Transcription Factor Binding Site Prediction by the Use of Profile Hidden Markov Models

FINAL REPORT

prepared for the
Austrian Marshall Plan Foundation

submitted by:
Michael Aigner BSc.

Salzburg, September 2012

Acknowledgement

I would like to express my gratitude to my supervisor Stefan Wegenkittl, for investing time and great effort throughout the development of this project.

Great thanks go to my internship supervisor Saad Mneimneh at the City University of New York, who supported me while working on this project and was a incredible help in any situation during my internship at Hunter College.

Furthermore, I want to thank the Austrian Marshall Plan Foundation, for making this internship possible by a generous scholarship.

Special thanks go to my family, who made my studies possible and gave me all possible support.

Preface

This report emerged in the course of a research project on behalf of the Austrian Marshall Plan Foundation, conducted during an internship at the City University of New York, by Michael Aigner. It includes extracts of the master thesis *Transcription Factor Binding Site Prediction by the Use of Profile Hidden Markov Models* [1] and summarizes methods and results of the corresponding research.

Table of Contents

Acknowledgement ii

1. Introduction 1

2. Selected Background Theory 2

2.1. Transcription Factor Binding Sites 2

2.2. Data, Formats and Protein Representation 3

2.2.1. Alphabets 3

2.2.2. FASTA Format 3

2.2.3. Data 4

2.3. Consensus Sequence and Position Weight Matrix 4

2.4. Hidden Markov Models 5

2.4.1. Profile HMMs 5

2.4.2. Smith-Waterman Style pHMM 6

2.4.3. Parameterization of pHMMs 6

2.5. Decoding Algorithms 7

2.5.1. Null Models 7

3. Application to TFBS prediction 8

3.1. pHMM from Ungapped MSAs of TFBSs 9

3.1.1. Parameterization of pHMMs from Ungapped MSAs 9

3.2. Approaches of TFBS Prediction with HMModeler 10

3.2.1. Defining Positive and Negative Samples 11

3.2.2. Implementation 1: Threshold Estimation for TFBS Prediction, by the Use of Bayesian Classifier 11

3.2.3. Implementation 2: Threshold Estimation for TFBS Prediction for Sequence Segments, with a Fixed Length, by the Use of Bayesian Classifier 14

3.2.4. Implementation 2 for Particular TFBS Profiles 17

3.2.5. Implementation 3: Scoring Visualization of Single Sequences, Based on the Sliding Window Method 21

4. Conclusion 23

Bibliography 25

1. Introduction

Understanding homology among proteins starts with sequencing those proteins themselves. Commonly, protein sequences are represented by strings of amino acids, nucleobases or peptides. A considerable and continuously growing amount of protein sequences has already been explored and made available in online databases like PFAM [2], Swiss-Prot [3], and Astral [4]. Furthermore, the JASPAR Database [5] provides a considerable collection of sequenced *Transcription Factor Binding Sites* (TFBSs). Well known and with a big impact to the field, the Human Genome Project [6] finally paved the way to deal with sequences of the human genome. By this, studying the statistics of sequence data has become possible. Organizing and classifying the vast amount of sequence data which has already been explored and gathered, is part and the first step to understand and to gain knowledge out of the data, see [7, p. 1]. Organization and classification in this context is exactly the approach of protein homology search. To classify sequences, no matter of which representation, means to proof the similarity of one sequence to a set of others. This method is currently the cheapest and therefore the preferable for examination in the first place.

From the point of view of a computer scientist, the first approach for showing similarities is to compare those strings, compute distances and, based on the resulting measures, classify them. Fortunately, this approach is not out of touch with reality. The "similar sequence - similar structure - similar function paradigm" [8] implies that the similarity of two sequences of proteins (respectively strings) is an indicator for a similarity of 3D structure and function. Exploring similarities among sequences is considered to reveal relationships between proteins of similar biological function. Therefore it should come as no surprise that "most of the problems in computational sequence analysis are essentially statistical" [7, p. 1].

In the early 90s a group around A. Krogh and D. Haussler introduced profile Hidden Markov Models (pHMMs) [9], adopting HMM techniques which have been used in speech recognition. HMMs had been used in biology before, but their paper had a dramatic impact, because HMM technology was well-suited to “profile” methods for searching databases using multiple sequence alignments (MSAs). Since then, bioinformatic groups use pHMMs as the underlying formalism for sequence profile analysis, see [10].

Decoding algorithms are used to find the sequence of hidden states (path) of pHMMs given an obtained sequence. In addition, by the use of decoding algorithms, the probability of that path both occurring and producing the obtained sequence can be estimated.

The probability whether a protein sequence contains a TFBS can be described by the resulting probabilities (scores) of decoding algorithms. Furthermore, the path can be used to determine a location of the TFBS within the obtained sequence.

The basis for the implementation is the HMModeler framework. This framework provides, among others, tools for parsing protein sequence formats and parameterization and training of pHMMs.

2. Selected Background Theory

2.1. Transcription Factor Binding Sites

Transcription is one of the fundamental processes of life. It is the first step in a procedure that translates the Desoxy-ribonucleic-acid (DNA), into proteins. This process is needed to enhance or repress the production of specific proteins to a certain time and amount. Imbalance of this systems lead, in the worst case, to severe effects. One of which is known as the major burden of mankind – cancer. [11]

The key element in transcription and regulation are transcription factors (TF). These proteins, referred to as *trans*-elements, interact with specific regions of the DNA known as *cis*-elements in order to enhance or suppress the transcription.

An important paradigm in this context is that TFs bind to defined short stretches of DNA. These elements (typically 6–12 base pairs) so-called *transcription factor binding sites*, are scattered throughout the genome. Often they are localized near the starting site of transcription, known as promoter, marking the beginning of a gene, but they are also found several hundred base pairs off.

A major question in current research is whether potential binding sites are functional and under what circumstances. In order to test this biological relevant state, they have to be identified first, which is challenging, considering that a simple string based search of the DNA sequence within a large genome finds large numbers of matching sequences. This task is further complicated by the fact that the most TFBSs are defined by a sequence that contains ambiguous bases and several proteins compete over one binding site. [12]

Driven by increasing availability of sequence datasets, development of powerful tools for the search and identification of such elements is of highest interest.

2.2. Data, Formats and Protein Representation

Proteins are coded as sequences of either amino acids or nucleobases. Therefore, several string representations have been developed, from which we will address two: the *Amino Acid Alphabet* and the *Nucleic Acid Alphabet*.

2.2.1. Alphabets

Alphabets in computer science are roughly just a set of characters. In bioinformatics, they contain a set of characters that is used to represent biological sequence information as a single-letter code.

However, the most common alphabets are the ones containing amino acids or nucleobases, which will be discussed below. For both representations, several notations exist. Commonly they follow the IUPAC/IUBMB coding [13]. In addition, several approaches of reduced amino acid alphabets exist, which are not going to be addressed.

2.2.1.1. Nucleic Acid Alphabet

The term “Nucleic Acid Alphabet” has been adopted from IUPAC/IUBMB. For the sake of better understanding we continuously use the term “nucleobases” in order to refer to adenine, cytosine, guanine and thymine. The basic *nucleic acid alphabet* after IUPAC/IUBMB consists of the five characters "AGTCN", standing for these four nucleobases each represented by their first letter. Additonally, the character "N" is standing for an arbitrary nucleobase. The *nucleic acid alphabet* is used by the *.sites* file type of the JASPAR database, which is going to be dealt with. While not recommended by IUPAC, JASPAR *.sites* files contain "X" as well for unknown acids, see [14]. In terms of biological sequencing "N" and "X" have a different meaning, but from the computer sciences perspective, "N" and "X" have to be handled equally as long as no further information is available.

2.2.2. FASTA Format

The FASTA format is a common text-based format for representing either nucleobase, amino acid or peptide sequences or sets thereof. FASTA files with a set of sequences are called multi-FASTA files, see [13]. Although, the FASTA format does not specify the alphabet to be used for the sequence data, it is recommended to use the IUPAC single-letter codes [13]. Since FASTA format thus has less restrictions, it can be seen as a meta format. Several sub formats representing more specific sets of sequences use the FASTA format as well.

2.2.2.1. Files of Type .sites

Files of this type are provided by the open access JASPAR CORE database (The JASPAR database).

The *.sites* file represents the raw data for all sites of [5] for the construction matrix models of TF DNA-binding preferences. Since files of the type *.sites* represent TFBSs, the alphabet has been extended by using both lower case and upper case characters. Uppercase characters describe the exact location of the binding site itself. The following example illustrates that:

```
>MA0036GATA2 1
cgatcAGATAggctgcctcgg
>MA0036GATA2 2
caGGATActtgacttgtggt
```

The length of the binding sites is always constant in one file. Obviously the sequences are not aligned with respect to the binding sites. This leads to the conclusion that the alignment information for training the HMM lies in the upper case letters only.

2.2.3. Data

For all test implementations the data set of 28 *.sites* files containing TFBSs, provided by JASPAR [5] has been used. Each file represents a set of sequences containing binding sites for one particular transcription factor. Such a set is denoted as *TFBS profile*. The marked binding sites have a length of 5 to 19 nucleobases.

2.3. Consensus Sequence and Position Weight Matrix

Common methods, for TFBS prediction are the *consensus sequence* and the *position weight matrix* (*pwm*).

In general, the consensus sequence represents a set of example sites by matching the example sites closely but not necessarily exactly. The number of allowed mismatches, the ambiguity of the consensus sequence and the sensitivity and precision of the representation are in a relation to each other.

Creating consensus sequences is quite easy, but it is not optimal for predicting the occurrence of new sites. Several methods for generating consensus sequences have been compared by Day and McMorris in [14], see [15]

The *pwm* is an alternative to the consensus sequence, which represent a multiple sequence alignment as a matrix where each cell represents a score for a nucleobase in the particular position. *Pwms* give more significant results for TFBS prediction than consensus sequences,

but have one major disadvantage. With *pwms* it is not possible to map insertions or deletions of nucleobases, caused by evolutionary mutations. A common way to map those mutations is the use of HMMs.

2.4. Hidden Markov Models

HMMs are stochastic models named after A. Markov, because of the underlying theory of Markov Chains. In general, HMMs are a tool to efficiently describe a stochastic model for sequences. These models distinguish internal states π_i from emitted symbols x_i. The intern state sequence is called the path π and follows a Markov chain. In bioinformatics, such chains are used to abstractly describe the skeleton or backbone of a family of sequences. For the determination of the path, given an observed sequence, decoding algorithms are used.

Some parameters of HMMs are transition probabilities which are distinguished between the types transition probabilities a_{kl} and emission probabilities e_k. More precisely, we have

$$a_{kl} = P(\pi_i = l \mid \pi_{i-1} = k)\ , \tag{1}$$

where π_i are the state variables and k and l are the actual states. Therefore the transition probability a_{kl} is defined as the probability of a transition from state k in the i-1th position of the path, to state l in the ith position. Similarly,

$$e_k(b) = P(x_i = b \mid \pi_i = k)\ . \tag{2}$$

Thus, the emission probabilities e_k are defined as the probability that symbol b is emitted when being in state k.

The emitted sequence x is also called the observation. Considering protein homology search the observation represents a sequence of amino acids or nucleobases.

In general, an HMM has no restriction in terms of possible transitions from any state to any other. These models are called fully connected models. Profile HMMs (pHMMs) can be considered as a group of more stringent models, with a particular architecture.

2.4.1. Profile HMMs

Commonly, pHMMs are used in protein homology search for aligning and distance measuring (scoring) of test sequences to a given sequence dataset. Therefore, the architecture of pHMMs should correspond to the criteria of the set the model is going to be trained with and the desirable opportunities for the alignment.

The goal of aligning a sequence to a set of others is to find a way to align the sequence such, that a column-wise best possible match of the symbols emerges. Since parts of some

sequences can be missing (deletions) or additional parts can occur (insertions), this leads to three possible state types for pHMMs, see [16].

Match states are those states which are emitting, at column i, a symbol b with the probability $e_i(b)$. ***Insert states*** are emitting symbols as well, but since these are additionally emitted to the model columns, this output is somewhat in between model columns. Therefore, the emission probabilities for insert states are taken from a background distribution of the respective alphabet, called *qas*. ***Delete states*** emit the symbol "-" which represents a gap. Delete states are also called *silent states*, see [16].

2.4.2. Smith-Waterman Style pHMM

Additionally to the original pHMM this model has two flanking simple self-looping models. Both are consisting of a *begin state B* (resp. an *end state E*) represented by a square, an *insert state IB* (resp. *IE*) represented by a diamond and a silent state *SB* (resp. *SE*) represented by a shaded circle. The corresponding transition probabilities are summarized as a_{FI} (transitions to the flanking insert states) and a_{FS} (transitions to the flanking silent states) where $a_{FI} = 1 - a_{FS}$.

These flanking models are connected to the incorporated original pHMM via the silent states which allow a transition to each of the match states ($a_{SB\,M_i}$ and $a_{M_i\,SE}$).

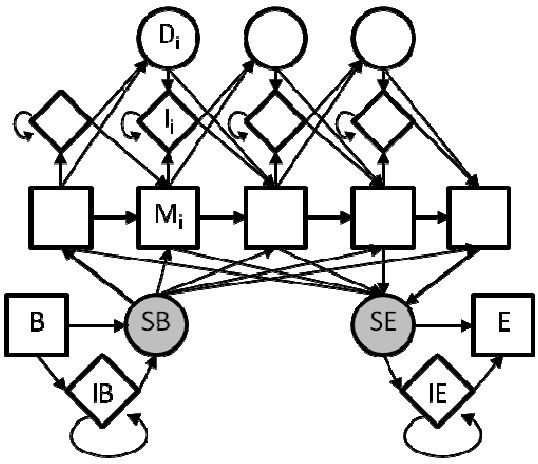

Figure 1: Smith Waterman pHMM

2.4.3. Parameterization of pHMMs

Parameterization of pHMMs means to incorporate prior expert knowledge by setting parameters either to affect the automatic calculation/estimation of probabilities or to even substitute the estimation process by assigning user defined values. Configuration of Null models (see Chapter 2.5.1) is considered a parameterization as well. The aim of parameterization is to make the distribution, over the whole space of sequences peak around the desired group of positives (e.g. protein families or TFBSs), see [7, p. 107].

Parameters which are currently in use by the HMModeler are: *pseudo counts* for transition probabilities; *background distribution* or *distribution mixtures* for emission probabilities; *fixed transition probabilities* for the flanking models and there from outgoing intro transition probabilities; and the *expected model length* (eml), which takes account into the flanking transition probabilities and the simple null model.

2.5. Decoding Algorithms

Decoding algorithms tell us "what the observation sequence 'means' by considering the underlying states" [7, p. 107]. So, the aim of decoding algorithms is to find an intern path π through the HMM given an observed sequence x, which means a mapping of $x \rightarrow \pi$, see [17] and [18]. In the case of the Viterbi algorithm the found intern path is the most probable path π^*.

Furthermore, decoding algorithms find the probability that an HMM produces this sequence. This probability is usually used for deriving so-called *scores* for an HMM-sequence tuple.

For general information about these algorithms please see literature like [7] and [19].

2.5.1. Null Models

In order to assess significance of scores, those must be comparable. By decoding a sequence we get a probability that an HMM produces a certain sequence, but this probability is dependent on the length of the testing sequence. Since either all $a_{kl} \leq 1$ and all $e_l \leq 1$, the likelihood for a sequence s decreases with $L = |s|$. This means that sequences with a variable length L are not comparable up to here. To make the scores comparable, a correction method is needed, see [20].

The correction of the score is done by dividing the probability of the decoding algorithm result by the result of the null model (resp. subtracting in the log-space).

2.5.1.1. Simple Null Model

This model follows the approach of using a different model and scoring with the same sequence.

The simple null model is equal to the Smith-Waterman style pHMM flankings. By correcting a score with the result of this model, length dependency of both the begin and the end cycle cancel out. Furthermore, the length dependency of the insert states of the core model are canceled out concerning the emission probabilities, see [21].

2.5.1.2. Reverse Sequence Null Model

The reverse sequence null model, follows a different approach by changing the sequence (by reversing it) and scoring it with the same model.

Fortunately, the reverse model offers several benefits like easy implementation and a sufficient correction of length dependencies. Obviously, it is more time-consuming than the simple null model.

2.5.1.3. Resulting Scores

Resulting from the combination of decoding algorithm and null model, we can assign to each tested sequence a set of six scores. Therefore, we write $\tilde{V}(s) = V^E(L)$ and $\tilde{F}(s) = F^E(L)$ of the sequence s where $V^E(L)$ is the Viterbi score and $F^E(L)$ is the Forward score; $\tilde{S}(s)$ denotes the score of simple null model of sequence s; and s^{-1} stands for the reversed sequence s.

	Viterbi algorithm		Forward algorithm	
No null model	*viterbi score*	$\tilde{V}(s)$	*forward score*	$\tilde{F}(s)$
Simple null model	*simple corrected viterbi score*	$\tilde{V}(s) - \tilde{S}(s)$	*simple corrected forward score*	$\tilde{F}(s) - \tilde{S}(s)$
Reverse sequence null model	*reverse corrected viterbi score*	$\tilde{V}(s) - \tilde{V}(s^{-1})$	*reverse corrected forward score*	$\tilde{F}(s) - \tilde{F}(s^{-1})$

Table 1: Resulting scores of combination of decoding algorithm and null model

3. Application to TFBS prediction

Simply put, TFBS prediction means locating particular sections of sequences which are highly likely TFBSs. By Viterbi decoding of a testing sequence we get, on the one hand the score $P(\pi^*)$ and on the other hand the path π^*. The path π^* may contain a section of match states which would mark the most probable section of the TFBS, in the case we use a pHMM trained by TFBSs only. So, one way to locate and score a TFBS in a sequence is decoding it and determine the match states section as TFBS with the probability of $P(\pi^*)$.

3.1. pHMM from Ungapped MSAs of TFBSs

Since JASPAR sequences do not contain any gaps and the marked TFBSs are continuous, every column of the MSA can be seen as a model column for the pHMM and we do not expect any insertions. This leads to a very particular version of the pHMM.

As long as we do not modify the model by parameterization towards higher delete and insert probabilities, but even increase the pseudo counts for match to match transitions, we can expect a model with match to match transition probabilities close to 1[1]. Further we can assume that the model has to be entered in the first column to assure an equal length of the TFBS to the example TFBSs.

In terms of decoding this means that each nucleobase of one section of the testing sequence will be mapped to a match state and therefore is part of the *match area*.

Thus, we can take advantage of flanking models to shift the alignment to the most probable region. This leads to the following conclusion:

In the case of decoding a sequence by using a pHMM from an ungapped MSA of TFBSs with length L, a *match area* μ is the most probable continuous section producing the decoded sequence for the location of a TFBS, with length L, too.

3.1.1. Parameterization of pHMMs from Ungapped MSAs

For the intro transition probabilities we have set the probability of a transition from the silent begin state to the first match state to $a_{SB\,M_0} = 0.99$. By doing this, we expect the path π^* of the Viterbi algorithm to enter the pHMM in the first model column. We also increase the pseudo counts for match to match transitions to 100 and set pseudo counts for all other transitions to 1.

Intro transition probabilities	**Flanking transition probabilities**	**Pseudo Counts**
$a_{SB\,M_0} = 0.99$ $a_{SB\,M_{j\neq 0}} = \frac{0.01}{N-1}$	$a_{FI} = \frac{2-(\lvert s\rvert - \lvert TFBS\rvert)}{(\lvert s\rvert - \lvert TFBS\rvert)}$ $a_{FS} = 1 - a_{FI}$	$match\ to\ match = 100$ $insert\ to\ insert = 1$ $delete\ to\ delete = 1$ $flanking\ transitions = 1$ $intro\ transitions = 1$

Table 2: Parameterization for pHMMs

Thus, the pHMM have been drastically simplified as visualized in Figure 2. Based on this parameterization, the grey marked transitions and states are highly unlikely to be applied by

[1] Due to calculation issues, ε pseudocounts for all states are used. Thus, every transition probability is $1-\varepsilon$ at the maximum.

the decoding algorithms and consequently we ensure to achieve continuous match areas from the first to the last match state of the model.

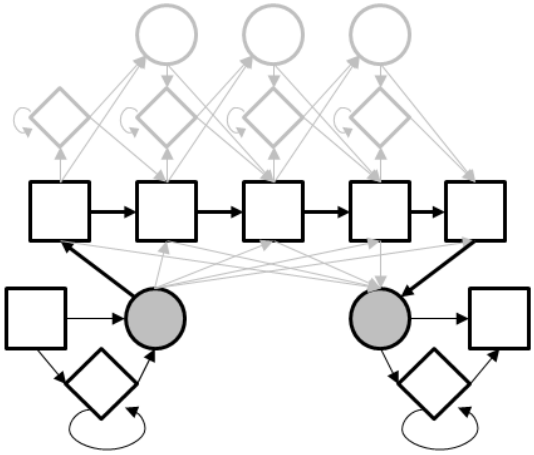

Figure 2: Reduced Smith Waterman pHMM

3.2. Approaches of TFBS Prediction with HMModeler

In taking advantage of these findings, a testing scenario can be sketched as follows:

1. Parsing: parse a *.sites* file and create a MSA
2. Parameterization: set a high amount of pseudo counts for match states and set a high probability for entering the pHMM in the first column.
3. Training: leave one sequence of the MSA out and train the pHMM by calculating emission probabilities e_k and transition probabilities $a_{k\,l}$.
4. Decoding: decode the omitted sequence in order to the trained model, to determine a match area and the corresponding scores.
5. Classifying: Check whether the match states are corresponding to the original position of the TFBS. If so, the score represents a positive sample, otherwise it could be seen as a negative.
6. Threshold estimating: Repeat steps 1 to 5 and train a Bayesian classifer in order to estimat thresholds for positve scores.

Following this procedure, we get either a positive or a negative sample and the corresponding score per sequence. By decoding an unknown explorative sequence we can use the threshold to predict whether the sequence contains a TFBS and expose with the match area where it may be located.

This is a first result but may not be satisfying, because of the following concerns: a) negative samples describe only scores which are a result of a decoding, where the match area does not match the original TFBS position, which is not tantamount with a confirmed negative sample; b) we do not know the significance of the resulting scores; c) we get one match area within one sequence, but possibly the sequence contains several likely positions of TFBSs.

3.2.1. Defining Positive and Negative Samples

We need to answer the question what exactly is meant by a positive or a negative sample. Eventually scoring thresholds should be used to decide on the basis of a score whether a sequence contains a TFBS or not. To train a bayesian classifier we need at least two groups of samples: scores associated with sequences containing a TFBS (positives) and scores with sequences not containing a TFBS (negatives). This would be the ideal case, but proving whether a sequence do not have a TFBS is unrealistic, since this requires a lot of real-world laboratory experiments for each sequence.

Unfortunately, real-world biology makes it even more difficult, since so-called *repeats* of TFBSs within the example sequences of JASPAR exist. Thus, in one sequence several potential TFBSs can occur, whereas only one is marked as such. This means there is a considerable probability that the decoding of a sequence results in matching repeats and consequently gains a high score, but has to be labeled as a negative sample, since it is not marked as a TFBS.

However, currently there is no way for us to prove whether a negative sample is a de facto negative sample. Since we expect a higher $P(\pi^*)$ for positive samples, we can conclude that wrongly labeled negative samples raise a threshold, but we do not know by which factor.

Although the definition is not entirely satisfying we stick to the follwing one:

A *positive sample* describes a sequence-scores tuple where the match area coincides with the original location of the TFBS.

A *negative sample* describes a sequence-scores tuple where the match area does not coincide with the original location of the TFBS or a tuple where the sequence does not even contain a known TFBS.

3.2.2. Implementation 1: Threshold Estimation for TFBS Prediction, by the Use of Bayesian Classifier

In order to get a first understanding of scores the above sketched testing scenario has been implemented. The used data set has been the full set of the 28 JASPAR *.sites* files. For each *.sites* file we have executed steps 1 to 5. The parameterization (step 2) was set as described in Chapter 3.1.1, except the flanking transition probabilities. Due to performance issues those were set to $a_{FI} = \frac{2-(x-|TFBS|)}{(x-|TFBS|)}$, where $x = \overline{|s|}$ for each TFBS profile. The *eml* for the simple null model was set to $\overline{|s|}$. For the training (step 3) of pHMMs 10% of the sequences of each MSA have been taken out. These 10% amount to 2,513 sequences. The classification (step 5) followed the definition of Chapter 3.2.1. Noticeable at step 5, for 16 files all TFBS have been found correctly, so they do not produce negative samples. The remaining 12 files make up

540 negative samples, whereas only 5 files make up for 90% of those negative samples. In other words the majority of the negative samples are resulting from a minority of the tested files and may not be representative for the rest of the files. Another consequence is, that we cannot implement a threshold estimation for those 16 files (and consequently for the particular TFBS profiles, since the dataset contains no sequences causing a negative sample). Furthermore, we have a positive/negative ratio of approximately 4:1.

For estimating the threshold for positive scores (step 6) a naive Bayes classifier has been trained by the Matlab function `classify(sample, training, group, 'quadratic')`. The parameter `'quadratic'` specifies the type of discriminant function. The training of the classifier was done with 70% randomly chosen samples. The testing of the classifier was done by the remaining 30%. The feature vector has been two dimensional, with the features *simple corrected viterbi score* and *reverse corrected viterbi score*.

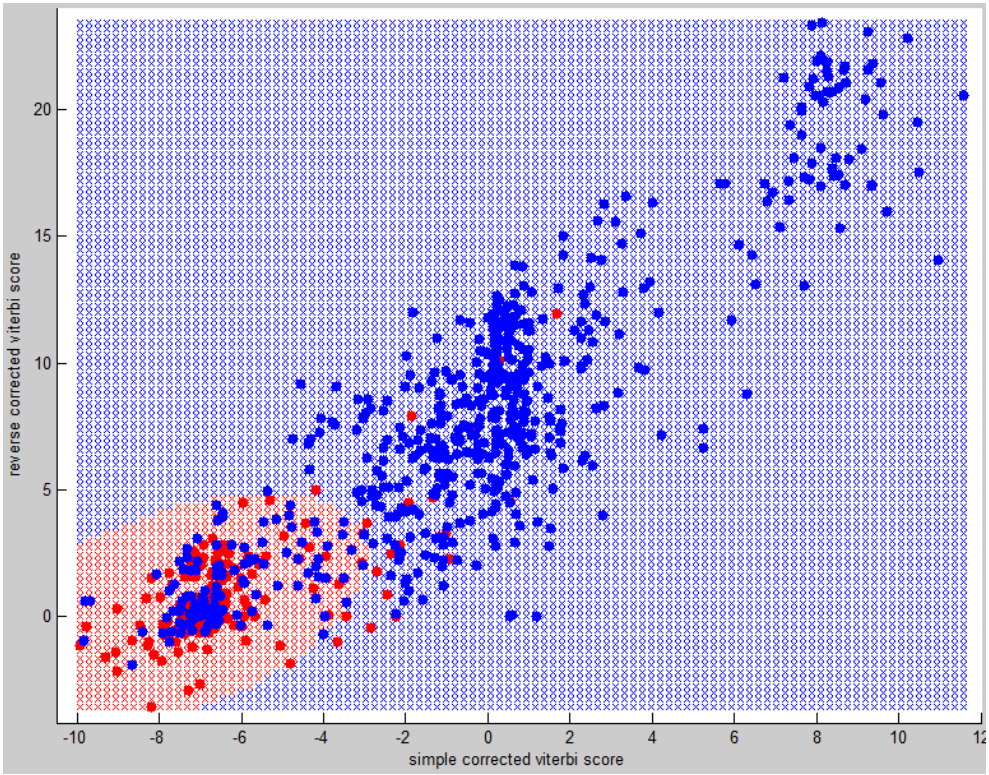

Figure 3: Decision regions and testing samples of Implementation 1

Figure 3 shows the scatter plot of the trained classifier and the testing data. The blue surface describes the positive decision region, the red surface describes the negative decision region. Accordingly, the blue dots describe positive samples and the red dots negative samples.

Due to the positive/negative ratio, the negative decision region is much smaller than the positive, which is true for the testing data, but may not for explorative data. A correlation of the two dimensions is obvious, but also expected, since both dimensions are related to the

Viterbi score. We also expected a higher threshold in consequence of wrong negative classification, so it is not surprising, that false positives are less likely than false negatives.

With the Matlab function *plotconfusion*, we created the classification confusion matrix shown in Figure 5. The confusion matrix can be interpreted as described in Figure 4.

# True Positives (TP) $\frac{TP}{P+N}$	**# False Positives (FP)** $\frac{FP}{P+N}$	$\frac{TP}{TP+FP}$ $1-\frac{TP}{TP+FP}$
# False Negatives (FN) $\frac{FN}{P+N}$	**# True Negatives (TN)** $\frac{TN}{P+N}$	$\frac{TN}{TN+FN}$ $1-\frac{TN}{TN+FN}$
$\frac{TP}{TP+FN}$ $1-\frac{TP}{TP+FN}$	$\frac{TN}{TN+FP}$ $1-\frac{TN}{TN+FP}$	$\frac{TP+TN}{P+N}$ $1-\frac{TP+TN}{P+N}$

Figure 4: Legend for confusion matrices

478 63.4%	**23** 3.1%	95.4% 4.6%
114 15.1%	**139** 18.4%	54.9% 45.1%
80.7% 18.3%	85.8% 14.2%	**81.8%** **18.2%**

Figure 5: Confusion matrix of implementation 1

The confusion matrix represents the outcome of the testing of the Bayesian classification. For practical biological purposes the most interesting classification outcome is the sensitivity (SENS, marked in a green circle) and the false rejection rate (FRR, marked in a red rectangle).

By implementing a varying threshold for moving the decision boundaries, the sensitivity can be increased by lowering the FRR and vice versa. The threshold can be adjusted in order to do cost optimization, depending on the costs of FPs and FNs. In terms of biological research these costs can be diverse.

After calculating the log of all Bayesian posteriori probabilities for all points, the threshold is computed iteratively with 100 steps from the lowest to the highest probability. Plotting of the sensitivity against the FRR over the threshold is called receiver operating characteristic (ROC) and shown in Figure 6. The marked data point in Figure 6 represents the threshold equal to the Bayesian classification shown in Figure 3 and Figure 5.

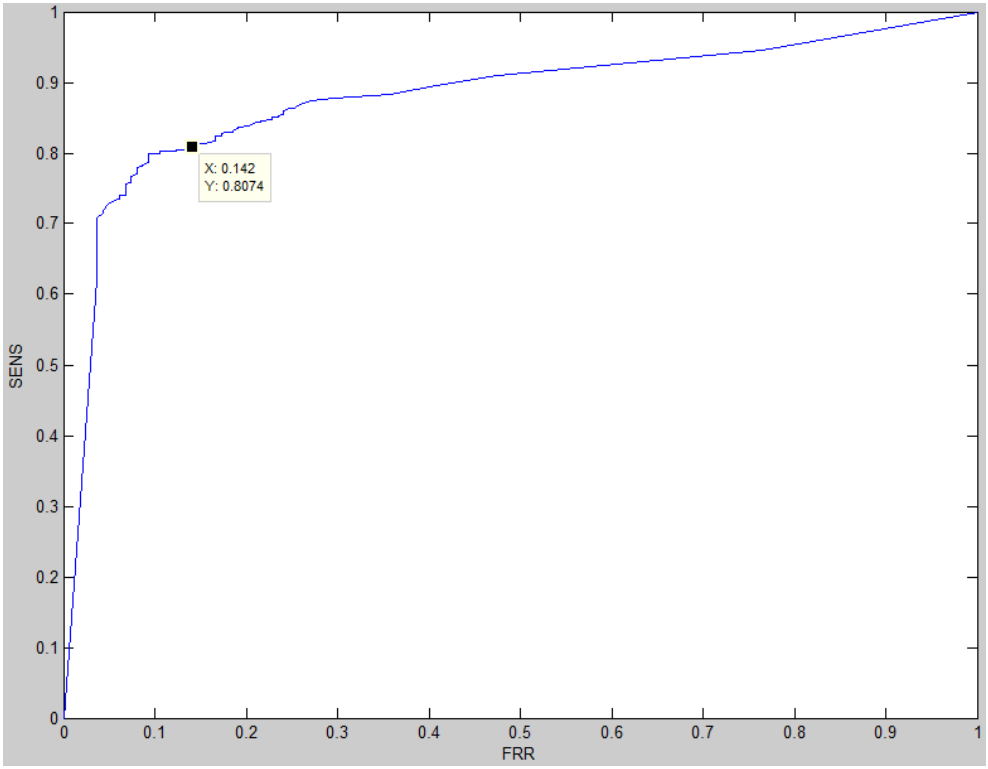

Figure 6: ROC graph of implementation 1

An optimum in terms of classification would be a rectangular graph corresponding to aforementioned sensitivity of 100% and FRR of 0%.

Although, the results of the first test scenario are far better than random classification, they are not sufficient for actual application to TFBS prediction. Thus, the test implementation needs to be modified in order to solve aforementioned problems and probably achieve more satisfying results.

3.2.3. Implementation 2: Threshold Estimation for TFBS Prediction for Sequence Segments, with a Fixed Length, by the Use of Bayesian Classifier

Another way of generating a testing set for threshold estimation is not to decode whole sequences, but to slice the testing sequences into segments of equal length. In other words, we define a window $w_{k\to l}$ where k is the sequence index of the first nucleobase position of the window and l is the last nucleobase position. The size of $w_{k\to l}$ is $|w| = l - k = x * |TFBS|$ where $x \geq 1$. We slide this window nucleobase-wise (for each position i) over the testing sequence. For each window, we execute steps 4 and 5 of the above testing scenario. Thus, one sample for each scored window or $|s| - |w|$ samples for each sequence will be produced. Relative to the window size, a number of windows will not contain the actual TFBS and result in a negative sample. Consequently this procedure leads up to a higher number of negative samples.

By doing this we can eliminate several of our concerns: a) due to the fact that negative samples are produced for each sequence (resp. each TFBS profile), for one thing, negative samples are less dependent on the particular TFBS profile and for another thing, we can implement threshold estimations for each TFBS profile; b) since we get a higher amount of negative samples, on the one hand the impact of wrong classification is reduced, and on the other hand the positive/negative ratio corresponds more to real-world application scenarios; c) the impact of length dependencies of scores will be reduced, due to a lower length variability of the testing sequences.

Of course, computational effort increases, by using this method, since steps 4 and 5 have to be repeated $|s| - |w|$ times, for each sequence.
The window size was set to $|w| = |TFBS|$. Consequently, we do not expect insertions in the flanking states. Due to calculation issues the flanking transition probabilities were set to $a_{FI} = \varepsilon$ instead of 0. Due to segmentation, the number of testing samples has been increased to 554,151 with 2,513 positive samples and 551,638 negative samples.

Besides aforementioned alterations, parameterization and execution of all other steps have been performed in conformity with implementation 1.

As we can see in Figure 7 the decision regions changed their size drastically in comparison with Figure 3. Due to the positive/negative ratio of approximately 1:220, the positive decision region, for this test, is (much) smaller than the negative one. Not surprisingly, the correlation of the two features is still present.

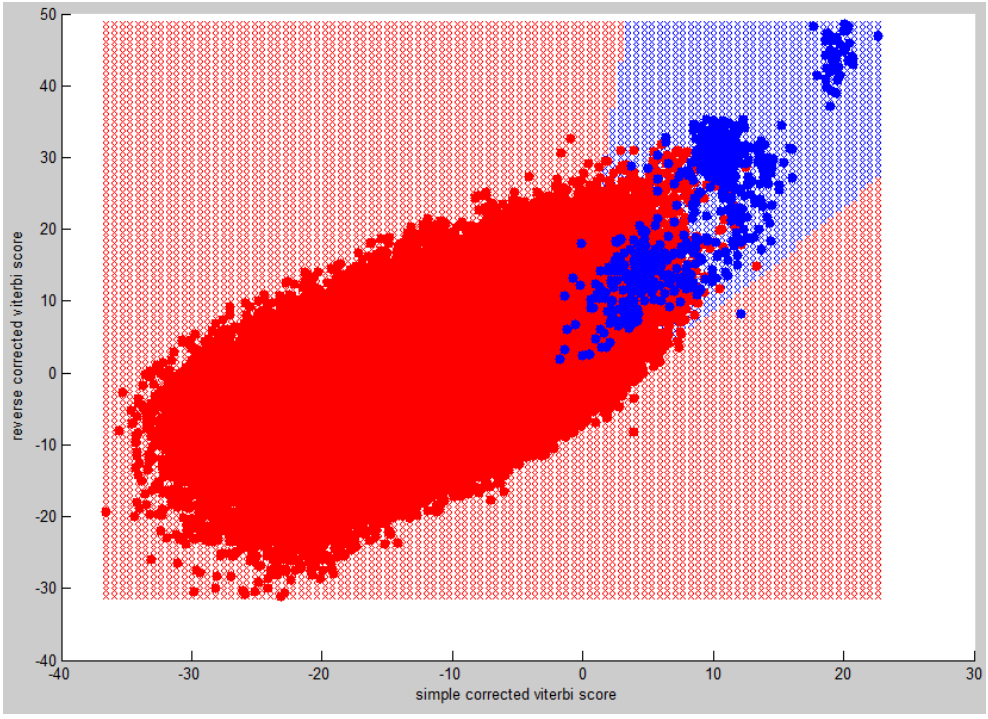

Figure 7: Decision regions and testing samples of implementation 2

The confusion plot (Figure 8) of the Bayesian classifier with a fixed threshold, shows that the test with a fixed window results in a higher recognition rate. The sensitivity has been increased to 98.4% and the FRR decreased to 3.5%.

742 0.4%	**5769** 3.5%	11.4% 88.6%
12 0.0%	**159723** 96.1%	100.0% 0.0%
98.4% 1.6%	96.5% 3.5%	**96.5%** **3.5%**

Figure 8: Confusion matrix of implementation 2

Considering the ROC plot of this test implementation, the improving of recognition is even more obvious. The resulting graph is much closer to the rectangular desired one. The marked data point in the middle represents the Bayesian classification of Figure 7 and Figure 8. The lower left data point shows that we can reach a sensitivity of 90.15% with a FRR of 1%. Accordingly, the upper right one shows a FRR of 7.64% with a sensitivity of 99%.

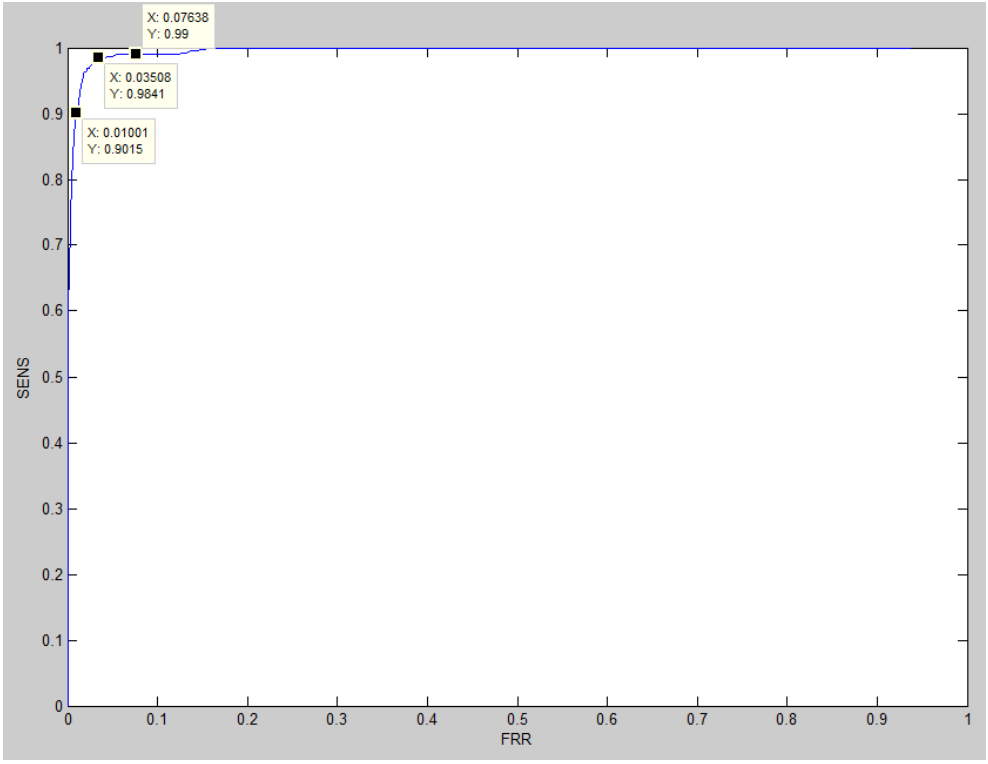

Figure 9: ROC plot of second test implementation

However, the estimated thresholds represent not one particular TFBS profile, but all profiles at once. For the application to explorative sequences, decoding of those has to be done for each profile, anyway. Therefore, threshold estimation for each TFBS profile may give more representative thresholds and thus, lead to higher recognition rates.

3.2.4. Implementation 2 for Particular TFBS Profiles

Since the sliding window test provides a set of positive and negative samples for each TFBS profile, this test is applicable for single TFBS profiles. The tests for particular profiles have been performed in conformity with the second test implementation for all steps including parameterization. In the following we show the tests for two single *.sites* files each representing one particular TFBS profile.

In a current parallel research project which has been done by Saad Mneimneh at the City University of New York, the significance of the corrected Viterbi scores has been examined theoretically. By the central limit theorem for sums of local Viterbi scores, an estimation of the expected values for the thresholds of the scores has been derived. The current version still requires a numerical approximation algorithm. Nevertheless, we incorporate preliminary results into implementation 2 for particular TFBS profiles (shown as green lines in Figure 10 and Figure 13). The proximity to our empirical results indicates this approach should be examined further and could eliminate the necessity of training examples for finding useful thresholds for the classification algorithm. The results of this research will be published within another paper.

Figure 10 shows the decision regions for the TFBS profile Klf4 of the MA0039.2.sites file. Obviously, there is a high number of false positives.

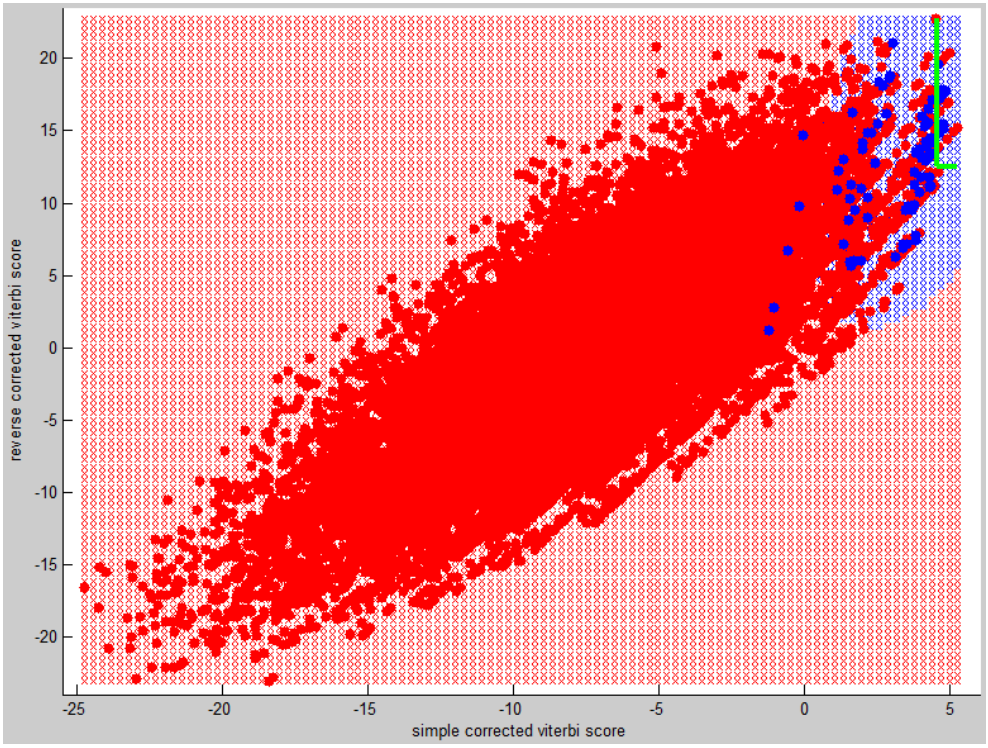

Figure 10: Decision regions and testing samples of implementation 2 for the Kfl4 profile

The confusion matrix (Figure 11) for this test confirms the high number of negative samples. The ratio of positive/negative test samples of approximately 1:194 (0.5% positives) shows that knowing the prior probabilities for positives and negatives, a random classification would be even better.

126 0.5%	**1327** 5.1%	8.7% 12.5%
5 0.0%	**24654** 94.4%	100.0% 0.0%
96.2% 3.8%	94.9% 5.1%	**94.9%** **5.1%**

Figure 11: Confusion matrix of implementation 2 for the Kfl4 profile

The ROC plot in Figure 12 also shows a worse performance than the ROC of implementation 2 (which was applied to all files).

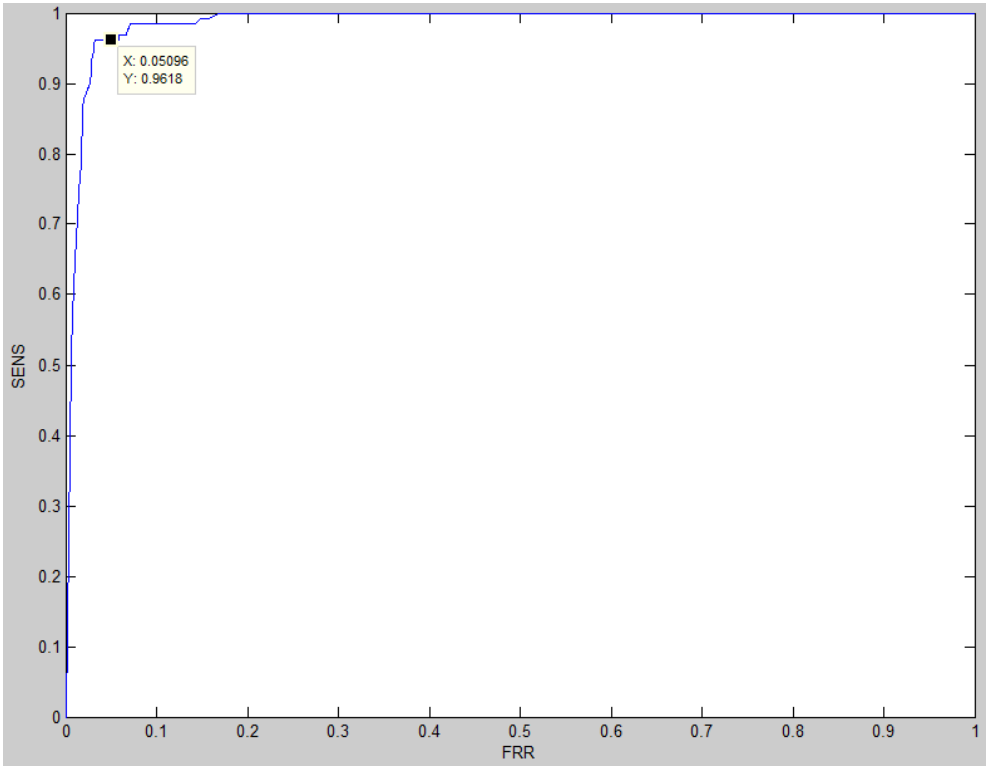

Figure 12: ROC of the test implementation 2 for the Kfl4 profile

Far more satisfying results of implementation 2 for particular TFBS profiles are given by the application to the CTCF profile of the MA0139.1.sites file. The decision boundary clearly separates (shown in Figure 13) positive and negative test data points. The misclassifications of four negative samples as positives can be neglected in terms of the ROC. The confusion matrix (Figure 14) demonstrates the excellent results, too. But coming back to the four false positives, the confusion matrix shows also, that every eighth as positive classified sample is false. Still, we do not know if those samples should not be labeled negative, because of discussed circumstances in Chapter 3.2.1.

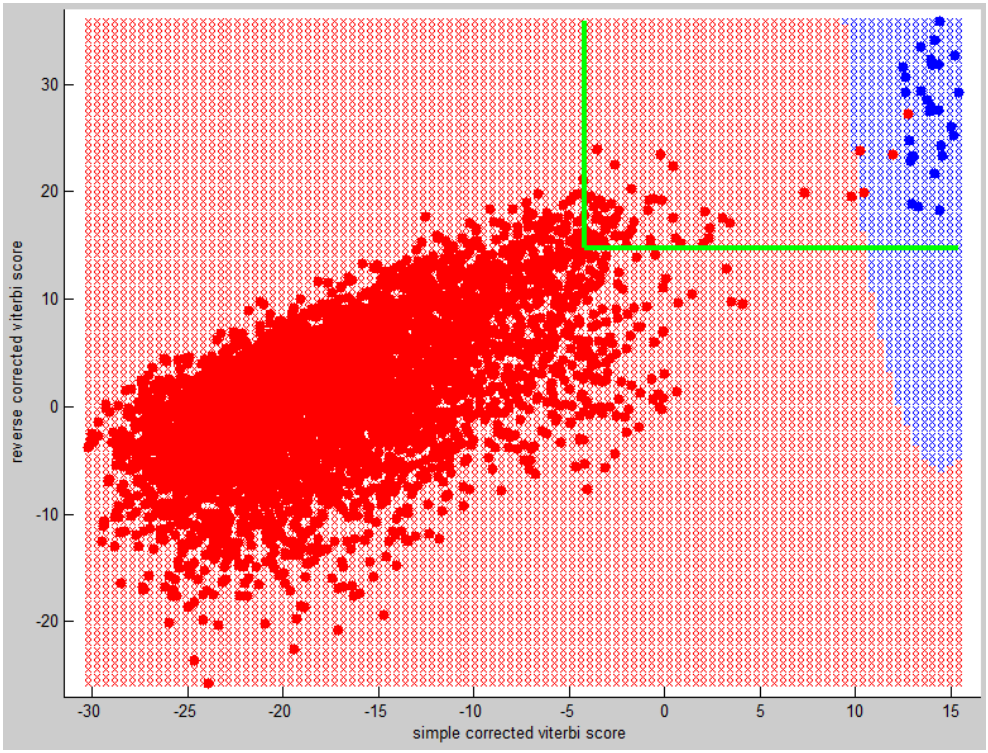

Figure 13: Decision regions and testing samples of test implementation for the CTCF profile

28 0.6%	**4** 0.1%	87.5% 12.5%
0 0.0%	**5039** 99.4%	100.0% 0.0%
100.0% 0.0%	99.9% 0.1%	**99.9%** **0.1%**

Figure 14: Confusion matrix of the test implementation 2 for the CTCF profile

The ROC (Figure 15) is close to the optimum of the aforementioned rectangular graph. In this case, we zoomed the ROC in order to show that it is still not perfectly rectangular. Again, the marked data point represents the threshold corresponding to the decision regions (Figure 13) and the confusion matrix (Figure 11).

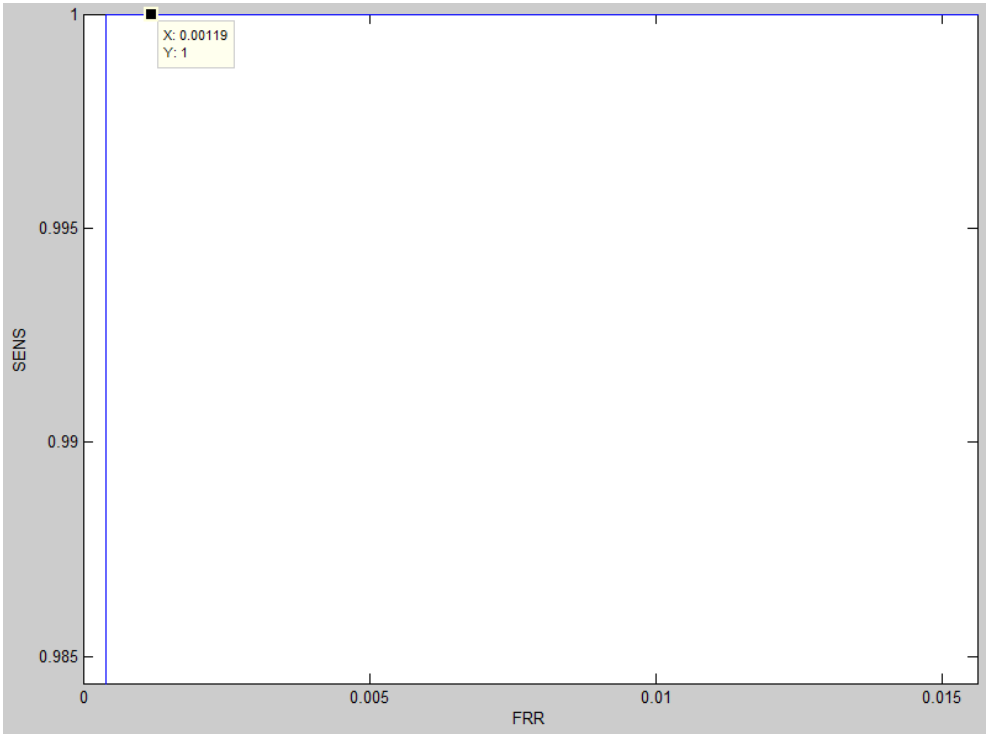

Figure 15: Zoomed ROC of the test implementation 2 for the CTCF profile

3.2.5. Implementation 3: Scoring Visualization of Single Sequences, Based on the Sliding Window Method

A third test implementation visualizes the window test for one particular sequence. This test does not follow the sixth step of threshold estimation. Rather, a visualization of likely TFBS locations shall be given. For doing this, we have been using the resulting Viterbi scores and the match area of the Viterbi algorithm.

The procedure follows steps 1 to 5 alike the second implementation, but in this case for only one TFBS profile. Furthermore, only one sequence has been left out for decoding.

The resulting scores of each window can be shown as graphs over the tested sequence. By doing this, we give a visualization of the scoring peaks of eventual TFBSs.

In order to produce a continuous graph for the whole sequence we have been computing the *average of the simple corrected Viterbi score* for each nucleobase position. Thus, we assign to each nucleobase position i the following score

$$s(i) = \frac{\sum_k^l \tilde{V}(w_{k\to l})}{|w|}, l \leq\ i \geq k \tag{3}$$

where $\tilde{V}(w_{k\to l}) = V^E(L)$ of the sequence segment $w_{k\to l}$.

This works as follows: We move the window, as described in the second implementation, over the sequence. For each position of the window we decode its sequence. We assign the

resulting score, of each window, to each nucleobase within the window. After decoding the whole sequence, we take the average of the assigned scores for each nucleobase. The graph of the *average simple corrected viterbi score* over the sequence gives a continuous graph of the probability for each nucleobase being a part of a TFBS.

Another way of visualizing the *simple corrected viterbi score*, is to assign the score not to each nucleobase, but to the one in the middle of the window. In this case, we do not have to take an average, because we assign only one score for each position. For the sake of clarity of the visualization, we artificially decreased this score by -10. Furthermore, now values would be assigned to the nucleobases of the positions 0 to $\frac{|w|}{2}$ and $|s| - \frac{|w|}{2}$ to $|s|$, since those do not occur as the middle of a window. Therefore we assign an initial value of -10 to these nucleobase positions. For all other nucleobase positions we assign the score

$$s(i) = \tilde{V}\left(w_{i-\frac{|w|}{2} \to i+\frac{|w|}{2}}\right) \tag{4}$$

We can also visualize likely locations of TFBSs, by showing the absolute frequency of each nucleobase being a part of the match area, see (5). This does not work for a window size $|w| = |TFBS|$, since it is highly likely that the nucleobases of the testing sequence, will be processed by match states equally often. But by increasing the window size we can apply this method. We decided for a window size of $|w| = 1.5 \cdot |TFBS|$ as a tradeoff between speed performance and significance of the results. The absolute frequency of being a part of the match area, does not give information about how likely a particular region is a TFBS, but rather gives the information which regions, are highly unlikely a TFBS. This graph usually results in a very recognizable shape of ups and downs, for which we call this a *Manhattan I* graph.

$$s(i) = \#(i \in \mu) \tag{5}$$

A window size of $|w| > |TFBS|$ let us also reconsider the assignment of the Viterbi scores to the nucleobases. Since we expect, that several regions of the testing sequence are potential TFBSs we expect those to be marked as match areas, more often than the surrounding regions. The match area thus is "more responsible" for the score than the surrounding inserts. So it comes naturally to mind to assign the scores only to the nucleobases which are part of the corresponding match areas. Since we assign a variable amount of scores to each nucleobase, again we have to take the average. We call the resulting graph *Manhattan II* and define it as shown in (6).

$$s(i) = \begin{cases} i \notin \mu \rightarrow -15 \\ i \in \mu \rightarrow \dfrac{\sum_k^l \tilde{V}(w_{k \rightarrow l})}{\#(i \in \mu)} \end{cases} \quad (6)$$

Another graph (orange) visualizes the most likely locations after the *viterbi simple corrected score* as ranks. We assign a value of 10 to the most probable region, and decrease the value for every further one by 1. But every position being part of the match area once, has a value of 1 at least. If likely regions overlap, the higher rank will be preserved.

Fehler! Verweisquelle konnte nicht gefunden werden. shows a visualization of the above described test implementation for the sequence *MA0139.1 CTCF 3*. The TFBS (marked by the peak of the green graph) has been classified correctly (orange graph). But, another region (first peak of orange graph) shows a high probability for being a TFBS as well. This region may be a repeat, whereas the sequence of that region is not equal to any reference TFBS of the MA0139.1.sites file. Obviously, this visualization makes it easy for a biological researcher to determine likely locations for TFBSs.

4. Conclusion

In summary, the use of pHMMs and the Viterbi algorithm is a qualified method for predicting TFBSs. In order to predict all likely locations of TFBSs, segmentation of testing sequences brings an added value to these methods. Furthermore, the specific use of match areas leads to new methods to assess and visualize likely TFBS locations.

Training of pHMMs by ungapped MSAs, in terms of TFBS prediction is highly similar to usual methods like position weight matrices and consensus sequences. But using those provides the options to use the Viterbi algorithm and finally to extend the method in order to map gaps and insertions to the model.

Classification of sequences (or sequence segments) as non TFBSs is not possible with current methods and makes up one of the hitches for using pattern recognition methods in this context.

The reimplementation of the decoding algorithms and separation of alphabets allows the framework to decode sequences notated in arbitrary alphabets. Furthermore the implementation of the class *Decoder*, provides several new methods for a convenient access to the output of the decoding algorithms. These methods concern the computation matrices, the backtrack path, the match area and the full set of scores.

A seamless integration of the methods in HMModeler is planned. Furthermore, the methods should be extended by a multi-step procedure for realigning the sites of the JASPAR database allowing gaps (deletions) and insertions.

Bibliography

[1] Aigner, M., 2012. *Transcription Factor Binding Site Prediction by the Use of Profile Hidden Markov Models*. Master Thesis, Salzburg University of Applied Sciences, Salzburg.

[2] Wellcome Trust Sanger Institute, 2012. *PFAM* [Online]. Available at: http://pfam.sanger.ac.uk/ [Accessed 8. August 2012].

[3] SIB Swiss Institute of Bioinformatics, 2012. *UniProtKB/Swiss-Prot* [Online]. Available at: http://web.expasy.org/docs/swiss-prot_guideline.html [Accessed 9. August 2012].

[4] University of California, Berkeley, 2012. *ASTRAL* [Online]. Available at: http://astral.berkeley.edu/ [Accessed 8. August 2012].

[5] University of Copenhagen, 2012. *The JASPAR database* [Online]. Available at: http://jaspar.cgb.ki.se/ [Accessed 9. August 2012].

[6] U.S. Department of Energy Genome Program, 2012. *Human Genome Project Information* [Online]. Available at: http://www.ornl.gov/sci/techresources/Human_Genome/home.shtml [Accessed 8. August 2012].

[7] Durbin, R., et al., 1998. *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids*. Cambridge: Cambridge University Press.

[8] Orengo, C. A., Jones, D. T., and Thornton, J. M., 2003. *Bioninformatics: Genes, Proteins and Computers.* Oxford: BIOS Scientific Publishers Ltd.

[9] Krogh, A., et al., 1994. Hidden Markov Models in Computational Biology: Applications to Protein Modeling. *Journal of Molecular Biology*, 5 (235), 1501-1531.

[10] Washington University in St. Louis, 2012. *Profile HMMs* [Online]. Available at: http://www.csb.yale.edu/userguides/seq/hmmer/docs/node9.html [Accessed 10. August 2012].

[11] Alberts, B., et al., 2008. *Molecular biology of the cell*. New York: Garland Science.

[12] Hughes, T. R., 2011. *A Handbook of Transcription Factors.* New York: Dordrecht.

[13] Leonard, S. A., 2002. IUPAC/IUB Single-Letter Codes Within Nucleic Acid and Amino Acid Sequences. *Current Protocols in Bioinformatics*, Appendix 1A-Appendix 1B.

[14] Day, H., and McMorris, F.R., 1992. Critical comparison of consensus methods for molecular sequences. *Nucleic Acids Research*, 20 (5), 1093-1099.

[15] Stormo, G. D., 2000. DNA binding sites: representation and discovery. *Bioinformatics*, 16 (1), 16-23.

[16] Graf, R. J., 2010. *Multiples Sequenzalignment mit Hidden Markov Modellen*. Master Thesis, Salzburg University of Applied Sciences, Salzburg.

[17] Hütt, M.-T., and Dehnert, M., 2006. *Methoden der Bioinformatik: Eine Einführung*. Berlin Heidelberg: Springer.

[18] Reiter Horn, D., Houston M., and Hanrahan, P., 2005. Supercomputing 2005 *Proceedings of the ACM/IEEE SC 2005 Conference. ClawHMMER: A Streaming HMMer-Search Implementatio. Seattle, 12-18 November, 2005*. New York: The Association for Computing Machinery.

[19] Viterbi, A. J., 1967. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory*, 13 (2), 260-269.

[20] Barrett, C., Hughey, R., and Karplus, K., 1997. Scoring hidden Markov models. *Bioinformatics*, 13 (2), 191-199.

[21] Auer, F. F., 2009. *Scoring Schemes and Parameter Prediction for Profile HMMs*. Diploma Thesis, Salzburg.