sequences, each of them contains at least one gene leading to 168 genes which are divided into 1028 exons. The genes are taken from all five chromosomes of *Arabidopsis Thaliana* and to make sure that none of the gene prediction programs were trained with genes included in this data set, only genes which had not been used in other *Arabidopsis Thaliana* data sets up to the year 1999 were used. This test set is called *AraSet*.

Every single sequence contains always 2000 nucleotides before the first gene starts and another 2000 nucleotides after the last gene stops. However, to minimize the risk that these 2000 nucleotides contain fragments of other genes, the prediction starts always 300 nucleotides before the first gene starts and ends 300 nucleotides after the last gene stops as shown in Figure 7.1

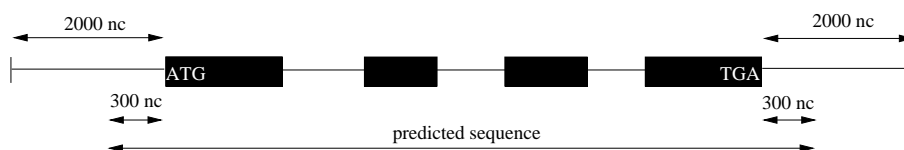


Figure 7.1: Structure of the sequences in AraSet

Since the training can not be done on the test set, an other data set is needed for training. For this purpose the set *AraClean*¹ of Korning *et al.* [21] has been used. *AraClean* contains 144 sequences each of them containing exactly one gene, which is always located on the forward strand. All in all the set contains 878 exons.

¹ *AraClean* and *AraSet* are available from
<http://www.charite.de/molbiol/bioinf/bioinf/Datenbanken/at.html>

7.2 Performance of the different classifiers and coding region identifying methods

7.2.1 Training

To train the three classifiers SVM, k -Nearest Neighbor and Parzen Window, the set *AraClean* has been analyzed with the coding region identifying methods Fourier, Entropy Deficiency, SRMeasure, HexaCount, Z-Curve, IMM and DFA. With the output of these methods three training sets were built to train the different classifiers. Since the amount of time needed to perform a prediction gets really high for the Parzen window and k -Nearest Neighbors classifiers when the training set is too large, these two set were chosen to have a smaller size than the SVM trainings set. Table 7.1 shows the number of vectors in every used training set.

TrainingSet	Number of	
	coding vectors	non coding vectors
SVM	1549	1549
Parzen Window	360	360
k -Nearest Neighbors	360	360

Table 7.1: Sizes of the classifier training sets

Based on these three training sets six-fold-cross-validations have been performed to get the parameters of each classifier in respect of maximizing the correlation coefficient. In Table 7.2 the found parameters are shown for each of these classifiers. The used threshold for the probabilistic output was 0.5. The kernel used for the SVM is a Radial Basin Function.

Classifier	Parameters
SVM with RBF Kernel	$C = 128$ $\gamma = 0.0167$
k -Nearest Neighbor	$k = 33$
Parzen Window	$h = 0.4$

Table 7.2: Parameters for each classifier

The number of support vectors of the SVM is 986. This is almost a third of the training set, which is an indicator that many of the training vectors point in the same direction, unheeded to which region they actually belong to. In the worst case almost every third training vector was misclassified.

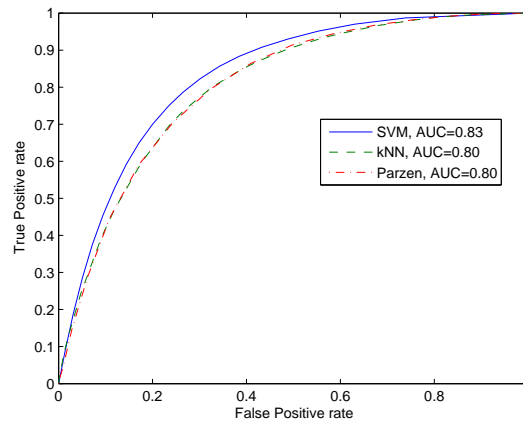


Figure 7.2: ROC–Curve of the different classifiers. All features have been used for input

7.2.2 Discussion

All the results shown here are based on *AraSet*. Figure 7.2 shows the ROC–Curves of the classifier by using all features as described before. As one can see the area under the curve of the SVM is with 0.83 slightly bigger than of the other two classifiers ($AUC = 0.80$). But still, by setting the threshold at its optimal level more than 8% of the nucleotides predicted as coding are non coding in reality, while 11% of the nucleotides predicted as non coding are coding.

To evaluate which of the coding region identifying methods generates the most useful information, experiments as follows have been done. The SVM has been trained by using always only one coding region identifying method as input. After the training was done, the *AraClean* test set was used to evaluate the performance of this specific coding region method. To do that the threshold was set to the optimal level by using the ROC–Curve and then the binary output was analyzed. The results for every method are shown in Table 7.3.

The output of the SVM by using only the DFA as input is by far the worst. As the very low correlation coefficient $CC = 0.104$ shows, the prediction is almost at random. This performance of the DFA would probably be better if the set contained more longer exons.

The very new SRMeasure method which takes advantage of the Fourier phase performs better than the very well known Fourier method which prediction is based on the magnitude. Although the difference of the correlation coefficient CC between these two methods is not enormous.

Interestingly, the best working method is the 20 year old HexaCount method. Even the Z–Curve is clearly outperformed by HexaCount, which is very impressive since the Z–Curve has an output with the dimensionality ten (see 3.5) instead of one like all the other methods. By using all methods together it can be seen that the performance is clearly

	TP	TN	FP	FN	<i>Acc</i>	<i>Sp</i>	<i>Sn</i>	<i>CC</i>
DFA	106969	209963	151602	95759	0.562	0.528	0.414	0.104
Entropy	80102	292894	68671	122626	0.661	0.395	0.538	0.223
Fourier	107650	249634	111931	95078	0.633	0.531	0.490	0.218
IMM	168084	138530	223035	34644	0.543	0.829	0.430	0.221
SRMeasure	127070	238290	123275	75658	0.647	0.627	0.508	0.276
Z-Curve	137454	281356	80209	65274	0.742	0.678	0.631	0.450
HexaCount	138266	303431	58134	64462	0.783	0.682	0.704	0.525
All	141964	314340	47225	60764	0.809	0.700	0.750	0.579

Table 7.3: Comparison of the performance between the different coding identifying methods.

limited by HexaCount. It seems as HexaCount already contains most of the information other methods also contain. However, there is still an improvement of the correlation coefficient *CC* by 10% if all methods are used as input features for the SVM instead of only using the HexaCount method.

7.3 Performance of the combination of GHMM and SVM

7.3.1 Introduction

Based on the results of the different classifiers the combination has only be done for the SVM and the GHMM since the SVM was evaluated to be the best classifier. To see if there is an improvement from the raw GHMM to the combination of the GHMM and the SVM all test have been done for the GHMM itself as well as for the combination. Following to that, the test have also been done with other well known gene prediction programs, namely *GenScan* and *GenMark.hmm*.

*GenScan*² was developed by C. Burge [6] in 1997. The main difference between the GHMM developed in this thesis and *GenScan* is the ability of *GenScan* to recognize promoters. It uses a content sensor to detect them as can be seen in Figure 7.3. The appearance of promoters is especially known in vertebrate DNA but not in plant DNA.

These days *GenMark.hmm*³ is known for being one of the best gene prediction programs available. It was developed by A. Lomsadze *et al.* [1] and the last version is from 2005. *GenMark.hmm*.

The GHMM used by *GenMark.hmm* does not differ too much from the GHMM used in this thesis. But the way the GHMM is used to predict is very different. While in this thesis

² *GenScan* is available from: <http://genes.mit.edu/GENSCAN.html>

³ *GenMark.hmm* is available from: <http://exon.gatech.edu/GeneMark/eukhmm.cgi>

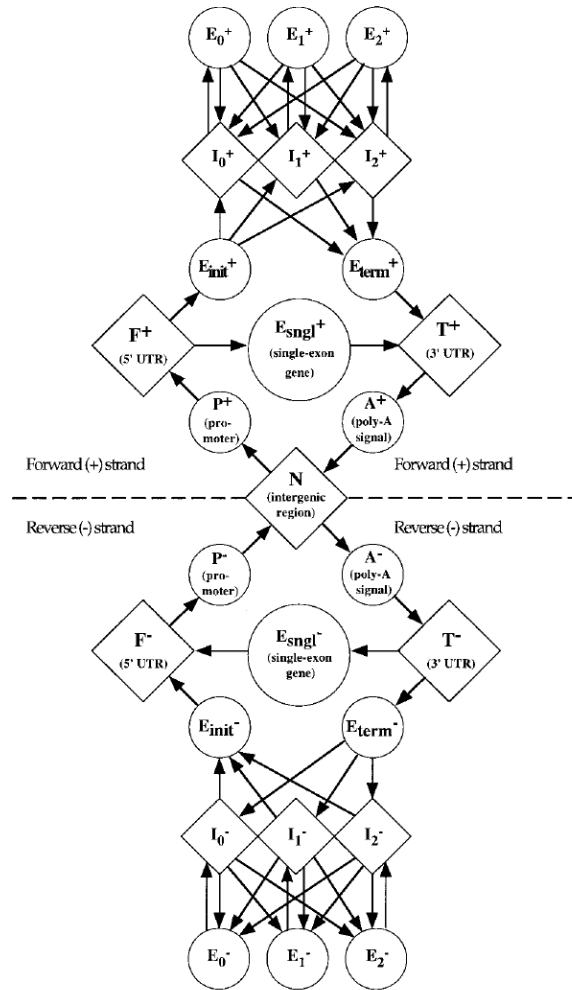


Figure 7.3: GHMM used by *GenScan*. Special sensors are used to identify promoters.

the parameters are learned a single time and then used to predict all the given sequences, *GenMark.hmm* uses a self-training algorithm. Every time a new sequence is analyzed the parameters of the GHMM are optimized until the best fitting ones are found. This unsupervised training qualifies *GenMark.hmm* even to predict genes in genomes where very little known data is available yet.

Two different tests have been generated to measure the performance of the combination of the GHMM and the SVM. First the first million of chromosome IV of *Arabidopsis Thaliana* was predicted. For the second test the *AraSet* was used. In [19] the prediction accuracy of different gene prediction programs was evaluated with this data set. Because both of the programs *GenScan* and *GenMark.hmm* have been optimized since they were used in that publication the results presented here differ a little bit.

To measure the prediction accuracy the two level measurement as introduced by Burset

and Guigo [16] has been used. The nucleotide level measure is described in section 4.5.2. On the exon level four different measures are defined as follows.

$$\text{Sensitivity } S_n = \frac{\text{number of correct exons}}{\text{number of actual exons}} \quad (7.1)$$

$$\text{Specificity } S_p = \frac{\text{number of correct exons}}{\text{number of predicted exons}} \quad (7.2)$$

$$\text{missing exons ME} = \frac{\text{number of missing exons}}{\text{number of actual exons}} \quad (7.3)$$

$$\text{wrong exons WE} = \frac{\text{number of wrong exons}}{\text{number of predicted exons}} \quad (7.4)$$

A predicted exon is only considered as being correct if both splice site match with the reality. If a predicted exon overlaps with the reality it is whether considered as wrong nor as missing. The missing and wrong exons are defined according to Figure 7.4.

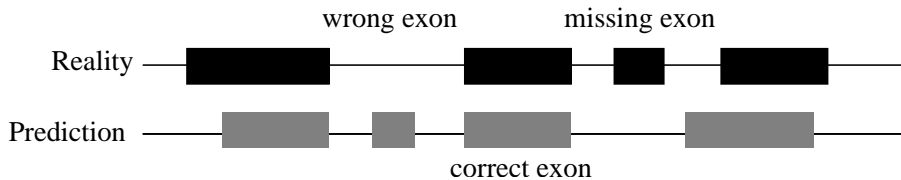


Figure 7.4: Exon level measures.

7.3.2 Training

The training was done twice, first a training set out of chromosome I of *Arabidopsis Thaliana* was formed. This training set was then used to train the SVM. The GHMM was also trained by using the chromosome I. After both the SVM and GHMM were trained, the first million nucleotides of chromosome IV were used to predict the genes.

To predict the genes of the *AraSet*, the training set for the SVM was formed by using *AraClean*. Table 7.4 shows the sizes for both of the training sets. Since *AraClean* contains only genes on the forward strand, it can not be used to train the GHMM. Therefore the Chromosome III of *Arabidopsis Thaliana* was used to train the GHMM for predicting the *AraSet*⁴. Since all genes in the *AraSet* are cut out, there is no length distribution of the intergenic region. The intergenic region has always the same length of 300 nucleotides. Therefore the intergenic length model of the GHMM was deactivated for predicting genes in *AraSet*.

⁴*AraSet* also contains 2 genes (Sequence number 41 and 42) out of chromosome III of *Arabidopsis Thaliana*. Therefore this two sequences were not used for prediction.

TrainingSet	Number of	
	coding vectors	non coding vectors
Chromosome I	1549	1549
<i>AraClean</i>	2425	2425

Table 7.4: Sizes of the training sets used to train the SVM.

7.3.3 Discussion

Table 7.5 and 7.6 show the results of the *AraSet* prediction and the prediction of the first million nucleotides of chromosome IV respectively. As one can see *GeneMark.hmm* outperforms clearly all other predictors. With a correlation coefficient *CC* over 0.89 on the nucleotide level and an average of 74.9% of the sensitivity and specificity on the exon level accuracy is very high.

	Nucleotide Level				Exon Level				
	Sn	Sp	Acc	CC	Sn	Sp	$\frac{(Sn+Sp)}{2}$	ME	WE
GeneMark.hmm	0.987	0.938	0.972	0.940	0.906	0.851	0.878	0.017	0.073
GenScan	0.882	0.916	0.929	0.844	0.646	0.707	0.677	0.166	0.087
GHMM	0.925	0.757	0.867	0.733	0.642	0.249	0.445	0.001	0.542
GHMM+SVM	0.887	0.832	0.895	0.777	0.326	0.357	0.341	0.170	0.147

Table 7.5: The accuracy of the GHMM and SVM combination compared to other gene prediction programs. The prediction is based on the *AraSet*.

	Nucleotide Level				Exon Level				
	Sn	Sp	Acc	CC	Sn	Sp	$\frac{(Sn+Sp)}{2}$	ME	WE
GeneMark.hmm	0.96	0.89	0.94	0.89	0.784	0.715	0.749	0.049	0.132
GenScan	0.76	0.85	0.87	0.72	0.642	0.800	0.721	0.262	0.171
GHMM	0.91	0.77	0.88	0.75	0.642	0.280	0.461	0.014	0.495
GHMM+SVM	0.91	0.82	0.90	0.79	0.550	0.387	0.468	0.140	0.110

Table 7.6: The accuracy of the GHMM and SVM combination compared to other gene prediction programs. The prediction is based on the first million nucleotides of chromosome IV of *Arabidopsis Thaliana*.

The GHMM itself suffers from a very high false positive rate, independent on which data is used. In both cases almost all of the potential signals are predicted, which results in a very low missing exon rate, but also in a very high wrong exon rate. By using the combination of the GHMM and SVM the false positives are down-trodden. This leads on the one hand to a much lower wrong exon rate (about 4 times lower), but on the other hand the missing exon rate is also increased. The main problem of the combination is the

accuracy on the exon level. Often the splice sites are not hit exactly. While this does not have a big influence on the nucleotide level accuracy it does have on the exon level.

With a closer look it can be seen, that often short exons are not recognized by the SVM. It seems like the resolution is not good enough. Probably the accuracy on the exon level could be increased by using smaller window sizes for the coding identifying methods.

Interestingly the combination of the GHMM and SVM works much better on the first million nucleotides of chromosome IV. Even *GenScan* is outperformed clearly on the nucleotide level, but also the missing exon (0.262) rate of *GenScan* is almost twice as high as of the combination (0.140) while the wrong exon rate of the combination is also lower.

There could be two reason for this. First of all the chances are high that the training set, made of chromosome I has almost the same statistical properties as the first million of nucleotides of chromosome IV. That would explain the rather good result of the combination. On the other hand it could be that the cut out genes of *AraSet* contain promoters, whereas the most of the genes in the first million nucleotides of *Arabidopsis Thaliana* do not. In that case the promoter sensor of *GenScan* would not work, which may lead to worse results. Nevertheless it could not be found any information about promoters in the *AraSet* nor in *Arabidopsis Thaliana* in general.

Chapter 8

Conclusion

8.1 Review

Since the amount of sequenced DNA data is growing very fast these days, the use of computerised gene finding methods has become indispensable. However, these methods are still not highly satisfying.

This thesis has mainly focused on two different techniques, the coding region identifying methods and the GHMM.

The coding region identifying methods are used to locate weak regions where genes could be. It has been shown that the prediction accuracy is increased by using different coding identifying methods as input for classifiers such as k -nearest neighbor, Parzen window and SVM instead of only using the methods for itself. Although it has also been shown that increasing the prediction accuracy is clearly limited by the best coding identifying method.

The implemented GHMM uses eight different signal sensors and eleven content sensors. However, the GHMM itself suffers from a high false positive rate. To lower this high positive rate a combination with the probabilistic output of a classifier has been introduced. This probabilistic output is used as additional information for the content sensors. It has been shown that this combination leads to a lower false positive rate, but also some splice site signals get lost. Although the obtained results with the combination on a widely used test set show some improvement, they can not keep up with topical gene prediction programs.

8.2 Outlook

To further improve the accuracy of the combination SVM and GHMM, several possibilities are supposable. Other coding identifying methods could be used for the classifier. However, it is questionable if there are better methods available these days. In a next step the window lengths of the used methods could be adjusted to get a higher resolution of the classifier output. By using more content sensors, for example a promoter sensor, to rarefy the GHMM, the accuracy could be further increased.

Chapter 9

Software

9.1 Introduction

The computation of all coding region identification methods needs an enormous amount of computing power and several gigabytes of harddisk space are needed only to store the results of one chromosome. Therefore the Matlab implementation did not seem to be the best choice. As a result an own software has been developed. The software is divided into two parts, a server and a client application.

9.2 Server

The server is used to analyze entire chromosomes or only short parts of it. It is written in C/C++ to make it as fast as possible, the speed benefit compared to Matlab is about factor 60. The whole code consisted about 12000 code lines. It has been designed to be compilable on every popular platform such as Unix or Windows by using Qt. Qt is a platform independent framework of the company Trolltech¹. It is freely available for non commercial use. For the SVM the LIBSVM of [7] is used. This is a very fast implementation of a SVM, which source code is available for non commercial use.

The server is operated by the use of a configuration file. This file specifies which data should be precomputed for later use. Dependent on the size of the data which has to be precomputed the precomputing can take up to several days time.

Once some data is precomputed the server opens a port and waits for incoming client commands to send the wanted data.

Following the used configuration file is explained in detail.

¹<http://www.trolltech.com/>

9.3 Servers configuration file

The configuration file is built of sections and subsection written as `[section name]`. This sections or subsections then contain commands for the program. This commands are according to the syntax `command = value`.

9.3.1 [ALL]

In the first section `[ALL]` contains information valid for the whole program. In this sections are paths defined where the data are located.

- `DATAPATH` where to store the feature data
- `PREDICTIONPATH` where to store the classifier and GHMM results
- `GENBANKPATH` where the gene bank files has been stored used for reference

Additional to this path commands the command to specify which genome type should be taken for training has been added. The `GENOMETYPE` command in this section sets the training genome type for all following sections.

9.3.2 [FEATURE]

The second section `[FEATURE]` contains all necessary information for the precomputing of the features. The first command `ENABLE` enables the whole section. Possible values are `YES` or `NO`. The `NO` disables the whole section and no feature calculation will be done.

The first subsection `[TRAIN]` contains the train information for the feature section. Some features need a gene bank file to successfully train on. The gene bank file is specified by the gene bank path defined in the `ALL` section added the sub directory specified with the `GENOMETYPE` command from the `ALL` section as well. In this path the gene bank file with the name composed as `"SUBDIR + NUMBER + .gb"` needs to be present.

Following the feature specified subsections has to be configured. For every feature the same commands are available. Every feature can be switched on or off with the `ENABLE` command in the feature subsection. Then for every feature a `WindowSize`, `StepSize` and `FilterLength` command allows to set these values as desired.

At the end of the `FEATURE` section there is the `[PREDICT]` subsection. Here the commands for the prediction is located. The `GENOMETYPE` command specifies the sub folder from the `DATA` command in the `[ALL]` section. The `SUBDIR` command is one half of the the next sub folder. The other half of the sub folder name is the `NUMBER` command. `NUMBER` commands can be multiply defined in this section then all the defined numbers are calculated. The `FROM` and `TO` commands set the prediction to a defined range. If the value of one of these commands is set to `-1` the whole sequence will be predicted. The nucleotide sequence comes from a gene bank file which is located at the `GENBANKPATH` in the sub folder of the `GENOMETYPE` value of this section and the file is named according to `"SUBDIR + NUMBER + .gb"`.

9.3.3 [CLASSIFIER]

The next section [CLASSIFIER] contains all necessary information for the classifiers. Here as well all the classifier computation can be disabled by setting the `ENABLE` command to `NO`.

The train section for the classifier section contains the information where the data to train the classifier is available. Unlike the section before the paths of the precomputed features are needed.

The next subsection [USEDFEATURES] specifies which feature should be used in the classifier. A single feature can be switched on or off by set the value to `YES` or `NO`. This makes it easy to switch a feature of to check how important it is for a good prediction.

In the classifier sub sections every classifier can be switched on or off for the prediction. There are also classifier specific commands available.

- `NNCount` the neighbors count used by the kNN algorithm to make the prediction
- `h_PARAMETER` how the different gauss distributions are shaped for the Parzen algorithm
- `CParameter` and `GammaParameter` used by the SVM described in section 4.3

For all the classifiers a separate training set will be made. The size of the training set are specified with the three commands `TRAIN_FROM` , `TRAIN_TO` and `TRAIN_STEP_SIZE`. The training set size should be 600 for kNN, 600 for Parzen Window and 3000 for the SVM. If they are larger the prediction takes a lot of time. The size of the training set can be seen in the output of the program after it has been made.

The prediction sub section works almost as in the previous section. Except that the result files are stored in the `PREDICTIONPATH` and not in the `DATAPATH`. But also more than one `NUMBER` can be used to predict more than one sequence of a test set.

9.3.4 [GHMM]

In the [GHMM] section all information for the different GHMMs are provided. This section can also be enabled or disabled by the `ENABLE` command. If reference data in form of a gene bank file is available the reference can be written to a file for compare issues. As above this task can be switched on with `YES` or off with `NO`.

The `TRAIN` subsection defines on what data the sub models of the GHMMs should be trained.

There are three different GHMMs available:

- [IMM_ALONE] The content sensors are normal IMMs
- [SVM_ALONE] The content sensors is the classifier data as described in section 6.2
- [IMMSVM_COMBINED] The IMM and classifier combination as described in section 6.3

Every GHMM can be switched off. For the GHMMs with a classifier the classifier needs to be specified. The path is the same as the path of the results is. The command `ClassifierMax` and `ClassifierMin` set the values for Equation 6.1.

The prediction section has the same meaning as it has in the `[CLASSIFIER]` section discussed above.

9.3.5 `[SERVER]`

`[SERVER]` is the section where the server can be switched on or off. And with the command `log` a log file name can be specified which then is in the same directory as the program itself.

9.3.6 `[MAKEFASTA]`

At the end there is another section `[MAKEFASTA]` which is used to generate fasta files out of gene bank files. These files are used to paste into the web interface of the well known gene predictors.

9.4 Client

The client is used to show all the precomputed data of the server. By a reason of platform independence the client is written in Java with a total of about 3000 code lines. The client provides a GUI as shown in Figure 9.1. The GUI is mainly divided into four parts:

1. Coding identification methods
In this part the values of the different coding identification methods are shown. The methods are chosen by the drop down menu on the left side.
2. Probabilistic classifier output
In this part the probabilistic output of the different classifier is shown. By clicking on the radio buttons on the left the respective classifier output is shown.
3. Binary classifier output
By clicking on the “ROC-Curve” button the optimal threshold for the binary output of the chosen classifier can be set. The binary output is then shown in this part of the client window while the statistics are shown on the right side.
4. GHMM prediction output
In this part the output of different gene prediction program as well as the output of the combination presented in this thesis can be shown. After choosing an entry of the drop down menu the statistics will be updated.



Figure 9.1: GUI of the client

9.5 Communication between Clients and Server

The communication between clients and server is based on a protocol. This protocol consists of a header, which is sent every single time when a client and the server are exchanging data. The header has the following format:

```
Header
{
    long command;
    long startPos;
    long stopPos;
    long pixelW;
    long pixelH;
    long genomeType;
    long chromosome;
    long dataType;
    long ROCFilter;
    long ROCThreshold;
    long payloadLength;
}
```

All the variables are of the type `long` even if their values margin would allow a shorter data type. This makes sure that the header has always the same size, independent of the used compiler. Since in C/C++ no rule exists how to arrange different data types in a `struct`, different header sizes would result.

If now the client wants to have data from the server, it first has to set the header and then send it to the Server.

- `command`: specifies which data should be displayed on the client. {SVM, KNN, PARZEN, FOURIER, SRMEASURE, HEXACOUNT, ZCURVE, DFA, ENTROPY, IMM, GHMM_IMM, GHMM_SVM, GHMM_IMM_SVM, GHMM_GENSCAN}
- `startPos`: start position of the data corresponding to `chromosome`
- `stopPos`: stop position of the data corresponding to `chromosome`
- `pixelW`: the width in pixels of the client to draw the data. This is used by the server to precompute the data. If there are more data points available than pixels, then only selective data points are send back to the client to save bandwidth. If there are more data points than pixels, the server sets `pixelW=1` and 0 otherwise.
- `pixelH`: the height of the client in pixels to draw the data. Currently not used by the server.
- `genomeType`: the type of the genome of which the data is wanted. {ARA, ARASET, ARACLEAN}
- `chromosome`: the number of the chromosome of which the data is wanted.

- `dataType`: what group of data is wanted. {PREDICTION, FEATURE, REFERENCE, ROC, BINARY, GHMM, STATISTICSBINARY, STATISTICSGHMM}
- `ROCFilter`: length of the filter for the binary output. Used by the ROC-Curve.
- `ROCThreshold`: Threshold to generate the binary output. Used by the ROC-Curve.
- `payloadLength`: The size of the payload. Is set by the server.

If now for example the client wants to display (number of available pixels are 500) the SVM prediction of the chromosome IV from position 1000 to 2000 the header has to be set as follows:

```
Header.command=SVM;
Header.startPos=1000;
Header.stopPos=2000;
Header.pixelW=500;
Header.genomeType=ARA;
Header.chromosome=4;
Header.dataType=PREDICTION;
```

Figure 9.2 shows how the communication between a client and the server works. First of all the client tries to connect to the server by sending a header containing the request for some data. After the server has received this header, it will send back a new header to the client and the payload containing the requested data. As soon as the client has all data it sends a header with the command DISCONNECT to the server, which tells the server to quit the connection.

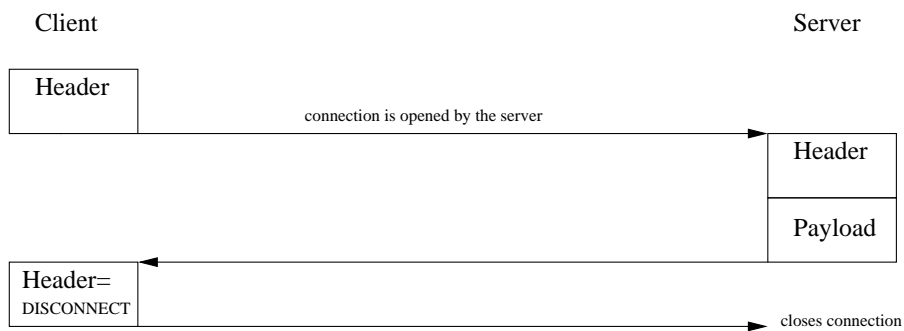


Figure 9.2: The handling of a request from a client to the server.

9.6 Gene program parser

Since every gene program uses its own output format for its prediction a parser has been written to convert all this different outputs to the format used in this thesis. The output of the following gene prediction programs can be converted:

- GenScan
- GeneMark.hmm
- Grail
- FGENEH

To program is executed by the command `genePredictionParser input output`.

Appendix A

Class diagrams of the Server application

In this appendix class diagrams of the server application are shown.

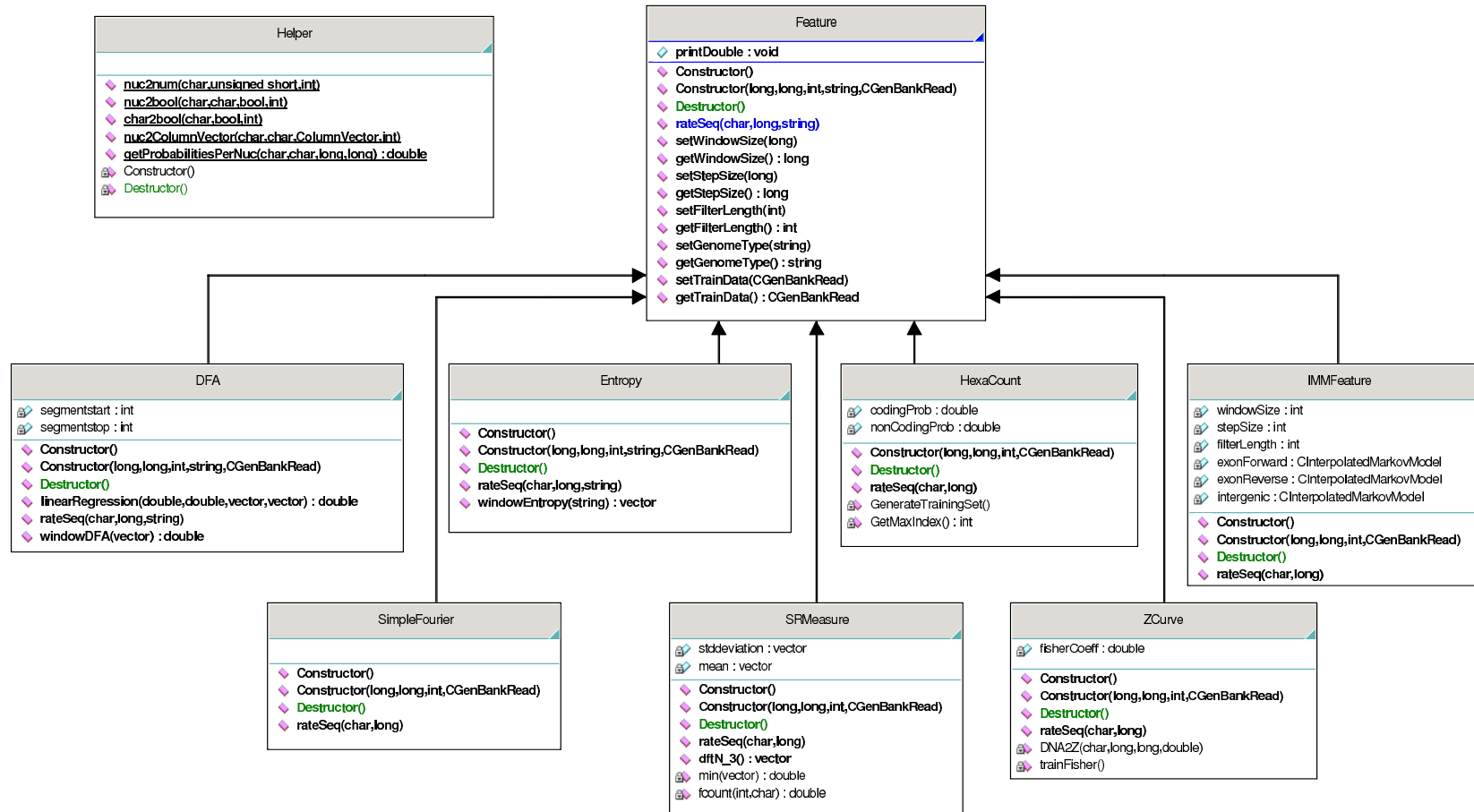


Figure A.1: Class diagram of the feature part

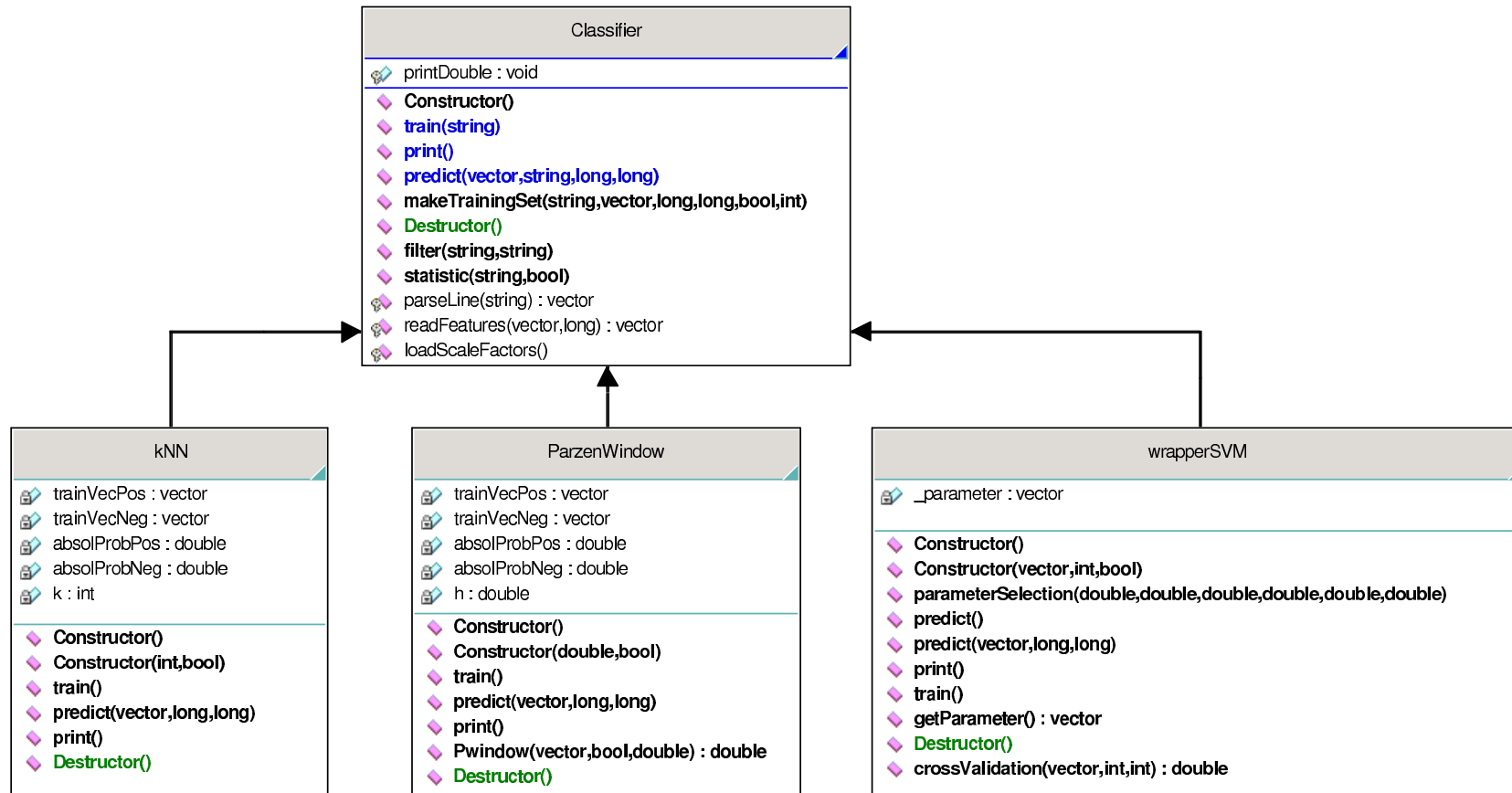


Figure A.2: Class diagram

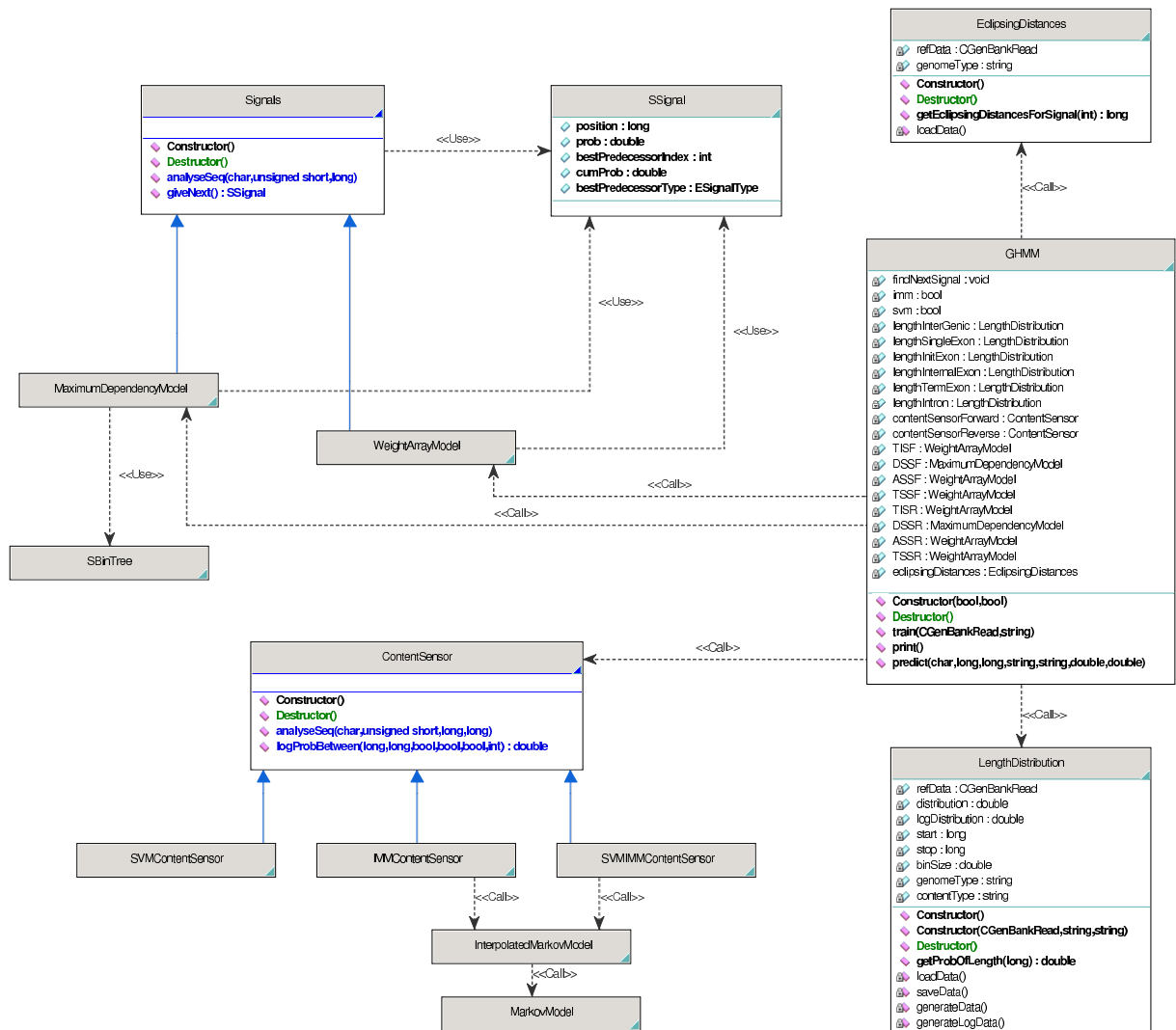


Figure A.3: Class diagram of the GHMM part

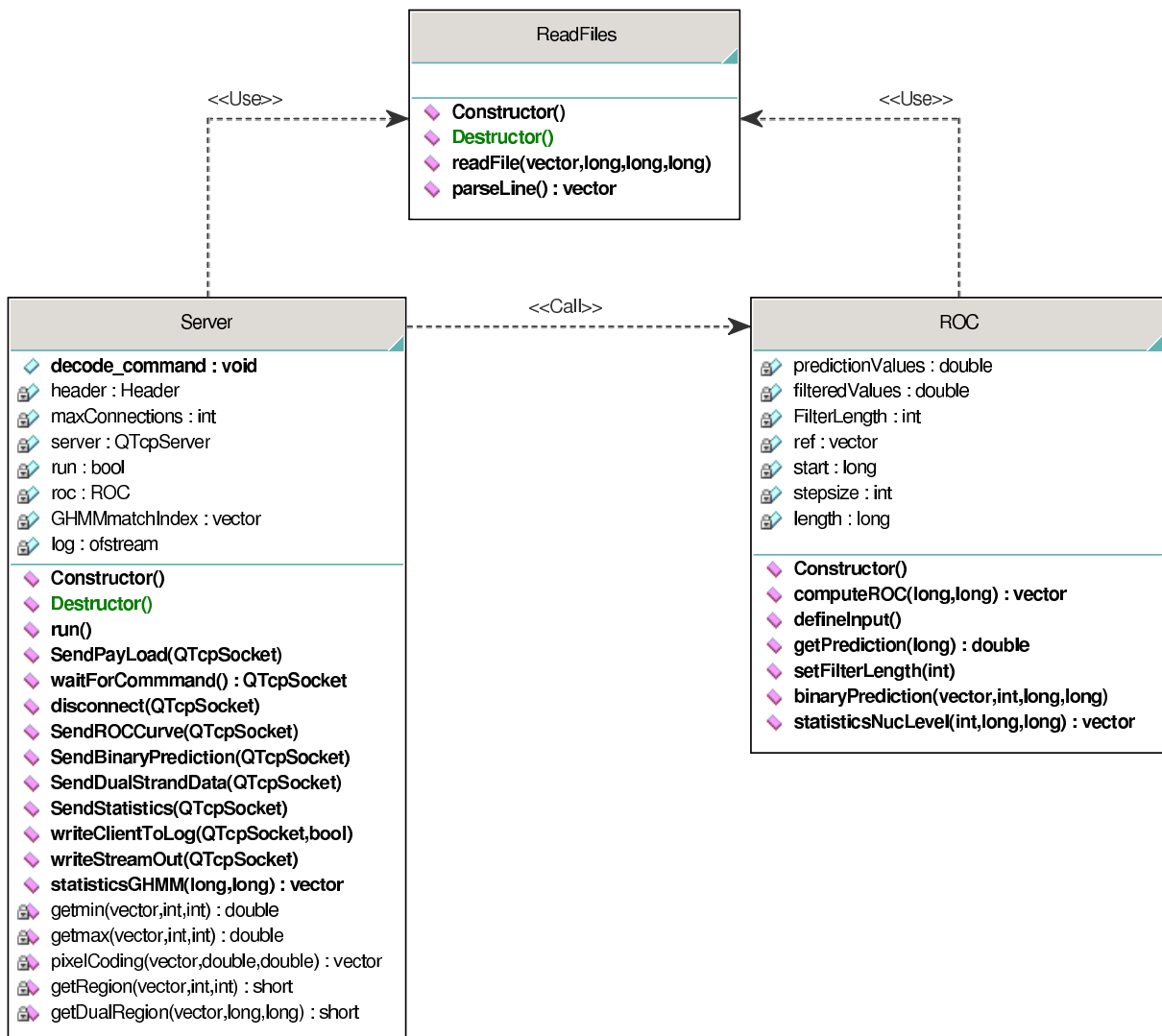


Figure A.4: Class diagram of the Server part

Appendix B

Class diagrams of the GUI application

In this appendix class diagrams of the GUI application are shown.

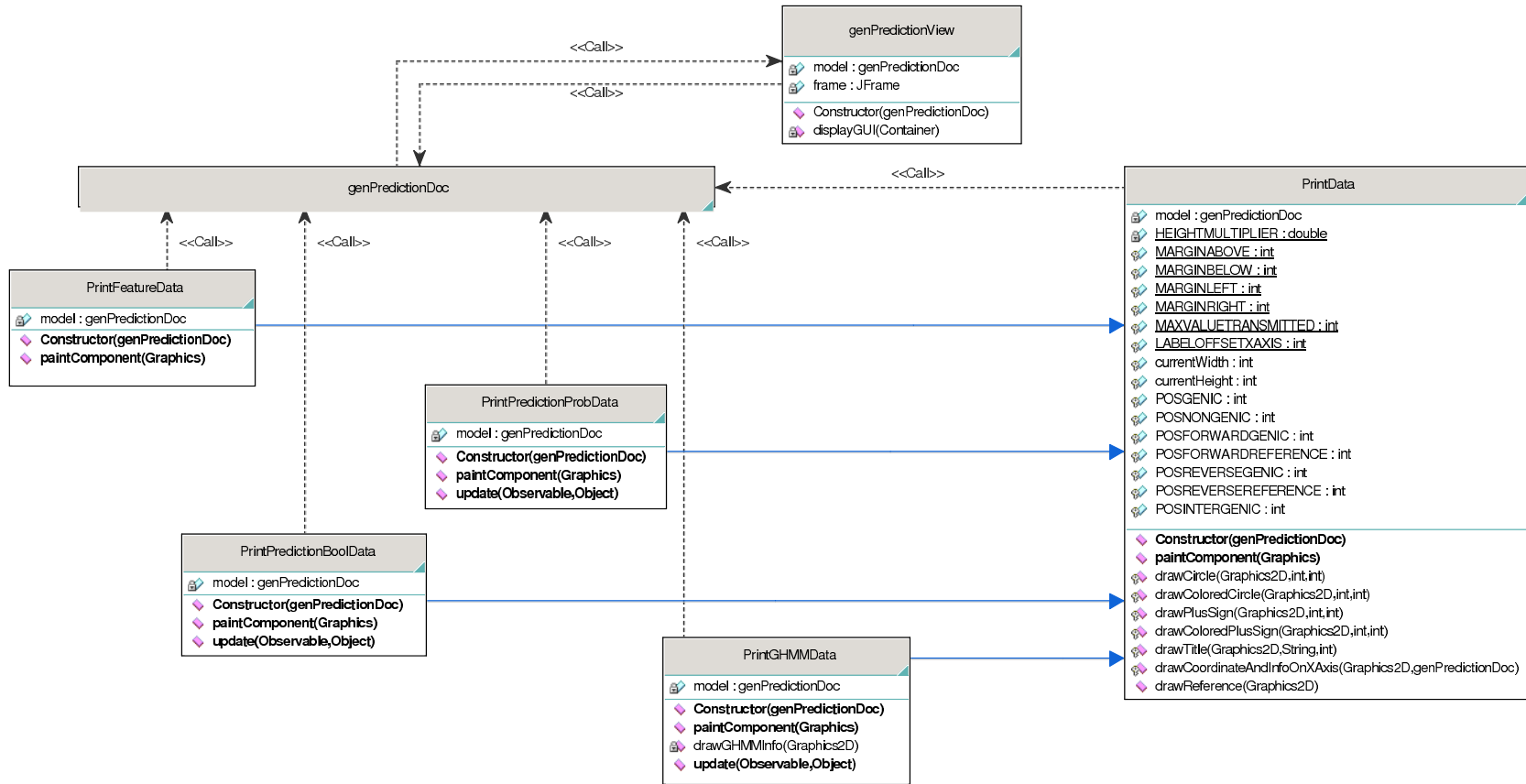


Figure B.1: Class diagram of the view part

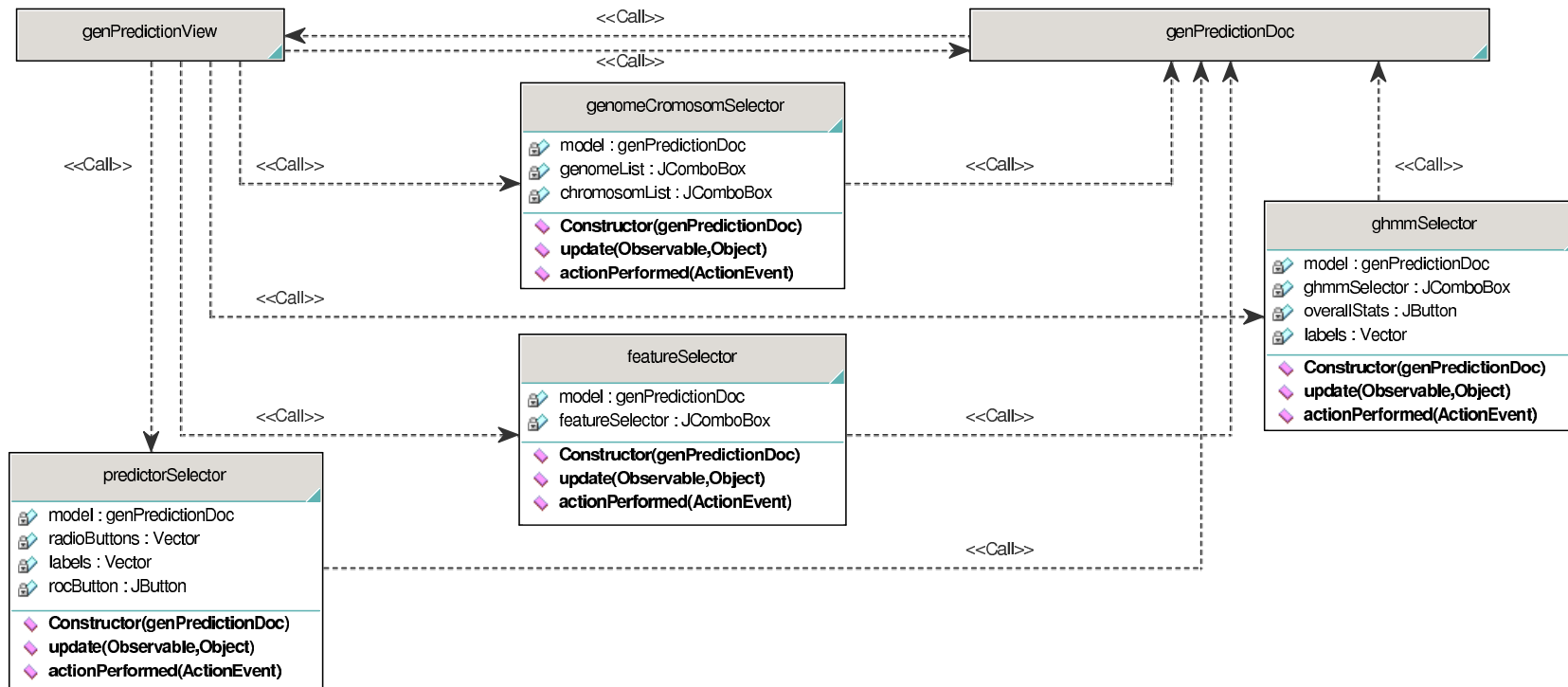


Figure B.2: Class diagram of the selectors part

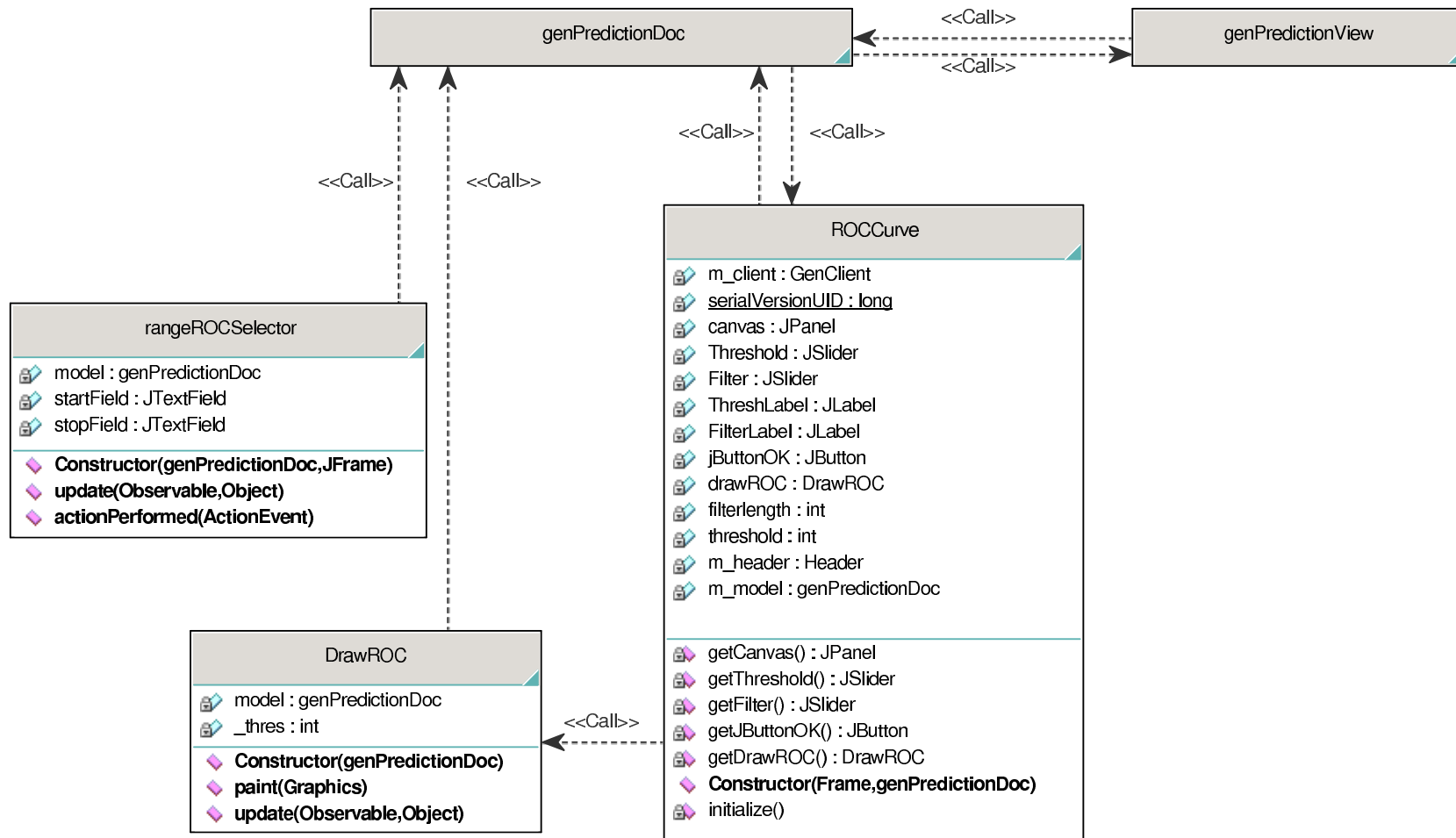


Figure B.3: Class diagram of the ROC part

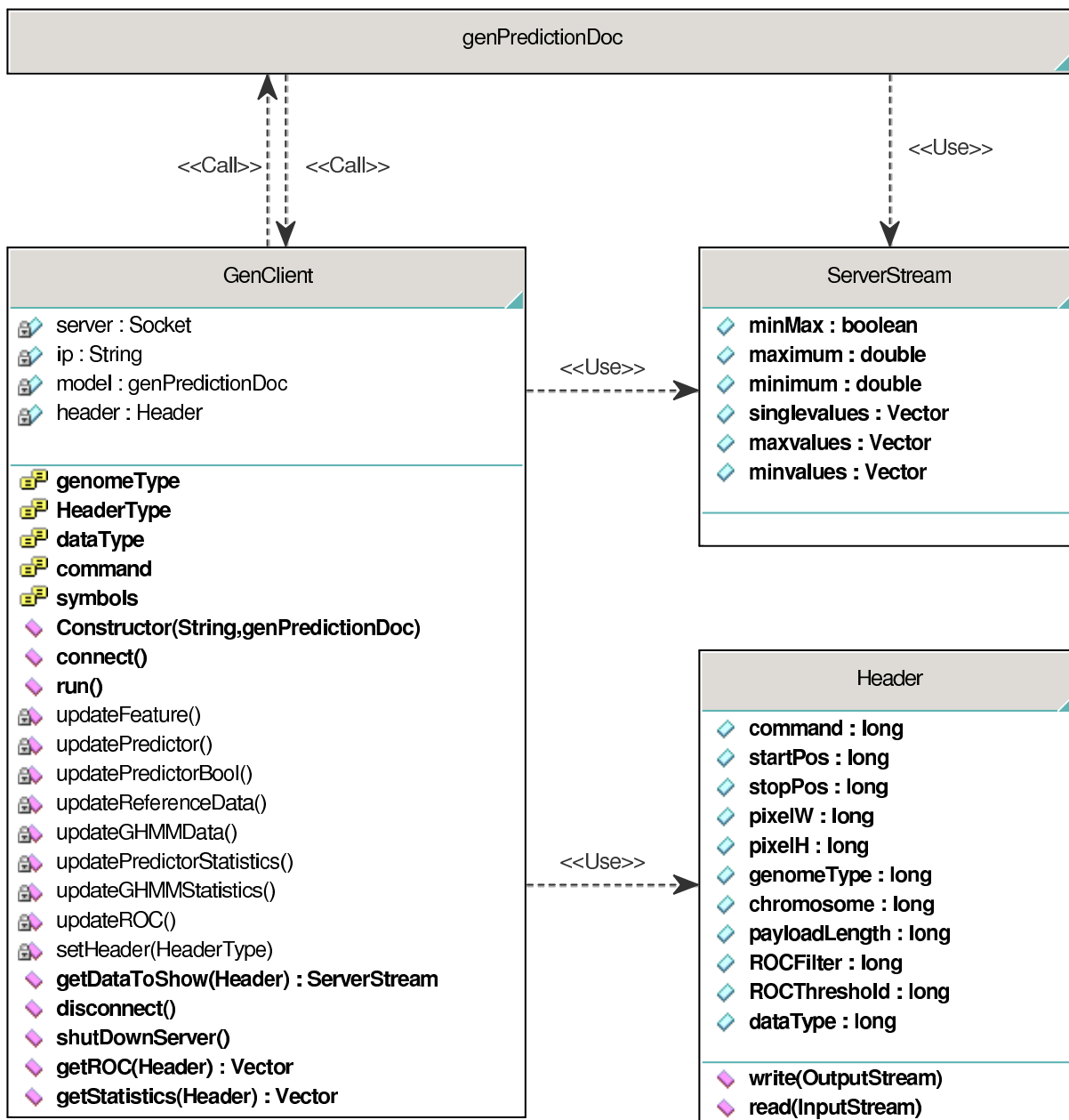


Figure B.4: Class diagram of the Server part

Bibliography

- [1] A. Lomsadze, V. Ter-Hovhannisyanyan, Y. O. Chernoff and M. Borodovsky. Gene identification in novel eukaryotic genomes by a self-training algorithm. *Nucleic Acids Research*, Vol. 33, No 20:6494–6506, 2005.
- [2] H. Almagor. Nucleotide Distribution and the Recognition of Coding Regions in DNA Sequences: an Information Theory Approach. *Theoretical Biology*, 117:127–136, 1985.
- [3] M. Azbel. Random two-component one-dimensional ising model for heteropolymer melting. *Phys Rev Lett.*, 31:589–592, 1973.
- [4] B. Alberts, A. Johanson, J. Lewis, M. Raff, K. Roberts and P. Walter. *Molecular biology of the cell*. Garland Science, fourth edition edition, 2002.
- [5] C. J. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining und Knowledge Discovery*, 2:121–167, 1998.
- [6] C. Burge and S. Karlin. Prediction of Complete Gene Structures in Human Genomic DNA. *Journal of Molecular Biology*, 268:78–94, 1997.
- [7] C. Chang and C. Lin. LIBSVM: a library for support vector machines. 2001. Software available at <http://www.csie.ntu.edu.tw/~cjlin/libsvm>.
- [8] C. Strässle and M. Boos. Prediction of Genes in Eukaryotic DNA. *Hochschule Rapperswil*, 2005.
- [9] C. Zhang and J. Wang. Recognition of protein coding genes in the yeast genome at better than 95 *Nucleic Acids Research*, 28, No. 14:2804–2814, 2000.
- [10] D.Kotlar and Y.Lavner. Gene Prediction by Spectral Rotation Measure: A New Method for Identifying Protein-Coding Regions. *Genome Res*, 13:1930–1937, 2003.
- [11] T. Eitrich. *Support-Vektor-Maschinen und ihre Anwendung auf Datensätze aus der Forschung*. Forschungszentrum Jülich, 2003.
- [12] J. Gao, Y. Qi, Y. Cao and W. Tung. Protein coding sequence identification by simultaneously characterizing the periodic and random features of DNA sequences. *Journal of Biomedicine and Biotechnology*, 2:139–146, 2005.
- [13] J. W. Fickett and C. Tung. Assessment of protein coding measures. *Nucleic Acids Research*, 20:6441–6450, 1992.
- [14] JM. Claverie and L. Bougueleret. Heuristic informational analysis of sequences. *Nucleic Acids Research*, 14:179–196, 1986.

- [15] L. Rabiner and B. Juang . *Fundamentals of Speech Recognition*. Prentice Hall PTR, 1989.
- [16] M. Burset and R. Guigo. Evaluation of gene structure prediction programs. *Genomics*, 34:353–367, 1996.
- [17] M.O. Zhang and T.G. Marr. A weight array method for splicing signal analysis. *Oxford Journal of Bioinformatics*, 9 number 5:499–509, 1993.
- [18] N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines*. Cambridge University Press, 1 edition, 2000.
- [19] N. Pavy, S. Rombauts, P. Déhais, C. Mathé, D. V. V. Ramana, P. Leroy, P. Rouzé. Evaluation of gene prediction software using a genomic data set: application to Arabidopsis thaliana sequences. *Journal of Bioinformatics*, Vol. 15:887–899, 1999.
- [20] P. Beck and P. Frei. Gene Prediction with a Combination of GHMM and SVM. *Hochschule Rapperswil*, 2006.
- [21] P. G. Korning, S. M. Hebsgaard, P. Rouzé and S. Brunak. Cleaning the GenBank Arabidopsis thaliana data set. *Nucleic Acids Research*, Vol. 24:316–320, 1996.
- [22] J. C. Platt. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. Technical report, Microsoft Research, 1999.
- [23] S. Tiwari, S. Ramachandran, A. Bhattacharya, S. Bhattacharya and Ramakrishna Ramaswamy. Prediction of probable genes by fourier analysis of genomic sequences. *Computer Applications in the Biosciences*, 13 No. 3:263–270, 1997.
- [24] S.L. Salzberg, A. L. Delcher, S. Kasif and O. White. Microbial gene identification using interpolated markov models. *Nucleic Acids Research*, 26 No 2:544 – 548, 1998.
- [25] R. Voss. Evolution on long-range fractal correlation and 1/f noise in dna base sequences. *Physical Review Letters*, 68:3805–3808, 1992.