Sequence-based protein classification:

binary Profile Hidden Markov Models and propositionalisation

A thesis submitted in fulfilment
of the requirements for the degree
of
Doctor of Philosophy

in
Computer Science

at
The University of Waikato

by
Stefan Mutter

THE UNIVERSITY OF
WAIKATO
Te Whare Wānanga o Waikato

Department of Computer Science
Hamilton, New Zealand
April 2011

Attraversiamo – Let's cross over[1]

[1] from E. Gilbert. *Eat, pray, love* (page 346). London, England: Bloomsbury Publishing, 2006.

Abstract

Detecting similarity in biological sequences is a key element to understanding the mechanisms of life. Researchers infer potential structural, functional or evolutionary relationships from similarity. However, the concept of similarity is complex in biology. Sequences consist of different molecules with different chemical properties, have short and long distance interactions, form 3D structures and change through evolutionary processes.

Amino acids are one of the key molecules of life. Most importantly, a sequence of amino acids constitutes the building block for proteins which play an essential role in cellular processes.

This thesis investigates similarity amongst proteins. In this area of research there are two important and closely related classification tasks – the detection of similar proteins and the discrimination amongst them. Hidden Markov Models (HMMs) have been successfully applied to the detection task as they model sequence similarity very well. From a Machine Learning point of view these HMMs are essentially one-class classifiers trained solely on a small number of similar proteins neglecting the vast number of dissimilar ones. Our basic assumption is that integrating this neglected information will be highly beneficial to the classification task. Thus, we transform the problem representation from a one-class to a binary one.

Equipped with the necessary sound understanding of Machine Learning, especially concerning problem representation and statistically significant evaluation, our work pursues and combines two different avenues on this aforementioned transformation. First, we introduce a binary HMM that discriminates significantly better than the standard one, even when only a fraction of the negative information is used. Second, we interpret the HMM as a structured graph of information. This information cannot be accessed by highly optimised standard Machine Learning classifiers as they expect a fixed length feature vector repre-

sentation. Propositionalisation is a technique to transform the former representation into the latter. This thesis introduces new propositionalisation techniques. The change in representation changes the learning problem from a one-class, generative to a propositional, discriminative one. It is a common assumption that discriminative techniques are better suited for classification tasks, and our results validate this assumption.

We suggest a new way to significantly improve on discriminative power and runtime by means of terminating the time-intense training of HMMs early, subsequently applying propositionalisation and classifying with a discriminative, binary learner.

Acknowledgements

I am deeply grateful to anyone who has helped me, supported me, given me the strength to persevere, or has been simply there for me.

My sincere thanks go to my two main supervisors Bernhard Pfahringer and Geoff Holmes who have provided the guidance and academic expertise and experience necessary to complete this thesis and who taught me valuable research skills. Both of you brought together a unique and invaluable mix of attention to detail without losing the bigger picture. Bernhard and Geoff, thank you very much. I also want to thank my third supervisor Michael Mayo for his role on the supervisory panel.

Thanks for financial support go to the Computer Science Department at the University of Waikato and its Machine Learning Group, especially Bernhard Pfahringer, for providing a scholarship. I am also very grateful to the Livestock Improvement Corporation (LIC) for their funding through the Patrick Shannon Doctoral Scholarship.

I also want to express a big thank you to Jacqui Elphick who provided excellent technical support and became a friend. To all other technical and administrative staff goes a big thank you, too.

Additionally, I want to thank fellow, present and former, PhD candidates especially Andrea Schweer and Judy Bowen. Andrea and Judy, thanks for your help, academic expertise and support as friends. To all colleagues at the Faculty of Computing and Mathematical Sciences a big thank you.

Last but not least, I want to say sincerely thank you for the support from my family and friends. This thesis could not have been completed without you. Special thanks go to the Lowe family (my NZ family) for their continuous, generous support and the Heron family for allowing me to go on a writing retreat to their lovely family batch in Te Mata, Coromandel at no cost.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This thesis presents research to further advance the discrimination between, and the detection of, similar proteins. This statement not only summarises the thesis' work, it also highlights four key aspects of the problem.

- First, it identifies the subject of the research, namely proteins. These key molecules of life are represented by their sequence of amino acids.
- Second, the key to this problem is sequence similarity.
- Third, the detection of similar proteins involves two classes – those which are similar forming the so-called target class and those which are not. We label the latter sequences as negative and the former as positive. This is a binary, discriminative problem.
- And last, the discrimination between similar proteins implies that there are two or more different classes of similar proteins that need to be distinguished from one another. Thus, this constitutes a multi-class, discriminative problem.

This list already reflects an inherent property of this area of research. The first two observations are from Biology and Bioinformatics whereas the last two originate from the Machine Learning and Statistical Modelling communities. Both communities have put in considerable effort to successfully advance their methods, and to bridge the gap between the communities by combining methods. From a Bioinformatics point of view, Larrañaga et al. (2006) see Machine Learning as a crucial tool to generate knowledge from data. Frank et al. (2004, 2005) coming from a Machine Learning background introduce Bioinformatics as

an important field of application for Machine Learning. To bridge the aforementioned gap a sound understanding of both Machine Learning and Bioinformatics is necessary to utilise well-suited tools in correct ways and to define, represent and evaluate a problem in this area of research.

For any scientific research it is important to find the best way to define a problem or to formulate a specific research question. The reason is that the question asked pre-determines its answer or pre-determines and potentially limits the scope of the search; apart from the fact that the question and the answer need to be scientifically correct and able to be validated. For work that uses methods from Machine Learning, Statistical Modelling and Bioinformatics, it is essential to use the techniques from the different communities in a sound way and combine them meaningfully and not arbitrarily.

We will first introduce the Machine Learning nature of the problem and then look at techniques from Bioinformatics for similarity.

1.1 The Machine Learning nature of the problem

A prominent feature of most problems that deal with the detection of, or discrimination between similar proteins, is dataset imbalance.

Figure 1.1 provides an intuitive grasp of the problem. Like the tip of an iceberg, the number of proteins belonging to the positively labelled target class is small compared to the vast amount of negative data; that is all proteins that are dissimilar. Machine Learning approaches that try to infer a decision boundary symbolised by the water level to discriminate between positively labelled similar proteins and negatively labelled dissimilar proteins have to take the dataset imbalance into consideration. This is true not only for learning but also for evaluation. An algorithm that figuratively predicts that the iceberg is entirely submerged into the water, meaning all proteins are dissimilar, will be correct for the vast majority of instances. However, such an approach is not useful at all for recognising the target class symbolised by the tip of the iceberg.

Machine Learning research offers four different strategies to deal with the situation (Chawla et al., 2004) and this thesis will employ all four of them in a sensible, problem-specific way.

The first strategy is to sample the negative class and only use representatives

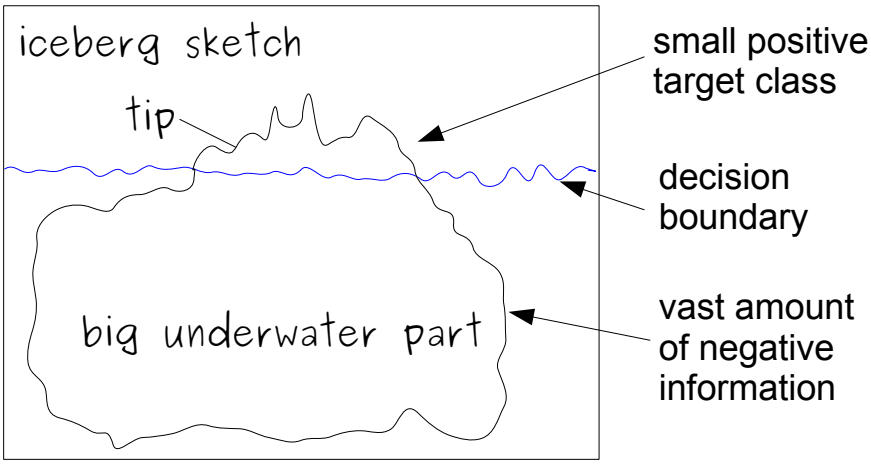


Figure 1.1. *Sketch of an iceberg. The figure compares the imbalanced nature of a dataset to an iceberg. In a discriminative task the waterline symbolises the decision boundary between the positive and the negative class.*

for training. This work will present a simple and fast way to sample the negative class that is competitive with more sophisticated but slower methods.

Secondly, the evaluation measure has to be chosen carefully. Simply looking at the percentage of correctly predicted instances as in the above example is not a sensible way to assess the predictive performance of an approach. The algorithms in this thesis are compared against one another using their area under the receiver operating characteristics curve (Bradley, 1997; Metz, 1978).

Another strategy is to select a number of features for the positive and the negative class that are strongly discriminatory. In order to exploit this strategy, methods are needed to extract features from proteins. This thesis will use existing approaches and introduce new ways to create discriminatory features.

The final strategy is to use one-class classification. In terms of the iceberg example, that means only building a model for the tip of the iceberg, completely ignoring the submerged part. Thus, imbalance is not a problem. In terms of classifying proteins, the model is built from information from the positive sequences only. At prediction time a so-called prediction cut-off decides class-membership. One-class classification is also known as outlier, anomaly or novelty detection. Its main use is in problems where there are no, or insufficient, negative examples. However, in our area of research there is a vast amount of negative

information. To establish the decision boundary adding negative information is helpful. Thus, we will transform a one-class approach to a binary one without ignoring the challenges faced by imbalance. This will lead to a better predictive performance, simply because it uses additional information. However, care has to be taken concerning the runtime of the binary approach as the straightforward incorporation of additional negative information increases runtime.

Independently from treating the task as a one-class or binary problem, the question is how to discriminate between ice from the tip and the underwater part of the iceberg, or how to discriminate between positive instances and negative ones to establish the decision boundary. Figuratively and literally, similarity is the key to answer this question. Bioinformaticians have done extensive research in this area.

1.2 Proteins and similarity

The concept of similarity is complex in Biology. Concerning amino acid sequences, it goes far beyond the fact that some amino acids are more closely related to one another than to others due to their chemical nature. Parts of the sequence form local, regular 3D structures like helices. On top of these local substructures the protein has an overall 3D structure. Therefore, even distant amino acids in the sequence can be closely related due to the local or overall shape of the protein. Thus, the amino acid sequence does not contain all the important information about a protein. However, its information can be accessed far easier than the 3D structure of a subpart or the entire protein. Additionally, compared to the small number of known structures, there is a vast amount of unknown sequences. In these cases, sequence similarity is a key to infer new knowledge as the underlying principle is that similar sequences lead to similar structures and similar structures might link to a similar function of the protein in the organism.

However, the concept of sequence similarity is even more complex as the biological world is not stable but changes continuously through evolutionary processes. There is an evolutionary pressure on proteins and proteins better adapted to their function and environment perform better and therefore, are more vital. But, even though the evolutionary pressure is on the protein as a

whole, evolution of proteins works through changes in the amino acid sequence. Different sequences and parts of the sequences might evolve at different rates. For example, a region of a sequence responsible for a vital function of the protein is most likely conserved. Thus, sequence similarity can be used to determine evolutionary relationships.

Hidden Markov Models have proven to be very successful in modelling sequence similarity especially remote ones (Söding, 2005). Krogh et al. (1994) introduced a Hidden Markov Model specialised for proteins. They call it a Profile Hidden Markov Model (PHMM). Because of PHMMs' success in detecting even remote similarities, they are the basic model used in this thesis. Apart from the fact that they capture the concept of sequence similarity well, these graphical models allow probabilistic scoring of similarity.

1.3 Bridging the gap

This section brings together knowledge of the nature of the problem from Machine Learning and the highly specialised sequence similarity model from Bioinformatics.

From a Machine Learning perspective on PHMMs, there is one fundamental observation. PHMMs are trained on instances belonging to the positive target class only. Therefore, they are a one-class classifier. This observation forms the basis of our work. Research has used this fact implicitly (Jaakkola et al., 1999). It is only recently, that scientific work explicitly acknowledges PHMMs as one-class classifiers (Bánhalmi et al., 2009; Mutter et al., 2009).

A one-class classifier needs a prediction cut-off for detection or classification. This cut-off is not solely based on the probabilistic scoring provided by the PHMM. Researchers use the odds of the score against a score from a null hypothesis (otherwise called a null model). These null models vary in complexity starting from treating each amino acid in each protein equally likely.

The fact that PHMMs are one-class classifiers has implications on the kind of problems that can be solved by them. Being a one-class classifier it can only address a binary problem. It is therefore well suited for the binary detection task or protein classification if the underlying problem is binary. Multi-class datasets in protein classification need to be transformed into binary ones to be able to

take advantage of PHMM modelling. The proposed transformation creates n binary datasets out of a multi-class dataset with n classes. In each of the binary problem specifications, one class is treated as the positive target class and all others together form the negative class. Therefore, these datasets have the same structure as the binary ones used in the closely related detection task. This thesis claims that this transformation is not only advantageous because PHMMs can be used to model similarity, but also because biological research questions are in most cases binary. For example, the question of whether a protein can be found in the nucleus or not will be asked more often than the question of whether a protein can be found in the nucleus or in the cytoplasm.

Figure 1.2 gives a first broad graphical overview of the ideas presented in this thesis and what the thesis seeks to achieve. Light gray coloured models refer to

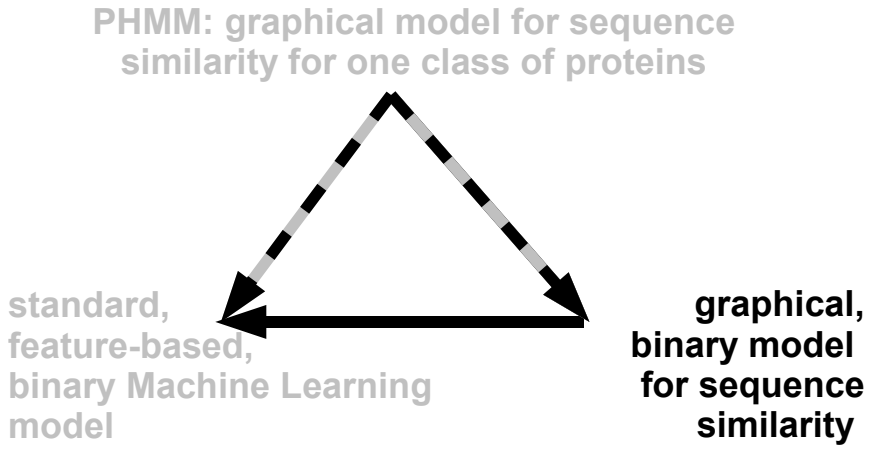


Figure 1.2. *A broad, graphical introduction to the ideas of this thesis on how to bridge the gap. References to existing methods and models are coloured light gray, whereas black indicates new ideas and models presented in this thesis. Arrows with both colours mark an extension or a new use of an existing technique.*

existing models, whereas new models presented in this thesis appear in black. Newly developed methods are symbolised by a black arrow, whereas an arrow with both colours indicates an extension or new use of an existing technique. As the figure shows, this thesis will transform a PHMM into a binary classifier. We will refer to this classifier as a binary PHMM.

Another area of research is to make the information captured in a PHMM available for standard, highly optimised Machine Learners as Figure 1.2 shows.

Jaakkola et al. (1999) have presented an approach that creates numeric features from a PHMM for positively and negatively labelled sequences. Therefore, not only does the input representation change from a graph to numeric features, but also the problem definition changes from one-class to binary. We will propose a different approach that results in fewer features. Nonetheless, these features are competitive concerning their predictive performance.

Chapter 2 will introduce the concepts behind and around the ideas presented in this work. As a consequence, by the end of that chapter, Figure 1.2 will be completed by concrete techniques that will be used, optimised or newly introduced in this thesis.

At first sight it might seem a good idea in terms of predictive performance to make use of the vast amount of negatively labelled sequences. However, this optimisation has an impact on runtime. It is one of the objectives of this thesis to keep this impact as small as possible. It is therefore notable that this thesis will show a strategy that improves both runtime and predictive performance compared to the performance of a PHMM.

This is one of the objectives and contributions of this work. The following section will introduce the objectives, whereas Section 1.5 highlights the thesis' contributions.

1.4 Thesis statement and objectives

This thesis argues that transforming a PHMM – a very successful, probabilistic, graphical one-class model for sequence similarity – in different but sensible ways into a binary classifier is beneficial in terms of predictive performance and sometimes even in terms of runtime.

The work in this thesis strictly follows the principle of preferring simpler ideas whenever these ideas lead to competitive results.

The discussion of the previous sections can be summarised into the following objectives:

- To investigate closely the related areas in Bioinformatics and Machine Learning for sound problem definition, representation and evaluation.
- To compare results in a meaningful way.

- To make use of the vast amount of negative instances as time efficiently as possible.
- To compare different null models and investigate the influence of negative information on the null model.
- To investigate PHMM training more closely by evaluating after each iteration of the training algorithm.
- To find new ways to create features and different ways of joining them together.
- To investigate the influence of features built from a model based solely on negative instances.

1.5 Contributions

This thesis contributes in the following ways to the research communities in Machine Learning and Bioinformatics:

- A new approach for binary classification by extending an existing one-class classifier that leads to better predictive performance.
- A demonstration that a simple sampling strategy is competitive compared to more sophisticated methods and training with full information.
- A derivation showing that negative information can act like a null model.
- A new strategy for classification based on PHHMs that optimises predictive performance and time efficiency.
- The development of simpler ways to create features.
- An extension of feature generation to binary PHMMs.
- Providing information about statistical significance to compare results.

The contributions of this thesis are broadly reflected within its structure, outlined in the next section.

1.6 Thesis structure

The next chapter discusses concepts in biological sequence classification. It will introduce related work from Bioinformatics and Machine Learning, in particular, PHMMs. Related approaches will be evaluated according to criteria derived from our analysis of Machine Learning techniques. At the end of the chapter, we will outline our proposed approach in more detail.

Chapter 3 introduces the datasets used in our experiments. Additionally, it presents the setup of all classifiers and provides more details about evaluation, especially how statistical significance is calculated.

The following three chapters provide all information about the experiments themselves and their results. Chapter 4 compares the original PHMM implementation with its binary counterpart. It also introduces the idea of sampling the negative class. In accordance with preferring simple ideas whenever they work at least equally well, experiments will show that the simplest and fastest sampling idea leads to competitive results compared to other sampling strategies and to using the full negative information. Another contribution presented in this chapter mathematically proves that a PHMM trained solely on negative instances is equivalent to a null model. Thus, the chapter introduces a problem- and sequence-specific model that acts like a null model.

The second experimental chapter, Chapter 5, proposes ways to create features from PHMMs trained on the positive class only and compares it to an existing approach. Simpler ways of creating features lead to competitive results. The most remarkable contribution of this chapter is the proposal of a strategy that increases predictive performance of the model and decreases runtime. Results suggest to train a PHMM only for one iteration of its training algorithm and create features from this partially trained PHMM. These features discriminate competitively to the fully trained basic PHMM. In addition, it is faster than to fully train a PHMM.

Chapter 6 introduces how to create features from binary PHMMs. These approaches are not always superior to the ones presented in Chapter 5 in terms of predictive power even though the underlying PHMMs perform significantly differently in most cases.

The thesis' conclusions are presented and summarised in Chapter 7. This thesis ends with an outline of future work.

1.7 Summary

From a sound understanding of the underlying problem representation and taking a Machine Learning perspective, the thesis' main contributions stem from transforming a one-class classification problem characterised by a vast amount of negative data into a binary problem. These new binary models exploit the information from the positive and the negative class, therefore, leading to an overall positive effect of adding negative information. Not only does predictive performance take advantage of the added information, this thesis will introduce an interesting new strategy that boosts discriminative power and does not increase runtime even though a vast number of negative instances are considered.

Chapter 2

Concepts in biological sequence classification

Sequence analysis plays a major role in Biology and Bioinformatics as the keys to understand life are found in sequences. DNA and RNA are sequences of nucleotides. Proteins are built from a chain or sequence of amino acids. This work addresses the classification of proteins based on this sequence. Thus, the term sequence refers to a sequence of amino acids.

In general, biologists differentiate between four different structural aspects of proteins. The primary structure, or primary sequence, of a protein refers to its amino acid sequence. The secondary structure, or sequence, contains regularly repeating local structures such as alpha helices, beta sheets and turns, whereas the tertiary structure is the overall 3D shape of a single protein. Structures formed by several protein molecules are referred to as quaternary structure. For most proteins, we only know their primary sequence as it is experimentally more expensive to determine the secondary, or, especially, the tertiary structure. This is why most approaches in protein classification are based on the primary sequence only. Some approaches do use the secondary structure as well, either experimentally determined ones or predicted ones. However, in this thesis, we only use the primary sequence. Therefore, within this thesis, the terms sequence, amino acid sequence and primary sequence are synonyms and all refer to the primary sequence of amino acids of a protein.

This chapter first introduces the concept of sequence similarity which is the key to inferring knowledge about structure, function or evolutionary relationship of a newly identified protein from the pool of proteins with known structure

and function. The task is to identify proteins that are significantly similar, the so-called homologs. Des Jardins et al. (1997) refer to similarity as the gold standard for homology detection and protein classification. The chapter will show from previous work that Hidden Markov Models capture similarity well, especially of remote homologs (Madera and Gough, 2002). However, they are considerably slower than other methods (Söding, 2005).

Subsequently, the chapter introduces Profile Hidden Markov Models (PHMMs) (Durbin et al., 1998; Krogh et al., 1994). The underlying graph of these models is especially designed to model similarities in proteins. Apart from the fact that PHMMs have been shown to be well-suited to remote homology detection and protein classification, we further motivate their use by presenting additional advantages. Consequently, we show how our approach deals with challenges of PHMM based modelling. Null models are the important concept introduced in this section, they play a crucial part in scoring a sequence with a PHMM and this thesis contributes to the area of null model research.

In order to achieve advances in the area of remote homology detection and protein classification it is crucial to understand the Machine Learning theory and concepts around PHMMs and the tasks to which these models are applied. First of all, a deeper understanding of the Machine Learning aspects allows categorisation of existing, related approaches and seeing where they fall short or where their assumptions are incorrect. Second, it enables us to build our approach with a sound understanding of the nature of the learning problem, its representation and evaluation. The second half of the chapter will cover these aspects starting with Machine Learning theory. In the following, this theory will be applied to define five criteria to evaluate previous work in the area of remote homology detection and protein classification. At the end of this chapter, we introduce our approach.

2.1 The basic concept of sequence similarity

“Evolution is a tinkerer and not an inventor” (Durbin et al., 1998; Jacob, 1977). Sequences are not newly invented, they change in evolutionary processes. Therefore, sequence similarity is a key to understand and infer structural, functional or evolutionary relationships (Karplus et al., 1998; Notredame, 2002). The most

reliable way to infer this knowledge about structure and function is by direct biological experiment (Durbin et al., 1998). However, these experiments are expensive and time consuming. Additionally, there is a vast amount of unclassified proteins with known primary sequence.

Amino acid sequences that are considered similar are called homologous sequences. Consequently, the task to identify similar and therefore related proteins is often referred to as homology detection.

Traditionally, there are methods to detect similarities in strings in Computer Science. However, biological sequences evolve in a complex molecular world. The crucial concept in homology detection is alignment.

It captures the three different ways sequences evolve.

- A specific position in the sequence can be altered. In Biology this phenomenon is called a point mutation. From a Computer Science viewpoint this is just a change of a single character in a sequence string.
- Parts of the sequence can be deleted.
- Parts of the sequence can be newly inserted.

If there is no difference in a specific region of a sequence, this region is labelled as conserved. Generally a conserved region may indicate a conserved function or structure. These conserved sequence positions are called matches. Therefore, in an alignment, we can find matches, insertions and deletions. By convention, matches are displayed as upper case letters, whereas inserts are represented as lower case letters. A deletion is symbolised by a dash. Alignments are always displayed with equal length so that conserved regions are below one another. This representation facilitates analyses by human experts. In this thesis, we align amino acid sequences globally, meaning we align the sequences as a whole. Figure 2.1 shows a simple, artificially created pairwise alignment.

The significance or relevance of an alignment is given by a score. However, scoring an alignment is not an easy task. As Durbin et al. (1998) point out, the scoring system requires careful thought and can be very complex. They argue

```
MMFFA           DDAAA--E
MMFFArrnsstnnrrEDPFMLWE
```

Figure 2.1. *A sequence alignment for two short artificially created amino acid sequences.*

that developing sensitive scoring schemes and evaluating the significance of alignment scores is a statistical task. We will show in the course of the thesis that problem representation, classification and evaluation techniques from Machine Learning can advance alignment based sequence classification.

Homology detection is the basis for classifying proteins. Homologous sequences form a family or class of proteins. Homology detection employs similarity scores from alignments to detect family or class membership. Classification discriminates between members of different classes or families based on similarity scores.

Certainly, there have been approaches in both homology detection and classification that use additional sources of information other than the primary sequence, e.g. Bystroff et al. (2000). Methods that are solely based on primary sequence information are the most popular.

The standard approach in homology detection starts with a single sequence and searches a database for homologs that are either used as the result or to refine the query. In classification, to the contrary, a fixed set of sequences is established prior to training and testing.

As methods in homology detection form the basis for classification approaches, we will succinctly discuss them. It is common to divide homology detection methods into three categories:

- methods that align two single sequences,
- methods that align a sequence with a so-called profile and
- methods that align two so-called profiles.

The ordering is consistent with the chronological order these techniques were developed. Eddy (1998) utilises the first two categories in his review paper about PHMMs in 1998. The same categorisation forms the basis of Wang et al. (2001)'s work on protein classification in 2001, whereas Söding and Huang and Bystroff in 2005 and 2006 respectively use all three categories (Huang and Bystroff, 2006; Söding, 2005).

2.1.1 Sequence-sequence based methods

Altschul et al. (1990)'s basic local alignment search tool (BLAST) is a widely

used tool to compare two sequences based on similarity. It uses a fast heuristic to approximate a pairwise alignment like the given in Figure 2.1. To score the alignment the tool uses probabilistic scoring matrices that are calculated with substitution rates for amino acids in closely related proteins. BLAST optimises a measure of local similarity and additionally estimates statistical significance of its scores.

The common procedure to detect homologs for a query sequence in a database or dataset of sequences is to create a pairwise alignment with each sequence and derive a score for each of these alignments. The homologs are the sequences that score high.

BLAST is by far the most popular tool to search for homologous sequences even though the methods from the next section have shown to be more sensitive (Biegert and Söding, 2009; Eddy, 1998) and are therefore better suited for remote homology detection.

2.1.2 Sequence-profile based methods

BLAST uses pairwise alignments. However, Park et al. (1998) point out that, if sequence similarity falls below 30%, pairwise methods are unlikely to detect relationships. Therefore multiple (homologous) sequences are aligned in one so-called Multiple Sequence Alignment (MSA). The MSAs in our experiments are built for at least 59 and up to 2589 sequences.

In general, an MSA represents or constitutes a model for a family or class of proteins. They are not only used in (remote) homology detection, they are also a model of a classifier. The common representation of an MSA in remote homology detection and classification is a profile. It contains the probability of each amino acid along the sequence positions in the MSA (Söding, 2005). Therefore it is a richer source of information than a single sequence. The alignment of a sequence to a profile has led to great improvements in sensitivity compared to a pairwise alignment of sequences (Söding, 2005). This approach is especially successful in the so-called twilight zone of sequence similarity (Doolittle, 1981) where sequence identity is smaller than 25% (Söding, 2005). Huang and Bystroff (2006) stress the need for more accurate alignments in this scenario. From the 23 datasets that we use in this thesis, 15 are from within the twilight zone.

There are two common profiles. The first one, called PSI-BLAST (position-specific iterated BLAST), extends BLAST to the concept of profiles and was developed by Altschul et al. (1997). The second profile representation uses Hidden Markov Models (Rabiner, 1989).

PSI-Blast

The PSI-BLAST algorithm starts with a single sequence and searches a database for homologous sequences. An MSA is built from these sequences. The output is a matrix with a column for each position in the alignment and a row for each amino acid. The entry refers to the probability of the occurrence of a specific amino acid in a specific alignment position. The database search process is iterated. In all but the first iteration the MSA is used to search the database. Subsequently, the MSA is updated using significant BLAST hits and a new matrix is calculated. PSI-BLAST uses weights for the sequences. In our approach we do not need to scan a protein database as we use already existing classification datasets. All sequences in our datasets are relevant. Thus, we do not weight input sequences.

Hidden Markov Models

Hidden Markov Models were first introduced in the field of speech recognition (Rabiner, 1989). But they have been very successful in biological sequence applications (Durbin et al., 1998; Krogh et al., 1994). They are a probabilistic, graphical model that serves the dual purpose of modelling a class of proteins in remote homology detection and discriminating against the class in protein classification. Because of their probabilistic nature, they allow scoring an alignment probabilistically. The most common Hidden Markov Models are the so-called Profile Hidden Markov Models (PHMMs) (Eddy, 1998). Krogh et al. (1994) designed them according to the structure of an alignment with match, insert and delete states. They form the basis of all our approaches. We will introduce them in detail in Section 2.2.

Several studies have shown that PHMMs perform better in remote homology detection and quality of alignments than profiles like PSI-BLAST. However, they are slower (Eddy, 1998; Krogh et al., 1994; Söding, 2005). Madera and Gough (2002) revealed in their study that PSI-BLAST is 30 times faster but performs

worse than an equivalent PHMM based approach. Their work also stresses the importance of a good quality alignment. In addition, the graphical structure of Hidden Markov Models and Profile Hidden Markov Models allows the construction of features for a sequence. In Machine Learning this feature construction process is called propositionalisation and we will introduce it in Section 2.4.4. Jaakkola et al. (1999) use propositionalisation of PHMMs in their approach to detect remote homologs. Liao and Noble (2002) use propositionalisations as well, however, pairwise alignments are the basis of their approach. These pairwise alignments do not have the rich graphical structure of a Profile Hidden Markov Model and therefore, allow the construction of fewer features. Liao and Noble (2002) use the score of the alignments.

2.1.3 Profile-profile based methods

In remote homology detection there is another category of methods that we will summarise briefly for completeness. Instead of using a sequence and a profile, two profiles are compared.

Yona and Levitt (2002) motivate a profile-profile based method, because the idea behind profiles is to achieve a concise, robust and powerful statistical representation of a protein family or class. They use the Jensen-Shannon divergence (Yona and Levitt, 2002) between probability distributions on profiles. The advantage is that this metric is proportional to the negative log likelihood that the two distributions represent samples drawn from the same source distribution.

Söding (2005) introduced a profile-profile method based on two PHMMs by generalising the scoring from a sequence alignment to a PHMM. His PHMM-PHMM alignment leads to a significantly higher sensitivity and better alignment quality than simple profile based methods.

2.2 Profile Hidden Markov Models

Hidden Markov Models (Rabiner, 1989) enjoy widespread use in Bioinformatics from gene finding to protein classification (Cherry, 2001).

A Hidden Markov Model describes a probability distribution over a poten-

tially indefinite number of sequences (Eddy, 1998). It consists of a sequence of states which are connected by transitions. Each of these state transitions has a probability and the probabilities of all transitions leaving a state sum up to one. Thus, the state-sequence forms a first order Markov chain. Each state has a symbol emission distribution for generating or emitting a symbol of a common alphabet. In this thesis, the alphabet is defined over all possible amino acids and can generate amino acid sequences. It is therefore a generative, probabilistic, graphical model.

Figure 2.2 illustrates a simple example of a Hidden Markov Model from Eddy's review paper on Profile Hidden Markov Models (Eddy, 1998). It consists of

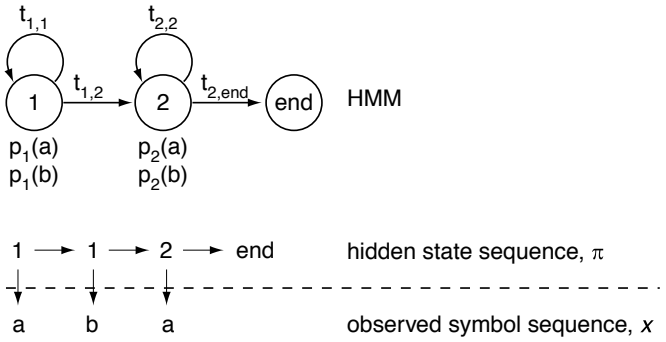


Figure 2.2. *A simple Hidden Markov Model. The figure is taken from Eddy (1998). It models sequences with two symbols 'a' and 'b' with two regions of different character distribution. It shows an example for a possible hidden state sequence π and an observed sequence of symbols. Note that the symbol sequence could have been generated by a different state sequence (1-2-2) with most likely a different probability.*

two states modelling regions with different character distributions. The states emit two symbols 'a' and 'b'. In this example the starting state to enter the HMM is state 1. We go through the model emitting or generating a sequence. In Figure 2.2 we observe the sequence 'aba'. However, we do not know from which states the symbols were emitted. The state sequence is hidden. This is a characteristic of a Hidden Markov Model. It is possible to infer the state sequence probabilistically. In the case of our simple example the possible state sequences are '1-1-2' or '1-2-2'. Most likely the two state sequences will have a different probability.

Krogh et al. (1994) developed a Hidden Markov Model specialised for protein modelling. It is called a Profile Hidden Markov Model (PHMM). Successful applications include homology detection and protein classification (Eddy, 1998; Haussler et al., 1993; Jaakkola et al., 1999). There are two well-known software tools that create and use PHMMs. These are SAM (Hughey et al., 2003; Karplus et al., 1998) and HMMER (Eddy, 1998). Other important applications are so-called PHMM libraries e.g. the Pfam database (Finn et al., 2010) and the PROSITE profile database (Sigrist et al., 2002). They contain a large collection of protein families represented by Hidden Markov Models and they allow researchers to determine whether or not the sequence under consideration contains one or more known domains.

This section introduces PHMMs emphasising their benefits and identifying challenges faced when using PHMMs as a tool to model and classify proteins. A PHMM is a generative, graphical, probabilistic representation of an MSA.

Figure 2.3 shows the graphical structure of a small PHMM. At first sight, we

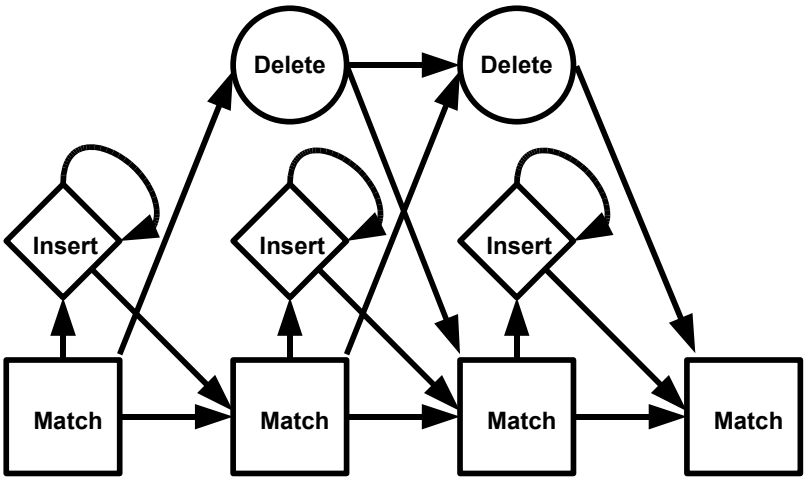


Figure 2.3. *The graphical structure of a PHMM*

can identify three different sets of states: match, insert and delete states. As the names suggest they model the three distinct states of any biological sequence alignment. For a single position these three states are mutually exclusive. A column is defined as a set of match, insert and delete states representing a specific position in an alignment. All but the first and last column of a PHMM consists of one match, insert and delete state. The first column has only a match

and insert state, the last column has only a match state. As a consequence all global alignments represented by a PHMM in this thesis begin and end with a match state. The PHMM from Figure 2.3 has four columns, whereas the PHMMs used in our experiments have 35 columns. Throughout this thesis the terms columns of a PHMM, length of a PHMM and number of match states of a PHMM are used interchangeably as they describe the same entity.

The delete states are non-standard states as they emit no symbols at all. Insert and match states emit symbols representing amino acids. In our PHMMs there are no transitions from insert to delete states and vice versa as these transitions are very unlikely in a biological setting and we only train emissions in match states (Durbin et al., 1998). We align sequences globally to our model. This means that the first sequence position is always modelled by the match state in the first column and the last sequence position by the match state in the last column.

Calculations in PHMMs in this thesis are performed in log space to avoid underflows (Durbin et al., 1998). For programming purposes we define $\log(0) = -\infty$.

Rabiner (1989) identifies three basic problems for Hidden Markov based modelling. In the context of PHMMs and protein modelling these problems are:

- How to choose a state sequence which is optimal in some sense. In a PHMM context, this question is how to calculate the best alignment of a test sequence to the PHMM. The answer determines which parts of a sequence are matches and insertions and whether or not some deletions have occurred and how likely the alignment is.

- How to compute the probability of an observed amino acid sequence given a PHMM. This problem is often referred to as scoring and it is crucial in remote homology detection or protein classification as the score expresses a measure of similarity. As we do all calculations in log space, the score will be the log likelihood of the sequence given the PHMM. However, we will see that more sophisticated scoring solutions are necessary.

- How to train the PHMM to represent a class or family of proteins from previously unaligned sequences which belong to this class or family.

All three problems can be solved by dynamic programming approaches. The first two problems are concerned with scoring and the last with training a PHMM.

We use the notions, formulae and descriptions from Durbin et al. (1998)'s book to further introduce the aspects of PHMMs that are important for this thesis, namely scoring and training. A succinct overview of the advantages and disadvantages of PHMMs follows. The presentation of PHMMs finishes with identifying challenges in scoring. For a comprehensive presentation of the topic, refer to Durbin et al. (1998).

2.2.1 Basic scoring algorithms

The first and the second questions are both related to scoring a sequence x under a PHMM H.

The biggest problem when using the log likelihood or the raw probabilities for scoring is that both are dependent on the length of the sequence (Durbin et al., 1998). One possible alternative is to use so-called log-odds scoring. In addition to the log likelihood $ll(x)$ of the sequence x under the PHMM H, we also calculate the log likelihood of the same sequence x under a so-called null model *NM*. The null model models the null hypothesis. It deliberately excludes the mechanism or pattern being tested (Gotelli, 2001). There are many different null models and they are a crucial part of scoring. Section 2.2.5 introduces them in more detail. The log-odds score $lo(x)$ is defined as follows:

$$lo(x) = ll_H(x) - ll_{NM}(x)$$

Therefore, the log-odds score allows the following interpretation.

- If $lo(x) = 0$ then the sequence x is equally likely being generated by the null model *NM* and the PHMM H.
- If $lo(x) < 0$ then the sequence x is more likely being generated by the null model *NM*.
- If $lo(x) > 0$ then the sequence x is more likely being generated by the PHMM H.

However, interpreting 0 as a prediction cut-off does not necessarily lead to the

highest accuracy. The best prediction cut-off has to be determined experimentally.

For the first problem of determining the best alignment of a new sequence a definition for optimality is required. This thesis defines the best alignment of a sequence x under a PHMM H as the one with the highest log-odds score, meaning the path π^* through the PHMM with the highest log-odds score.

$$\pi^* = \arg\max_{\pi} lo(x, \pi)$$

This path is well-known as the Viterbi path. It can be calculated using dynamic programming by implementing the following formulae from Durbin et al. (1998). Let $V_j^M(i)$ be the best path matching subsequence $x_{1\ldots i}$ to the submodel up to state j, ending with x_i being emitted by match state M_j. Similarly $V_j^I(i)$ is the score of the best path ending in x_i being emitted by insert state I_j and $V_j^D(i)$ for the best path ending in delete state D_j. Let $e_S(x_i)$ be the emission probability of x_i in match state or insert state S, q_{X_i} be the emission probability of x_i under the null model and let $a_{S_j S_{j+1}}$ be the transition probability from state S_j to state S_{j+1}. Then we can write the following three formulae defining a dynamic programming problem

$$\begin{aligned}
V_j^M(i) &= \log \frac{e_{M_j}(x_i)}{q_{x_i}} + \max \begin{cases} V_{j-1}^M(i-1) + \log a_{M_{j-1}M_j} \\ V_{j-1}^I(i-1) + \log a_{I_{j-1}M_j} \\ V_{j-1}^D(i-1) + \log a_{D_{j-1}M_j} \end{cases} \\
V_j^I(i) &= \log \frac{e_{I_j}(x_i)}{q_{x_i}} + \max \begin{cases} V_{j-1}^M(i-1) + \log a_{M_j I_j} \\ V_{j-1}^I(i-1) + \log a_{I_j I_j} \end{cases} \\
V_j^D(i) &= \max \begin{cases} V_{j-1}^M(i) + \log a_{M_{j-1}D_j} \\ V_{j-1}^D(i) + \log a_{D_{j-1}D_j} \end{cases}
\end{aligned}$$

These equations calculate the log-odds score of the best path. The path itself can be constructed by storing pointers to the entry that maximises the log-odds score at any sequence position x_i.

The answer to the second problem can be found in a similar way by using the so-called forward algorithm that, given a sequence x, calculates its probability

$P(x)$. This probability is defined as

$$P(x) = \sum_{\pi} P(x, \pi)$$

Therefore, instead of using the path that maximises the probability, we sum over all paths. This means to calculate the forward score, we just have to replace the maximum in Viterbi's formulae by sums. Care has to be taken when summing up in log space, because the logarithm of a sum of probabilities cannot be calculated from the logs of the probabilities without exponentiation and log functions. We follow Durbin et al. (1998)'s advice to implement this in a computationally efficient way. The log-odds of the forward score is the overall log-odds of the sequence x given the PHMM H. In the future, we will refer to this score just as the log-odds score.

The same log-odds score can be calculated by the backward algorithm as well. It works on the same principle as the forward one. It just starts summing up from the end of the sequence and model and goes backwards to the start.

Another important application of the forward and backward algorithm is to calculate the probability that observation x_i came from state k. This is the posterior probability $P(\pi_i = k|x)$. We use the posterior probability as a feature in some of our approaches. It is calculated according to the following formula. Let $f_k(i)$ be the probability of the observed sequence up to and including x_i, requiring that $\pi_i = k$, calculated by the forward algorithm, and let $b_k(i)$ be the probability of the observed sequence $x_{i+1}...x_L$ of length L, requiring that $\pi_i = k$, calculated by the backward algorithm.

$$P(\pi_i = k|x) = \frac{f_k(i)b_k(i)}{P(x)}$$

The asymptotic runtimes of the Viterbi, forward and backward calculations are all in $O(n \cdot k)$ where n is the length of the sequence and k is the number of states. The reason is that all dynamic programming matrices have a size of $n \cdot k$ and to calculate an entry we check at most three others. This is specific to PHMMs. For general HMMs the runtime is in $O(n \cdot k^2)$ as potentially there could be a transition from one state to all other states. As Figure 2.3 shows the maximum number of successor states in a PHMM is three.

2.2.2 Training

To train a PHMM from a set of unaligned sequences we use the so-called Baum-Welch algorithm which is a derivative of the expectation-maximisation (EM) algorithm (Baum, 1972; Durbin et al., 1998).

As a consequence it consists of two steps. First it calculates the expected number of times each transmission and emission is used given the training sequences. This involves a standard forward and backward calculation for each sequence. We add pseudo-counts to prevent these expected numbers from being 0. In a second step, the maximisation step, we use maximum likelihood estimators for each transition and emission to calculate their new values. Overfitting can occur when insufficient training data is provided. Training stops when the difference in the log likelihood between the old and new model is less than a pre-defined threshold.

We define A_{kl} as the expected number a transition a_{kl} is used by all training sequences in all positions as

$$A_{kl} = \sum_j \frac{1}{P(x^j)} \sum_i f_k(i) a_{kl} e_l(x_{i+1}) b_l(i+1)$$

where $\frac{f_k(i)a_{kl}e_l(x_{i+1})b_l(i+1)}{P(x)}$ is the probability that $a_{kl}$is used in position i in sequence x. Similarly the expected number of times symbol c appears in state k is defined as

$$E_k(c) = \sum_j \frac{1}{P(x^j)} \sum_{i|x_i^j=c} f_k^j(i) b_k^j(i)$$

The maximum likelihood estimator for a transition a_{kl} is given by

$$a_{kl} = \frac{A_{kl}}{\sum_{l'} A_{kl'}}$$

and the one for an emission $e_k(c)$ by

$$e_k(c) = \frac{E_k(c)}{\sum_{c'} E_k(c')}$$

The Baum-Welch algorithm will converge to a local optima. It is not guaranteed that the optima is a global one (Durbin et al., 1998).

As we calculate for each sequence in each iteration the forward and the backward algorithm, the runtime of the Baum-Welch algorithm for one sequence in one iteration of PHMM training is in $O(n \cdot k)$. Baum-Welch training is time intense especially compared to fast heuristic approaches like BLAST (Cherry, 2001).

2.2.3 Advantages

In a survey paper, Cherry (2001) discusses advantages and disadvantages of Hidden Markov Models in Bioinformatics. This section and the following one about disadvantages introduces his findings and shows how this thesis' work takes advantage of the benefits and reacts to the challenges of Hidden Markov Models. The points raised by Cherry apply to general Hidden Markov Models and therefore to PHMMs as well.

Statistical Grounding

Hidden Markov Models are built upon sound statistical theory. In terms of PHMMs for remote homology detection and protein classification, they make it possible to probabilistically score the degree of similarity represented in the alignment.

Graphical representation

All Hidden Markov Models have an underlying probabilistic, graphical structure of states and transition. In particular, PHMMs have a probabilistic graph covering the structure of a sequence alignment. Any structured graph is a source of information itself. Machine Learning provides tools to access the information in a graph and transform it into a representation that allows further evaluation. These so-called propositionalisation techniques will be introduced in Section 2.4.4.

Incorporating of Knowledge

First of all, in every PHMM prior knowledge has already been incorporated as its graphical structure is adapted to sequence alignments.

Another way to add prior knowledge is during initialisation. Due to the fact that the Baum-Welch algorithm only guarantees to converge to a local optima, careful parametrisation can help to avoid local optima and find the global one. Emission and transition probabilities can be initialised in a problem-specific manner.

Additionally, statistics incorporated into the training process can constrain the training (Brown et al., 1993).

Finally, it is possible to integrate sources of information other than the primary sequence in a Hidden Markov Model. Bystroff et al. (2000) include emission for the secondary structure and other chemical properties in each state of their Hidden Markov Model.

2.2.4 Disadvantages

There are some challenges to face when using Hidden Markov Models. In the following we will identify these challenges and show how this thesis responds to them.

Local optima

The pitfall in HMM training is that the Baum-Welch algorithm only guarantees to converge to a local optima. Haussler et al. (1993) suggest a problem-sensitive initialisation of the parameters of the model. The intuitive idea behind this approach is that when parameters are initialised near to the global maximum, it is less likely that the training algorithm will get trapped in a local optimum.

Durbin et al. (1998) use simulated annealing (Kirkpatrick et al., 1983). It allows getting away from local optima through stochastic choice of configuration, as opposed to always maximising the expectation. A similar, simple way is to introduce noise to counts of estimated numbers and reduce the amount of noise per iteration (Haussler et al., 1993).

This thesis does not address the problem of local optima when building a PHMM directly. However, one contribution of this thesis are methods to construct a fixed length feature vector out of a PHMM and use the resulting vector as input for standard, propositional learners. These learners can potentially compensate for the PHMM trapped in a local optima.

Markov principle

Standard Hidden Markov Models are essentially a first order Markov chain. Therefore, they can only model dependency in a very restricted way. The emission of a symbol at time t only depends on the emission at time $t-1$. This is of course a limited and incorrect assumption for most problems. In protein modelling long distance relationships in the primary sequence exist because of the three dimensional structure of the protein. However, PHMMs have shown to be a successful tool in this area of research (Durbin et al., 1998; Eddy, 1998; Karplus et al., 1998; Park et al., 1998). Therefore, this thesis does not address challenges imposed by the Markov principle and uses PHMMs that are first-order Markov models. Nonetheless, on the other hand, in other areas of research this limitation is crucial e.g. RNA folding problems (Cherry, 2001).

Speed

As Söding (2005) and Cherry (2001) point out, Hidden Markov Models are fairly slow in training and evaluation as they need to calculate various dynamic programming matrices based on all possible paths through the model.

One way to speed up PHMMs is to reduce the length of the PHMM. In the experiments conducted in this thesis, all PHMM have only 35 columns.

This thesis optimises protein classification in different ways. Traditionally, one PHMM on the positive class has been trained. This thesis suggests to train an additional PHMM on the negative instances. This leads to a higher discriminative power. However, to deal with the increase in runtime, we introduce sampling techniques for the sequences of the negative class.

Overfitting

Overfitting can occur in PHMM training. Brown et al. (1993) and Haussler et al. (1993) argue that the problem arises when there are too many free parameters to estimate from a small training set. The latter use a form of regularisation that is closely related to the model prior and can be used to bias the model in some specific direction. The former uses Dirichlet mixture priors. This approach reduces dimensionality by clustering the amino acid distribution into prototypical classes of distributions. Park et al. (1998) use sequence weighting in addition

to Dirichlet mixture priors.

There is another simple way to reduce the number of free parameters in a PHMM: restricting the number of columns. Brown et al. (1993) estimate a PHMM with 401 columns using 88 sequences. Their best parameter to sequence ratio is observed when estimating a PHMM with 147 columns using 400 sequences. In this thesis the number of columns is set to 35. Therefore, there are fewer parameters to estimate and the problem of overfitting is less apparent. We do not use regularisation or Dirichlet mixture priors to avoid overfitting. Our worst parameter to sequence ratio is 35 columns to 59 sequences, the best one is estimating the same length PHMM with 2589 sequences.

2.2.5 Concepts and challenges in scoring

Scoring an alignment is a crucial but non-trivial task as the score defines the measure of similarity between the sequence and the MSA represented by a PHMM. PHMMs have the advantage of having a probabilistic score. In this section we will have a brief look into three different areas that influence scoring. The first aspect of scoring is concerned with the input sequences that are used to train the model during the Baum-Welch training process. Karchin and Hughey (1998) suggest weighting training sequences to achieve a model that generalises better. The advantage of their methods is that weighting works quickly. Training sequences in this research project though all have equal weight.

As explained in Section 2.2.1, log-odds scoring is applied in PHMMs. It serves two purposes. First, it replaces the length-dependent log likelihood score. Second, it evaluates the score of a Hidden Markov Model in relation to a null hypothesis.

Barrett et al. (1997) identify two questions:

- What should the null model be? and
- What prediction cut-off should be used to consider a sequence a match to the model?

Neither question is easy to answer and for both questions there may not be a universal answer.

Choice of null models

In their work on scoring Hidden Markov Models Barrett et al. (1997) define a null model as a null hypothesis which is usually a simpler statistical model intended to represent the universe of sequences as a whole, rather than the group of interest. Gotelli (2001) has a different view on null models. Like Barrett et al. he considers a null model to represent the null hypothesis. However, he defines it as a model that deliberately excludes the mechanism being tested. Where Gotelli emphasises the deliberate exclusion of the mechanism being tested, Barrett et al. stress the universal representation of a null model. This thesis contributes to this general null model discussion. It introduces newly developed quasi null models that shed some light onto this discussion.

Barrett et al. (1997) additionally categorise null models into sequence-specific null models, model-specific null models, null models specific to both sequence and model and global null models.

Global null models are fixed for all scoring sequences and all models. An example of a global null model is to assume that each amino acid is equally likely in each position. We refer to this null model as the uniform null model.

A model-specific null model is fixed for all scoring sequences but different for each model. The distribution of amino acids in the training set or the geometric average of the match state probabilities are examples of this category.

The amino acid frequencies of a scored sequence is an example of a null model that is sequence-specific but the same for each model.

Barrett et al. (1997) did not investigate null models that are specific to both sequence and model. Their study showed that model-specific null models perform better than sequence-specific or global ones. They identify a null model based on the geometric average of the match state probabilities to be performing the best. However, the results differ from problem to problem. For example, for the ferredoxins problem the uniform null model works particularly well.

Karplus et al. (1998, 2005) introduce and test the reverse sequence null model. It is a null model specific to both sequence and model as it reverses the amino acid sequence of any test sequence and scores the reversed sequence against the same PHMM. Consequently, this type of null model corrects for length and composition biases and some other, more subtle effects (Karplus et al., 2005). They show that this null model is superior to model-specific null

models in remote homology detection. However, there are cases when the reverse sequence null model does not perform well, for example in the presence of directed motifs.

This thesis compares global, model-specific and a quasi null model that is specific to both model and sequence.

Prediction cut-off

The second problem, to define an optimal prediction cut-off value is important in any practical application of HMMs in homology detection or protein classification. It is evident that this cut-off depends on the specific detection or classification task. An optimal prediction cut-off maximises the accuracy of the model. Therefore, comparing the predictive power of different approaches based on accuracy is problem-dependent and might not generalise well. Additionally, it is an optimisation problem itself to find the right cut-off. Thus, this thesis does not calculate a prediction cut-off and consequently does not compare approaches based on their problem specific accuracy. In contrast, it compares the ability to rank the sequences correctly based on their similarity. The ranking is independent of the prediction cut-off and does not change when a different cut-off is chosen. We assume an approach to be favourable if it leads to a better similarity-based ranking of the sequences. The measure to evaluate this ranking is the so-called area under the receiver operating curve (AUC) (Bradley, 1997).

2.3 Summary

We introduced the basic concept of similarity and showed that sequence similarity, especially remote similarity, can be best captured by the use of Hidden Markov Models. For remote homology detection and protein classification there exists a specialised form of Hidden Markov Model, a so-called Profile Hidden Markov Model (PHMM). In addition, we discussed different forms of null models and that the right choice of null model is important for achieving good results.

One of the aims of the thesis is to contribute in protein classification by proposing new methods that discriminate better. However, there are several questions. First of all, what do we mean by 'better'? Secondly, how can we use

a PHMM as a classifier in a sensible way? To answer the second question we need to further investigate what the nature of the classification problem is that we want to solve. What type of classifier is actually a PHMM? What information is captured in a PHMM, and how can we assess it?

2.4 Machine Learning concepts

This section introduces concepts in Machine Learning stemming from the following three basic questions for any classification problem.

- How to define the classification problem?
- How to represent the data?
- How to evaluate?

Sound answers to these questions for remote homology detection and the protein classification task allow the definition of criteria to compare existing approaches. Not only will we gain evidence for limitations and shortcomings in existing approaches, but in addition, this thesis will show that a sound understanding of Machine Learning and adequate use of its methods can further advance protein classification.

The first three sections look into how to represent the classification problem. The following section discusses data representation whereas the last section focusses on evaluation.

2.4.1 One-class and binary classification

From a Machine Learning point of view a PHMM constitutes a one-class classifier. Yoon (2005) defines one-class classification as a classifier C that is only trained on one class of training instances, the so-called target class. There exists a measure $d(X)$ for the distance of any test sequence X to the target class. At testing time a prediction cut-off Θ on $d(X)$ decides about the class membership of X. Following this definition, a PHMM is a one-class classifier for the class or family it has been trained on. We will refer to this class as the positive class. The distance measure is the log-odds score and Θ is a problem-specific

prediction cut-off for accepting a sequence as a homolog. As discussed in Section 2.2.5, it is a difficult problem to infer the optimal Θ. In the remainder, we will call the standard PHMM as introduced earlier in this chapter a one-class PHMM. In this way we clearly identify a PHMM as a one-class classifier and, using this nomenclature, differentiate a one-class PHMM from binary extensions we propose later in this thesis.

One-class classification is often called outlier or novelty detection. Hempstalk (2009) categorises one-class classification into two groups. The first extends multi-class classification methods, whereas the second is based on density estimation. In the latter category, a statistical distribution is fitted to the data from the positive target class. Test instances with a low probability can be identified as not belonging to that class (Pearson, 2005). A PHMM models the joint probability distribution over sequences and states. Several other algorithms to estimate densities are used in one-class classification i.e. Parzan estimators or Gaussian mixture models (Clifton et al., 2007; Tarassenko et al., 1995). Tax (2001) provides a detailed overview of one-class classification.

Schölkopf et al. (2000) proposed a classification-based approach to one-class classification using a Support Vector Machine. It is implemented in the open-source software libSVM (Chang and Lin, 2001). We used this implementation on features created from a one-class PHMM. However, in our case, there was no substantial gain in AUC compared to a basic one-class PHMM based approach. Therefore, in this thesis we restrict the presentation of the one-class classification approach to pure PHMM based classification.

The fact that a PHMM is a one-class classifier is essential to this thesis. This has implications on the learning problem. A one-class PHMM can only solve a binary classification problem. It decides about class membership using a model built from one class only. Homology detection is indeed a binary problem. A protein either belongs to the class or family under consideration or it does not. In these binary settings, the class under consideration is referred to as the positive class, all other proteins form the negative class. In this thesis a one-class PHMM is trained on the positive class if not otherwise mentioned.

Usually the datasets for protein classification consist of more than two classes. Therefore, if we want to use PHMMs in protein classification, the problem needs to be transformed into a binary one. Basically, there are two approaches to this

task. In a one-vs-all setting, there is one positive target class and all other classes from the multi-class dataset are combined into the negative class. The other possibility is to define a binary classification problem for each pair of classes from the original multi-class problem. We strongly prefer the former method of transformation. First, as the positive target class is quite small, a pairwise approach leads to small datasets. In addition, the pairwise approach does not completely reflect a typical biological research question. Most of the time researchers in Biology are not interested in whether a protein belongs to class *A* or *B*. It is more important to decide whether it belongs to *A* or not and if not, whether it belongs to *B* or not. Using this setup, we can use highly optimised techniques from homology detection, e.g. PHMMs.

Common one-class classification approaches are characterised by the absence of negative training examples. However, in protein classification there is an abundance of them. Thus, this thesis will extend the pure one-class PHMM approach to a binary one. Bánhalmi et al. (2007) generate artificial negative examples based on the boundaries from a one-class classifier trained on the positive class. They subsequently train a Support Vector Machine on the binary dataset. They report an advantage of the binary approach. However, it is not clear whether the advantage is significant. As we have plenty of negative examples, we will use them instead of creating artificial ones.

To summarise, this thesis works on binary datasets for protein classification. They result from a transformation of multi-class datasets using the one-vs-all approach. These datasets are either used in a one-class or binary classification setting.

It is important to note, that the one-vs-all binary datasets are highly imbalanced. We will introduce techniques to deal with them in the next section.

2.4.2 Dataset imbalance

The binary datasets in protein classification and the databases in homology detection are characterised by the presence of a vast amount of negative sequences compared to the number of positive ones. Consequently, it is important to use Machine Learning techniques that are able to deal with dataset imbalance. In their editorial on learning from imbalanced data Chawla et al. (2004) discuss four techniques to deal with this challenge.

- First they propose one-class classification. In our work, the pure one-class PHMM approach follows this technique. In addition, we extend the basic one-class classification scheme by using two one-class classifiers to solve a binary problem that is characterised by class imbalance. This thesis proposes training a one-class PHMM on the positive class and a second one separately on the negative class. Thus, class imbalance does not negatively influence the performance.

- Second they propose sampling techniques. It is either possible to oversample the positive class or to undersample the negative class.

- The final recommendation is to use feature selection. Section 2.4.4 introduces feature generation techniques based on PHMMs. Features generated and selected carefully can achieve a higher separability between the positive and the negative class.

- They additionally propose a fourth technique that deals with evaluation of imbalanced datasets and will be discussed in Section 2.4.5.

This thesis explores all four approaches to deal with imbalanced datasets. For sampling, we undersample the negative class instead of oversampling the positive class. First of all there is a vast amount of negative data available, and secondly by undersampling the huge negative class, our approach will have a better performance concerning runtime. This is important as PHMM approaches are slow. However, as Drummond and Holte (2003) point out, undersampling of a class may lead to a higher variance due to non-determinism in the sampling process.

2.4.3 Generative and discriminative models

The previous section introduced a one-class and binary classification problem. This section exploits another important aspect of modelling. It is the distinction between generative and discriminative classifiers.

A Hidden Markov Model is a generative learner as it defines a joint probability distribution over a sequence x and all possible paths π through the model. According to Drummond (2006), the term generative refers to the fact that, having learnt the full joint probability, it is able to generate an artificial sequence

belonging to the model. We conducted a number of experiments with artificially created sequences. The idea was to use low scoring ones as negative examples. However, the PHMMs we use contain only 35 columns, the average sequence length in our datasets is much bigger. Artificial sequences generated by our PHMMs are very unlikely to be of the same length than the ones used for training because it is not very likely to continuously stay in an insert state in one column for a long time. In addition, we do not learn emissions in insert states and therefore, the uniform distribution of amino acids in insert states might differ considerably from the actual distribution. We believe these to be the reasons for the poor performance of this approach.

Using a PHMM in remote homology detection is a purely generative approach. To decide about family membership, the posterior probability of the sequence given the model is calculated. A PHMM like an HMM in general has underlying constraints to make the modelling of the full joint probability feasible. All the HMMs in this thesis are so-called first order Markov Models as an event i only depends on the previous event $i-1$. Due to this constraint it is possible to effectively calculate the posterior probability for a sequence X even though there is an exponential number of paths π. One possible solution is the forward algorithm as discussed earlier in this chapter. The aforementioned constraint is not true when modelling proteins, but PHMMs have proven to be successful nonetheless.

Commonly in homology detection, as Jaakkola et al. (2000) explain, there exists a PHMM H_1 and a corresponding null model H_0. Both calculate the posterior probabilities $P(X|H_1)$ or $P(X|H_0)$ respectively for a sequence X. Classification or discrimination is usually based on the log odds score $lo(X)$

$$lo(X) = \frac{P(X|H_1)}{P(X|H_0)}$$

If there is more than one one-class PHMM, then the posterior probability for the sequence X is calculated separately. Discrimination is based on the differences in these probabilities as Jaakkola and Haussler (1999) explain.

However, in classification we are interested in discriminating between classes and therefore interested in $P(H_1|X)$, the so-called class conditional probability.

Generative models can calculate this probability using Bayes' Rule.

$$P(H_1|X) = \frac{P(X|H_1)P(H_1)}{P(X)}$$

Discriminative models directly use the conditional probability. Vapnik (1998) argues that "one should solve the classification problem directly and never solve a more general problem as an intermediate step." And it is the general consensus in the Machine Learning community that discriminative approaches perform better than generative ones if better discrimination is the aim and fully labelled data is available (Brefeld et al., 2005; Jaakkola and Haussler, 1999; Ng and Jordan, 2002).

There are two fundamental ways to change from generative to discriminative modelling:

- change the model itself or
- extract features of the generative model or generate features from the raw input that can be used as input for a discriminative classifier.

Sometimes a third possibility is mentioned, that is to train a generative classifier discriminatively. However, as Minka (2005) argues there is only one correct likelihood and one correct way to train a generative model. Discriminative training changes the model itself and should therefore belong to the first category, as a new discriminative model is used. In addition, Minka's work gives a theoretical basis to hybrid approaches between generative and discriminative learning.

Generative learners have their discriminative counterparts. Ng and Jordan (2002) compare naïve bayes and its discriminative partner logistic regression. The discriminative counterpart of Hidden Markov Models are so-called Conditional Random Fields. As we want to profit from research and optimisation on sequence similarity and homology detection we do not use Conditional Random Fields or change our models to them. This way, the approach presented in this thesis can be used with PHMMs. Thus, taking into account the above, we follow the approach to extract features from PHMMs instead of alternatively using Conditional Random Fields.

One approach is to generate features from the raw input sequences directly.

A simple feature representation for example just uses the counts of each amino acid in the sequence. Other approaches derive chemical properties from each amino acid in the sequence. Examples are the hydrophobicity or whether an amino acid is charged or not. Additionally, researchers have transformed the amino acid sequence by pairs or other n-grams of amino acids. Depending on n, the number of features created by n-grams grows exponentially. All the above mentioned feature-creating processes have in common that they do not use similarity as defined by a sequence alignment. The next section will introduce methods on how to generate features from a PHMM. In the process the PHMM is considered as a graph, e.g. a form of structured data that can be exploited. Propositionalisation is the summary term for techniques that transform structured data into features.

2.4.4 Propositionalisation

Propositionalisation is the transformation of a complex input structure into a fixed length feature vector. The resulting representation is thus called propositional and a Machine Learner that uses it is referred to as a propositional learner. These learners are the standard in Machine Learning and therefore highly optimised. Apart from propositional or fixed length feature vector representation, some Machine Learning practitioners also refer to this representation as an attribute-value one.

As Kramer et al. (2001) point out, propositionalisation decouples features from model construction. Thus, it adds flexibility. It originates from the area of Relational Learning and Inductive Logic Programming (ILP). Kramer (2000) defines Relational Learning as learning from "examples described in terms of relations." A simple example is to think of cross-linked database tables where information of one training example is spread over multiple tables. A propositional representation is then according to our example a flat table with one entry per example.

Figure 2.4 gives a graphical insight into propositionalisation for PHMMs. The figure shows a PHMM and the training sequences as inputs into the propositionalisation process and its output is a fixed length feature vector representation. The PHMM is trained on one class, whereas the resulting propositional representation is binary as positive and negative instances alike are propositionalised.

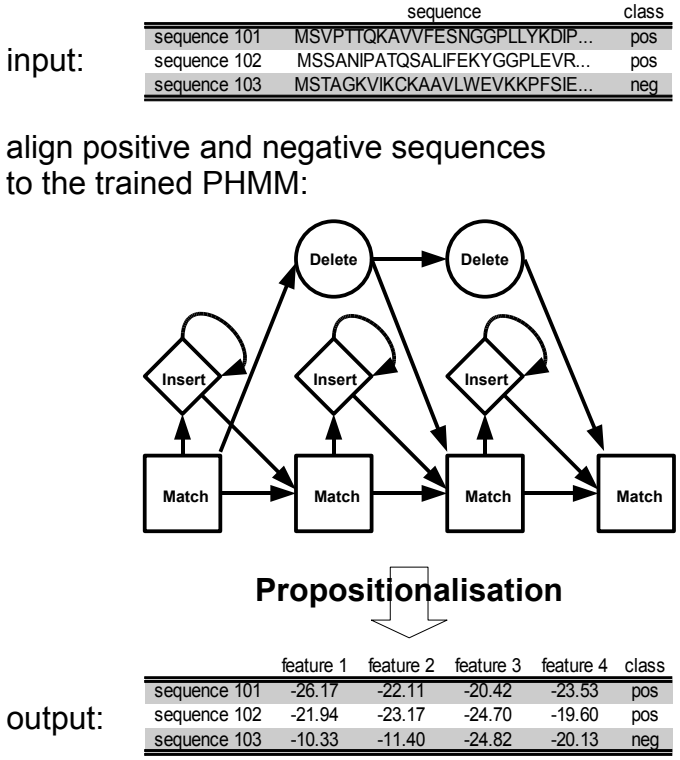


Figure 2.4. *The input and output of the propositionalisation process of a one-class PHMM.*

One of the most popular techniques to use on structured data is to define a kernel function on them that bridges the gap between structured data like a PHMM and a fixed length feature vector as seen in Figure 2.4.

Kernels

The presentation in this section follows Gärtner (2003)'s survey of kernels on structured data. He categorises the approaches into two distinct groups.

- Kernels based on the syntax of the representation and
- Model-based kernels that have some knowledge about the relationship between instances. This knowledge can be a generative model.

The most prominent representative of the latter group of kernels on generative models are Fisher kernels introduced by Jaakkola et al. (1999, 2000) on

top of PHMMs. However, these kernels work on Hidden Markov Models in general. Intuitively, Sewell (2008) describes the idea behind the Fisher kernel in the following way: It expresses in what directions a sequence stretches the parameters of the model. This is captured by a gradient vector. A similar gradient means that the sequences under consideration adapt the model in a similar way. Therefore, the Fisher kernel expresses similarity based on a PHMM i.e. a model for a Multiple Sequence Alignment.

The basis of Jaakkola et al.'s work is the so-called Fisher score vector U_X for a test sequence X (Jaakkola et al., 2000). It is the gradient of the log likelihood with respect to the parameters ζ of the generative Hidden Markov Model.

$$U_X = \nabla_\zeta \log P(X|\zeta)$$

This is a fixed length vector representation extracted from a Hidden Markov Model.

To derive a kernel function from the Fisher score, the similarity or distance of the Fisher score vectors U_X and $U_{X'}$ for sequences X and X' needs to be quantified. Jaakkola et al. define this distance $k(X,X')$ as

$$k(X,X') = \sqrt{\frac{(U_x - U_{X'})^T (U_x - U_{X'})}{2\sigma^2}}$$

where σ is a scaling parameter set to the median Euclidean distance between the gradient vectors corresponding to positive training sequences and the closest gradient vector from a negative training sequence. This distance measure on Fisher score vectors is the input into a radial basis function (RBF) kernel $K(X,X')$.

$$K(X,X') = e^{-(k(X,X'))^2}$$

However, in their implementation Jaakkola et al. (2000) only use the gradient vector for the emissions in the Fisher score vector. They do not use the information from the gradient of the log likelihood with respect to the transitions. Our work does exactly the same. Additionally, we do not learn emission probabilities in insert states. These probabilities are constant and therefore do not add information. Consequently, the Fisher score vectors in this thesis are built from the gradient of the log likelihood with respect to the emissions in match states.

Jaakkola et al. (2000) also use a 9-component Dirichlet mixture decomposition of the emission probabilities. Let $e_s(x_i)$ be the emission probability of symbol x_i in state s, then it is decomposed into nine components according to the following formula

$$e_s(x_i) = \sum_{j=1}^{9} c_{j,s} e_s^j(x_i)$$

where $\sum_{j=1}^{9} c_{j,s} = 1$. Using a mixture decomposition allows abstracting away from residue identity. However, it is not entirely clear from the reference in Jaakkola et al. (2000)'s work which 9-component mixture decomposition they used. Therefore, this thesis' implementation of Jaakkola et al.'s work does not use any Dirichlet mixture decomposition.

Sewell (2008) gives an overview of Fisher kernels. They have not only been used successfully in Bioinformatics, Dick and Kersting (2006) for example use them on relational data.

Syntax-based kernels have been applied in protein classification as well. Hua and Sun (2001) use a very simple string kernel just counting how many times a certain amino acid occurs in a sequence. A more sophisticated syntax kernel has been developed by Saigo et al. (2004). It is a so-called convolution kernel. The main idea is to define a kernel on composite objects by kernels on parts of the object. A convolution string kernel for example could use common subsequences of a string. Saigo et al.'s string alignment kernel is a convolution kernel mimicing local alignment scoring schemes.

2.4.5 Evaluation

Evaluation is an essential component in Machine Learning research. A standard approach to compare two or more classifiers is based on the percentage of correctly classified test instances. This measure is known as accuracy.

When dealing with imbalanced datasets, accuracy may lead to unjustified conclusions. The binary datasets in this thesis are imbalanced. Dataset *enzyme_8*, for example, consists of 2.2% positively labelled instances. Any classifier just predicting the majority class evaluated on the training set achieves a performance of 97.8% accuracy. Even though this result seems impressive, the classifier completely fails to distinguish positive from negative sequences.

In addition, assessing the performance of a one-class PHMM with accuracy requires explicitly defining a prediction cut-off Θ. It is a difficult problem to find an optimal cut-off point. The receiver operating characteristic or ROC curve is a plot of the true positive rate versus the false positive rate for varying prediction cut-offs. Chawla et al. (2004) suggest using ROC curves to evaluate imbalanced datasets. The area under the ROC curve, known as AUC, is a commonly used single value summary for a ROC curve especially in Medicine, Radiology and Bioinformatics as well as Machine Learning (Hand, 2009). The calculation of AUC does not require any user-set cut-off for predictions. On top of that, there is an intuitive interpretation of the AUC. It equals the probability that a classifier ranks a randomly chosen positive instance higher than a randomly chosen negative one. Thus, the AUC is a measure concerning the quality of the ranking of instances. When the classifiers introduced in this thesis are used in a practical setting, a prediction cut-off needs to be defined. However, Rosset (2004) argues that in high-uncertainty situations AUC may be a well-suited performance measure even when the goal is to find the optimal prediction cut-off.

When comparing the AUC of two classifiers on the same dataset an important question is to know whether or not a difference in AUC is meaningful. Of course in Bioinformatics, a meaningful difference should reflect some biological reality, but that is hard to measure. The standard way in Machine Learning to compare results is to test for statistical significance. A result like a difference in AUC is considered statistically significant if it is unlikely to have occurred by chance. The results presented in this thesis are tested for significance using the method from DeLong et al. (1988). More information will be provided in Section 3.3.1. The meaningfulness of statistical tests, null hypothesis and null models have been discussed in the literature. However, as Demšar (2006) points out, there are no established alternatives.

There are limitations of using AUC as a singe value measure to assess a classifier's performance. Hand (2009) points out that if ROC curves cross, the AUC is misleading. One curve can have a higher AUC whereas the other one may be superior for most prediction cut-offs. In our analysis we do not know in general whether the corresponding ROC curves intersect. However, we analysed the ROC curves for the 117 most important experiments and the results suggest that AUC in combination with statistical significance prevents misleading results

in 95% of our cases. Figure 2.5 for example shows the ROC curves for PHMM-based classification using different PHMM models for dataset *euk_3*. The AUC

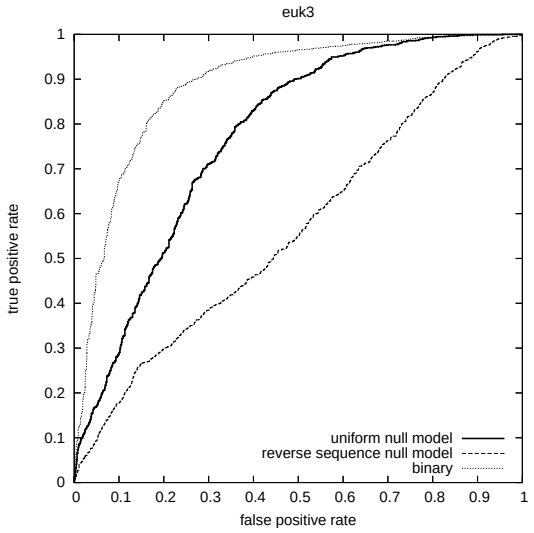


Figure 2.5. *ROC curve for euk_3 for three different ways of PHMM-based modelling that lead to statistically significant differences in AUC.*

of all of these models is statistically significantly different and the ROC curves show a clear domination pattern. Another case when AUC in combination with statistical significance interprets AUC results correctly is given in Figure 2.6 for dataset *pro_2*. The AUC for the binary PHMM approach is 0.875 and in terms of AUC the propositional learner seems to outperform the PHMM with an AUC of 0.893. However, there is no statistically significant difference in AUC and the ROC curves cross.

In only 5% of all results, there is a statistically significant difference in AUC but the respective ROC curves cross and Figure 2.7 gives an example. However, in half of these cases the cross or crosses appear close to the origin or when the true positive and false positive rates are both almost 1.0.

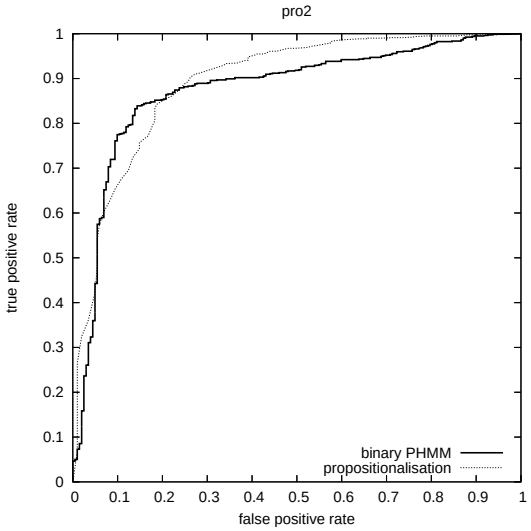


Figure 2.6. *ROC curve for pro_2 for binary PHMM classification and propositionalisation.*

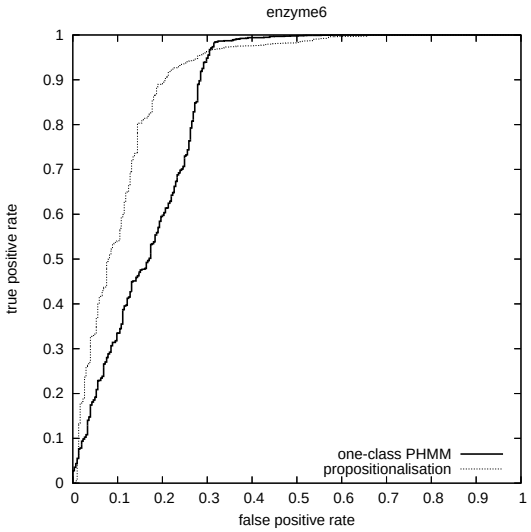


Figure 2.7. *ROC curve for enzyme_6 for one-class PHMM classification and propositionalisation.*

Hand finds another problem with AUC. It evaluates different classifiers using different metrics, more details on this can be found in his work (Hand, 2009). Despite these criticisms and deficiencies, AUC is a standard measure for performance assessment and comparison in Bioinformatics, thus, we use it in our approach.

2.5 Comparison of sequence classifiers

This section compares and categorises previous approaches and our own approach according to three fundamental questions from Section 2.4

Even though these questions are fundamental for any classification approach, the criteria that we will derive to categorise how actual approaches answer these questions are specific to protein classification. In this respect, care has to be taken if the criteria were to be applied to another class of problems.

The introduction of the following criteria provides a basis to compare different approaches concisely and consistently and allows us to demonstrate where research is necessary.

2.5.1 Criteria

We developed five criteria to compare and categorise approaches in protein classification. The first three criteria deal with matters concerning the definition of the classification problem. Subsequently, we introduce one criterion to differentiate between different data representations. The last criterion categorises the evaluation strategy.

Criterion 1: Using additional information

Most approaches in remote homology detection and protein classification use solely the amino acid sequence as input to decide similarity or class membership. We consider as additional information any information that cannot be derived from the amino acid sequence. Therefore chemical properties of amino acids or their count are not considered additional information. These cases are covered by criterion 4. However, some methods use other sources of information like the secondary structure of proteins in addition to the amino acid sequence.

Criterion 2: Generative or discriminative classification

As explained in Section 2.4.3, PHMMs are generative classifiers. However, in a discriminative task like classification, a discriminative approach often leads to better results. For remote homology detection generative models, especially PHMMs, are used. However, researchers have developed discriminative methods for protein classification. It is an important design decision whether the model for a class itself is important or whether discrimination is the sole task to perform. This thesis introduces methods to take advantage of a PHMM in a discriminative task.

Criterion 3: Multi-class, binary or one-class classification

This criterion covers an important aspect of the problem definition. To get a reasonable answer for a given problem, it is important to ask the right question. Machine Learning provides a sound framework for the right choice of classification problem. It is essential to know whether a classification task is inherently a multi-class problem or a binary question solved with a binary or one-class classifier. Section 2.4.1 showed that the question answered by remote homology detection and PHMMs is a binary question. For a sequence of unknown family or class, biologists ask whether the sequence belongs to a specific family or not. As an example we look at the subcellular localisation problem. Proteins can be found in subcellular localisation A, B or C. Thus, there are three classes. For a protein x for which the class has not been determined yet, biologists typically ask the following binary questions:

- Does x belong to class A or not?
- Does x belong to class B or not?
- Does x belong to class C or not?

Three one-class PHMMs are well-suited to answer this question. Generally speaking, these questions are much more likely than the question whether sequence x belongs to class A or class C. Barrett et al. (1997) build separate PHMMs for the remote homologs detections for globins, EF-hands and ferredoxins. Thus, they do not consider the problem as a multi-class problem, but as three

binary problems. They use a one-class classifier to solve each of them. In addition, the binary training sets are built using the one-vs-all strategy. One class from the multi-class set is identified as positive and all others form the negative class. A different transformation is pairwise, binary classification. However, this transformation is not in accordance with the intuitive problem definition.

Criterion 4: Sequence based or feature based classification

PHMMs take a sequence of amino acids as input. However, standard, propositional classifiers train on fixed length feature vectors. By regularising, feature vectors are created from PHMMs as explained in Section 2.4.4. This transformation enables the use of highly optimised, propositional classifiers. PHMMs are a form of structured, probabilistic data that allows creating a lot of potential features. Their main advantage is that these PHMM based features are related to the notion of similarity. Alternatively, approaches create features directly from the input sequence of amino acids. Examples are simple counts, n-grams, or chemical properties. It is not necessarily clear whether the notion of similarity captured by these features is sufficient for the problem.

Criterion 5: Evaluation and statistical significance

Sound evaluation is an important aspect for any Machine Learning task. With this criterion we differentiate between different measures for predictive power and record whether or not any indication of statistical significance of results is given.

2.5.2 Previous approaches

This section introduces 13 previous approaches in remote homology detection and protein classification. The last two approaches detect remote homologs, all others perform protein classification.

Haussler et al. (1993)

Haussler et al. work from unaligned sequences and build a PHMM for globins. Therefore it is a generative, sequence based approach that does not use any

additional information. The task is modelled as a binary classification problem deciding whether a sequence is a globin or not. The learner is a one-class PHMM. An accuracy based performance evaluation is reported without statistical significance. To the best of our knowledge this is the earliest work to use a PHMM as a classifier.

Reinhardt and Hubbard (1998)

Reinhardt and Hubbard use neural networks on a feature based representation. The fixed length feature vector is simple. Each entry contains the fraction of a specific amino acid. They do not use any additional information. The performance is evaluated by accuracy. There are no reports for statistical significance, however Reinhardt and Hubbard show the standard deviation of the accuracy. They transform the multi-class problem in subcellular localisation prediction into a binary one. But they use a pairwise transformation resulting in 21 binary classification problems for 7 classes. As mentioned earlier, it is more likely that a biologist looking at their dataset would ask the question whether or not a sequence can be found extracellular in eukaryots or not. However, the paper answers questions like whether a protein is extracellular in eukaryots or can be found in the nucleus.

Hua and Sun (2001)

Hua and Sun transformed the multi-class subcellular localisation problem into binary ones using the one-vs-all strategy. This problem definition reflects the biological problem. Their approach is discriminative and uses the same features as Reinhardt and Hubbard (1998). They train a Support Vector Machine with linear, polynomial and RBF kernels. Again accuracy is the measure for predictive power. Statistical significance is omitted in their work. We use their datasets in our approach. Their simple feature approach works well and leads to good accuracies. They report an accuracy for each binary problem.

Chou and Elrod (2003)

The other dataset that we use in this thesis is from Chou and Elrod's work. It is a multi-class dataset with 16 classes. Again they use a discriminative approach,

the so-called covariant discriminant algorithm. It is based on amino acid counts. Thus, it is a feature based approach. The core of the algorithm is a vector for each class containing the mean fraction for each amino acid for all proteins of the class under consideration. For each test sequence, the distance to all 16 vectors is calculated and the class that minimises the distance is returned. Chou and Elrod define their problem as a multi-class problem. However, the components of their classifiers are built from one class, only even though they do not state this fact explicitly. To evaluate their approach they use the true positive rate for each class and the overall true positive rate. Depending on the application this measure might be sufficient. However, in general, the true positive rate is not very meaningful without knowledge of the type I error. No statistical significance is reported.

Doderer et al. (2006)

Doderer et al. employ a combined strategy on subcellular localisation prediction by using the sequence and features generated thereof. In a first step they do a BLAST search with the test sequence. If they find a homologous sequence of sufficient score in the training set, then its localisation is used to classify the test instance. Recall, BLAST works well when sequences have a high similarity. In a second step for dissimilar test sequences, a fixed length feature vector is produced with information encoded in the primary sequence. No additional information is used. The representation trains a decision tree using error-correcting output codes to transform the multi-class problem into various binary ones. Doderer et al. claim their transformation with error-correcting output codes is more sensitive to minority classes. And indeed the accuracy for the smaller classes in one of their datasets increased considerably. To the contrary, training one-class classifiers circumvents the problem of minority classes at training time. In a practical setting, at prediction time, it is, however, necessary to assign a prediction cut-off Θ that is sensitive to the minority class. As pointed out earlier, AUC-based evaluation is independent of Θ. Additionally, the approach of Doderer et al. compares their results to one based on amino acid counts and pairs only. Their similarity search by BLAST for detecting homologies is more sophisticated than simple counts. Also they compare the accuracy of their hybrid approach to the one when only similarity detected from BLAST searches is

used. The overall accuracy is only slightly higher and it is not clear whether or not it is significant. In general, evaluation is based on accuracy, precision and recall. There is no test for statistical significance. The questions that arise from this work are whether a multi-class problem and approach is suited for protein classification and whether accuracy is the right measure when dealing with imbalanced classes. The results of this thesis suggest treating protein classification as a binary problem where one-class classifiers like PHMMs work well. In addition, when facing class imbalance, AUC is a better measure to compare the power of classifiers.

Des Jardins et al. (1997)

Des Jardins et al. predict enzyme functions and state that sequence similarity is the gold standard for protein function classification. They compare their approach to BLAST. However, PSI-BLAST or Hidden Markov Models are more sensitive tools. Like Doderer et al. (2006) they search for homologs in the dataset via BLAST and use the known function of the homologous sequence as prediction. For their own approach, they construct features from the sequence but additionally use other sources of information. In this case, features for secondary structure elements are constructed. The paper introduces three classification problems. One is binary, two are multi-class datasets. Again the multi-class datasets are presented to the Machine Learning classifiers – naïve bayes, decision trees and instance-based learning – as multi-class problems. As we pointed out earlier, this problem characterisation does not follow intuition from typical biological research questions. They report accuracy and no statistical significance. For the binary problem the BLAST based approach and their Machine Learning approach perform equally well. For the multi-class problem, sequence analysis outperforms their approaches. The differences appear to be significant but there is no information available to draw firm conclusions.

Yakhnenko et al. (2005)

In this work the input for the protein classification task are not features but the sequence of amino acids itself. In addition, Yahnenko et al. use k-order Markov chains and their discriminative counterpart instead of Hidden Markov Models. They compare the k-order discriminative Markov chain approach to

Support Vector Machines trained on $(k+1)$-grams. They, therefore, compare generative and discriminative approaches using accuracy, specificity and sensitivity but without providing evidence for statistical significance. We use the same dataset on protein subcellular localisation. For this task they learn binary classifiers which would imply that they use the one-vs-all-strategy. In most cases a higher order generative model is competitive with discriminative models. Yet the discriminative models outperform the generative ones slightly. Overall the three proposed methods perform similarly.

Borgwardt et al. (2005)

Borgwardt et al. use a discriminative approach with sequence and additional secondary structure information. They introduce graph kernels to the protein classification task. Each protein is represented by a graph built from its amino acid sequence, its secondary structure and some chemical properties of its primary sequence. The graphs are compared by means of a kernel. Classification is done with a Support Vector Machine. As in des Jardins et al. (1997), they use a binary problem to decide whether a protein is an enzyme or not. Additionally they introduce a multi-class problem to predict the enzyme class out of six classes. The problem is transformed into six one-vs-all binary problems. They report accuracy and some statistical significance values for the binary problem.

Wang et al. (2001)

This paper introduces a feature based approach with Bayesian Neural Networks without the use of any additional information. Performance is measured on the basis of precision, specificity and sensitivity and no statistical significance is given. Wang et al. work with binary classification problems deciding whether a protein belongs to a family or not. Most importantly, they compare their approach to SAM which is a PHMM implementation. However, because they are dealing with accuracy they have to define a prediction cut-off for the one-class PHMM from SAM which is trained on the positive instances only. Instead of defining one prediction cut-off, they define two assuming incorrectly a one-class classifier could actively decide about class membership for the negative class too. There is a cut-off for predicting a test sequence positively and a cut-off to assign a negative class label to the instance. If the score is between the two

cut-off values, the test sequence remains unclassified. We believe this misunderstanding arises because SAM has not been identified correctly as a one-class classifier. One contribution of this thesis is to clearly identify PHMM based classification as one-class classification and therefore, define the learning problem correctly. The PHMM based methods perform competitively using Wang et al.'s evaluation.

Jaakkola et al. (1999)

Jaakkola et al. propose a propositionalisation for PHMMs using Fisher score vectors for remote homology detection. We explained Fisher score vectors in Section 2.4.4. Their ideas work on general Hidden Markov Models as well. Therefore it is a discriminative approach using features from a generative, probabilistic, graphical model trained on sequences. They do not use any additional information. They evaluate their approach on a non-standard measure called the rate of false positive (RFP). The RFP for a positive test sequence is defined as the fraction of negative test sequences that score as good or better than the positive sequence. No statistical significance is reported. In Jaakkola et al.'s approach the PHMMs are implicitly treated as one-class classifiers. They use a Support Vector Machine with an RBF kernel as discriminative classifier. Their approach is the closest approach to ours one, as we also propositionalise one-class PHMMs. However, our propositionalisations are simpler as they lead to shorter feature vectors, but the predictive power of our discriminative models is competitive. In addition, we explicitly use the one-class PHMM in a one-class setting and see propositionalisation not only as a method to transform a generative model to a discriminative one, but also to change from a one-class to a binary problem. Additionally, we introduce a normalisation step for the feature vectors generated by Jaakkola et al.'s method that increases AUC.

Saigo et al. (2004)

This work uses string alignment kernels for remote homology detection in a discriminative way. As in Jaakkola et al.'s and our work, propositionalisation of an alignment is used. However, Saigo et al. propositionalise a pairwise alignment as opposed to a MSA represented by a PHMM. They do not use any additional information and evaluate with AUC and show ROC curves. However, there is

no statistical significance reported. The classifier is a Support Vector Machine. Saigo et al. state that scores from a pairwise alignment tend to be weaker to those obtained by profile based methods. In addition, a PHMM has a lot of features to construct due to their graphical nature. However, in accordance with Jaakkola et al.'s results and the results presented in this thesis, they observe an increase in predictive power when changing from a generative alignment based method to a discriminative one.

Bánhalmi et al. (2009)

Bánhalmi et al. compare one-class classification of proteins to binary classification. They use a discriminative, feature-based approach. Each feature is an average similarity score from one of the superfamilies represented in the training set. Averaging over scores is used as a way to reduce the number of features and to cope with imbalance. The results for the reduced feature sets are better. Some of their experiments include information about the protein structure as an additional source of information. However, they have conducted experiments solely based on average BLAST scores. Binary classifiers used were Random Forests, Neural Networks and Support Vector Machines. The results are compared using AUC. However, no statistical significance is given for the AUC values. Binary and one-class classification methods perform similarly. Bánhalmi et al. conclude that feature based one-class classification is an alternative to binary protein classification. They stress the fact that they do not need an MSA. However, their features are built from averaging over pairwise alignment scores.

Liao and Noble (2002)

Liao and Noble provide a good example of an approach that combines Machine Learning and Bioinformatics. The evaluation of their work following the thesis' criteria also reflects this. They use merely the primary sequence information and define the problem as a binary, discriminative classification problem. To the best of our knowledge, it is the only other approach apart from this thesis that uses AUC in combination with statistical significance. They follow Jaakkola et al. (1999)'s approach and transform the sequence information into a propositional representation. This propositionalisation step is called vectorisation. As

Jaakkola et al. do, they use a Support Vector Machine as propositional classifier. On a conceptual basis they extend Jaakkola et al.'s view as their work is clearly based on adding negative information. The practical difference is that they do not use PHMMs, instead they calculate a pairwise alignment score using the Smith-Waterman algorithm for each pair of sequences. Feature vectors with these scores are used to train the Support Vector Machines. They regard their approach as simpler as they do not need to define a PHMM. However, their pairwise alignment algorithm needs fewer parameters. Additionally, instead of fully training a MSA represented by a PHMM, they derive m^2 pairwise alignments if m is the number of sequences. Our approach based on PHMMs will show that significant increases in AUC can be achieved by not fully training a PHMM. Liao and Noble see propositionalisation as a way to include negative information, we will show that negative information can already be introduced in the PHMM learning step. Their results significantly outperform Jaakkola et al.'s approach. However, the performance of Jaakkola et al.'s work depends on the PHMM. As far as the paper reveals they train a PHMM with its length set to the standard value which is the average sequence length. This fact, in combination with very small positive training sets makes the PHMMs likely to overfit. In our approach, we restrict the PHMMs' size to 35. Jaakkola et al. do not restrict the PHMMs' length, however, they search for additional positive training sequences using a database search. Thus, this way the PHMMs, like the ones defined in this thesis, are less likely to overfit. The biggest drawback of Liao and Noble's approach is runtime as it is considerably slower than PHMM based approaches. Let n be the sequence length, then their approach with m^2 pairwise alignments is in $O(n^2 \cdot m^2)$ as each Smith-Waterman computation is in $O(n^2)$. In comparison, a PHMM needs $O(n \cdot m \cdot k)$ time for m sequences of length n and a PHMM with k states. As Liao and Noble point out, the PHMMs they use for comparison have approximately as many columns as the sequence length. However, in our case the number of columns is restricted. This leads to an increase in runtime compared to this thesis' approach of $m \cdot \frac{n}{k}$. This constitutes a vast increase. Liao and Noble use specialised hardware for their approach. This thesis, on the other hand, introduces methods to significantly improve an Jaakkola et al.'s approach by significantly reducing the runtime. Our approach works on standard hardware.

2.5.3 Proposed approach

This thesis uses PHMMs as a basic tool in protein classification. PHMMs model sequence similarity and have been successfully used in the closely related task of (remote) homology detection.

The use of PHMMs has implications on the way this thesis defines the classification problem and represents the data. A PHMM is a one-class classifier, therefore, it answers a binary question. All datasets used in this work are binary sets derived from multi-class problems. One class from the original multi-class dataset is used as the positive target class, all others form the negative class.

As Figure 2.8 illustrates, this research aims at transforming the one-class classification problem to a binary one. The reason behind this idea is that there

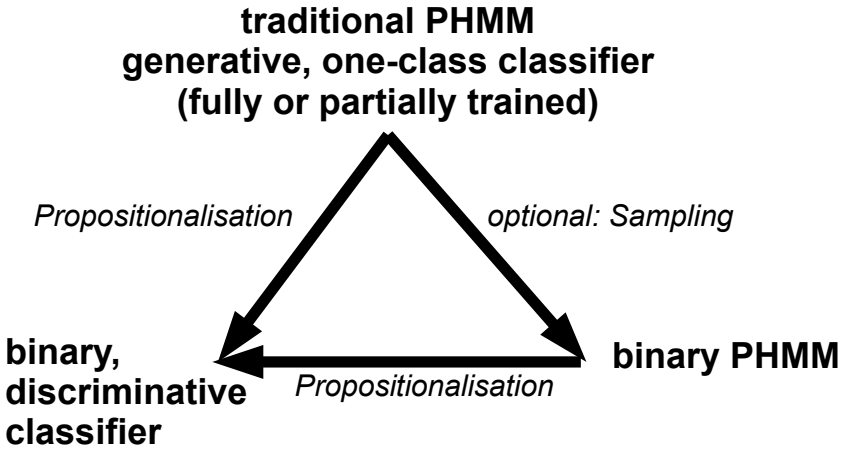


Figure 2.8. *The thesis' scheme*

is a vast amount of negative data compared to the small number of positively labelled sequences used to build the one-class PHMM. As discrimination is the overall goal in protein classification, the task will benefit from adding negative information. The first idea is to add another one-class PHMM trained separately on the negative instances. The separation ensures that the dataset imbalance does not negatively influence performance. Together we refer to the positive one-class PHMM and the negative one-class PHMM as a binary PHMM. Sampling strategies that originate from the community based around learning from imbalanced sets allow for a time efficient transformation of a one-class PHMM

to a binary one.

This thesis also introduces new approaches for propositionalisation. Propositionalisation allows the transformation of a structured input like a PHMM into a fixed length feature vector representation. It changes the model from a generative PHMM to a discriminative model better suited for classification. Our research results in new, simpler approaches to propositionalise a PHMM with competitive predictive power. Additionally, this thesis, like Liao and Noble (2002), explicitly interprets propositionalisation as a tool not only to change from generative to discriminative modelling but also as a tool to change from one-class to binary classification. The latter change is as important as the first as it allows using a vast amount of negative information.

Finally, binary PHMMs can be propositionalised using the same basic ideas as one-class PHMMs.

This thesis bridges a gap between Bioinformatics and Machine Learning as it brings together the PHMMs used successfully for homology detection and employs them on a classification task by carefully defining and looking at the Machine Learning characteristics and solutions for the underlying classification problem.

We evaluate our approach using AUC and report statistical significance of results. Thus, we compare the ability to rank sequences correctly by different methods and are not concerned with optimising the prediction cut-off for practical use of the models. Providing statistical significance is crucial to further advance protein classification as it is not clear from much of the related research whether or not their proposed methods lead to a significant improvement. As we discussed earlier, it is a different question whether a statistically significant improvement is biologically meaningful or not. The answer to this question is beyond the scope of this thesis. However, to compare Machine Learning approaches in homology detection and protein classification statistical significance is essential, but often neglected.

2.6 A note on alignment quality

Several studies reveal that alignment quality is crucial for remote homology detection and protein classification when amino acid sequences are used as in-

put (Jaroszewski et al., 2002; Madera and Gough, 2002; Söding, 2005). This thesis' results agree with these findings when PHMM-based classification is used. However, it is a very interesting finding that propositionalisation works well on the first iteration of the training algorithm. It is consistently competitive and in some cases, even leads to the best discriminative power as our experiments in Section 5.2 reveal. This is surprising as the alignment quality is poor in early iterations of the training algorithm. Figure 2.9 gives a graphical impression of this phenomenon.

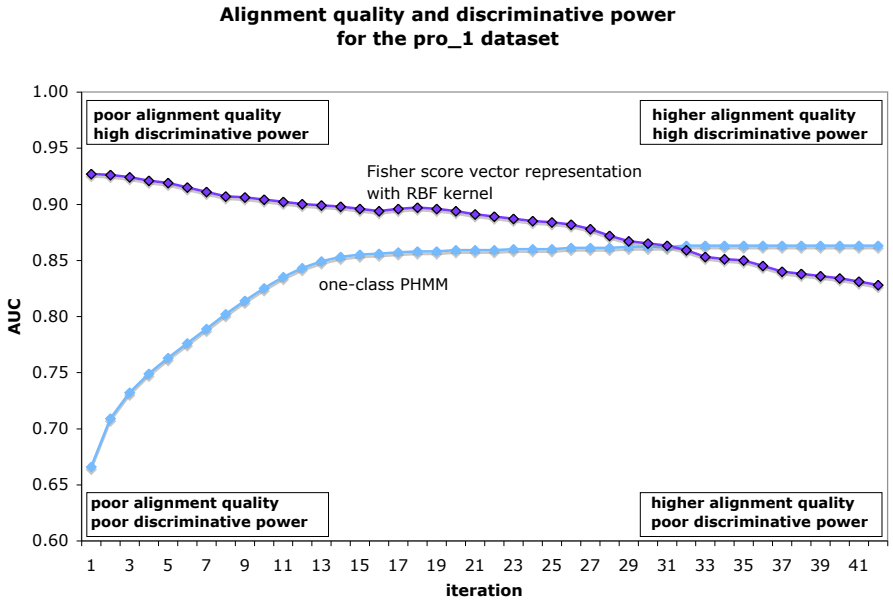


Figure 2.9. *Comparison of AUC for the pro_1 dataset for one-class PHMM classification and Jaakkola et al. (1999)'s approach. Alignment quality increases with number of iterations in the Baum-Welch training algorithm.*

It displays the AUCs for one of our datasets (*pro_1*) during the iterations of the Baum-Welch training algorithm comparing a basic one-class PHMM approach with its propositionalisation according to Jaakkola et al. (1999). In early iterations the alignment quality is poor but increases during the training process. The one-class PHMM reacts to poor and higher alignment quality as expected with poor and higher discriminative power. The propositional approach works already well on a poor quality alignment. This is true for all our datasets. In some datasets like *pro_1* the AUC actually decreases when the alignment quality increases. Liao and Noble (2002) report significant success of their method on

the basis of a comparison to propositionalisations from high quality alignment. This thesis will show that propositionalisation of a poor quality alignment improves predictive performance significantly.

2.7 Summary

This chapter introduced the basic concepts in biological sequence classification and related them to concepts in Machine Learning.

Sequence similarity is the key to understanding functional, evolutionary and structural relationships. PHMMs have been proven to model sequence similarity well. It is one of our fundamental assumptions that a sound knowledge from Machine Learning about how to define the protein classification problem, how to represent the data and how to evaluate the approaches will further advance protein classification.

This chapter has introduced the basic and fundamental ideas and related approaches in protein classification. The following chapters will introduce our experiments.

Chapter 3

Experimental setup

The previous chapters introduced protein classification, defined the objectives of the thesis and laid the groundwork for our research and contributions. The following three chapters will present the experiments. Their setup is identical throughout all three chapters. Additionally, all experiments use the same datasets. Therefore, this chapter introduces not only the datasets and their key properties but also the basic settings for all experiments.

Attention must be paid to the special evaluation strategy. Typically, in a non-incremental Machine Learning setting, the classifier is evaluated after full completion of training. However, in this thesis, we evaluate the PHMM after each iteration of the Baum-Welch training algorithm. This algorithm stops training when the differences between training iterations in the overall log-odds are sufficiently small. Evaluating not only after convergence but after each iteration is expensive in terms of runtime. However, it offers the following insights:

- It is not trivial to find the right threshold for the differences in the overall log-odds scores. Evaluating after each iteration of the Baum-Welch training algorithm circumvents this problem.

- Additionally and more importantly, it clears the way for a new direction of research. PHMM training is time intensive. This thesis proposes methods that stop PHMM training at early stages and use faster Machine Learning techniques to compensate for a loss in predictive power.

As opposed to previous approaches, this thesis reports statistical significance results for AUCs.

All experiments are performed using WEKA (Hall et al., 2009). Runtime analysis is based upon execution on 3 GHz Intel© 64-bit machines with 2 GB of memory. There has been no exclusive access granted to these machines. Thus, results concerning runtime have to be treated with some care. This thesis interprets the runtime results as a very strong indication of the real runtime behaviour.

The software tool StAR (Vergara et al., 2008) calculates statistical significance of our results at a 0.05 level. Its negligible runtime is not included in the timing results.

This chapter comprises three sections. First we introduce the datasets. The following section provides an overview of the classifiers and their setups. Finally, Section 3.3 introduces the evaluation strategy.

3.1 Datasets

All datasets belong to the field of protein classification. The task is to assign the protein to a specific class using only its primary sequence that is the sequence of amino acids. Therefore, the datasets consist of a string representing the sequence and a nominal class attribute.

All datasets form multi-class problems. However, they are transformed into binary ones. The sequences from the class under consideration form the positive class. Sequences from all other classes are treated as instances from the negative class. The resulting binary datasets are highly imbalanced featuring small positive classes.

We build PHMMs on the negative class as a whole. So far, in previous approaches, PHMMs for each class have been trained separately. However, in the negative class all but one of the original classes are combined. These one-class PHMMs on the negative class represent the potentially diverse negative class and not just one specific family or class of proteins. However, as experiments will show, these PHMMs increase the discriminative power when used as classifiers or as part of a classifier.

The following sections introduce three multi-class datasets and the resulting 23 binary classification problems.

3.1.1 Protein localisation

Protein localisation is the problem of determining where inside or outside a cell a mature protein can be found. The datasets are from Reinhardt and Hubbard (1998). They have also been used by other researchers (Chou and Elrod, 1999; Guo et al., 2004; Hua and Sun, 2001). A major characteristic is their high sequence similarity inside classes (Nakai, 2000). The first dataset addresses protein localisation in prokaryotes (pro). It consists of 997 instances belonging to three different classes. A protein is either classified as cytoplasmic, periplasmic or extracellular. We transform this multi-class problem into three binary classification tasks. In each binary setting, one class is used as the positive class and all the remaining classes are combined into one negative class. All other datasets are pre-processed in the same way. For the remainder of the thesis, we use the dataset name, followed by an index for the positive class, e.g. *pro_0* refers to the prokaryote dataset treating class with index 0 as positive and all remaining instances as negative. The eukaryote dataset (euk) consists of 2427 sequences containing four different classes. They are cytoplasmic, extracellular, mitochondrial and nuclear respectively.

The datasets *pro_0* and *euk_0* refer to the same localisation as *pro_1* and *euk_1* do. For the remaining datasets the localisations are different even though they agree on the index of the positive class.

3.1.2 Enzyme classification

In enzyme classification proteins are categorised not according to location but according to function. The dataset we use was introduced by Chou and Elrod (2003). In contrast to the ones used for protein localisation, they are characterised by low sequence similarity inside each class (Chou and Elrod, 2003). For 15 of the 16 classes the sequence similarity is smaller than 25% and therefore the classification problem is in the the so-called twilight zone of sequence similarity (Doolittle, 1981). Only class 14 with an average similarity of around 27% is out of this zone.

We use the UniProt identifiers and UniProt database (release 2010_4 from the 19th of March 2010)[1] (Apweiler et al., 2004; Consortium, 2010) to retrieve the

[1] http://www.uniprot.org/

primary sequences. Chou and Elrod (2003) checked for duplicates. However, as we use a recent version of the UniProt database, we retrieve a slightly different number of sequences per class and therefore repeat their post-processing step. We find duplicates and remove them. This results in a dataset of 2643 unique amino acid sequences containing 16 classes. The UniProt identifiers for these proteins are listed in Appendix A.

All binary datasets except *pro_0* share an important property. They are highly imbalanced with a small positive class. This is typical for protein classification. Table 3.1 gives an overview of our 23 binary datasets and the size of their positive class. The datasets are used in a one-class classification setting trained on

Table 3.1. *Overview of datasets and their respective size of the positive class. The percentage numbers are rounded.*

dataset	positive instances		dataset	positive instances		dataset	positive instances	
	number	%		number	%		number	%
enzyme_1	317	12.0%	enzyme_9	254	9.6%	pro_0	688	69.0%
enzyme_2	216	8.2%	enzyme_10	93	3.5%	pro_1	107	10.7%
enzyme_3	194	7.3%	enzyme_11	154	5.8%	pro_2	202	20.3%
enzyme_4	130	4.9%	enzyme_12	96	3.6%	euk_0	684	28.2%
enzyme_5	113	4.2%	enzyme_13	257	9.7%	euk_1	325	13.4%
enzyme_6	305	11.5%	enzyme_14	152	5.8%	euk_2	321	13.2%
enzyme_7	64	2.4%	enzyme_15	84	3.2%	euk_3	1097	45.2%
enzyme_8	59	2.2%	enzyme_16	155	5.9%			

the minority class. This thesis will show different ways to introduce negative information so that the discriminative power of the classification process can be increased. In addition, we introduce techniques beneficial to runtime when compared to fully training a PHMM on the whole dataset.

3.2 Classifier setup

The baseline classifier in our work is a PHMM trained on the positive class only. Furthermore in Chapters 5 and 6 standard Machine Learning classifiers are used as well. This section first describes the setup for the PHMMs and introduces the settings for the standard Machine Learning classifiers afterwards.

3.2.1 Profile Hidden Markov Models

PHMMs have been introduced in Section 2.2. Their graphical structure and their initial parameters are the same in all experiments. Our PHMMs represent global alignments of proteins.

Setting the length parameter

Sections 2.2.3 and 2.2.4 about the advantages and disadvantages of PHMM-based modelling and classification have already highlighted a few issues concerning the length of a PHMM and this section will expand on this discussion.

In this thesis each PHMM consists of 35 columns, i.e. 35 match states. The number of columns equals the expected length of the conserved region in a class of proteins. This length depends on the problem domain and thus, prior knowledge from a biological expert or heuristics are used to find the optimal number of columns. In many practical cases, the length and other parameters of a PHMM are determined by a pre-exisiting Multiple Sequence Alignment (MSA). Durbin et al. (1998) discuss a dynamic programming algorithm called *MAP model construction algorithm* on MSAs as an alternative to expert knowledge and use of heuristics. In our case, the input sequences are unaligned, so we cannot use any of these techniques. We have no prior knowledge concerning the optimal number of columns for the unaligned sequences from the *enzyme* datasets. The situation is different for the *prokaryote* and *eukaryote* datasets as work from Emanuelsson (2002) publishes the length of common protein sorting signals for localisations in eukaryotes. They vary in length from 3 to 70 residues.

Additionally, there are other considerations as well when deciding about the length of a PHMM. From a Machine Learning point of view the length of a PHMM can like any other parameter be estimated using a set of different lengths and cross-validation. However, an in-depth estimation is not feasible as PHMM training is time-intense. We did preliminary experiments with a few different lengths that clearly show that there is an optimal length. Shorter PHMMs might lead to faster but more inaccurate results, as well as longer PHMMs who require more training time. However, we do not know the optimal length for our datasets.

Researchers like Liao and Noble (2002) train PHMMs setting their length to

the average sequence length. These PHMMs are very slow in training. Additionally, especially if the PHMMs are trained on a small class of examples, the PHMMs are likely to overfit as there are a lot of parameters to estimate using a small set of training sequences only (Cherry, 2001). Brown et al. (1993) for example build PHMMs with up to 401 columns. However, this PHMM is trained on 88 sequences only. Opposed to our work, they use Dirichlet mixture priors to avoid overfitting. Therefore, a small length addresses not only the problem of overfitting but also influences training time positively.

However, it can happen that our PHMMs only model part of the conserved region, leading to a suboptimal alignment. On the other hand, if the length of the conserved region is smaller than the number of columns, the resulting alignment could suffer in quality as well.

Our datasets are imbalanced with a small positive class. Therefore, the one-class PHMM trained on the positive class is prone to overfit if the PHMM is too long. The situation is different for the negative class. There are a lot of instances to estimate the parameters of the one-class PHMM for the negative class. These instances are from different classes and consequently are more diverse and harder to align. As we pointed out before, the length of the PHMM equals the length of the expected conserved region. A long PHMM for the negative class might therefore aggravate the problem of finding a good alignment in a diverse environment.

Note that for the datasets *pro_0* and *euk_0* as opposed to all other datasets, there might not be a pattern to align in the positive class. The positive class of these two datasets consists of proteins that are located in the cytoplasm. Nakai (2000) argues that it is plausible to see the cytoplasm as a default location. However, he also points out that there have been cases where cytoplasmic target or retention patterns have been found. The results of this thesis support the argument of Nakai (2000) to see the cytoplasm as default localisation for the proteins in our datasets.

A PHMM with a suboptimal length can converge to a local optimum instead of a global one. This thesis proposes subsequent propositionalisation to allow leaving the local optimum.

Setting all other parameters

The PHMMs are trained from unaligned sequences using the Baum-Welch algorithm, a special case of the EM algorithm. It guarantees convergence to a (local) optimum. Its convergence criterion is a sufficiently small change in the overall log-odds score. This score is normalised by the number of residues in a sequence. The threshold for this normalised score is set to 0.0001[2]. This is a strict convergence criterion. However, because we evaluate after each iteration of the Baum-Welch training algorithm, our approach reveals that stopping the training after fewer iterations can be beneficial to the predictive power of the model.

In the case of sampling, training is stopped after at most 100 iterations. Results will show that the AUC has levelled out at this stage.

If not explicitly stated differently, all emission and transition probabilities are initialised uniformly. We do not train for emissions in insert states. The emission probabilities of the standard uniform null model are initialised in the same way. They remain constant throughout the training process. In addition to the symbols representing the 20 standard amino acids, four other symbols can be emitted in each match or insert state or in the null model. These are:

- the symbol 'B' representing asparagine or aspartic acid,
- the symbol 'Z' for glutamine or glutamic acid,
- the symbol 'U' that stands for selenocysteine and
- the symbol 'X' meaning the amino acid at the residue is unknown.

The symbols 'O' and 'J' representing pyrrolysine and leucine or isoleucine respectively are never found in our datasets and therefore not part of the symbols used as emissions.

3.2.2 Other classifiers

The standard learners used are a linear Support Vector Machine (SVM) (Platt, 1998), Random Forests (Breiman, 2001) and bagged, unpruned C4.5 decision

[2]The threshold used was proposed by A. Krogh in a personal e-mail communication.

trees (Breiman, 1996). Preliminary experiments used a larger set of classifiers including boosting pruned C4.5 decision trees (Freund and Schapire, 1996), naïve bayes (John and Langley, 1995), k-nearest neighbour classification (Aha et al., 1991) and SVMs with polynomial kernels and radial basis function (RBF) kernels. However, their performance was not competitive compared to the learners presented in this thesis.

The complexity parameter for the linear Support Vector Machine and the number of features used in the Random Forests are estimated using an internal 10-fold cross-validation. All Random Forests consist of 100 trees. We use the same number of unpruned decision trees in bagging.

The only parameter in the approach from Jaakkola et al. (2000) is the complexity parameter c of WEKA's SMO. We use the standard setting and set c to 1. In addition, when we use the Fisher score vector representation with Random Forests, each consists again of 100 trees. The number of features is set to WEKA's standard value.

3.3 Evaluation strategies

All experiments are performed using one 10-fold cross-validation. The de-facto standard performance evaluation in Machine Learning is averaging over ten runs of 10-fold cross-validation. However, due to the runtime of approaches, this is not feasible. This is why one run of 10-fold cross-validation is used as a compromise. In previous work, like the one from Jaakkola et al. (1999) and Liao and Noble (2002), explicit training and test set splits are used.

Predictive power of a classifier is measured using the area under the receiver operating characteristic (ROC) curve (AUC) as motivated in Section 2.4.5.

As opposed to previous work, the PHMMs are evaluated not only after convergence of the training process, they are also evaluated after each iteration of the training algorithm. This evaluation offers new insights into the training process. Not only does it allow us to find a good threshold to stop training, more importantly it gives us the opportunity to experiment with not fully converged PHMMs. In the experiments presented in Chapters 5 and 6 the PHMMs themselves are used as input and through propositionalisation they provide features for a standard Machine Learning classifier. Results show that it is not only ben-

eficial for runtime but also in terms of AUC to stop PHMM training after a few iterations and to propositionalise.

3.3.1 Statistical significance

Rosset (2004) developed limited theory based exact moment calculations for the differences between two AUC scores on the same test set. These calculations are only feasible with knowledge of the underlying probability structure. This work compares the exact computations to the two most common approximate significance tests derived by Hanley and McNeil (1983) and DeLong et al. (1988). The former is a parametric test, whereas the latter is non-parametric. Based on the results of the study, Rosset recommends using the approach from DeLong et al. We follow this recommendation.

The freely available software tool StAR (Vergara et al., 2008) calculates the statistical significance of AUC scores implementing the approach from DeLong et al. (1988). We test significance after the last iteration of the Baum-Welch training algorithm for solely PHMM based approaches. The statistical significance of the proposed propositional extensions of PHMMs is tested after the first and the last iteration of training. Providing information about statistical significance of differences in AUCs is a contribution of this thesis. As we pointed out in Chapter 2, previous approaches do not report statistical significance with the exception of the work from Liao and Noble (2002).

3.4 Summary

This chapter described the experimental setup in detail. The following three chapters will introduce the experiments and interpret their results.

We use 23 binary datasets from two problem domains. They have been created from three multi-class problems. All but one of these binary datasets are highly imbalanced. There is a vast amount of negatively labelled sequences compared to the positive ones. This is a characteristic of datasets in protein classification.

As opposed to previous approaches we evaluate after each iteration of the Baum-Welch algorithm. This research will be introduced in Chapters 5 and 6.

Chapter 4

One-class and binary classification with Profile Hidden Markov Models

This chapter addresses protein classification with PHMMs as the sole classifier. From a Machine Learning point of view it introduces the baseline one-class classification problem and classifier. It is one of our fundamental observations that PHMMs are basically generative one-class classifiers. Contrary to the general use of one-class classification in the absence of negative data during training, these one-class PHMMs train on highly imbalanced datasets with a vast amount of negative data compared to a small number of positive instances.

This thesis exploits ways to transform the one-class PHMM and therefore the one-class classification problem into a binary one. Our basic assumption is that the presence of negative data and its use during training increases the model's ability to discriminate between the positive and the negative class. There are two fundamental ways to achieve this:

- We transform the basic one-class PHMM to a binary model.
- We change the representation of the model through propositionalisation.

As Figure 4.1 shows, this chapter addresses one-class classification and discusses the former problem transformation. It transforms the one-class PHMM to a binary model that uses the negative data during training. Using the vast amount of negative training data causes an increase in training time that makes these models less appealing for practical use. However, as Figure 4.1 already suggests, sampling the negative class provides a technique to optimise predictive power in a way that is more sensitive to training time.

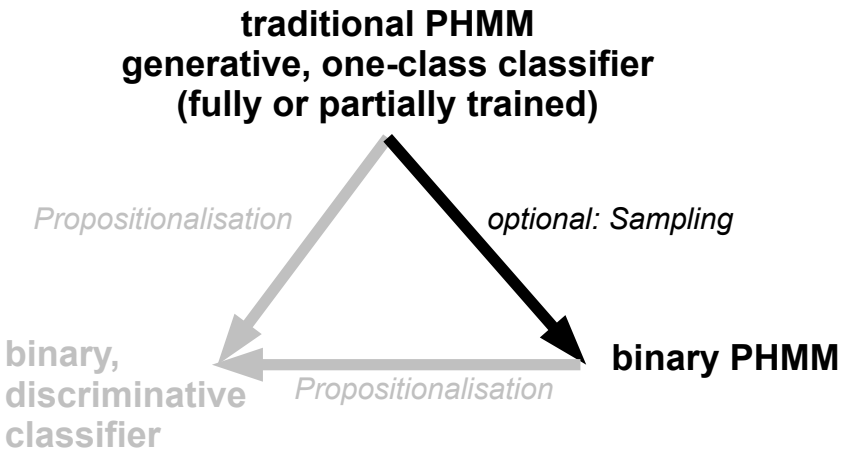


Figure 4.1. *The relevant part of the thesis' scheme*

This chapter discusses three contributions. First, it introduces one-class PHMMs. Consequently, it transforms these classifiers into binary ones by training an additional PHMM exclusively on the negative data. At testing time the binary classifier assigns either a positive or a negative class label to the test sequence depending on which of the two PHMMs produces a higher score for the sequence under consideration. As experiments will show, the inclusion of negative information in this simple way is beneficial to the predictive power of the classification models. However, the binary models need more training and testing time.

Second, to reduce training time we introduce the idea of sampling the negative class. We show that sampling does not necessarily decrease the predictive power of the models. There are two important observations:

- Simple and fast sampling strategies work best.
- The combination of binary classification and sampling makes this approach attractive with respect to predictive power and runtime.

In general, sampling is not merely a strategy to decrease training time, it is also applied to deal with imbalanced datasets (Chawla et al., 2004).

Third, this chapter links binary PHMM classification to null models. These models are an important concept in Bioinformatics. We show that the PHMM for the negative class can be interpreted as a null model for the one-class PHMM

trained on the positive class. In the one-class classification case the standard null hypothesis or null model treats each amino acid equally likely. This is a uniform null model. Other null models use, for example, the reverse sequence (Karplus et al., 1998, 2005). This constitutes a null model that is specific to both sequence and model. Another kind of null model used in the thesis is model-specific. It is the background distribution of amino acids in the training set. The PHMM trained on the negative class can act as a null model which is specific to both sequence and model and leads to better predictive performance. This chapter reinforces the emphasis on using a good null model.

In addition to the three aforementioned contributions there is an important, more general (fourth) observation. In the context of binary protein classification problems, the findings suggest that simple ideas to extend the traditional one-class classification model work best regarding both predictive power and runtime. These findings are not unique to the protein classification problem. In other areas of Machine Learning and Data Mining simple ideas have shown their potential (Keogh et al., 2005) in solving practical problems.

Section 4.1 discusses the baseline one-class PHMM model. In the next section these models are transformed to binary PHMM classifiers. Section 4.3 introduces the sampling ideas, whereas Section 4.4 shows how binary classification relates to null models. Section 4.5 evaluates models experimentally and discusses results. The chapter concludes with a summary of findings and contributions.

4.1 The baseline one-class classification

One-class classification is a technique widely used in Bioinformatics (Bánhalmi et al., 2009). Primarily, this technique has been designed for learning if only examples from one class are present during training. In many cases and especially in protein classification the assumption that no negative information is available during training does not hold. However, one-class classification is a popular approach even though it does not make use of all information that is available. The reasons for its popularity are as follows:

- The data is highly imbalanced.
- The emphasis is on the model itself, even though these models are used to discriminate between classes on proteins as well. In a discriminative setting it is beneficial to include negative information (Jaakkola et al., 1999).

The next two subsections explore the reasons in more detail.

4.1.1 Dataset imbalance

Dataset imbalance is a common problem in many real world datasets (Chawla et al., 2004). There are many approaches to deal with imbalance on a data and algorithmic level as Chawla et al. (2004) point out in their editorial. On a data level instances can be sampled. There are two main strategies:

- oversampling and
- undersampling

This thesis exploits undersampling techniques on the negative class. At an algorithmic level instances can be weighted, the evaluation strategy can be adapted or one-class classification can be used (Chawla et al., 2004). It is a simple but an extreme measure to ignore one class during training, especially if the ignored class is the majority class, and the latter is exactly the case in binary protein classification. The class of interest, the positive class, is the minority class, in datasets like *enzyme_8* only 2% of the instances are positive. It can be hard to define which proteins to include in the negative set of instances. If the aim is to answer the question whether an oxidoreductase acts with sulphur or not, all oxidoreductases not acting on sulphur should be included in the negative set, but what about protein of other classes of enzymes? In general it is easy to define a positive set of proteins but harder to define a good set of negative instances. Thus, ignoring the negative class during training by using one-class classification simplifies the problem. As a side effect it also speeds up the training time, as models are only trained on a small number of positive instances.

4.1.2 Emphasis on the model

A trained PHMM represents a family or class of related proteins. It is a model for those proteins. Researchers in Biology and Bioinformatics are primarily interested in the model, even though they use the model to decide about class membership. The model allows them to describe the protein class. In the case of the PHMM it is a generative, probabilistic model. It enables the construction of new (artificial) members of the class of proteins and, more importantly, offers a probabilistically scored consensus sequence for the protein class. In addition, it allows the construction of profiles. In brief, the model itself is very interesting for research and allows new conclusions about the class of protein under consideration. Of course these generative models can be used, and are used, to decide about class membership. An unknown protein is aligned to the PHMM and the score gives insights into how likely it is that this amino acid sequence belongs to the class of proteins represented by the PHMM.

4.1.3 At testing time

This subsection emphasises again the important concepts when scoring a test sequence by a one-class PHMM. A full discussion of scoring a sequence with a PHMM can be found in Section 2.2.1. In addition, we assume the one-class PHMM to be fully trained. Section 2.2.2 describes the training process. The first part of this subsection covers the scoring itself, whereas the second part discusses how to evaluate the scores and therefore the predictive power of the one-class PHMM.

Scoring

One-class classifiers need a similarity measure. In the case of a one-class PHMM this measure is calculated as the log-odds score for each test sequence. Odds scores describe the ratio of the probability of the test sequence given the one-class PHMM divided by its probability under a null model.

The scoring of a test sequence X of length n in a one-class PHMM with k states is in $O(n \cdot k)$.

Evaluation

Assume there is a protein called "DH16". As a first step we determine how similar DH16 is to the positive class by calculating its log-odds score $lo(DH16)$. Subsequently, protein DH16 is classified as positive if and only if $lo(DH16) > \Theta$, meaning its log-odds score is greater than a problem specific threshold Θ.

4.1.4 An example one-class Profile Hidden Markov Model

Figure 4.2 illustrates how a one-class PHMM works based on a small example. The task is to decide whether or not a sequence built out of the characters 'L' and

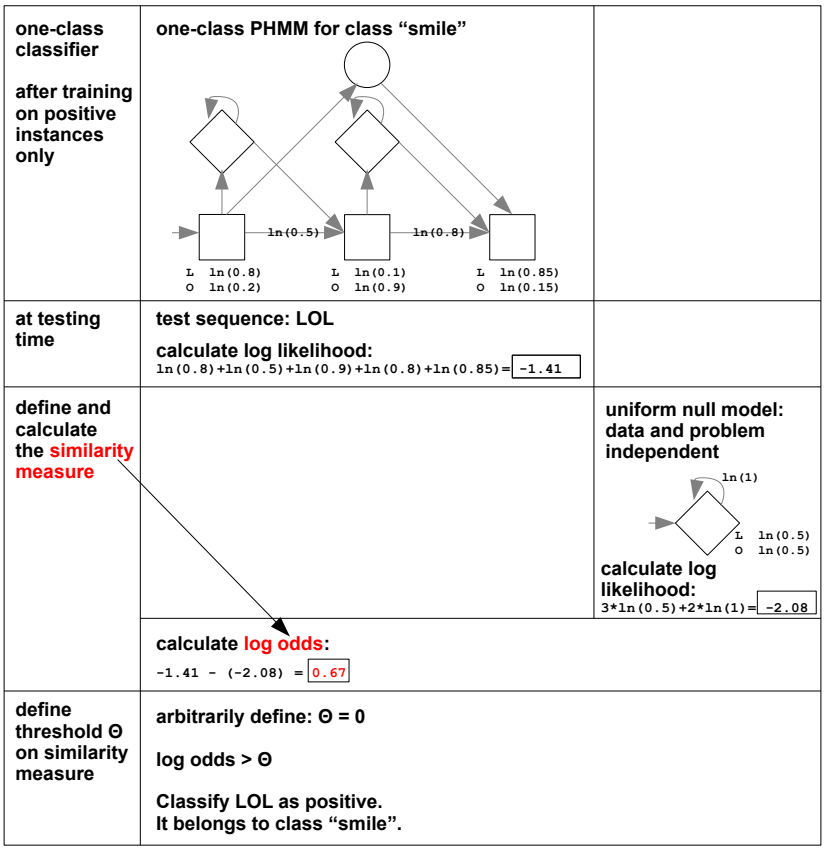


Figure 4.2. *An example of one-class PHMM classification. The positive class is "smile". Training and test sequences are built from characters 'L' and 'O'. The one-class PHMM has three match states. The one-class PHMM is shown after training. Only parameters that influence the classification of the test sequence 'LOL' are shown.*

'O' belongs to the class *smile*. This is the positive class. The negative class *non smile* includes any other sequence. It is a one-class classification problem so the classifier decides only whether or not a sequence is positive. The trained one-class PHMM has three match states and is shown after training is completed. The parameters of the model that are needed to decided about the class membership of the test sequence 'LOL' are shown in the figure. There is only one possible path for the sequence 'LOL' through the PHMM.

As a first step the length dependent log likelihood $ll_{PHMM}(X)$ of the test sequence X given the PHMM for the positive class is calculated. The similarity measure of one-class PHMMs is the log-odds score $lo(X)$. It is calculated by subtracting the log-likelihood of the uniform null model $ll_{null}(X)$ from $ll_{PHMM}(X)$: $lo(X) = ll_{PHMM}(X) - ll_{null}(X)$

To decide whether the sequence 'LOL' belongs to class *smile*, its log-odds score must be above a defined threshold Θ. By default Θ can be set to 0. However, there is no guarantee that 0 is the best Θ for a particular problem. In this small example we use the default threshold. As Figure 4.2 shows $lo(\text{'LOL'}) = ll_{PHMM}(\text{'LOL'}) - ll_{null}(\text{'LOL'}) = -1.41 - (-2.08) = 0.67$ using 0 as a threshold on the log-odds score, the test sequence 'LOL' is classified as positive.

The one-class PHMMs used in the experiments follow this example with the exception that Θ is not explicitly defined. We use AUC as an evaluation measure and therefore have no need to define the problem specific Θ. To calculate the AUC, all test sequences need to be sorted according to their log-odds score.

Note that in order to calculate the accuracy of a one-class PHMM, Θ needs to be defined and strategies implemented to find an optimal value for it. In this work we do not implement these strategies as we do not evaluate based on accuracy.

4.1.5 Calibration of logarithmic scores

At testing time, the one-class PHMM outputs the log-odds for each test sequence. Consequently, the test sequences are ranked based on their log-odds scores. The higher the score and therefore, the higher the ranking, the more likely it is for a test sequence to belong to the positive class. This ranking allows us to construct the ROC curve and thus calculate the AUC.

In WEKA (Hall et al., 2009) test instances are ranked based on their class

probability. The evaluation of our methods in WEKA thus requires the transformation of log-odds scores into probabilities. The transformation has to ensure that:

- The ranking based on log-odds scores remains without changes and that
- For all possible scores a probability can be calculated.

Logistic functions ensure these properties. They are strictly monotonic, take any real number as input value and output a real number between 0 and 1. Figure 4.3 shows an example of a logistic function. Therefore, the one-class

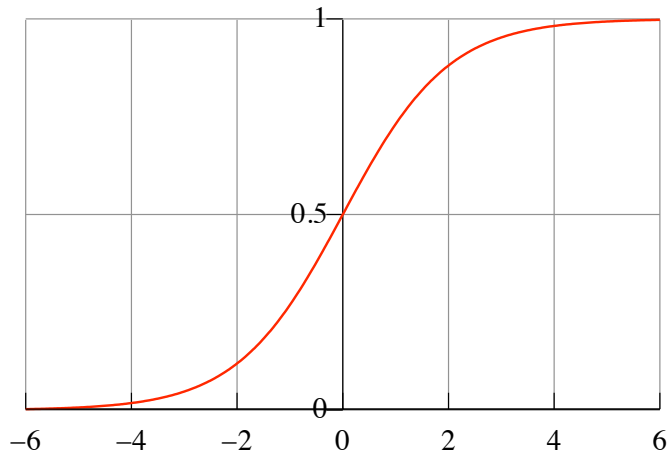


Figure 4.3. *An example of a logistic function taken from Wikipedia (2010)*

PHMM algorithm learns a logistic function on the log-odds scores. To calculate the AUC it uses the probabilities of the trained logistic model. WEKA requires positive and negative instances to learn a logistic function. In the one-class classification case training is restricted to the positive class and thus, we cannot use the real negative instances even though the calibration has no influence on the AUC values as the transformation is strictly monotonic. Therefore, artificial negative examples are created. For each positively labelled amino acid sequence, we build a negative sequence of the same length and amino acid distribution. The position of the specific residues are changed randomly though.

This calibration of logarithmic scores is necessary because of the way WEKA calculates AUCs. However, it is implemented so that it does not change the log-odds based ranking. Therefore it has no influence on the resulting AUC values.

4.2 Binary classification with Profile Hidden Markov Models

So far PHMMs have been used as one-class classifiers. This is intuitive from a biological perspective.There is class imbalance and more emphasis on building a model which represents and identifies a family of related proteins. However, from a Machine Learning perspective, in a discriminative classification task, the information provided by negative instances has been shown to be useful (Jaakkola et al., 1999; Liao and Noble, 2002). The protein classification problem is characterised by imbalanced data sets. This differs from standard one-class classification where negative information is missing at training time (Tax, 2001).

This section shows how to transform a one-class PHMM into a binary PHMM classifier. The motivation is to increase the predictive power. However, in this particular learning scenario, any model for the negative class is trained on a huge amount of negative data. Therefore, training and testing time will increase.

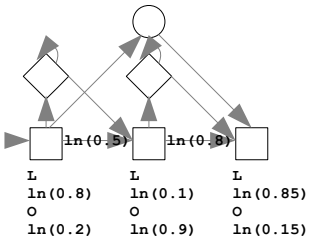
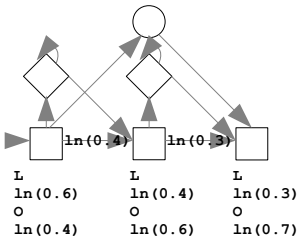
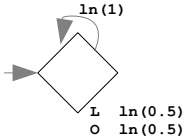
4.2.1 Combination of two one-class classifiers

In standard discriminative learning there is one model built on all training examples. Our approach is to build two models; one model for the positive instances and one model for the negative instances. Thus, we train two one-class PHMMs on a disjunct set of training instances. Both one-class PHMMs have the same graphical structure, e.g. the same number of match states.

At testing time, the two PHMMs output a log-odds score for the sequence. The class that maximises the score is predicted.

This is a simple extension of the previously introduced one-class PHMM. Figure 4.4 explains how the binary PHMM classifier works on the simple two class problem of *smile* and *no smile*. The binary PHMM depicted in this figure is fully trained and only the parameters that are necessary to score the test sequence 'LOL' are shown. As in the one-class case, we calculate the log-odds scores for both one-class PHMMs. In this example the test sequence 'LOL' leads to a log-odds score of 0.67 for the positive one-class PHMM as before. The cor-

<table>
<tr><td>one-class classifiers

after training a PHMM for each class separately</td><td>one-class PHMM for class “smile”
 
</td><td></td><td>one-class PHMM for class “non-smile”
 
</td></tr>
<tr><td rowspan="2">at testing time</td><td>test sequence: LOL</td><td></td><td>test sequence: LOL</td></tr>
<tr><td>calculate log likelihood:
ln(0.8)+ln(0.5)+ln(0.9)+
ln(0.8)+ln(0.85)= -1.41</td><td></td><td>calculate log likelihood:
ln(0.6)+ln(0.4)+ln(0.6)+
ln(0.3)+ln(0.7)= -3.50</td></tr>
<tr><td>uniform null model: data and problem independent</td><td></td><td> 

calculate log likelihood:
3*ln(0.5)+
2*ln(1)= -2.08</td><td></td></tr>
<tr><td>define and calculate the similarity measure</td><td>calculate log odds:
-1.41 - (-2.08) = 0.67</td><td></td><td>calculate log odds:
-3.50 - (-2.08) = -1.42</td></tr>
<tr><td>predict class with maximum log odds</td><td colspan="3">The one-class PHMM for the positive class “smile” maximises the log-odds for test sequence LOL.

Classify LOL as positive.
It belongs to class “smile”.

score for LOL is 0.67 or its normalised probability is 0.89</td></tr>
</table>

Figure 4.4. *An example of binary PHMM classification. The binary PHMM consists of two one-class PHMMs; one for the class smile and one for the class non smile. Training and test sequences are built from characters ‘L’ and ‘O’. The PHMMs have three match states. The binary PHMM is shown after training. Only parameters that influence the classification of the test sequence ‘LOL’ are shown.*

responding score for the negative one-class PHMM is -1.42. Up to this point the example follows the one-class case. To predict the class label for 'LOL', we predict the class with the higher log-odds score. This is again the positive class.

As pointed out earlier to evaluate all our approaches including the binary PHMM, we use AUC. WEKA requires a probability for each test instance in order to calculate AUC. In the case of binary PHMM classification there is no need for a calibration of logarithmic scores as we did in the one-class case. We have two log-odds scores lo_{pos} from the positive one-class classifier and lo_{neg} from the one-class PHMM for the negative class. We can use standard normalisation to define $P(X|H_{binary})$ the probability of the test sequence X given the binary PHMM H_{binary}.

$$P(X|H_{binary}) = \frac{e^{lo_{pos}(X)}}{e^{lo_{pos}(X)} + e^{lo_{neg}(X)}}$$

All test sequences are sorted according to this probability for calculating AUC.

The assignment of the class label in accordance with the maximum score $\max(lo_{pos}(X), lo_{neg}(X))$ is independent of the null model as the following equations show:

$$\begin{aligned} \max(lo_{pos}(X), lo_{neg}(X)) &= \max(ll_{pos}(X) - ll_{null}(X), ll_{neg}(X) - ll_{null}(X)) \\ &= \max(ll_{pos}(X), ll_{neg}(X)) \end{aligned}$$

Instead of calculating the log-odds scores, the correct class label can be assigned by choosing the class that maximises the log likelihood. Figure 4.5 shows our 'LOL' example with this modification.

The log likelihood of a test sequence differs from its log-odds score. However, normalisation makes sure that both approaches lead to the same probability and therefore are equivalent concerning our evaluation strategy. The following equation shows it mathematically:

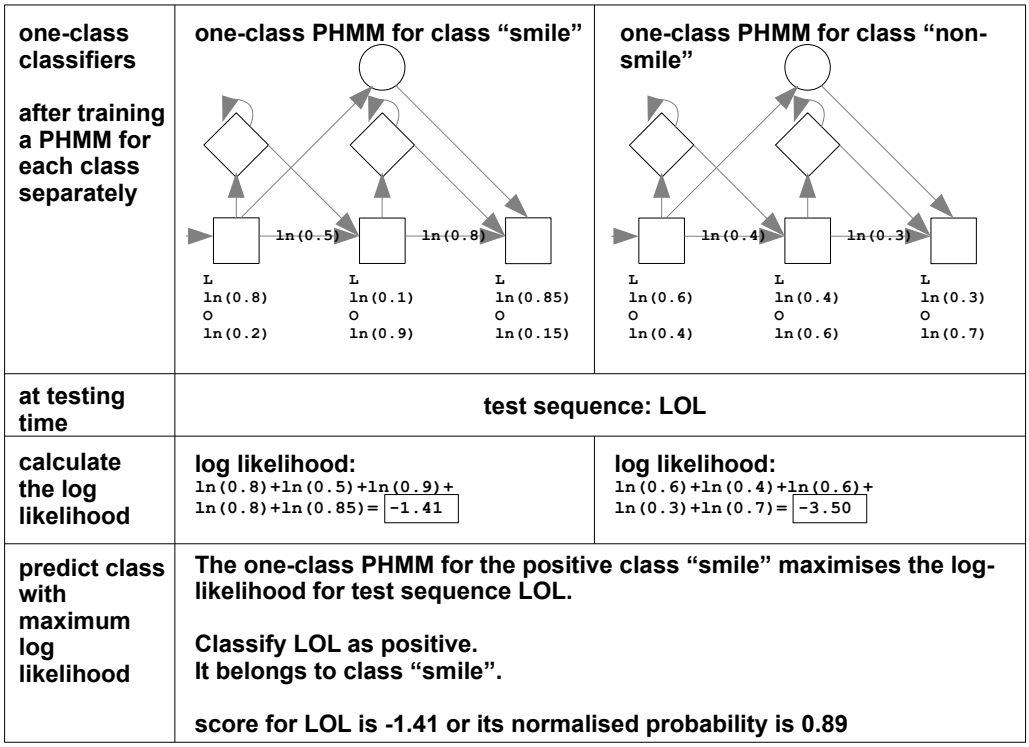


Figure 4.5. *An example of binary PHMM classification employing log likelihood*

$$
\begin{aligned}
P(X|H_{binary}) &= \frac{e^{lo_{pos}(X)}}{e^{lo_{pos}(X)} + e^{lo_{neg}(X)}} \\
&= \frac{e^{ll_{pos}(X) - ll_{null}(X)}}{e^{ll_{pos}(X) - ll_{null}(X)} + e^{ll_{neg}(X) - ll_{null}(X)}} \\
&= \frac{e^{ll_{pos}(X)}}{e^{ll_{null}(X)}} \cdot \frac{e^{ll_{null}(X)}}{e^{ll_{pos}(X)} + e^{ll_{neg}(X)}} \\
&= \frac{e^{ll_{pos}(X)}}{e^{ll_{pos}(X)} + e^{ll_{neg}(X)}}
\end{aligned}
$$

To summarise, we train a PHMM H_{pos} on the positive instances and separately a second PHMM H_{neg} on the negative instances. The PHMM H_{pos} is the PHMM from the one-class classification case. At classification time, a test

sequence X is assigned the class label of the corresponding PHMM that has a higher log likelihood. In order to calculate the AUC value, the log likelihood values are normalised to probabilities.

At an implementational level it can still be beneficial to use log-odds scores as they prevent numeric instabilities by shifting the resulting log likelihoods (Durbin et al., 1998). Our implementation follows this recommendation.

One advantage of this simple approach is that dataset imbalance does not influence the results.

4.2.2 Training and testing time

The large number of negatively labelled training sequences boosts the predictive power in a discriminative task. However, this optimisation comes at a cost. In this case, training and testing an additional one-class PHMM on a huge number of sequences requires more time than in the one-class setting.

A training sequence T of length n in a one-class PHMM with k states needs $O(n \cdot k)$ time in one iteration of the Baum-Welch training algorithm. In this binary setting there are more sequences. However, the negative class is more diverse. Thus, there are fewer iterations for training the one-class PHMM for the negative class. However, the vast increase in size of the training set leads to an increase in training time.

At testing time each of the two one-class PHMMs in the binary setting require $O(n \cdot k)$ to score a test sequence X of length n in a PHMM with k states. Practically, a one-class PHMM is twice as fast in scoring a test sequence than a binary PHMM, because in the binary PHMM, the one-class PHMM for the negative and the positive class, both, have the same number of states k. However, to calculate the AUC in WEKA, the one-class PHMM has to learn a logistic model on the output as well. This takes extra time, but training and evaluating a logistic model on one numeric and one binary target attribute is fast compared to PHMM runtime. For AUC calculation in general this overhead is not required.

4.3 Sampling the negative class

Ideally we want to find a smaller sample of the negative class that still allows discrimination. In this way, the model has a comparable AUC but is not much slower to train and test compared to the one-class PHMM. Additionally, sampling can, as in one-class classification, be interpreted as a strategy to deal with an imbalanced dataset.

All sampling strategies presented in this section belong to the class of undersampling methods.

There are two basic strategies to sample the negative class:

- *Directed undersampling* selects a number of negative sequences which have a known similarity to the positive class. This similarity is measured by the log-odds score and therefore the sampling is score based. Different score based approaches use different degrees of similarity.
- *Random undersampling* works without any prior knowledge of how the negative sequences relate to the positive class. This sampling strategy will be called uniform sampling.

Score-based sampling will be introduced in Section 4.3.1, whereas Section 4.3.2 gives more details about uniform sampling.

All sampling approaches are evaluated on all datasets except *pro_0*. In this particular dataset the positive class is the majority class.

In each iteration of Baum-Welch training a new sample of the negative class is taken. The sample size of the negative class equals the size of the positive class. In this way all proposed forms of undersampling make sure that the datasets are perfectly balanced after the sampling step. Figure 4.6 shows the percentage of instances used in one iteration of the training algorithm. The graphs in this Figure assume that the full dataset is used for training. As explained in Section 3 all experiments are performed using one ten fold cross-validation. As this Figure demonstrates, *enzyme_8* for example only uses 4% of the data in each iteration.

4.3.1 Score-based sampling

In score-based sampling, the one-class PHMM for the positive class assigns in a first step, a log-odds score to each negative sequence. Subsequently, negative

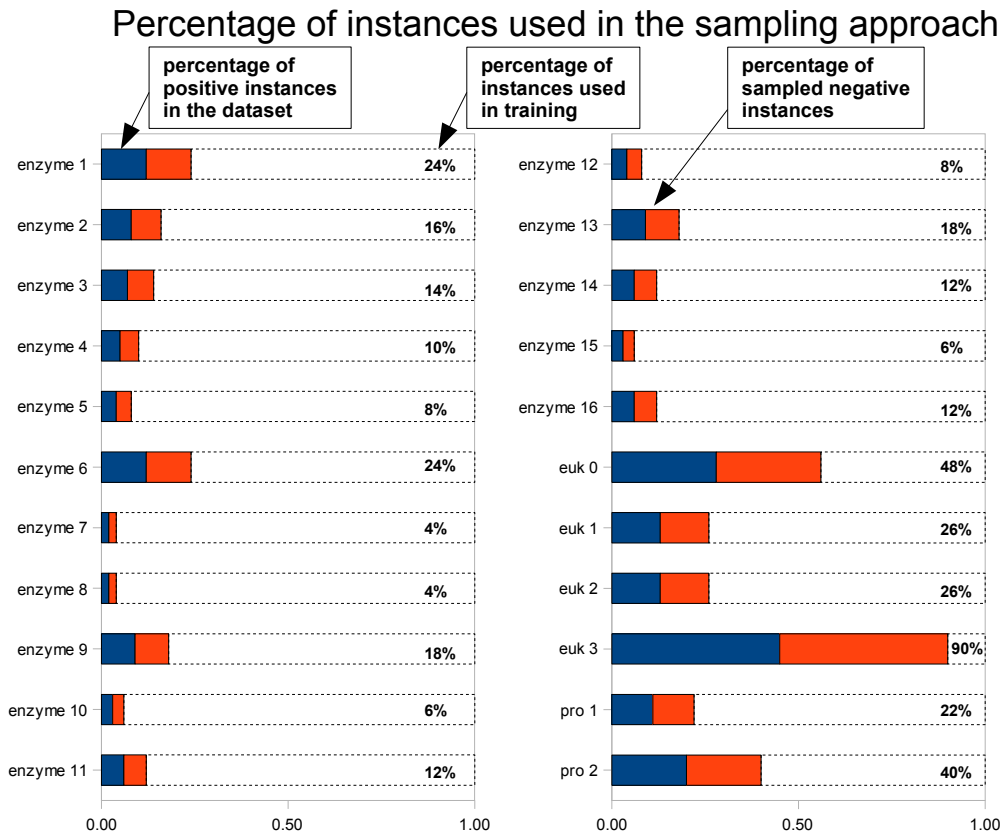


Figure 4.6. *The percentage of instances used in sampling in one iteration of the Baum-Welch training algorithm. The percentage of positive instances is coloured in blue, the negative in red. The full dataset is used for training in this figure.*

sequences are chosen at random depending on their score.

The basic idea is to guide the sampling step by providing a score that shows how closely related a particular negative sequence is to the positive class. Based on this similarity we use three different strategies:

- We take negative sequences that score highest with the positive one-class PHMM. These sequences are close to the discriminative decision surface and should provide more information.

- We take negative sequences that score lowest with the positive one-class PHMM. These sequences are identified as negative easily by the positive one-class model. They might contain information that allows modelling the negative class consistently.

- We take negative sequences and make sure that the sample contains high and low scoring ones and some that score in between. This sample should represent the negative class well. We call this approach stratified sampling.

In each training iteration, the positive one-class PHMM is trained first. Subsequently it is used to score the negative sequences. Samples from these negative sequences are chosen based on their score. The negative one-class PHMM then trains on the chosen negative training sample.

4.3.2 Uniform sampling

Uniform sampling is simple and fast. It is fast because the positive one-class PHMM does not need to score any sequence. This sampling method is simple because in each iteration of the Baum-Welch PHMM training algorithm we take negative sequences completely at random.

By drawing a new sample in each iteration of the training algorithm, uniform sampling can cover the full variance of the negative class.

4.3.3 Other sampling ideas

A trained one-class PHMM for the positive class is a generative model. Therefore it is able to generate new artificial sequences that belong to this family of proteins. The degree to which a newly generated sequence belongs to the positive class is expressed by the score of the sequence.

The idea is to create artificial sequences that are low scoring and therefore not likely to belong to the positive class and use these as artificial negative examples instead of the real world ones. This approach has more of a theoretical than a practical value, because in practice negative examples do exist. However, using only positive and artificially created examples keeps this approach as pure one-class classification. However, this idea did not lead to competitive results based on AUC and performed even worse than the basic one-class classification approach. In addition, generating sequences that score very low is not fast as most sequences that are generated by a one-class PHMM score well under the model that created them.

4.4 A null model interpretation

The previous section transformed the basic one-class PHMM approach to a binary one by training a separate one-class PHMM for the negative class. From a Machine Learning point of view the previous section discussed binary classification as opposed to one-class classification. However, from a biological perspective the PHMM of the negative class can be interpreted differently. It acts as a null model for the one-class PHMM trained on the positive instances.

Coming back to the example class *smile* and the classification task to assign a label to the sequence 'LOL', Figure 4.7 shows the algorithm when the negative one-class PHMM acts as a null model. A comparison of this example with the one presented in Figure 4.2 reveals that the log-odds score for the test sequence 'LOL' increases from 0.67 in traditional one-class classification to 2.09 with a negative one-class PHMM acting as null model. As discussed earlier finding the right threshold Θ on the log-odds score that decides class membership is a difficult problem. However, the increase in the log-odds score shows that the PHMM classifier with a one-class PHMM null model classifies the instance 'LOL' more likely as positive.

From a mathematical point of view both interpretations of our approach are equivalent under the AUC. We show that a change between these interpretations does not have any influence on the ranking of the test instances.

The AUC of the binary PHMM interpretation and the null model interpretation is the same, if and only if for any two instances X and Y when:

$$ll_{pos}(X) - ll_{neg}(X) \;>\; ll_{pos}(Y) - ll_{neg}(Y)$$

under the null model interpretation, the binary PHMM interpretation satisfies:

$$\frac{e^{ll_{pos}(X)}}{e^{ll_{pos}(X)} + e^{ll_{neg}(X)}} \;>\; \frac{e^{ll_{pos}(Y)}}{e^{ll_{pos}(Y)} + e^{ll_{neg}(Y)}}$$

This condition holds as the next equation shows:

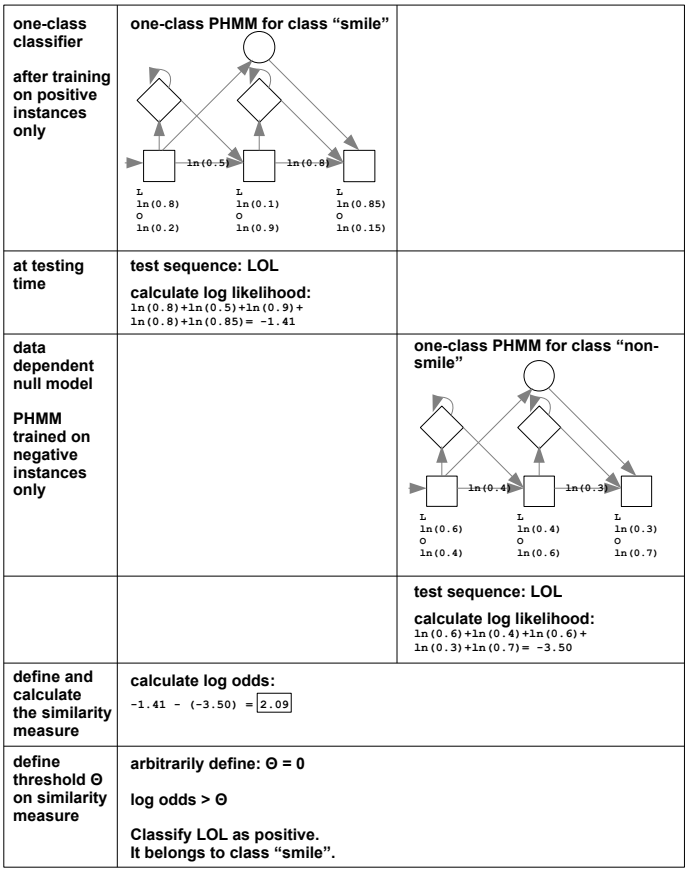


Figure 4.7. *An example of PHMM classification using the one-class PHMM built on the negative data as null model. Training and test sequences are built from characters 'L' and 'O'. The PHMMs have three match states. The binary PHMM is shown after training. Only parameters that influence the classification of the test sequence 'LOL' are shown.*

$$
\begin{aligned}
\frac{e^{ll_{pos}(X)}}{e^{ll_{pos}(X)} + e^{ll_{neg}(X)}} &> \frac{e^{ll_{pos}(Y)}}{e^{ll_{pos}(Y)} + e^{ll_{neg}(Y)}} \\
e^{ll_{pos}(X) - ll_{pos}(Y)} \cdot (e^{ll_{pos}(Y)} + e^{ll_{neg}(Y)}) &> e^{ll_{pos}(X)} + e^{ll_{neg}(X)} \\
e^{ll_{pos}(X)} + e^{ll_{pos}(X) - ll_{pos}(Y) + ll_{neg}(Y)} &> e^{ll_{pos}(X)} + e^{ll_{neg}(X)} \\
ll_{pos}(X) - ll_{pos}(Y) + ll_{neg}(Y) &> ll_{neg}(X) \\
ll_{pos}(X) - ll_{neg}(X) &> ll_{pos}(Y) - ll_{neg}(Y)
\end{aligned}
$$

In a practical implementation there might be numerical differences due to imprecision when calculating logs and transforming them again using the exponential function. In addition, if we follow the null model interpretation and implement the system in WEKA, the resulting log-odds scores need to be transformed again into probabilities by learning a logistic function. Our implementation follows the binary classification interpretation and uses the exponential function and normalisation to transform log-odds scores to probabilities.

Binary classification with two PHMMs outperforms the one-class case where only a PHMM trained on the positive instances is used. This statement can be worded alternatively saying that a sequence- and model-specific null model represented by a one-class PHMM on the negative class outperforms the uniform null model.

4.4.1 A global null model

In general, finding a good null model for a problem is a crucial and difficult task. As Gotelli (2001) points out, a null model is a model that deliberately excludes the mechanism being tested. The simplest way to achieve this in our setting, is to use a null model that assumes that every amino acid in a protein is equally likely to occur in any position. This is the uniform null model. It is independent of the training data.

4.4.2 A model-specific null model

An alternative to the uniform null model is to include prior information. The prior information available in this research work are the amino acid sequences in the training set. Therefore, it is possible to use their background distribution as an alternative null model. However, this might violate Gotelli (2001)'s condition.

A PHMM is a generative, probabilistic one-class classifier. This is true for a standard PHMM with a uniform null model. However, in the context of null models, it is the choice of null model that changes the PHMM classification from purely one-class on one extreme to a fully binary approach on the other. The former is the one-class PHMM trained on the positive instances and the latter is the binary PHMM. However, depending on the null model there could

be different kinds of PHMM that include more than the information from the positive training instances without fully training a model for the negative class, e.g. a null model that uses the background distribution from the negative class. This is not one-class classification anymore, even though we train a one-class PHMM. From a Machine Learning point of view the null model plays a crucial role in differentiating approaches that are purely one-class or binary. This is an important observation in the null model discussion.

4.4.3 Sequence- and model-specific null models

Section 2.2.5 introduced and motivated the reverse sequence null model from Karplus et al. (1998, 2005). They use the score of the reversed sequence from the positive one-class PHMM as null model score. It corrects for length and composition biases and constitutes a strict one-class approach. In most of our datasets the global, uniform null model outperforms the reverse sequence one. The terms reverse sequence null model and reverse null model are used interchangeably in this thesis and describe the same null model.

The negative one-class PHMM acts as a sequence- and model-specific null model as well. However, unlike the uniform and reverse sequence null models, that are inferior in terms of AUC, it constitutes a binary approach. A negative one-class PHMM as a null model satisfies Gotelli (2001)'s condition. In this supervised learning setting, we train the negative one-class PHMM exclusively on the negative instances. These instances have been labelled negative because of the absence of a particular mechanism or pattern that defines the positive class provided the labelling is correct.

4.5 Experimental results

This section presents the experimental results for one-class PHMMs and binary PHMMs on our datasets. It splits into two main sections. The first compares one-class and binary PHMM classification without sampling. The second shows the benefits of sampling the negative class. Chapter 3 explains the experimental setup and introduces the datasets in detail. As pointed out in that chapter, models are evaluated after each iteration of the Baum-Welch training algorithm and

not only after convergence. We provide details about the statistical significance of differences in their performances after full convergence.

4.5.1 One-class and binary classification without sampling

AUC

This section compares binary PHMMs to one-class PHMMs with a uniform and a reverse null model. Succinctly put, there is no dataset in our study for which a one-class approach performs statistically significantly better in terms of AUC after training is fully completed. The one-class PHMM with a reverse null model is only competitive on one occasion in terms of AUC compared to a binary PHMM. The one-class model with a uniform and thus simpler null model performs better and is competitive to binary PHMMs in four of 23 datasets after full convergence of the Baum-Welch training algorithm. In the following we will present the results in detail.

Figure 4.8 shows the AUCs for *enzyme_4*, *enzyme_8*, *euk_3* and *pro_1*. Note that each graph depicts the AUC for a specific dataset in the iterations of the training algorithm. These curves are not ROC curves. In addition, each graph contains a table with the p-values originating from our analysis for statistical significance after the last iteration of the Baum-Welch training algorithm. Significance is tested at the 0.05 level. The tables contain the p-value for each pair of approaches, printed in bold if the difference in AUC is significant after training is fully completed.

The results for *enzyme_4*, *euk_3* and *pro_1* are representative of most of our datasets. The binary PHMM achieves the highest AUC of all three approaches and its predictive power after the final iteration of the Baum-Welch training algorithm is statistically significantly better than the one-class PHMMs. When comparing the two one-class PHMMs with different null models, the simpler uniform null model performs better for all or most training iterations and leads to a significantly higher AUC after the one-class PHMMs' training is fully completed. The situation is slightly different for *enzyme_8* which is the dataset with the smallest number of positive instances. Again, the binary PHMM performs better, in terms of AUC through the entire training process, than the two one-class approaches. In the end, the differences are statistically significant.

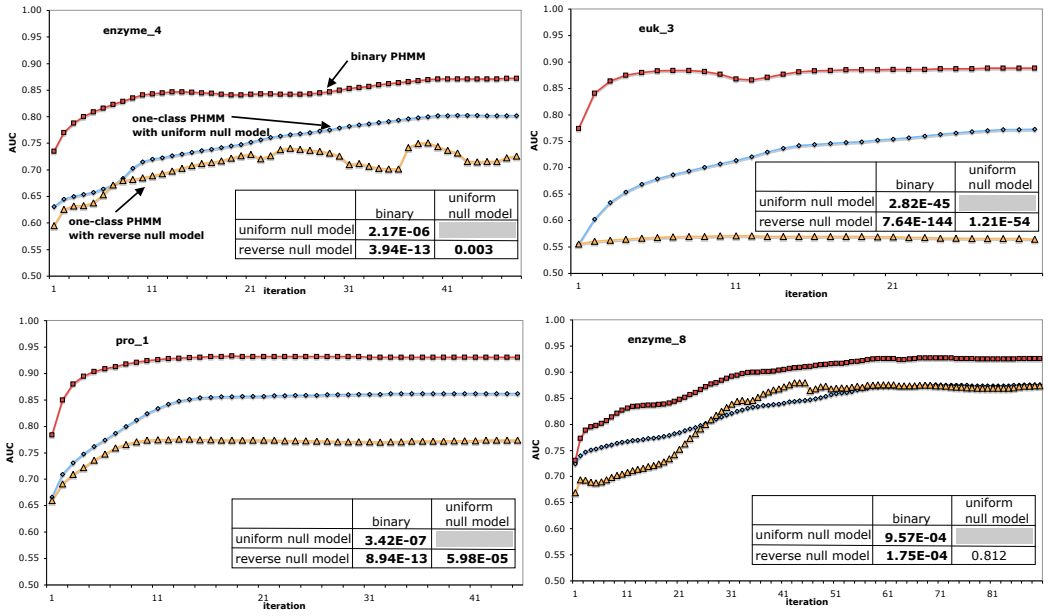


Figure 4.8. *AUC values for enzyme_4, enzyme_8, euk_3 and pro_1 in one-class and binary PHMM classification. In the one-class case, a uniform and a reverse null model is used. The AUC is calculated in each iteration of the Baum-Welch training algorithm. The table shows the p-value for each pair of approaches after the last iteration of the Baum-Welch algorithm. Statistical significance is tested at the* 0.05 *level. A significant difference is printed in bold.*

However, there is no statistically significant difference in the performance of the one-class PHMM with a uniform null model and with a reverse null model as the one-class PHMM with a uniform null model achieves an AUC of 0.876 and its reverse counterpart an AUC of 0.874. In contrast, the binary PHMM reaches an AUC of 0.926 after being fully training.

Similarly, as Figure 4.9 shows, there is again no statistically significant difference in the AUC of the one-class approaches for *enzyme_5*. However, for this dataset the binary PHMM does not perform statistically significantly different in terms of AUC when comparing the results of the final training iterations. The binary approach reaches an AUC of 0.837, just slightly but not significantly higher than the one-class PHMM with a uniform null model. Its AUC is 0.834. The one-class PHMM with a reverse null model achieves an AUC of 0.807. This is the only dataset where the binary approach does not outperform the one-class PHMM when a reverse null model is used. For *enzyme_1* the binary PHMM leads

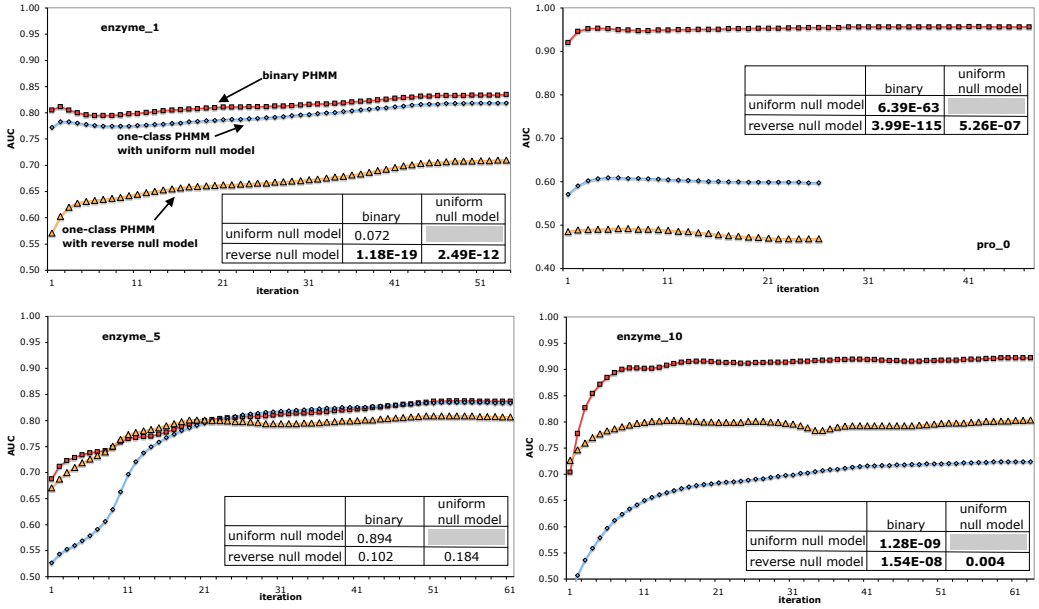


Figure 4.9. *AUC values for enzyme_1, enzyme_5, enzyme_10 and pro_0 in one-class and binary PHMM classification. In the one-class case, a uniform and a reverse null model is used. The figure follows the layout of Figure 4.8.*

to a higher AUC for all training iterations compared to the one-class models. However, after full convergence, the difference is only statistically significant compared to the reverse null model. The simpler one-class approach with a uniform null model performs equally well as the binary one. We encounter this exact behaviour in two more datasets, namely *enzyme_2* and *enzyme_14*. Their corresponding figures B.1 and B.2 can be found in Appendix B.1. Note, that *enzyme_10* is the only dataset where the one-class PHMM with reverse null model significantly outperforms the one-class PHMM with a uniform null model. But again, the binary approach is better than both of them.

The dataset *pro_0* is a special case. The one-class PHMMs with uniform and reverse sequence null model converge faster than the binary PHMM. For all other dataset except *euk_0* the one-class approaches and the binary approach converge at the same iteration. The reason is that for all but these two datasets, the one-class PHMM for the positive class, whose instances share a pattern, needs more iterations than its negative counterparts. As Section 3.2.1 explains, there might not be a pattern in the positive class of *pro_0* and *euk_0*. The binary PHMM significantly outperforms both one-class PHMMs even though the

positive class is the majority class and the negative class the minority class. This is the opposite situation to all other datasets and to our assumption that the positive class is the minority class. However, our approach works well due to biological reasons. The negative class, in this case, contains a common pattern, whereas there is likely none in the positive class. This is why the one-class approaches do not work well. Thus, this behaviour suggests that adding negative information is not only useful, when there are only a few positive instances. The information contained in the negative class can boost predictive performance in general.

Figure 4.10 displays the results for *enzyme_12*, *enzyme_13*, *enzyme_15* and *enzyme_16*. For all these datasets, the binary approach leads to higher AUCs

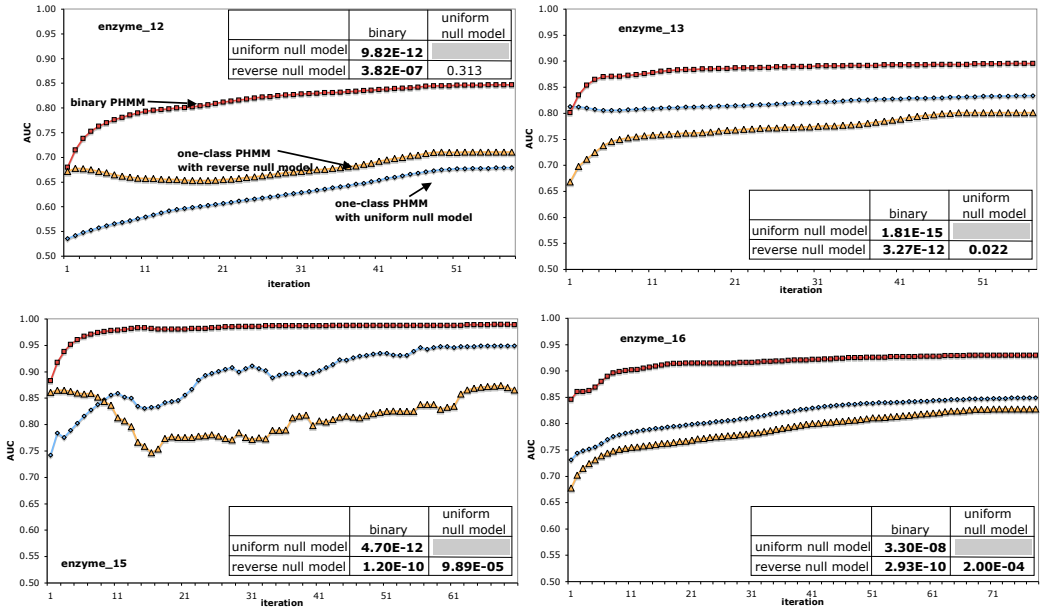


Figure 4.10. *AUC values for enzyme_12, enzyme_13, enzyme_15 and enzyme_16 in one-class and binary PHMM classification. In the one-class case, a uniform and a reverse null model is used. The figure follows the layout of Figure 4.8.*

than the one-class ones and the differences are again statistically significant after convergence of the training algorithm. The one-class PHMM with a uniform null model achieves significantly higher AUCs after training is completed compared to the one-class PHMM with a reverse null model for *enzyme_13*, *enzyme_15* and *enzyme_16*. For *enzyme_12* the reverse null model outperforms

the uniform one, but the difference is not statistically significant. Appendix B.1 shows the same behaviour for *enzyme_7*, *enzyme_9* and *pro_2*.

The graphs depicting the AUCs and p-values of all datasets that have not been presented in this section can be found in Appendix B.1.

Table 4.1 summarises the findings for fully converged PHMM models. A one-class PHMM with a reverse null model only leads for one dataset to a significantly higher AUC than the one-class PHMM with a uniform null model. On the contrary, the one-class PHMM with a reverse null model decreases AUC sig-

Table 4.1. *Comparison of AUCs for one-class PHMM classification with a uniform and a reverse null model and binary PHMM classification. The table indicates statistical significance at the* 0.05 *level compared to one-class PHMM classification with a uniform null model.*

dataset	one-class PHMM		binary PHMM
	uniform null model	reverse null model	
enzyme_1	0.819	0.710 (−)	0.835 (+)
enzyme_2	0.864	0.782 (−)	0.884 (=)
enzyme_3	0.679	0.621 (−)	0.783 (+)
enzyme_4	0.802	0.726 (−)	0.872 (+)
enzyme_5	0.834	0.807 (=)	0.837 (=)
enzyme_6	0.838	0.649 (−)	0.870 (+)
enzyme_7	0.759	0.779 (=)	0.847 (+)
enzyme_8	0.876	0.874 (=)	0.926 (+)
enzyme_9	0.909	0.923 (=)	0.962 (+)
enzyme_10	0.724	0.804 (+)	0.922 (+)
enzyme_11	0.941	0.914 (−)	0.955 (+)
enzyme_12	0.679	0.711 (=)	0.847 (+)
enzyme_13	0.834	0.801 (−)	0.896 (+)
enzyme_14	0.993	0.975 (−)	0.998 (=)
enzyme_15	0.949	0.866 (−)	0.989 (+)
enzyme_16	0.849	0.828 (−)	0.930 (+)
euk_0	0.600	0.503 (−)	0.831 (+)
euk_1	0.878	0.852 (−)	0.971 (+)
euk_2	0.769	0.722 (−)	0.872 (+)
euk_3	0.773	0.565 (−)	0.888 (+)
pro_0	0.589	0.469 (−)	0.956 (+)
pro_1	0.862	0.774 (−)	0.931 (+)
pro_2	0.800	0.804 (=)	0.875 (+)

nificantly for 16 datasets. The binary PHMM is the most successful model as it significantly increases AUC in 20 cases compared to a one-class PHMM with a uniform null model. There is no significant loss. For the three remaining datasets, the binary PHMM boosts predictive performance compared to a one-class PHMM. However, the difference is statistically not significant.

Generally speaking, it is an advantage in a discriminative task like protein classification to include negative information. However, this benefit clearly has some implications on runtime.

Runtime and iterations

Even though there is an increase in AUC when a binary approach is taken instead of a pure one-class approach, the vast amount of negative information leads to an increase in training time. Testing in the binary case theoretically needs double the time of the one-class setting. But our implementation in WEKA requires an additional logistic model in the one-class case. However, it is mainly training that contributes to the increase in runtime.

Figure 4.11 shows the runtime for the one-class and the binary setting for the datasets *enzyme_1* and *enzyme_8*. Note that even though care has been taken

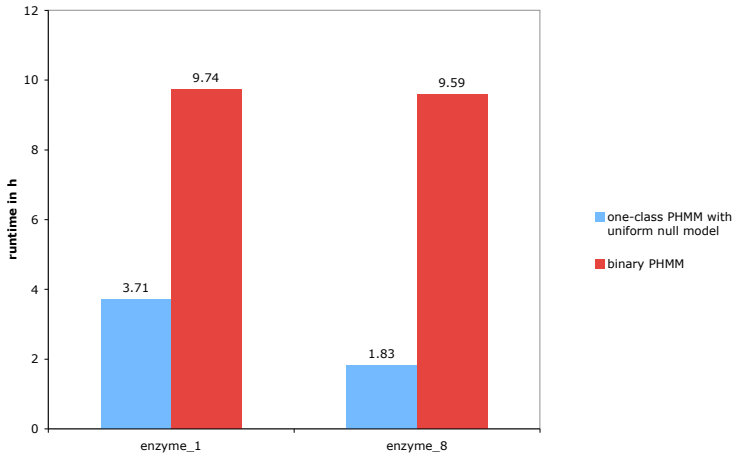


Figure 4.11. *Runtime for enzyme_1 and enzyme_8. The reported runtimes in hours include evaluation of the classifiers after each training iteration.*

when running experiments for timing, we did not have guaranteed exclusive access to the servers that ran our experiments. There are 317 positively labelled

sequences in dataset *enzyme_1*, whereas *enzyme_8* contains only 59 sequences with a positive class label. They represent the problems with the largest and smallest number of positively labelled instances from all *enzyme* datasets. Their runtime differs in the one-class case because *enzyme_1* has more than five times as many instances as *enzyme_8*. However, the difference in runtime is less than five times as *enzyme_8's* positive one-class PHMM needs 90 iterations to converge whereas the one for *enzyme_1* only takes 54.

We can clearly see in Figure 4.11 that binary PHMM classification needs considerably more time than its one-class counterpart. However, even though the differences are large, they are smaller than expected when looking at the number of sequences we additionally train in the negative one-class PHMM for binary classification. The reason is that the negative one-class PHMM of the binary classifier converges earlier than its positive counterpart. Its Baum-Welch training converges after 19 iterations for *enzyme_1* and after 20 for *enzyme_8*. The fact that the negative one-class PHMM converges much faster than the positive is true for all datasets except *pro_0* and *euk_0*. The negative class in the majority of datasets is more diverse and its PHMM converges therefore faster. However, the faster convergence cannot compensate for the increase in training instances.

AUC under other null models

So far, all experiments in one-class classification use a uniform or reverse null model. In the case of binary PHMMs, the negative one-class PHMM acts as a null model. Instead of learning another PHMM on top of the positive one-class PHMM, the background distribution of the amino acids can be used. There are three different ways to do this:

- Calculate the background distribution of amino acids in the positive training set. This is a one-class approach.
- Calculate the background distribution of amino acids in the negative training set and therefore, do not perform pure one-class classification anymore.
- Calculate the background distribution of amino acids in the entire training set and therefore, do not perform pure one-class classification either.

Figure 4.12 shows the results in terms of AUC and compares the three null models with the uniform null model from the previous section. For *enzyme_1*

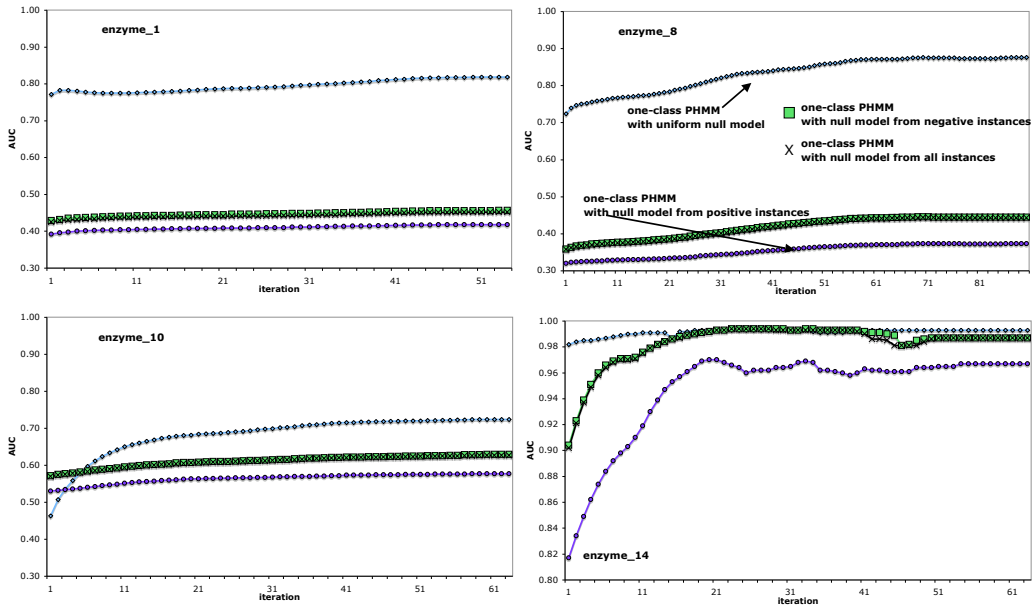


Figure 4.12. *AUC values for enzyme_1, enzyme_8, enzyme_10 and enzyme_14 in one-class PHMM classification. The figure shows the AUCs for the uniform null model and the null models built from the background distribution of the positive, the negative and all training instances. Please be aware that the y-axis is differently scaled for enzyme_14.*

and *enzyme_8* it is evident at first sight that the uniform null model clearly outperforms its three alternatives. Their AUC values are below 0.5 for the entire training process. After training is completed, the differences in AUC between the uniform null model and each of the three alternatives are statistically significant at a 0.05 level. The situation is different for *enzyme_10* and *enzyme_14*. These are the only two datasets for which there is no statistically significant difference in AUC when comparing the uniform null model to the ones created from the background distribution of the negative training instances and from the background distribution of all training instances. The AUCs of the uniform null model and the null model built from the background distribution of the positive training instances is statistically significantly different for these two datasets as it is the case for all other datasets as well. Therefore, when comparing these four null models, the uniform null model performs better by and

large. However, taking into account the results from the previous section, it is the binary approach where the negative one-class PHMM acts as a null model, that performs the best.

Looking at the differences between the three alternative null models in Figure 4.12, we see that the null model built from the background distribution of the positive instances performs the worst. This approach and the one with a uniform null model are pure one-class approaches. The null model based on the background distribution from the positive training instances performs the worst in all datasets and is always statistically significantly worse after training is finished compared to the null models built from the background distribution of the negative training instances and from all training instances. Here again, it shows that using or adding negative information improves performance.

The two approaches based on null models derived from the negative or all training instances perform almost the same in terms of AUC. This holds for all datasets. However, even though the difference is minimal, the null model that is solely based on information from the negative training instances achieves a statistically significant higher AUC compared to the null model that uses all training instances. This is true for all datasets except *enzyme_14* where both approaches perform equally well. Table 4.2 gives an overview of the final AUCs for the datasets in Figure 4.12.

Table 4.2. *AUCs for alternative null models for enzyme_1, enzyme_8, enzyme_10 and enzyme_14 after the last iteration of the Baum-Welch training. The term 'negative' refers to using a null model with the background distribution of the negative training instances, whereas the term 'all' stands for the null model built from the background distribution of all training instances.*

dataset	AUC negative	AUC all
enzyme_1	0.457	0.452
enzyme_8	0.445	0.443
enzyme_10	0.629	0.627
enzyme_14	0.987	0.987

Figure 4.13 illustrates the results for *enzyme_4*, *enzyme_5*, *euk_2* and *pro_2*. The results stress the dominance of the uniform null model over the alternatives.

Appendix B.2 contains the results for all other datasets.

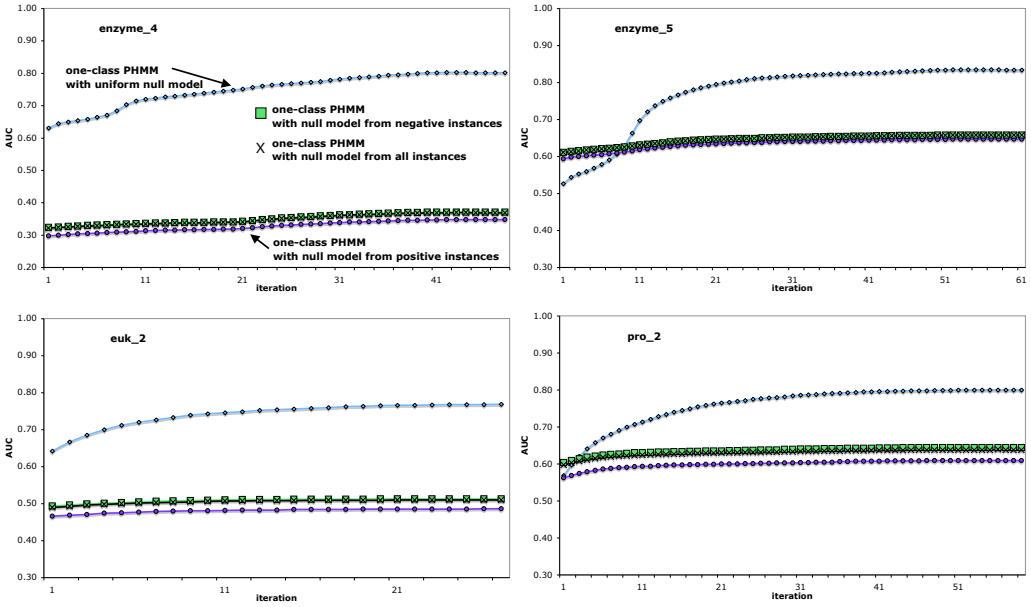


Figure 4.13. *AUC values for enzyme_4, enzyme_5, euk_2 and pro_2 in one-class PHMM classification. The figure shows the AUCs for the uniform null model and the null models built from the background distribution of the positive, the negative and all training instances*

Recall, all these alternative null models might suffer from the fact that they do not deliberately exclude the mechanism or pattern being tested as the background distribution might influence class membership.

4.5.2 Binary classification with sampling

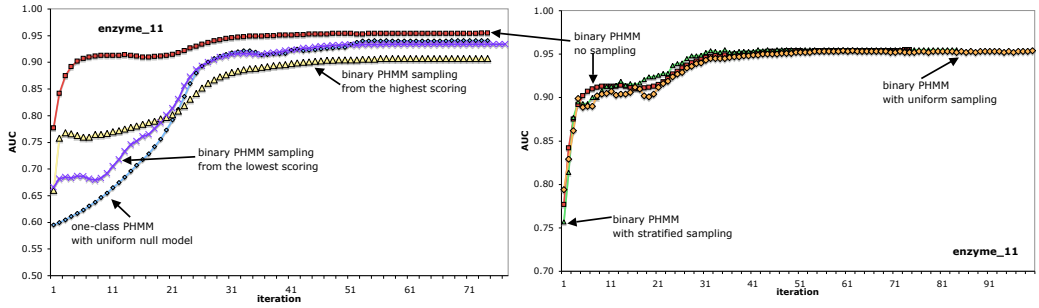
In this section the negative one-class PHMMs examine a sample of the negative class during training. We do not report results for the *pro_0* dataset, because its negative class is the minority class. Thus, there are 22 datasets in our experimental setup.

AUC

This section compares the four sampling approaches to one-class PHMM classification and the binary PHMM on the full training set based on their AUC values. The next section is concerned with runtime.

Training of PHMMs on the full training set stops after convergence. The criteria is a sufficiently small change in the overall log-odds score as Section 3.2.1 explains. The way sampling is implemented, the negative one-class PHMM trains on a different subset in each iteration of the Baum-Welch algorithm. This way, sampling covers the negative class better. However, the negative instances change in each iteration of the Baum-Welch algorithm. This might hinder convergence and therefore, training stops in the case of sampling after at most 100 iterations. Results will show that the AUC has levelled out at this stage.

Figure 4.14 shows the results for *enzyme_11*. The figure on the left compares



	binary no sampling	one-class uniform null model	binary uniform sampling	binary stratified sampling	binary sampling from the lowest scoring
one-class, uniform null model	**0.016**				
binary, uniform sampling	0.825	**0.022**			
binary, stratified sampling	0.819	**0.019**	0.762		
binary, sampling from the lowest scoring	**0.002**	0.243	**0.004**	**0.003**	
binary, sampling from the highest scoring	**1.99E-08**	**0.001**	**9.95E-07**	**1.09E-10**	**0.010**

uniform sampling statistically as good as a binary PHMM trained on a full dataset

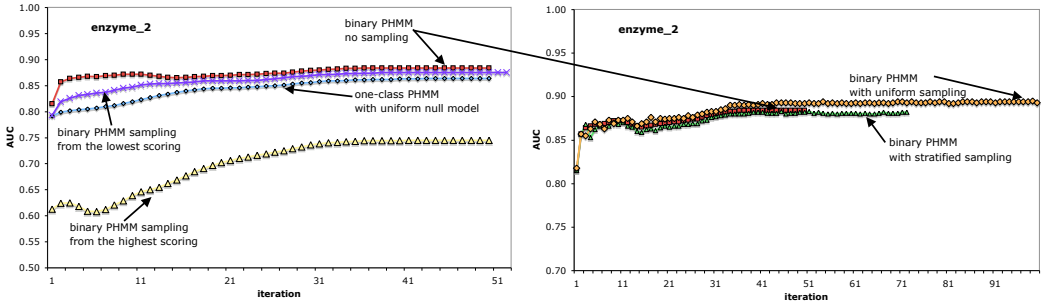
Figure 4.14. *Comparison of sampling approaches for enzyme_11. The figure on the left compares a one-class PHMM with an uniform null model, a binary PHMM trained without sampling, a binary PHMM sampling from the highest scoring negative instances and one that samples from the lowest scoring negatives instances. On the right, the binary PHMM without sampling is compared to uniform and stratified sampling combined with a binary PHMM. Note that the y-axes are scaled differently. The table below contains the p-values for all pairs of approaches after convergence. Bold print indicates a statistically significant difference at the* 0.05 *level. A green background colour indicates that there is no statistically significant difference between uniform sampling and no sampling. The absence of green colour shows that the binary PHMM with full information statistically significantly outperforms the uniform sampling approach.*

the predictive performance of one-class PHMM classification with a uniform null model to binary ones. The graph contains AUC values after each iteration of the Baum Welch training algorithm for standard binary PHMM classification when

the whole set of negative instances is used. In addition, it displays the results for sampling using the lowest and the highest scoring negative instances. The figure on the right contains the results for the standard binary PHMM again for reference, however, the scale of the y-axes are different. Additionally, the figure displays the AUC values for uniform and stratified sampling. The table underneath the figures contains the p-values for all pairs of approaches. Statistical significance is tested at a 0.05 level after the last iteration. Bold printed p-values indicate a statistically significant difference. One focus of our evaluation is to determine the relationship between fully training a binary PHMM and using the simple uniform sampling. Therefore, a green colour in the table indicates that uniform sampling is at least as good or better as fully training a binary PHMM based on statistical significance of the results.

For *enzyme_11* sampling from the highest and lowest scoring sequences decreases the AUC significantly compared to a standard binary PHMM. Training an additional PHMM from the lowest scoring sequences performs as well as one-class PHMM classification. However, the former trains an additional PHMM. When we look at the right-hand figure, the stratified and uniform sampling are competitive with standard binary classification in terms of AUC. The binary PHMM without sampling converges after 74 iteration, whereas the stratified sampling approach converges after 84 iterations. As explained before, training is stopped for uniform sampling after 100 iterations. The binary PHMM with uniform sampling has not converged, however its AUC has levelled out. After the training is completed there is no statistically significant difference between the standard binary approach and uniform and stratified sampling. Additionally, stratified sampling does not have a statistical significant advantage over uniform sampling even though its sampling is guided by a score. Seventeen of the 22 datasets share this relationship between uniform and stratified sampling. For three datasets uniform sampling significantly outperforms stratified sampling whereas for two datasets it is the other way around. More significantly, in 15 of 22 datasets the uniform sampling approach performs statistically equally well as using all training instances.

There is one dataset where uniform sampling leads to a statistically significant improvement in terms of AUC after training is completed compared to standard binary classification. This scenario is depicted in Figure 4.15 for *enzyme_2*.



	binary no sampling	one-class uniform null model	binary uniform sampling	binary stratified sampling	binary sampling from the lowest scoring
one-class, uniform null model	0.05				
binary, uniform sampling	**0.008**	**0.005**			
binary, stratified sampling	0.640	0.104	0.051		
binary, sampling from the lowest scoring	0.213	0.150	**0.011**	0.393	
binary, sampling from the highest scoring	**1.54E-10**	**6.62E-09**	**1.36E-20**	**5.14E-19**	**1.10E-13**

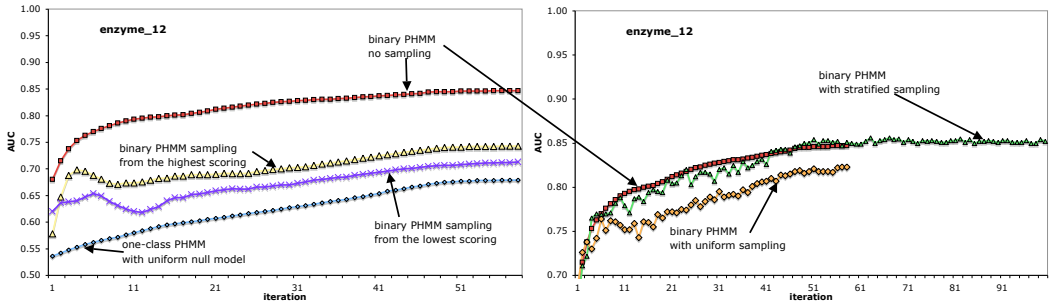
uniform sampling statistically **significantly better** than a binary PHMM trained on a full dataset

Figure 4.15. *Comparison of sampling approaches for enzyme_2. Note that the y-axes are scaled differently. The figure follows the layout of Figure 4.14.*

This is even more impressive because stratified sampling performs slightly worse than the standard approach but without a significant difference to binary PHMM classification on the full negative set. Again the different binary PHMMs converge after different iterations or their training is stopped after at most 100 iterations. Looking now at training an additional PHMM from a sample formed from the lowest scoring sequences, this approach leads to AUC values situated in between the one-class and the standard binary approach. There is no significant difference in AUC for the three approaches after training is completed. The strategy of sampling the lowest scoring sequences leads to a final AUC that is competitive compared to a binary PHMM without sampling for two datasets only, namely *enzyme_2* and *enzyme_14*. In all other cases, there is a statistically significant degradation in predictive performance. Figure B.17 in the appendix shows the results for *enzyme_14*. Using the highest scoring, negatively labelled sequence in the *enzyme_2* dataset leads to the worst results in terms of AUC. These negative sequences are the ones that score the most similar to positive ones. For all datasets, sampling with the highest scoring sequences is significantly worse in terms of AUC compared to not using any sampling at all.

So far the uniform sampling outperformed a binary PHMM trained on the positive instances or performed at least competitively.

As Figure 4.16 shows, the situation is different for *enzyme_12*. For this dataset



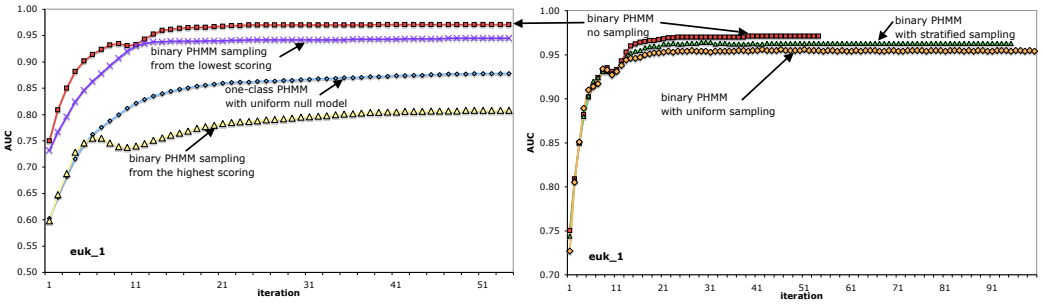
	binary no sampling	one-class uniform null model	binary uniform sampling	binary stratified sampling	binary sampling from the lowest scoring
one-class, uniform null model	**9.82E-12**				
binary, uniform sampling	**0.013**	**1.10E-08**			
binary, stratified sampling	0.506	**1.39E-11**	**0.015**		
binary, sampling from the lowest scoring	**1.68E-09**	0.054	**2.75E-06**	**3.99E-09**	
binary, sampling from the highest scoring	**3.90E-06**	0.072	**4.16E-04**	**3.70E-07**	0.353

Figure 4.16. *Comparison of sampling approaches for enzyme_12. Note that the y-axes are scaled differently. The figure follows the layout of Figure 4.14.*

sampling with the highest scoring sequences leads to higher AUC than using the lowest scoring ones. Both methods outperform the basic one-class approach. However, after training is completed none of these differences is statistically significant. On the other hand, all of these three techniques perform worse in terms of AUC throughout all iterations of Baum-Welch training when compared to a standard binary PHMM. In the end the differences are statistically significant. Uniform sampling significantly outperforms the aforementioned approaches. However, it is not as good as training a binary PHMM with complete negative information. Only stratified sampling outperforms the standard binary approach, but the difference in AUC is not significant. However, this time the binary PHMM with stratified sampling does not converge and the stratified sampling approach is stopped after 100 iterations. Uniform sampling is not only statistically significantly inferior to full binary training, it is also inferior compared to stratified sampling. The only other dataset where this is the case is *euk_1* and its results are summarised in Figure 4.17.

Stratified sampling performs better and statistically significantly outperforms uniform sampling after the last training iteration. However, opposed to *enzyme_12*, all sampling techniques including stratified sampling decrease AUC significantly after the final training iteration compared to using a binary PHMM without sampling.

The other datasets for which binary PHMM classification without sampling



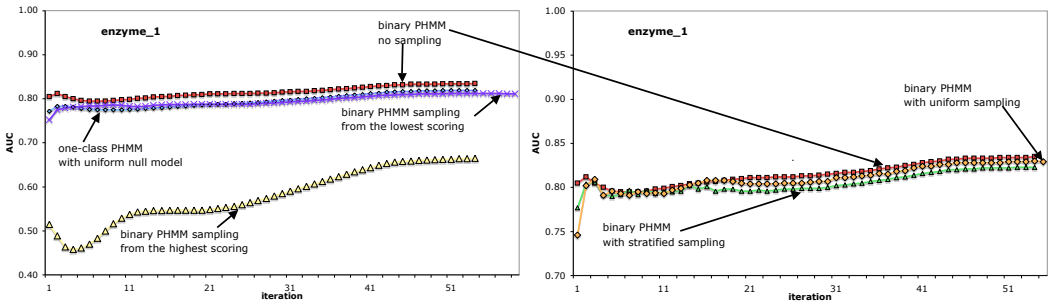
	binary no sampling	one-class uniform null model	binary uniform sampling	binary stratified sampling	binary sampling from the lowest scoring
one-class, uniform null model	**9.95E-20**				
binary, uniform sampling	**1.54E-09**	**1.56E-13**			
binary, stratified sampling	**5.45E-04**	**2.86E-16**	**7.09E-04**		
binary, sampling from the lowest scoring	**1.91E-07**	**9.93E-15**	0.091	**2.55E-04**	
binary, sampling from the highest scoring	**3.94E-70**	**0.000**	**2.27E-57**	**8.79E-64**	**9.96E-34**

Figure 4.17. *Comparison of sampling approaches for euk_1. Note that the y-axes are scaled differently. The figure follows the layout of Figure 4.14.*

is statistically significantly better than using uniform sampling are *enzyme_3*, *enzyme_13*, *enzyme_15* and *pro_1*. Their results can be found in Appendix B.3.

For *euk_1* sampling from the lowest scoring sequences consistently leads to higher AUCs than basic one-class classification and it is significantly better, whereas using the highest scoring sequences is even significantly worse than one-class classification.

Figure 4.18 displays the results for *enzyme_1*. For this dataset there is no



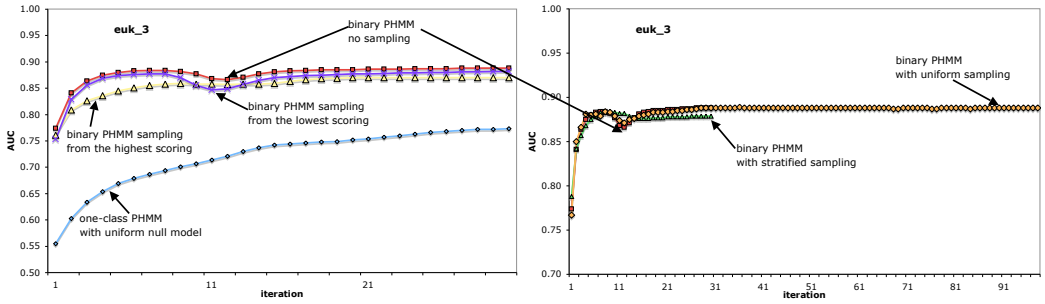
	binary no sampling	one-class uniform null model	binary uniform sampling	binary stratified sampling	binary sampling from the lowest scoring
one-class, uniform null model	0.072				
binary, uniform sampling	0.052	0.242			
binary, stratified sampling	**1.67E-05**	0.684	0.102		
binary, sampling from the lowest scoring	**3.40E-04**	0.261	**0.007**	0.113	
binary, sampling from the highest scoring	**1.40E-36**	**4.99E-19**	**7.45E-34**	**1.53E-34**	**5.83E-19**

uniform sampling statistically as good as a binary PHMM trained on a full dataset

Figure 4.18. *Comparison of sampling approaches for enzyme_1. Note that the y-axes are scaled differently. The figure follows the layout of Figure 4.14.*

statistically significant difference in terms of AUC for basic one-class, standard binary and binary PHMM classification with uniform sampling. Using stratified sampling leads to a significantly lower AUC when compared to the full binary approach. However, compared to the one-class and uniform approach, there are no statistically significant differences in their predictive power. The same holds when comparing the pair of approaches consisting of basic one-class classification and binary PHMM classification using a sample of the lowest scoring negatively labelled sequences. Again sampling with the highest scoring sequences is not competitive.

Figure 4.19 presents the results for *enzyme_3*. For this dataset all binary ap-



	binary no sampling	one-class uniform null model	binary uniform sampling	binary stratified sampling	binary sampling from the lowest scoring
one-class, uniform null model	**2.82E-45**				
binary, uniform sampling	0.790	**1.36E-42**			
binary, stratified sampling	**0.002**	**2.35E-32**	**7.56E-04**		
binary, sampling from the lowest scoring	**9.05E-06**	**9.88E-44**	**0.010**	0.504	
binary, sampling from the highest scoring	**8.05E-06**	**1.80E-22**	**3.51E-06**	**0.007**	**0.017**

uniform sampling statistically as good as a binary PHMM trained on a full dataset

Figure 4.19. *Comparison of sampling approaches for euk_3. Note that the y-axes are scaled differently. The figure follows the layout of Figure 4.14.*

proaches significantly outperform basic one-class PHMM classification in terms of AUC. However, all sampling approaches except uniform sampling perform statistically significantly worse than using all negative information in a binary PHMM. There is no statistically significant difference in AUC between standard binary PHMM classification and using simple uniform sampling. This approach to sample the negative class is also statistically significantly better than all other sampling approaches. However, the uniform sampling approach does not converge and is stopped after 100 iterations when the AUC has levelled out.

For figures presenting the results for all other datasets, refer to Appendix B.3.

Table 4.3 provides an overview of the results for the two most successful

sampling approaches by comparing the AUCs and their statistical significance for the binary PHMM classification without sampling to uniform and stratified sampling. There is no dataset for which stratified sampling significantly outper-

Table 4.3. *Comparison of AUCs for binary PHMM classification without sampling to uniform and stratified sampling. The table indicates statistical significance at the* 0.05 *level compared to binary PHMM classification without sampling.*

	binary PHMM		
dataset	without sampling	stratified sampling	uniform sampling
enzyme_1	0.835	0.823 (−)	0.829 (=)
enzyme_2	0.884	0.882 (=)	0.893 (+)
enzyme_3	0.783	0.768 (−)	0.764 (−)
enzyme_4	0.872	0.866 (=)	0.876 (=)
enzyme_5	0.837	0.843 (=)	0.834 (=)
enzyme_6	0.870	0.872 (=)	0.866 (=)
enzyme_7	0.847	0.836 (=)	0.844 (=)
enzyme_8	0.936	0.918 (=)	0.924 (=)
enzyme_9	0.962	0.961 (=)	0.961 (=)
enzyme_10	0.922	0.924 (=)	0.919 (=)
enzyme_11	0.955	0.955 (=)	0.954 (=)
enzyme_12	0.847	0.852 (=)	0.823 (−)
enzyme_13	0.896	0.889 (−)	0.886 (−)
enzyme_14	0.998	0.998 (=)	0.998 (=)
enzyme_15	0.989	0.986 (=)	0.985 (−)
enzyme_16	0.930	0.924 (−)	0.932 (=)
euk_0	0.831	0.822 (−)	0.828 (=)
euk_1	0.971	0.963 (−)	0.954 (−)
euk_2	0.872	0.850 (−)	0.868 (=)
euk_3	0.888	0.879 (−)	0.888 (=)
pro_1	0.931	0.931 (=)	0.920 (−)
pro_2	0.875	0.870 (=)	0.872 (=)

forms binary PHMM classification without sampling. However, it does perform equally well in 14 of 22 datasets and AUC is significantly decreased by the application of stratified sampling in eight datasets. Uniform sampling performs better overall. There are only six significant losses compared to binary PHMM classification without sampling. In one dataset, namely *enzyme_2*, uniform sampling even boosts AUC significantly. For 15 datasets both approaches perform without a significant difference. This clearly shows the power of uniform sampling in

terms of preserving predictive performance.

Runtime

The previous section showed that the simple uniform sampling strategy is highly competitive in terms of AUC compared to training the negative one-class PHMM on the full set of negatively labelled instances. This section investigates the runtimes of the proposed approaches. To achieve this, it compares again *enzyme_1*, the dataset from *enzymes* with the largest number of positively labelled sequences to *enzyme_8* which has the overall smallest positive class.

Figure 4.20 shows the runtime measured in seconds after training for one iteration of the Baum-Welch algorithm and subsequent evaluation of one 10-fold cross-validation. The different approaches differ in the number of PHMMs they

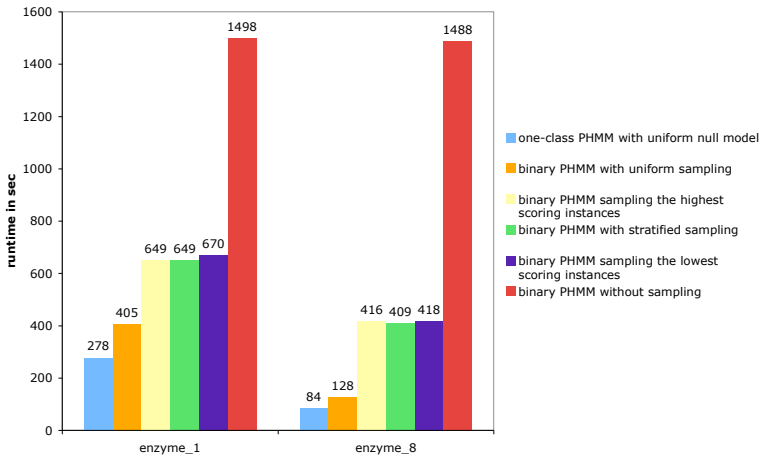


Figure 4.20. *Runtime in seconds for training and evaluating after the first iteration of Baum-Welch training for enzyme_1 and enzyme_8 for one-class PHMM classification and binary PHMM classification using the full negative training set and using the four proposed sampling approaches.*

train and in the training set. However, the test set is identical for all approaches. The results are as expected. Training a one-class PHMM on a small number of positive instances and evaluating on the test set leads to the fastest runtime. The binary PHMM trained on the full dataset requires the longest time by a large margin. Uniform sampling is the fastest sampling technique as it does not require scoring sequences. However, compared to pure one-class classification,

it requires the training of an additional PHMM. The three score-based sampling approaches need approximately the same runtime. This is not surprising as they score all negative sequences and then choose the same number of them to train on. They just employ a different metric for choosing the instances. Their runtime is considerably slower than uniform sampling. However, compared to binary PHMM classification without sampling, they are much faster.

Figure 4.21 displays the runtime results after Baum-Welch training is completed. The time is measured in hours and includes time for evaluating after each iteration of the Baum-Welch algorithm. The one-class PHMM is clearly

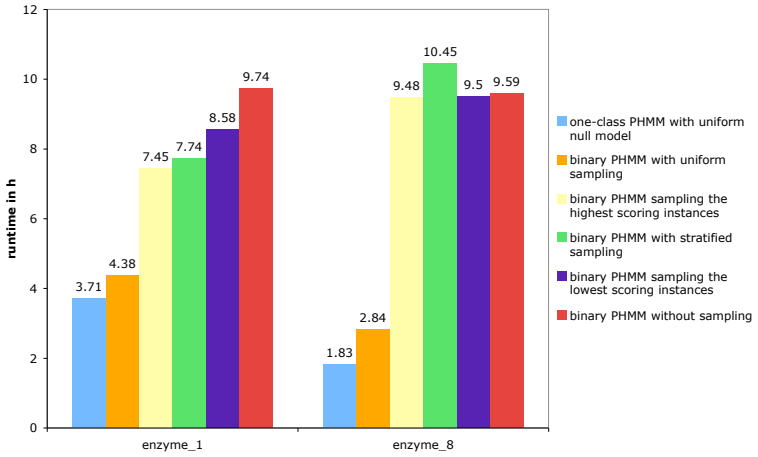


Figure 4.21. *Overall runtime for enzyme_1 and enzyme_8 in hours. The reported runtimes include evaluation of the classifiers after each training iteration for the one-class case and binary PHMM classification using the full negative training set and using the four proposed sampling approaches.*

the fastest, followed unsurprisingly by binary PHMM classification with uniform sampling. As the previous section revealed the competitive predictive power of the simple uniform sampling approach, this section confirms its favourable behaviour. Thus, binary PHMM classification with uniform sampling can be seen as an approach that optimises predictive power without a great loss in runtime compared to the standard one-class PHMM.

Looking at *enzyme_1*, the three score-based sampling approaches require considerably more runtime than uniform sampling. Sampling using the lowest scoring negative instances is more time-consuming than the other two sampling strategies. This is surprising when we look back at the results after one itera-

tion. Additionally, the gain in time made by score-based sampling compared to standard binary PHMM classification is smaller than expected and at most 2.3 hours. The situation is even worse for *enzyme_8*. In this case, sampling using the lowest or highest scoring sequences does not lead to a considerable improvement in runtime compared to training on all negative instances. Compared to the more than nine and a half hours runtime of the standard binary PHMM, the gain of around seven minutes from sampling is almost negligible. Additionally, stratified sampling requires more time than no sampling. To understand these runtimes, we have to look at the number of iterations needed to converge.

The runtime is influenced by both the number of sequences and the number of iterations necessary to converge. Figure 4.22 shows the overall number of iterations required by the Baum-Welch algorithm to converge. We see that for

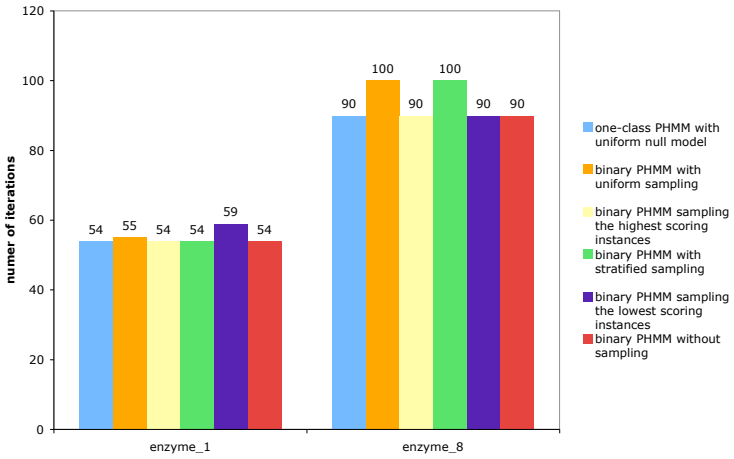


Figure 4.22. *Number of iterations in Baum-Welch training for enzyme_1 and enzyme_8.*

enzyme_1 the one-class PHMM requires 54 iterations to converge. Consequently, in all binary approaches the one-class PHMMs trained on the positively labelled sequences need the same number of iterations. The overall number of iterations in binary approaches can be higher than that, if the negative one-class PHMM converges at a later stage. This is the case when a sample of the lowest scoring instances is used. The approach needs 59 instead of 54 iterations. Thus, the runtime of sampling with the lowest scoring instances is higher than the one for stratified sampling or when a sample of the highest scoring instances is used. For *enzyme_1* and especially for *enzyme_8* uniform sampling requires

more iterations but because it is score-independent it does not influence the results negatively. However, to understand the runtimes for *enzyme_8*, we have to consider when the negative one-class PHMMs converge.

Figure 4.23 provides that overview. First of all, we see that in all sampling

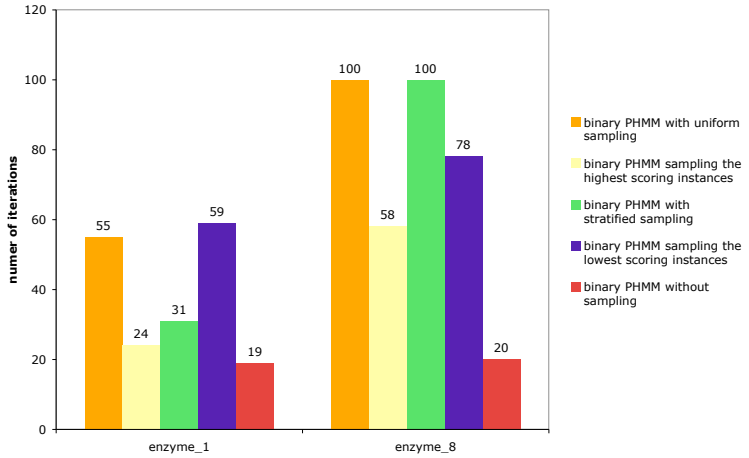


Figure 4.23. *Number of iterations of the negative one-class PHMM in binary PHMM classification approaches for enzyme_1 and enzyme_8*

approaches the negative one-class PHMM needs more iterations than training on the full negative set. However, the difference is much more pronounced for *enzyme_8*. This is important as asymptotically, the times to train, test or score a sequence are the same. If n is the length of the sequence and k is the number of states in the PHMM, then this time is $O(n \cdot k)$. But for training we need the forward and the backward algorithm and therefore, have to solve two dynamic programming problems each of size $O(n \cdot k)$ whereas for scoring we only need one of the above. As Figure 4.20 showed, if we only look at the first iteration, the difference between scoring and training is of practical relevance even though there is no difference asymptotically. However, the number of iterations that the negative one-class PHMM requires to converge plays a crucial role in score-based sampling.

Succinctly put, this section shows the advantage in runtime of uniform sampling compared to no sampling but also compared to score-based sampling approaches.

Sampling from all negative data

Sampling allows us to be able to deal with a vast amount of negative data. One idea is, therefore, to put all 23 datasets together and do binary classification on this dataset. This set has 6067 instances. Figure 4.24 gives a characteristic overview of the results using datasets *enzyme_4*, *enzyme_5*, *enzyme_7* and *enzyme_9*. The specific results for all other datasets can be found in Appendix B.4. In this setting we use a binary PHMM with uniform sampling of the new nega-

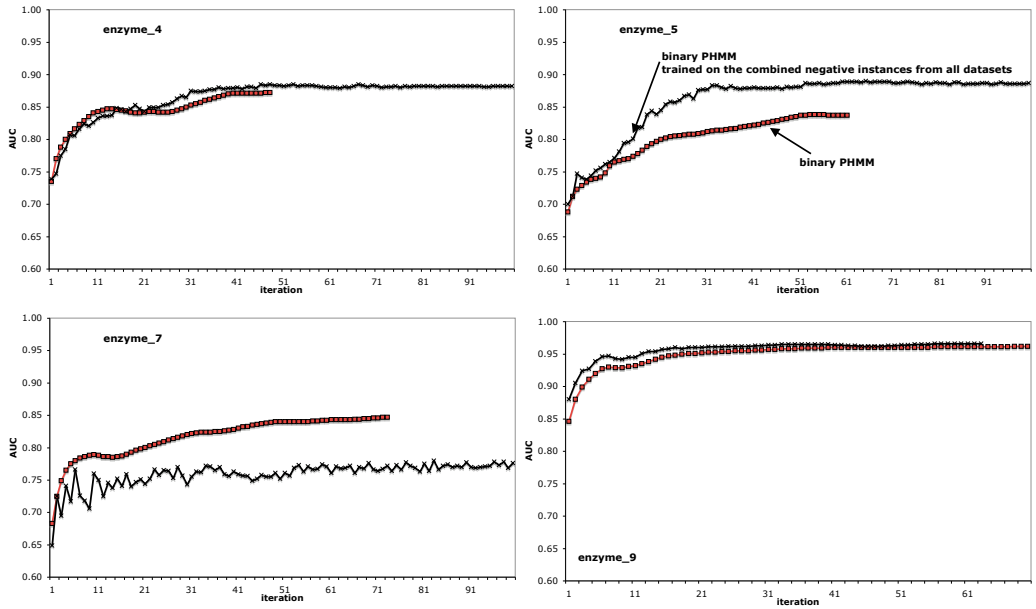


Figure 4.24. *AUC values for enzyme_4, enzyme_5, enzyme_7 and enzyme_9 for binary PHMM classification with no sampling and when taking all instances of the enzyme, pro and euk datasets as negative examples and subsequent use of uniform sampling.*

tive class. We do not report statistical significance as we compare two datasets of different size. Furthermore, as Figure 4.24 shows, the binary PHMM trained on the original dataset without sampling and the binary PHMM trained on all negative data with uniform sampling converge or stop their training at different iterations. First of all, the negative class is different. Additionally, as explained before, sampling is implemented in a way that the negative one-class PHMM trains on a different subset in each iteration of the Baum-Welch algorithm. This might hinder convergence and thus, training of the binary PHMM on all neg-

ative data with uniform sampling stops after convergence or after at most 100 iterations. The training of the binary PHMM without sampling stops after convergence.

The results for this new dataset are competitive in terms of AUC for *enzyme_4* and *enzyme_9*. It even outperforms the standard binary PHMM approach for *enzyme_5*. However, the situation is different when dealing with *enzyme_7*. In that case, the standard binary approach leads to higher AUC values. In general, uniformly sampling from all negative data leads to a competitive predictive performance in 13 of 23 datasets. Thus, the results also show that the question of whether adding more and potentially unrelated negative information can be helpful in establishing a decision boundary that is good enough to discriminate between remote members of the class and non-members is another non-trivial consideration.

This is an interesting setting for cases where there exists a positive class e.g. labelled proteins that belong to a class, but, where there are no explicitly defined negatives that are problem specific. As this result suggests, binary classification can improve the discriminative power of the model by using non problem-specific negative examples. Uniform sampling on this potentially huge set of negatives makes this approach practically feasible.

Comparison to previous approaches

The thesis' aim is to advance sequence-based classification with PHMMs. Consequently, the baseline classifier for all our experiments is the one-class PHMM trained on the positive instances only. This section, however, contrasts our PHMM based methods to previous work and thus, compares our results from one-class and binary PHMM classification to previous approaches, namely the work of Hua and Sun (2001), Guo et al. (2004) and Chou and Elrod (2003) that use the same datasets.

Sound comparisons are difficult to obtain as these approaches evaluate in different ways. Hua and Sun (2001) and Chou and Elrod (2003) evaluate with a jackknife test whereas Guo et al. (2004) uses a 5-fold cross-validation. All results in the thesis are obtained by means of a 10-fold cross-validation. Additionally, the previous approaches evaluate using accuracy and do not provide any information about statistical significance. Our work optimises AUC and cal-

culates statistical significance. Nonetheless, in the following, we will compare the approaches using accuracy calculated on different evaluation schemes. Additionally, the 16 datasets from *enzymes* are slightly different to the ones from the original dataset as the database they are retrieved from has changed as pointed out in Chapter 3.

Table 4.4 compares the accuracies for the *enzyme* datasets using a one-class PHMM and a binary PHMM trained on the full dataset with the results from Chou and Elrod (2003).

Table 4.4. *Comparison of accuracies for one-class PHMMs with uniform null model and the covariant discriminant algorithm (CDA) from Chou and Elrod (2003).*

dataset	CDA	one-class PHMM	binary PHMM
enzyme_1	0.48	0.64	**0.72**
enzyme_2	0.54	0.70	**0.88**
enzyme_3	0.43	0.44	**0.68**
enzyme_4	0.52	0.57	**0.88**
enzyme_5	0.45	0.68	**0.93**
enzyme_6	0.72	0.83	**0.85**
enzyme_7	0.47	0.65	**0.95**
enzyme_8	0.53	0.84	**0.96**
enzyme_9	0.85	0.89	**0.97**
enzyme_10	0.54	0.49	**0.95**
enzyme_11	0.79	0.76	**0.98**
enzyme_12	0.52	0.41	**0.86**
enzyme_13	0.75	0.68	**0.87**
enzyme_14	0.94	0.98	**0.99**
enzyme_15	0.70	0.86	**0.99**
enzyme_16	0.64	0.61	**0.89**

All datasets except *enzyme_14* are from the so-called twilight zone where sequence similarity is low. PHMMs are especially well equipped to detect remote similarities. Therefore, it is not surprising that the basic one-class PHMM with a uniform null model leads to a higher accuracy in 11 out of 16 times compared to the approach from Chou and Elrod (2003). The binary PHMM outperforms the one-class PHMM and CDA algorithm in all datasets.

The comparison also highlights a problem when evaluating with accuracy. A simple classifier that allways predicts the majority class achieve an accuracy of

0.88 for *enzyme_1*. Its AUC is 0.5. The accuracy of the CDA algorithm is 0.48 and the one of the binary PHMM is 0.72. Therefore, from an accuracy point of view, these classifiers appear to be inferior to the simple classifier that predicts the majority class. However, the AUC of the one-class PHMM is 0.82 and the binary PHMM achieves an AUC of 0.84. Succinctly put, looking at AUC reveals the algorithms true potential on these imbalanced datasets.

The situation in terms of accuracy is different for the *prokaryot* and *eukaryot* datasets as Table 4.5 shows. These two datasets have higher inside class sequence similarity than the *enzyme* datasets. The one-class PHMM with uniform null model performs worse compared to the approach from Guo et al. (2004) on all datasets. The accuracy of the one-class PHMM is slightly higher for *pro_1* compared to the accuracy achieved by Hua and Sun (2001). For *euk_2*, they achieve the same accuracy. However, using a binary PHMM increases accuracy drastically and leads to 4 wins and 3 losses compared to the best results from the two other approaches.

Table 4.5. *Comparison of accuracies for one-class PHMMs with uniform null model, binary PHMMs and the approaches of Hua and Sun (2001) and Guo et al. (2004).*

dataset	Hua and Sun	Guo et al.	one-class PHMM	binary PHMM
euk_0	0.77	**0.82**	0.48	0.73
euk_1	0.80	0.75	0.71	**0.95**
euk_2	0.57	0.61	0.57	**0.76**
euk_3	0.87	**0.89**	0.67	0.83
pro_0	0.98	**0.99**	0.64	0.92
pro_1	0.76	0.80	0.77	**0.90**
pro_2	0.79	0.79	0.62	**0.84**

4.6 Summary

In this chapter, we introduced basic one-class classification based on the observation that a one-class PHMM with a uniform null model is essentially a one-class classifier. We transformed one-class classification to binary classification in a simple way by adding a second one-class PHMM that is exclusively trained on the negative class. Typically, in protein classification, this class is much bigger

than the positive class. By training two one-class PHMMs separately this dataset imbalance does not influence the results. We empirically showed that binary models consistently achieve statistically significantly higher AUCs in almost all datasets after full convergence. However, because of the huge amount of negatively labelled sequences, the binary PHMM is slow in runtime. Therefore, we suggested sampling approaches.

We showed that sampling the negative class does not necessarily lead to a significant decrease in predictive power. To the contrary, the uniform sampling strategy leads to competitive results in 16 of 22 datasets in terms of AUC compared to using the entire negative information in a binary setting. It is worth pointing out that uniform sampling is a simple technique. It selects the negative instances at random without any prior scoring. Additionally, as opposed to score-based sampling, it guarantees that sampling improves runtime considerably.

In addition, a binary PHMM with sampling has the potential to successfully increase the discriminative power of one-class classification in the case where there are no specifically negatively labelled sequences.

In conclusion, a binary PHMM combined with score independent sampling leads to a very good ranking of test sequences based on AUC and the runtime overhead compared to a one-class PHMM is not excessive.

Chapter 5

Binary classification using propositionalisations of one-class models

A one-class PHMM represents a family or a class of proteins. The previous chapter investigated how to use and combine these models directly in a classification task. While the problem domain remains the same in this chapter, it adopts a different viewpoint on the one-class PHHMs. These models are no longer considered classifiers. In fact, this chapter takes a step back and considers a one-class PHMM as a graph and thus, as a form of structured data. In this way, a one-class PHMM is not a classifier but generates features for a classification process. This chapter introduces propositionalisation as a technique to generate these features, i.e. to make the information captured in a one-class PHMM accessible to standard Machine Learning classifiers. The main motivations for this approach are:

- To use discriminative techniques on top of the generative one-class PHMM
- To introduce negative information. The feature vector representation built from the one-class PHMM forms a binary dataset. Thus, all results presented in this chapter arise from binary classification.
- To allow the use of a wide range of classifiers on the data.

The basic assumption is that the presence of negative data through propositionalisation and the subsequent use of a discriminative classifier increases the ability to discriminate between the positive and the negative class.

Figure 5.1 shows how this approach fits into the overall framework of this thesis. This chapter and the previous chapter both use a generative one-class

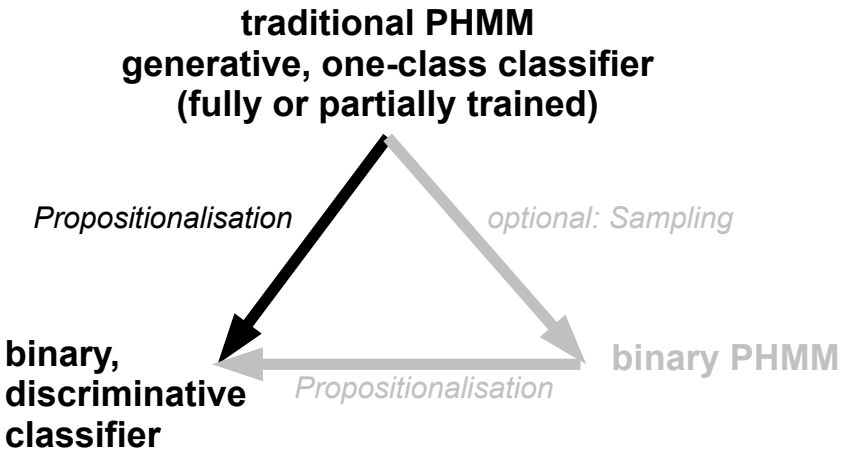


Figure 5.1. *The relevant part of the thesis' scheme*

PHMM as the basic model and both chapters present or will present the transformation to a binary classification problem to enhance predictive power. However, the proposed transformation is distinctly different. The previous chapter dealt with one-class PHMMs as generative classifiers. Here, we use them as a way to extract features from amino acid sequences. These features are either numeric or nominal and therefore, standard Machine Learning classifiers can be trained on them. We propositionalise on the basis of one-class PHMMs that are built using the positive instances only. The resulting datasets contain, however, positive and negative instances.

This chapter contributes to the thesis in three ways. First, it introduces new methods to propositionalise a one-class PHMM that result in more compact feature vectors than the previous approach by Jaakkola et al. (1999). The predictive power of our representations is competitive compared to one-class PHMMs.

Second, the new representations are tested on a range of different standard Machine Learning classifiers. This includes Support Vector Machines as Jaakkola et al. (1999) used but extends to Random Forests and bagged decision trees. As the experimental evaluation will show, propositionalisation in conjunction with Random Forests or bagged decision trees is competitive in terms of predictive power compared to the approaches from the previous chapter. The most intrigu-

ing finding that we will present in Section 5.2 is that our approach performs extremely well at the start of one-class PHMM training. In a nutshell our results suggest stopping the training of the one-class PHMM after the first or the first few iterations and use this one-class PHMM as input into the propositionalisation process. Not only does this procedure lead to a competitive AUC, it is also competitive in runtime, because training a one-class PHMM until convergence is slow.

Third, we improve on the results from Jaakkola et al. (1999) by normalising the propositional data before classifying. Their motivation to introduce propositionalisation is to switch from a generative to a discriminative learning problem. As they propositionalise positive and negative sequences, they implicitly also change the classification problem from a one-class to a binary one. This chapter explicitly sees propositionalisation as a tool to transform a one-class classification setting into a binary one as in Liao and Noble (2002).

Additionally, we show the flexibility of propositionalisation. The generated feature vectors can be combined to form new representations.

The following section will introduce different propositionalisations, whereas Section 5.2 will show experimental results. The final section will summarise our findings.

5.1 Propositionalisation

This section discusses methods to propositionalise a one-class PHMM. Figure 5.2 provides a graphical overview. It depicts the one-class PHMM as input into the feature generating process and shows an artificial propositional representation as output. The resulting dataset is binary. Note that all but one method that we propose are not limited to PHMMs. They will also work on HMMs in general.

In the following we will briefly highlight again the most important properties of the sufficient statistics of a PHMM and the Fisher score of a sequence. Jaakkola et al. (1999) use these scores as propositional representations. After that we will introduce our own methods.

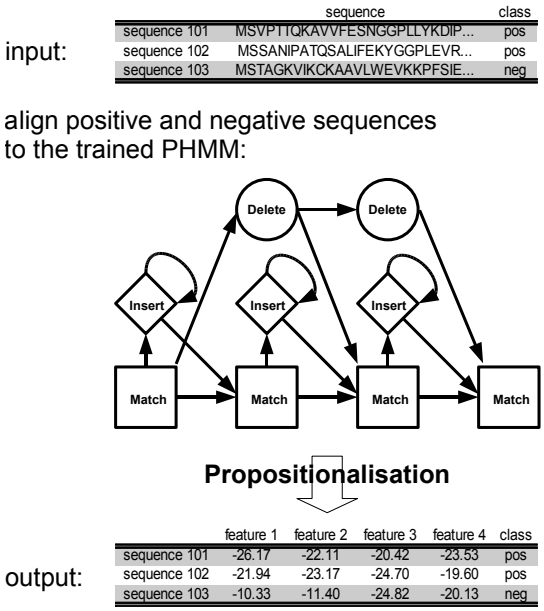


Figure 5.2. *The input and output of the propositionalisation process of a one-class PHMM.*

5.1.1 Fisher scores

One way to derive a propositional representation for a sequence uses a quantity that is analogous to the sufficient statistics of a HMM. The generated feature vector is the so-called Fisher score of the sequence. This method has been used by Jaakkola et al. (1999) on PHMMs and is explained in full in Section 2.4.4. We will summarise the main points here.

The sufficient statistics for a sequence tested on a HMM is a fixed length vector that contains an entry for each free parameter in the model. These are the posterior frequencies for all transitions and emissions. Thus, the sufficient statistics

- capture the extent to which each parameter is involved in generating the sequence from the HMM and therefore,
- provide a summary of the sequence in the parameter space of the model.

The sufficient statistics themselves are a propositional representation of a sequence. However, Jaakkola et al. (1999) go a step further and generalise the sufficient statistics to the so-called Fisher score. For a HMM H with parameters

θ and sequence X this score vector is defined as $U_x = \bigtriangledown_\theta logP(X|H,\theta)$. This fixed length vector representation therefore contains in each of its components the derivative of the log-likelihood score for sequence X with respect to a particular parameter. It can be derived from any HMM using standard forward and backward calculations. The magnitude of each component captures the extent to which a parameter contributes to generating the sequence X. This is the propositional representation we will compare our approaches to.

Like Jaakkola et al. (2000) we only use gradients with respect to the emission probabilities. It is important to note that because Baum-Welch training in this thesis does not update the emission probabilities in insert states, only the gradients with respect to the emissions in match states will be used. Those gradients derived from insert states are constant and do not add information to the propositional representation. In addition, the Fisher score vectors presented in this thesis are derived directly from the emission probabilities. Jaakkola et al. (2000) use a mixture decomposition that allows them to abstract from residue identity and takes affinities of amino acids into account. First of all, from their work it is ambiguous exactly which mixture distribution they used. In addition, our approaches work on the basis of residue identity and therefore, the comparison is fair.

Figure 5.3 shows a small example. The aim is to propositionalise the sequence 'WEKA' using a trained one-class PHMM. The calculation of the Fisher score is explained in Section 2.4.4. The figure shows the one-class PHMM that forms the basis for the propositionalisation process. There is no null model involved as the Fisher score works on the log-likelihood and not on the log-odds scores. Let l be the length of the sequence alphabet and k the number of columns in the one-class PHMM, then this technique creates $l \cdot k$ numeric attributes given that only emissions of match states are considered in the transformation process. In this example the Fisher score vector contains 12 non-class attributes. We derive two different propositional representations. The first is the Fisher score vector itself. The other normalises its components so that for all components i of the Fisher score vector U, $U_i \in [0,1]$. This is a small extension to Jaakkola et al.'s work. Experiments will show that the normalised Fisher score representation outperforms the original Fisher score. We use the normalisation procedure implemented as an option in WEKA's SMO algorithm.

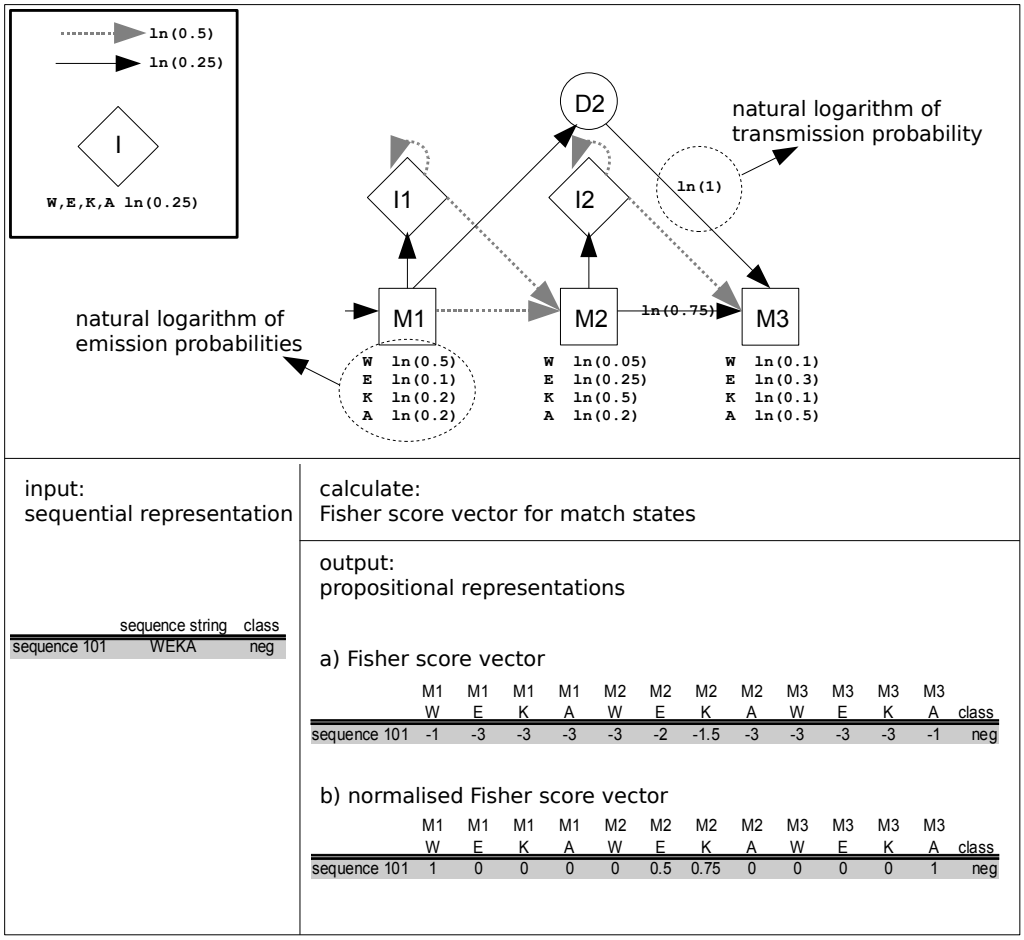


Figure 5.3. *Example on propositionalisation of the sequence 'WEKA' using the Fisher score method.*

Jaakkola et al. (1999) use their propositional representation as input to a Support Vector Machine with RBF kernel. The γ parameter of WEKA's RBF kernel is set, as proposed, to the median Euclidean distance between Fisher score vectors derived from positively labelled sequences and the closest Fisher score vector of the negative class. We follow their approach for the Fisher score and the normalised Fisher score representation.

5.1.2 Simple propositionalisations

This section presents simpler forms of propositionalisation. However, before doing that, we will define what the term *simple* means in this context.

Certainly, we understand a *simpler* process to be more intuitive and/or less complex. In this work, a propositionalisation A is defined as being *simpler* than a propositionalisation B if

- A's runtime grows asymptotically no faster than B's and/or
- A creates fewer attributes than B. Subsequent classification might be faster on a dataset with fewer attributes.

As we will explain in Section 5.1.4 the different propositionalisations presented in this chapter have the same asymptotic runtime. However, our propositionalisations create fewer attributes.

In the following, we will discuss three different ways to propositionalise. The first technique reduces a one-class PHMM to one numeric attribute. The second uses the best path through the one-class PHMM as a way to represent the model in a fixed length feature vector. The last approach creates one numeric attribute per state in the one-class PHMM.

Using path scores

The first propositionalisation reduces the feature space dramatically by mapping an amino acid sequence using a PHMM or a general HMM into two numeric attributes.

Figure 5.4 shows the propositional representations for the sequence string 'WEKA' as in the previous section. As the figure indicates, the two numeric attributes used in the logarithmic representation are the log-odds score of the sequence given the one-class PHMM and the log-odds score of the best or so called Viterbi path through the model. These log-odds scores depend on the null model as opposed to the Fisher score. Their calculation is explained in detail in Section 2.2.1. An additional propositional representation replaces each of the two logarithmic score values n_i with e^{n_i}. We refer to this representation as the exponential score as opposed to the logarithmic score. Both representations can be computed for any HMM.

These exponential scores are not normalised into real probabilities. The reason is a numeric issue. The overall log-odds ratio and the log-odds score of the best path can vary greatly. For example, let us consider the scores from Table 5.1. They show the propositional representation for the first instance of

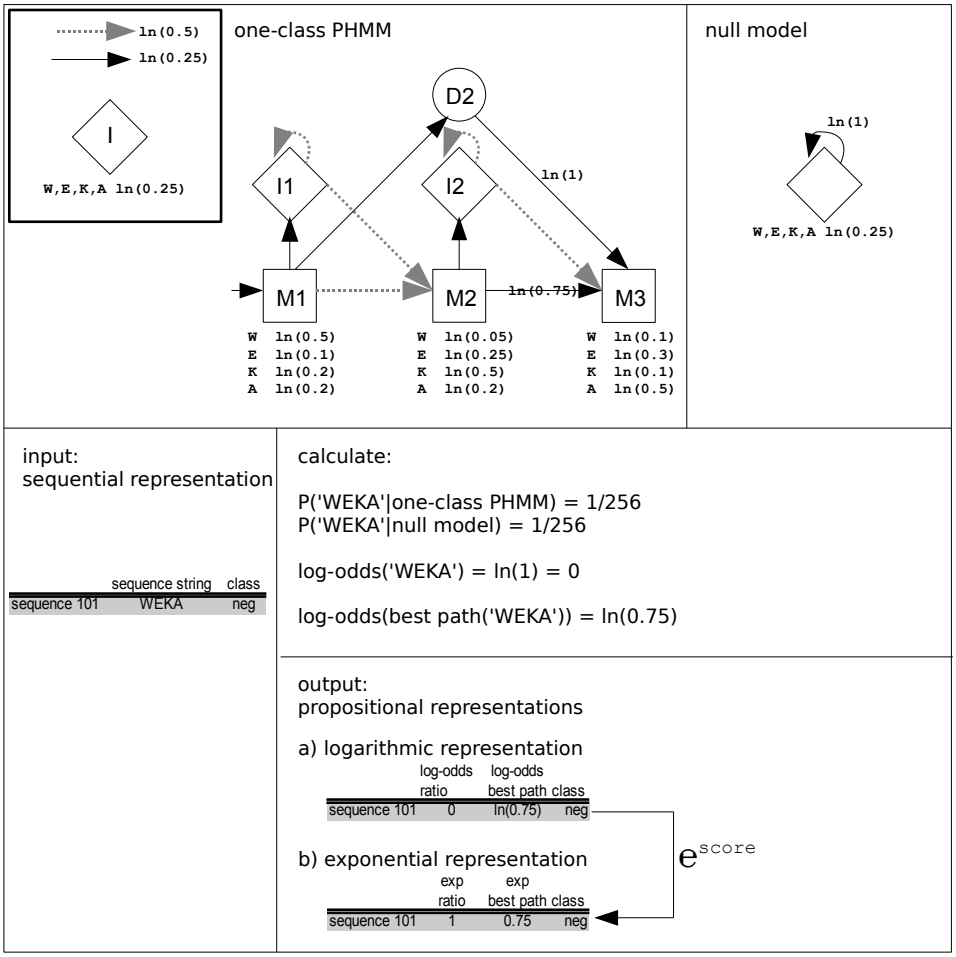


Figure 5.4. *Example propositionalisation of the sequence 'WEKA' using the log-odds ratio and the log-odds score for the best path of the sequence in a fully trained one-class PHMM.*

the *pro_2* dataset. Normalising these exponential values into real probabilities would lead to practical numeric problems as the probability for the best path would be evaluated to 0 and the overall probability would equal 1. This example is not an exception, it is extremely common for path scores. Therefore, normalising the exponential path scores is practically not an option.

Using the Viterbi path

The representations based on path scores depend on the null model. However, it is possible to construct a fixed length feature vector for a sequence X using

Table 5.1. *Propositional representation of the first instance from the pro_2 dataset. It shows the logarithmic and the exponential representation derived from the path scores. Values are rounded.*

representation	overall score	score for best path
logarithmic	9.9	-77.2
exponential	19206.6	2.9E-34

the Viterbi path, i.e the best path through the model independent of the null model.

For HMMs in general, the Viterbi path can be of arbitrary length. But propositionalisation requires a fixed length feature vector as output. A PHMM's structure is, however, made of a constant number of regularly built, re-occurring columns. These columns allow the definition of a fixed length feature vector.

We introduce our ideas using the example in Figure 5.5. Once again the sequence 'WEKA' is to be propositionalised. The strategy for creating a fixed length feature vector for a path of arbitrary length is to create only a constant number of attributes per column. We represent all but the last column with two attributes in the following way:

- One nominal attribute containing either the emitted amino acid symbol of the match state in the column or symbol for representing a delete state. In each column, the best path like all other paths passes either through the match or the delete state.

- One numeric attribute that counts how many times the insert state of the column has been visited. The default value is 0.

The last column is represented by one nominal attribute only. Its value is determined by the emission in the last match state.

In this way, a sequence is transformed into a fixed length feature vector. The length of the vector is $2k-1$ where k equals the number of columns. Therefore, in our example, the feature vector for the sequence 'WEKA' is $< W,1,K,0,A >$ representing the Viterbi path through M1, I1, M2 and M3.

This representation comes closest to a classical alignment representation as discussed in Chapter 2. A standard biological representation of the sequence

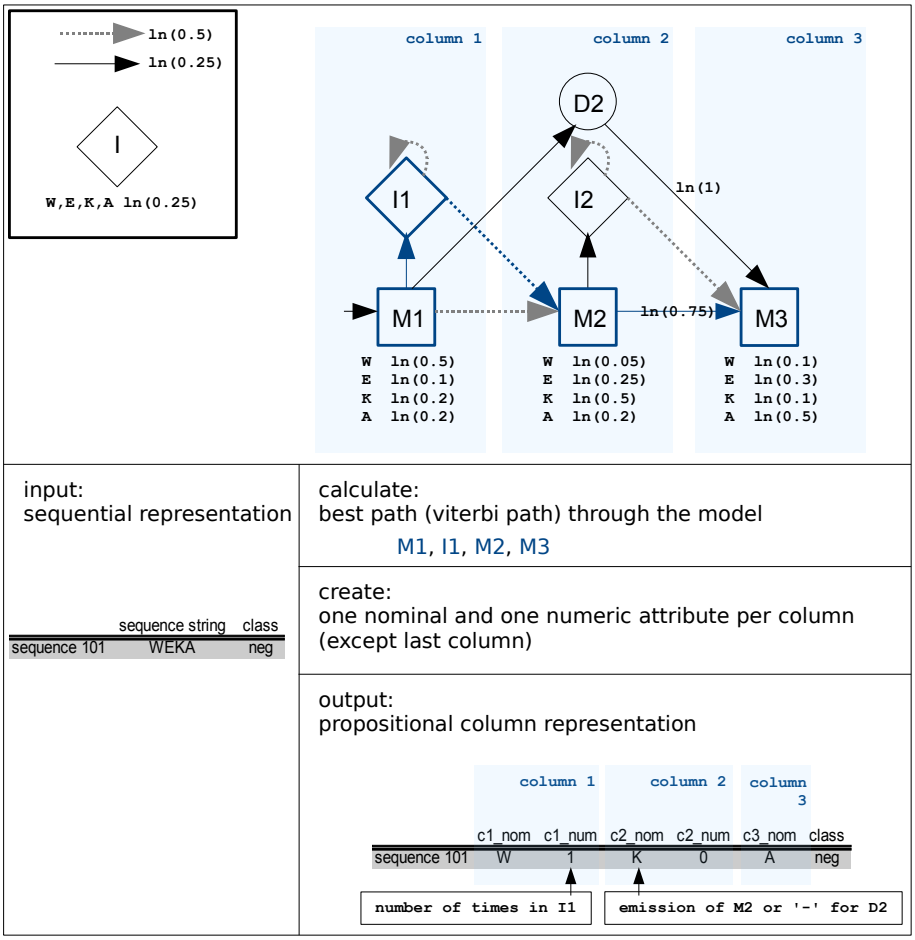


Figure 5.5. *Example propositionalisation of the sequence 'WEKA' by representing the best path with two attributes per column of the PHMM.*

'WEKA' aligned to the one-class PHMM is '`W e K A`'. A capital letter refers to an emission in a match state whereas a lowercase symbol indicates an insert.

Using one numeric attribute per state

The last representation that we introduce is similar to that one of Jaakkola et al. in that it is independent of the null model and works on PHMMs and on HMMs in general.

We calculate for each state the log of the posterior probability to be in the state when aligning sequence X to the model. Let $f_s(i)$ be the forward variable in state s at sequence position X_i as defined in Section 2.2.1. Analogously, let

$b_s(i)$ be the corresponding backward variable. In addition, the probability of the sequence X is $P(X)$ and can be calculated by the forward or backward algorithm as well. Durbin et al. (1998) define the posterior probability of being in state s when aligning sequence X as

$$P(s|X) = \sum_i \frac{f_s(i)b_s(i)}{P(X)}$$

These sums are independent of the null model and can be calculated for any HMM as they only use standard forward and backward calculations.

Figure 5.6 introduces the concept with our small 'WEKA' example. We first calculate the posterior probability of each state given the sequence 'WEKA' as shown. We align sequences globally and any alignment starts in the first match state and ends in the last one. Therefore, their corresponding posterior probability is always 1. This means they do not add any value towards the propositional representations and thus, they are not included. A feature vector containing the posterior probabilities has, therefore, $3k-5$ components where k is the number of columns. The minimum number of columns is 2. In our simple example there are 4 non-class attributes as M1 and M3 do not contribute useful information to the propositional dataset.

The first propositional representation uses the $\log P(s|x)$ for each state s except the first and last match states. We refer to this feature vector as a logarithmic states representation. Secondly, we use the probabilities but normalise them so that they sum to 1. This representation is referred to as a normalised probabilistic states representation. Figure 5.6 shows the corresponding feature vector for both representations.

All practically interesting propositionalisations using the posterior probability of each state are simpler than those created by Fisher scores because the feature vector of the former are smaller. Only if the alphabet size l equals 1 and there are more than 2 columns or l equals 2 and there are 5 are more columns, will the Fisher score vector be smaller. Practically, these cases are insignificant for protein classification as the alphabet size equals the number of amino acids. Thus we claim that all our propositionalisations are simpler than the one suggested by Jaakkola et al. as our methods lead to more compact feature vectors.

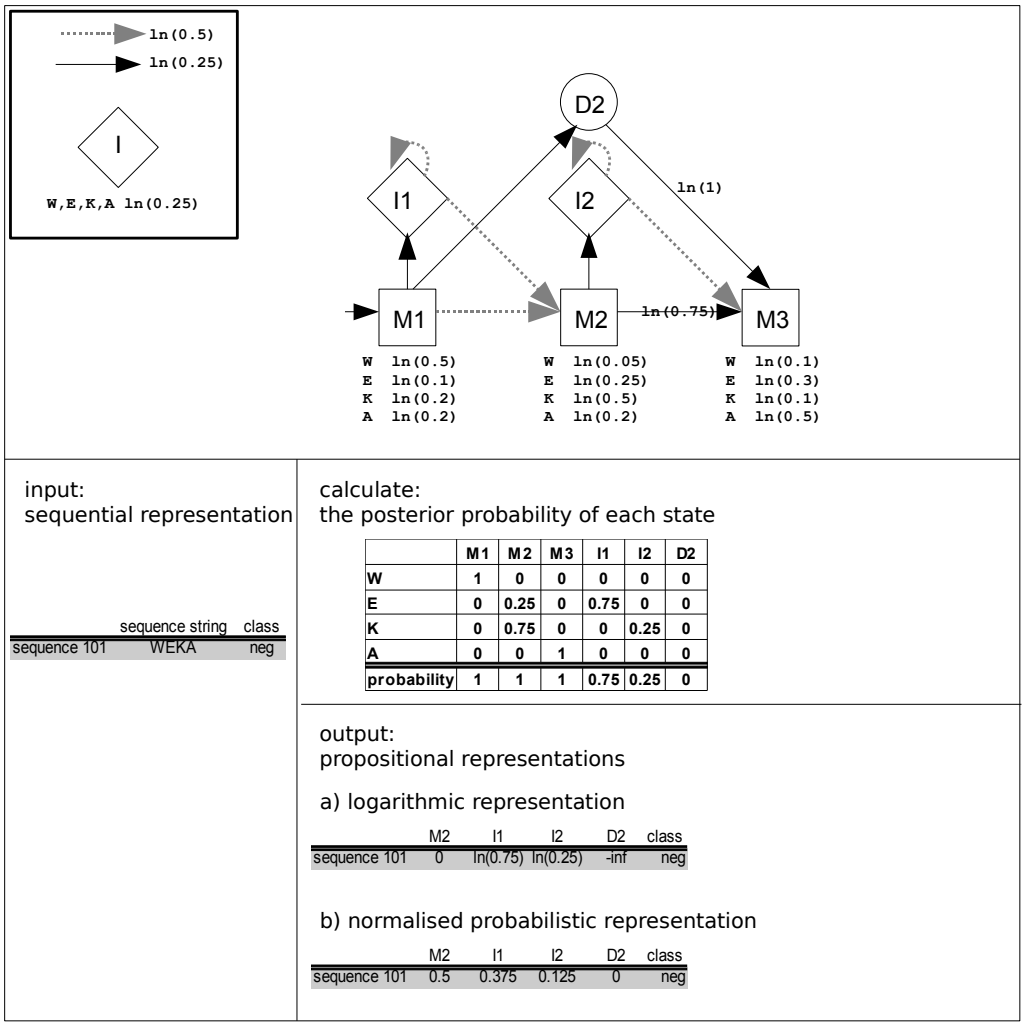


Figure 5.6. *Example propositionalisation of the sequence 'WEKA' using the posterior probability of each state given the sequence.*

Table 5.2 summarises the major features of the different strategies to propositionalise. They are ordered from complex to simple.

5.1.3 Binary classification and combining representations

The elegance of the propositional approach to protein classification using PHMMs as a form of structured data is manifest in three different ways.

First, the feature vector representation allows the use of any standard Machine Learning classifier and opens up the classification problem to a number of different optimisation strategies. In the experiment section we will show results for Support Vector Machines (SVMs) (Platt, 1998), Random Forests (Breiman, 2001) and bagged, unpruned C4.5 decision trees (Breiman, 1996). The hypothesis is that these classifiers can outperform the pure one-class PHMM classifier approach.

Second, even though the PHMM is built using only the positive instances, sequences from both classes are propositionalised using the one-class PHMM. Therefore, it introduces binary classification. As opposed to the one-class PHMM that is only trained on positive instances, propositional representations of training sequences from both classes are used on the standard Machine Learning classifiers. It is one of the thesis' fundamental assumptions that the introduction of negative information through propositionalisation increases predictive power.

Third, propositional representations can be combined to form new representations. As experiments will show, these are the most successful representations.

Table 5.2. *Summary of the properties of the different approaches to propositionalise. k is the number of columns in the one-class PHMM and l is the length of the sequence alphabet.*

representation	number of attributes	null model independent	applies to general HMMs
Fisher score vector	$k \cdot l$	yes	yes
posterior states probabilities	$3k-5$	yes	yes
Viterbi path	$2k-1$	yes	no
scores	2	no	yes

Two of them are especially worth mentioning. The first combines the logarithmic posterior states probabilities with the logarithmic path scores. We will refer to this representation as the combined logarithmic representation. The other joins the feature vectors from the normalised posterior states probabilities with the feature vector of the exponential path scores. For the rest of the thesis, this representation will be called the combined exponential representation.

Combining representations can increase predictive power. On the other hand the size of the feature vector will increase. However, this increase in size is small for the two combined representations. Both contain $3k - 3$ non-class attributes where k equals the number of columns of the one-class PHMM. Therefore they are still simpler than the Fisher score vector representation for the protein classification task.

The concept of combining representations is not only important to boost AUC in this chapter. It also builds the basic infrastructure for the approach that will be introduced in the next chapter.

5.1.4 Runtime

As mentioned earlier all propositionalisations have the same asymptotic runtime even though they create a varying number of attributes.

The Fisher score method and the approach using the posterior probability of each state both require the computation of the forward and backward matrices. Let n be the length of the sequence X and k be the number of states in the one-class PHMM, then the runtime of both propositionalisations is in $O(n \cdot k)$. We have to solve two dynamic programming problems of size $n \cdot k$.

In order to calculate the representation based on path scores we need the forward and Viterbi algorithm. The runtime for each is in $O(n \cdot k)$.

The Viterbi path based propositionalisation requires solving one dynamic programming problem, instead of two as in the aforementioned propositionalisations. However, the asymptotic runtime is still in $O(n \cdot k)$.

It is noteworthy that for a PHMM it is asymptotically as expensive to train one sequence in one iteration of Baum-Welch training as it is to test the sequence or to propositionalise it. However, testing and the Viterbi path propositionalisation only require the calculation of one dynamic programming matrix, whereas training and all other propositionalisations need two matrices to be calculated.

5.2 Experimental results

The experimental setup and datasets is as described in Chapter 3. All propositional classifiers are compared to one-class PHMMs. The next chapter will compare the propositional results to those from binary PHMMs. Models are either evaluated after each iteration of the Baum-Welch training algorithm or after its first and last iteration only due to the slow runtime of PHMM based approaches.

Figure 5.7 introduces the notation used to compare the performance of the propositional approaches to standard one-class PHMM classification. This chap-

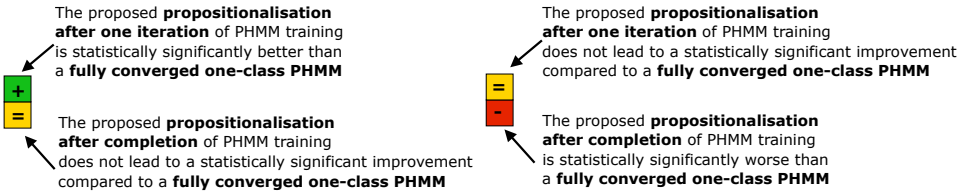


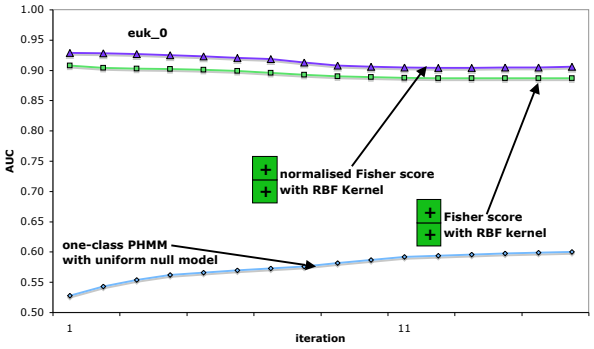
Figure 5.7. *Notation to compare statistical significance.*

ter tests for statistical significance at two distinct points during training. First, a one-class PHMM is trained for one iteration and subsequent propositionalisation allows binary classification with a standard Machine Learner. The AUCs of these experiments are compared to the AUCs of standard one-class PHMMs after the final iteration of its Baum-Welch training algorithm as only sufficiently trained PHMMs are competitive classifiers. The result of these statistical comparisons are reported in the upper square of the notation from Figure 5.7. The lower square displays similar information, but propositionalisation is performed after the PHMM has converged. A '+' sign indicates that the propositional approach under consideration leads to a significantly higher AUC than the fully trained one-class PHMM classifier. Consequently, a '−' sign represents the opposite situation. When there is no statistically significant difference in AUC, the '=' sign will be displayed in the corresponding square.

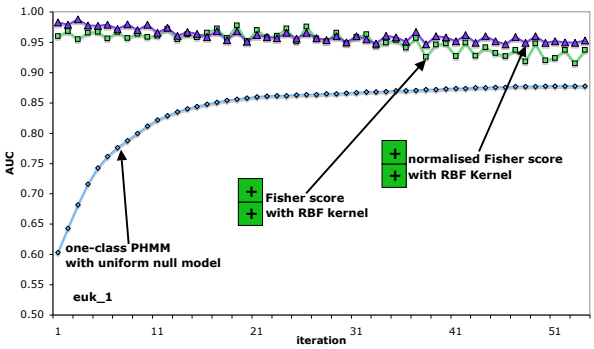
5.2.1 Predictive power

Fisher score vectors with Support Vector Machines

The presentation of results starts with the Fisher score based representations in combination with a Support Vector Machine and an RBF kernel. Figure 5.8 introduces the results for *euk_0* and *euk_1*. It compares one-class PHMM-based



	first Iteration	last Iteration	first Iteration
last iteration	**3.49E-07**		
first iteration	**8.49E-06**	0.682	
last iteration	**4.36E-12**	**1.61E-04**	**6.00E-05**



	first Iteration	last Iteration	first Iteration
last iteration	**4.37E-05**		
first iteration	**4.56E-06**	0.279	
last iteration	**1.25E-06**	0.059	**0.017**

Figure 5.8. *AUC values for euk_0 and euk_1 in one-class PHMM classification and propositional classification using Fisher score vectors. The AUC is calculated in each iteration of the Baum-Welch training algorithm. The table on the right contains the p-values. P-values printed in bold indicate a statistically significant difference in AUC at the* 0.05 *level. The meaning of the squares in the left-hand graphs is defined in Figure 5.7.*

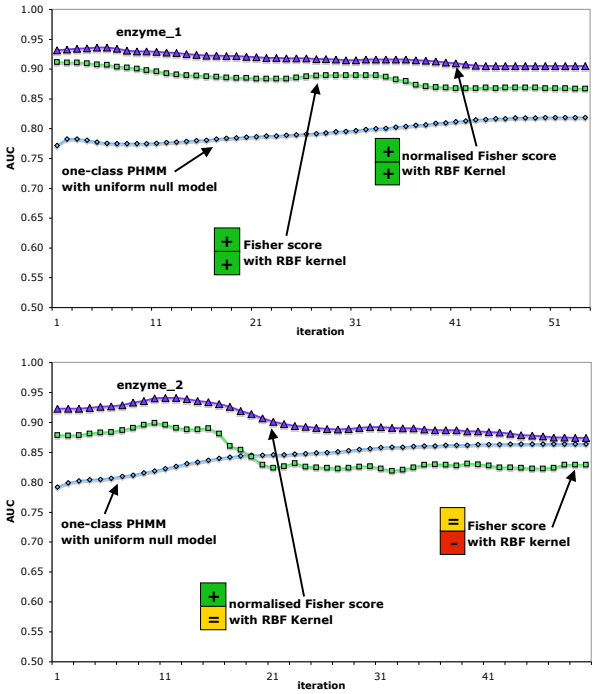
classification to propositional learning using Fisher score vectors directly and after a normalisation step. As the graphs show for both datasets, the propositional methods consistently outperform the one-class PHMM classification. For both datasets, the highest AUCs are achieved by stopping PHMM training after one iteration and subsequently using normalised Fisher scores with an RBF kernel. These AUCs and the slightly but significantly lower AUCs from Fisher score

vectors without an additional normalisation step significantly outperform the fully converged one-class PHMM. In addition, both methods also significantly outperform the one-class approach when propositionalisation is applied after PHMM training is completed. However, the predictive power of these propositional classifiers is significantly smaller than the ones built from the first PHMM iteration. For *euk_1* the difference in AUC between Fisher score vectors and their normalised version both built from a fully converged PHMM is not significant. Only two other datasets, namely *euk_2* and *pro_2* observe the same behaviour.

For all other 20 datasets, the normalised Fisher score vectors improve predictive power significantly when created from a fully trained one-class PHMM compared to the ones which omit the normalisation step.

For most datasets, normalisation of Fisher score vectors improve their performance considerably throughout the entire training process as Figure 5.9 shows for *enzyme_1* and *enzyme_2*. For *enzyme_1* both forms of propositional representation consistently lead to higher AUCs than one-class PHMM classification. The difference in AUC when propositionalised after the last, or even after only one, iteration of the Baum-Welch algorithm is significant. Additionally, the predictive power from propositionalisation built from the first iteration is considerably higher than the one from propositionalising a fully converged one-class PHMM. When comparing the propositional approaches with each other, the figure reveals that for both *enzyme_1* and *enzyme_2* normalisation leads to a statistically significantly better discrimination after the first and the last iteration of PHMM training. For *enzyme_2*, propositionalising after the final iteration without an additional normalisation step decreases AUC significantly compared to one-class PHMM classification. Normalisation, on the other hand, performs slightly but not statistically significantly better than the standard one-class approach. When looking at propositionalising after the first iteration instead of the last for *enzyme_2*, the normalised Fisher score significantly outperforms one-class classification. Omitting normalisation also increases AUC considerably. However, the difference is not significant compared to a fully trained one-class PHMM and it is significantly worse compared to the normalised representation.

Figure 5.10 depicts a similar situation for *enzyme_9*. Again, the normalised Fisher score representation is statistically significantly better than its unnormalised counterpart after the first and the final iteration. In this case, the



	first Iteration	last Iteration	first Iteration
last iteration	**0.005**		
first iteration	**0.012**	0.520	
last iteration	**6.42E-07**	**4.46E-04**	**2.57E-04**

	first Iteration	last Iteration	first Iteration
last iteration	**5.40E-04**		
first iteration	**1.00E-05**	0.783	
last iteration	**3.49E-09**	**0.003**	**0.003**

Figure 5.9. *AUC values for enzyme_1 and enzyme_2 in one-class PHMM classification and propositional classification using Fisher score vectors. The figure follows the layout described in Figure 5.8.*

normalised Fisher score already significantly outperforms the final AUC of the one-class approach after the first iteration and after the last iteration. When no normalisation is performed then propositionalisation after the first iteration boosts predictive performance significantly compared to a fully trained one-class PHMM. However, propositionalisation after the final iteration decreases AUC in a significant way. Other datasets with a similar result are *enzyme_7* and *enzyme_15*. Their graphs are shown in Appendix C.1.

The situation for *enzyme_10* is the same except that propositionalisation without normalisation after the final training iteration still decreases the AUC but not significantly compared to pure one-class classification. A similar behaviour can be found for *pro_1* in Appendix C.1.

Figure 5.11 compares the results for *enzyme_5* and *enzyme_6*. These are the first two datasets so far for which there is no significant advantage of normalisation of the Fisher score vector gained from a one-class PHMM after one iteration. The AUC is higher after normalisation, however, differences are not statistically

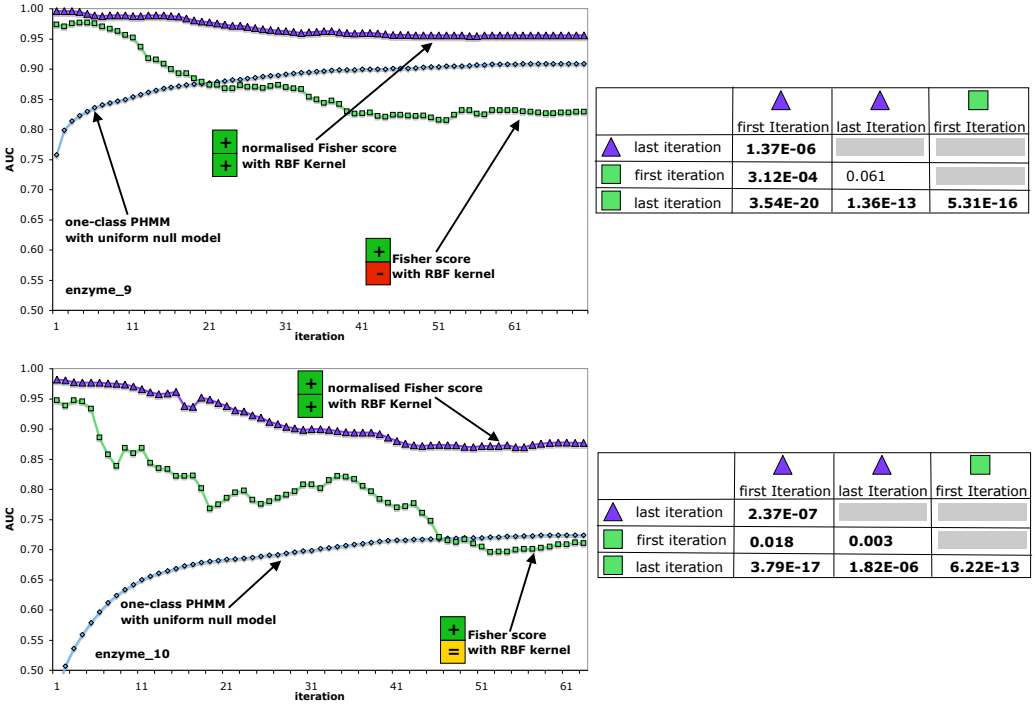
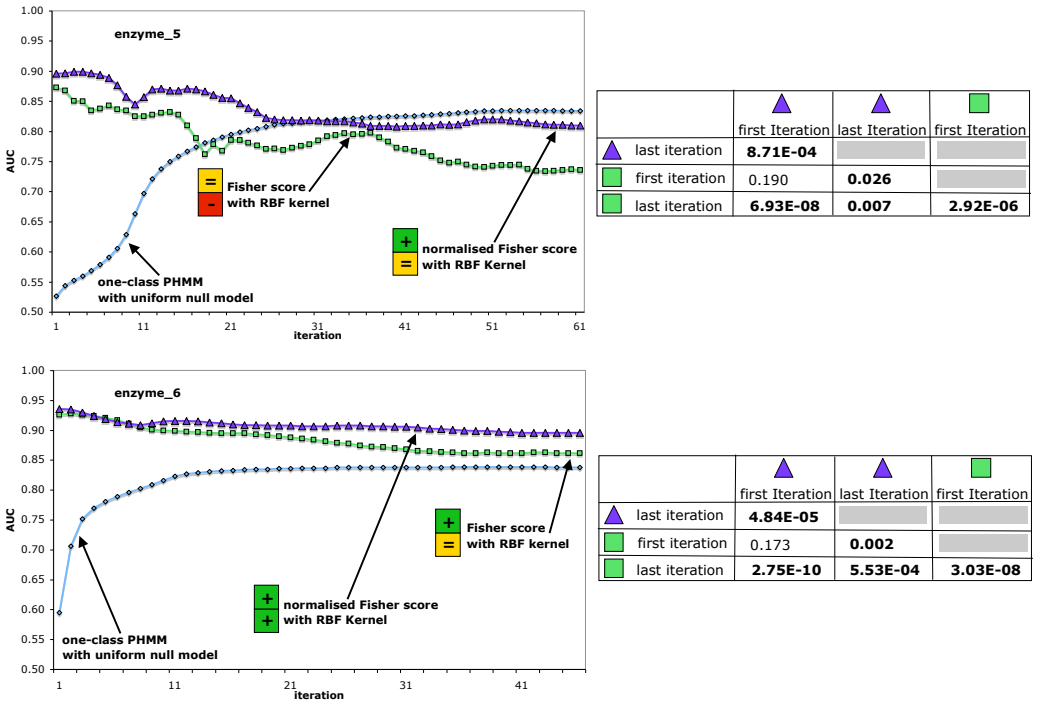


Figure 5.10. *AUC values for enzyme_9 and enzyme_10 in one-class PHMM classification and propositional classification using Fisher score vectors. The figure follows the layout described in Figure 5.8.*

significant. The same holds for only four other datasets. These are *enzyme_11*, *enzyme_13*, *enzyme_14* and *enzyme_16*. Again their results can be found in Appendix C.1. For all other 17 datasets normalisation of Fisher score vectors constructed after one iteration of one-class PHMM training lead to significantly higher AUCs than their unnormalised versions.

For *enzyme_6* both propositionalisations outperform one-class classification throughout the entire training process. However, when the Fisher score is not normalised the difference is only significant for propositionalisation after one iteration compared to a fully converged one-class PHMM. This also holds for *enzyme_16*.

The results are different for *enzyme_5*. Normalised Fisher scores gained after the first iteration of PHMM training significantly increase AUC compared to one-class classification with a fully trained PHMM. However, when propositionalised after the last iteration the AUC is decreased slightly but not signif-



	first Iteration	last Iteration	first Iteration
last iteration	**8.71E-04**		
first iteration	0.190	**0.026**	
last iteration	**6.93E-08**	**0.007**	**2.92E-06**

	first Iteration	last Iteration	first Iteration
last iteration	**4.84E-05**		
first iteration	0.173	**0.002**	
last iteration	**2.75E-10**	**5.53E-04**	**3.03E-08**

Figure 5.11. *AUC values for enzyme_5 and enzyme_6 in one-class PHMM classification and propositional classification using Fisher score vectors. The figure follows the layout described in Figure 5.8.*

icantly. Omitting normalisation leads to a significant decrease in AUC when the PHMM is fully trained before propositionalisation is performed. When the PHMM is propositionalised after the first iteration without normalising the resulting Fisher score vector, the AUC is improved compared to a fully converged one-class PHMM but the improvement is not significant. We observe the same pattern for *enzyme_8*, *enzyme_11* and *enzyme_14*. However, *enzyme_14* is the only dataset where the normalised Fisher score vector constructed on the basis of a one-class PHMM after one iteration leads to an increase in AUC that is not significant compared to a fully converged one-class PHMM.

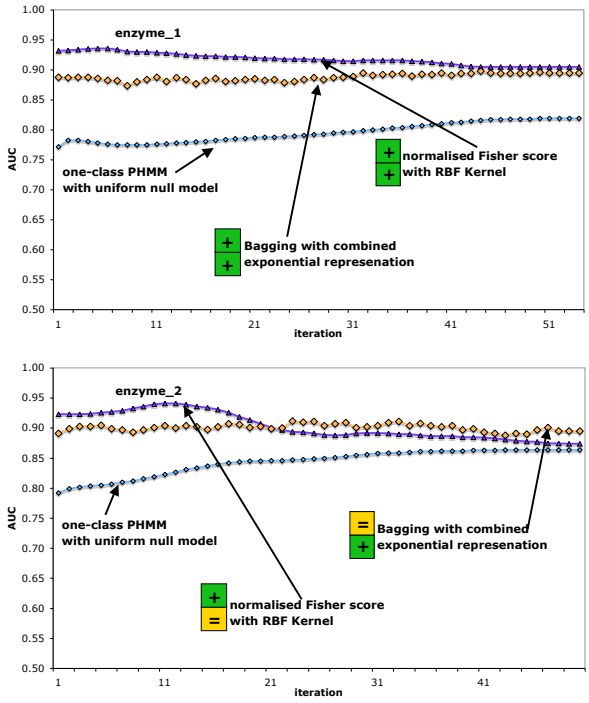
Appendix C.1 contains the results for all remaining datasets.

Bagging

The following three figures introduce results for bagging a combined exponential representation. It will be followed by a summary of the experiments pre-

sented to that point.

Figure 5.12 displays the results for *enzyme_1* and *enzyme_2* comparing the bagging approach with the normalised Fisher scores and one-class PHMM based classification. As in the case when normalised Fisher scores are used, bagging



	first Iteration	last Iteration	first Iteration
last iteration	**0.005**		
first iteration	**7.00E-06**	0.159	
last iteration	**2.30E-04**	0.385	0.495

	first Iteration	last Iteration	first Iteration
last iteration	**5.40E-04**		
first iteration	0.050	0.351	
last iteration	0.050	0.139	0.751

Figure 5.12. *AUC values for enzyme_1 and enzyme_2 in one-class PHMM classification and bagging with combined exponential representation. The AUC is calculated in each iteration of the Baum-Welch training algorithm. The table on the right contains the p-values. P-values printed in bold indicate a statistically significant difference in AUC at the* 0.05 *level. The meaning of the squares in the left-hand graphs is defined in Figure 5.7.*

for *enzyme_1* leads to a consistent improvement in terms of AUC compared to the one-class approach. The improvement is, however, smaller than the one obtained from normalised Fisher score vectors. The gap in terms of AUC between the two approaches is significant after the first iteration, whereas, when propositionalised after the final iteration, the difference in predictive power is not significant anymore. Compared to basic one-class classification, bagging based on a combined exponential representation built after just one iteration of PHMM training significantly outperforms a fully converged PHMM. The same

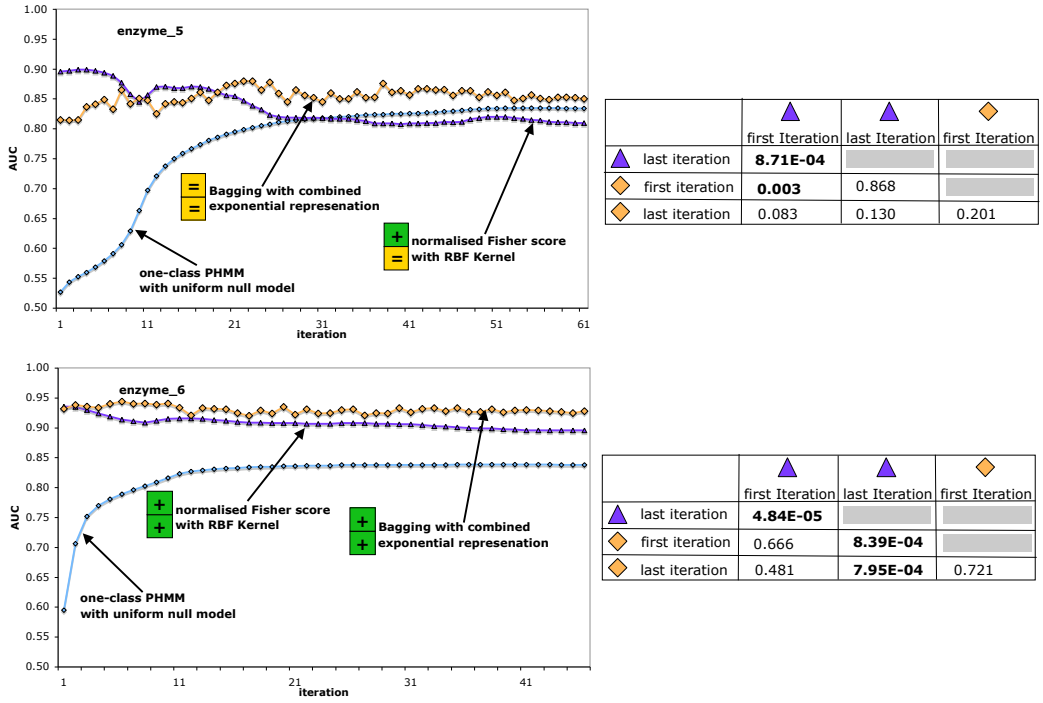
holds when the propositional learner is used after Baum-Welch training is completed. As opposed to normalised Fisher scores, the AUC from bagging after the final iteration (0.895) is slightly but not significantly higher than after bagging with a representation built on the basis of the first iteration (0.888).

For *enzyme_2* both propositional approaches perform with no statistically significant difference after the first and the last iteration even though the normalised Fisher score achieves a higher AUC after the first iteration and a lower one after the last. There are only three other datasets (*enzyme_4*, *enzyme_6* and *enzyme_14*) where there is no statistically significant difference in AUC after stopping a one-class PHMM after one iteration and subsequent propositionalisation.

For *enzyme_2* and six other datasets there is a slight but significant advantage in predictive power when bagging a combined exponential representation built after the last training iteration compared to normalised Fisher score vectors built from the same iteration. Bagging is significantly better than one-class classification with a fully converged PHMM when propositionalising after the last iteration. It improves AUC marginally when the first iteration builds the basis for the propositionalisation. For normalised Fisher score vector the situation is the opposite. This holds also for *pro_2*.

Figure 5.13 presents the results for *enzyme_5* and *enzyme_6*. For *enzyme_5*, compared to one-class classification with a fully trained PHMM, bagging increases predictive power. However, neither after the first, nor the last iteration is the difference in AUC statistically significant. Normalised Fisher score vectors calculated on the basis of a one-class PHMM after one training iteration lead to a significantly higher boost in AUC than bagging the combined exponential representation built from the same PHMM. Additionally, this increase in predictive power is also significant compared to a fully converged one-class PHMM. However, when propositionalised after the final iteration, both representations show no significant difference in AUC even though, in this situation, bagging outperforms all other approaches presented in this figure and the normalised Fisher score actually leads to a slight decrease in AUC.

The situation is different for *enzyme_6*. Bagging performs better than the normalised Fisher score vector with RBF kernel. After the final iteration the difference is significant. The only other dataset, where bagging with a com-



	first Iteration	last Iteration	first Iteration
last iteration	**8.71E-04**		
first iteration	**0.003**	0.868	
last iteration	0.083	0.130	0.201

	first Iteration	last Iteration	first Iteration
last iteration	**4.84E-05**		
first iteration	0.666	**8.39E-04**	
last iteration	0.481	**7.95E-04**	0.721

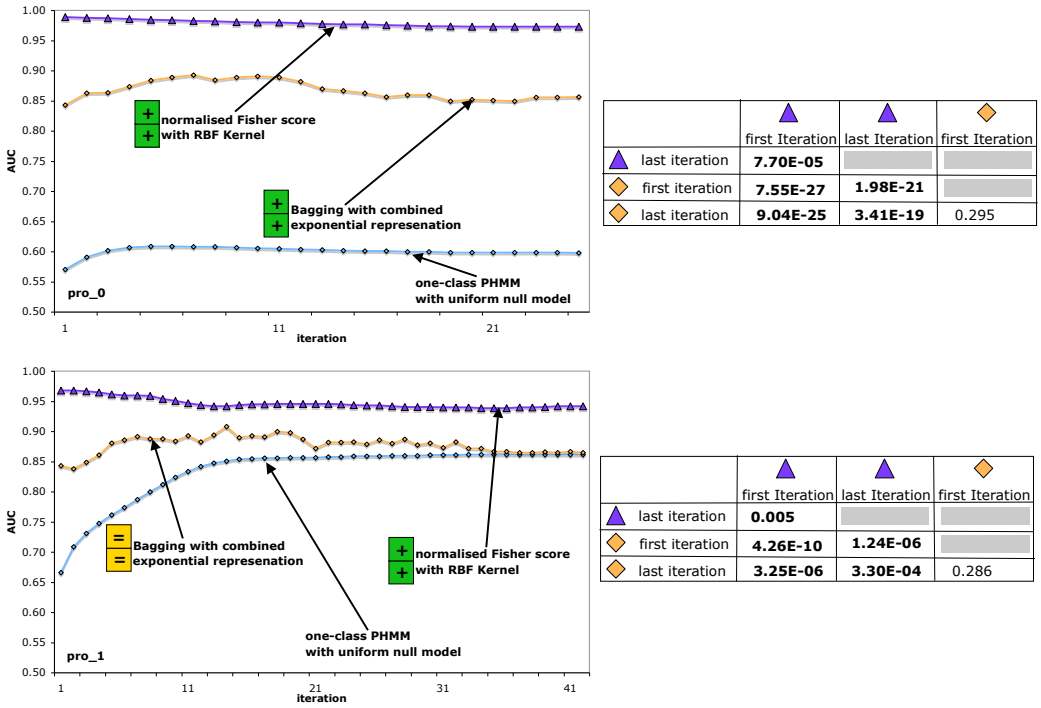
Figure 5.13. *AUC values for enzyme_5 and enzyme_6 in one-class PHMM classification and bagging with combined exponential representation following the layout defined in Figure 5.12.*

bined exponential representation built from a fully converged one-class PHMM is significantly better than normalised Fisher score vectors, is *enzyme_9*. Both representations perform better than standard one-class classification. The difference is of significance after the first and the last iteration.

The results for *pro_0* and *pro_1* are shown in Figure 5.14. For both datasets the difference in AUC between bagging and a Support Vector Machine classification based on normalised Fisher scores is larger than in the other datasets presented so far. These differences are statistically significant. For *pro_1* both propositional learners increase the predictive power significantly when propositionalising after the first and last iteration is compared to a fully trained one-class PHMM. The situation is different for *pro_1*. Bagging improves on AUC but the improvement is not significant. A similar situation holds for *enzyme_15*.

Appendix C.2 shows the results for all other datasets.

Table 5.3 summarises the results so far and provides an overview of statisti-



	first Iteration	last Iteration	first Iteration
last iteration	**7.70E-05**		
first iteration	**7.55E-27**	**1.98E-21**	
last iteration	**9.04E-25**	**3.41E-19**	0.295

	first Iteration	last Iteration	first Iteration
last iteration	**0.005**		
first iteration	**4.26E-10**	**1.24E-06**	
last iteration	**3.25E-06**	**3.30E-04**	0.286

Figure 5.14. *AUC values for pro_0 and pro_1 in one-class PHMM classification and bagging with combined exponential representation following the layout defined in Figure 5.12.*

cally significant differences in AUC compared to a one-class PHMM after training is fully completed. When Fisher scores are used for propositionalisation after one iteration of PHMM training, this leads more often to a statistically significant improvement in AUC compared to a fully converged one-class PHMM than propositionalising after the last iteration. This is indeed a very promising result as training a PHMM for just one iteration is fast compared to fully training a PHMM. It allows improving on AUC in a way that is sensitive to runtime. For bagging, we observe a statistical significant improvement after one iteration for 15 datasets. Contrary to the Fisher score situation however, there is a significant improvement for 16 datasets after the last iteration. However, the table also gives an indication of the superiority of a Support Vector Machine based approach with RBF kernel and normalised Fisher score vectors as input representation.

Table 5.3. *Summary of statistically significant differences of three propositional approaches after the first and last iteration compared to a fully converged one-class PHMM as baseline classifier. The propositional approaches are Support Vector Machine based classification with normalised and unnormalised Fisher scores and bagging with a combined exponential representation. A '+' sign indicates a significant advantage of the propositional methods, a '−' sign a significant disadvantage. If there is no statistically significant difference, a '=' sign is used.*

dataset	normalised Fisher score		Fisher score		Bagging	
	iteration		iteration		iteration	
	first	last	**first**	last	**first**	last
enzyme_1	+	+	+	+	+	+
enzyme_2	+	=	=	-	=	+
enzyme_3	+	+	+	+	+	+
enzyme_4	+	+	+	+	+	+
enzyme_5	+	=	=	-	=	=
enzyme_6	+	+	+	=	+	+
enzyme_7	+	=	=	-	+	=
enzyme_8	+	=	=	-	=	=
enzyme_9	+	+	+	-	+	+
enzyme_10	+	+	+	=	+	+
enzyme_11	+	=	+	-	=	=
enzyme_12	+	+	+	+	+	+
enzyme_13	+	+	+	+	+	+
enzyme_14	=	=	=	-	=	=
enzyme_15	+	+	+	-	=	=
enzyme_16	+	+	+	=	+	+
euk_0	+	+	+	+	+	+
euk_1	+	+	+	+	+	+
euk_2	+	+	+	+	+	+
euk_3	+	+	+	+	+	+
pro_0	+	+	+	+	+	+
pro_1	+	+	+	=	=	=
pro_2	+	=	+	=	=	+

Bagging in combination with the combined exponential representation as presented earlier is the most successful bagging approach. Using the combined logarithmic representation is slightly worse. Other representations can be used as well. However, they do not consistently lead to competitive AUCs as Figure 5.15 shows for *enzyme_8* and *pro_2*. The figure compares the result to

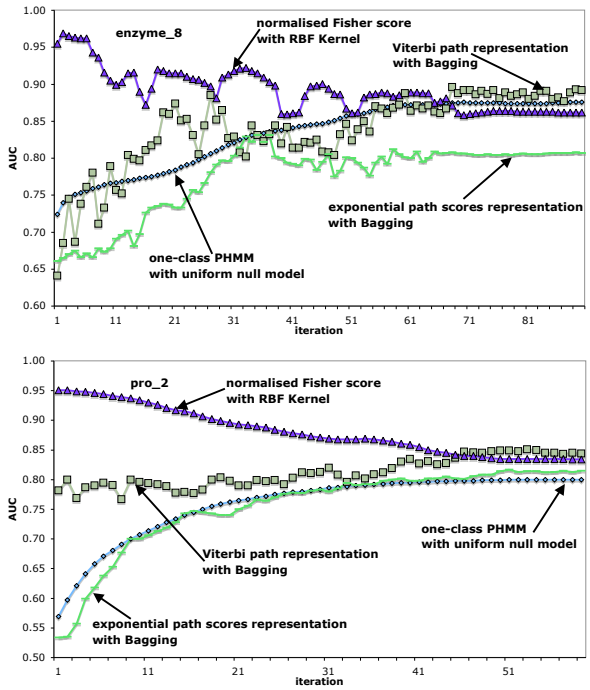


Figure 5.15. *AUC values for enzyme_8 and pro_2 for bagging with different propositional representations. Bagging is performed on top of a Viterbi path representation and one that uses exponential path scores. Both approaches are compared to one-class PHMM classification and propositionalisation with normalised Fisher score used in a Support Vector Machine with RBF kernel.*

one-class PHMM classification and propositional Support Vector Machine classification with an RBF kernel based on normalised Fisher score vectors. The Viterbi path representation is not consistently as competitive as the combined exponential one and the representation just based on path scores cannot outperform the one-class approach. This is why the detailed presentation of results in this thesis concentrates on the combined representation and Fisher scores.

Random Forests and linear Support Vector Machines

Random Forests and linear Support Vector Machines are only evaluated on propositional datasets originating from the first and final iteration of the one-class PHMM training algorithm. We trained linear Support Vector Machines on the combined logarithmic and the combined exponential representation. Using RBF kernels did not lead to promising results in preliminary experiments. In terms of AUC the combined logarithmic representation outperforms its exponential counterpart. Thus, the presented results are based on the combined logarithmic representation. When Random Forests are used, the combined exponential representation turns out be more successful. However, its advantage in AUC is not as pronounced as in the linear Support Vector Machine case. There are datasets for which both representations are competitive. Nonetheless, all results for Random Forests are based on the combined exponential representation.

The overall presentation of results will be spread over two tables. The first gives a detailed overview concerning the relationship in terms of AUC between the two propositional learners and the one-class PHMM after training is completed. Thus, it will give insights into whether or not these approaches are worthwhile in pursuing our goal of improving on one-class PHMM based classification. The second table compares the results to Support Vector Machine based classification using normalised Fisher score vectors. Therefore, it will reveal how these approaches compare to the powerful approach of training a one-class PHMM for one iteration and building normalised Fisher score vectors from it. These vectors are used with a Support Vector Machine and a RBF kernel.

Table 5.4 shows the resulting AUCs for all datasets and states whether or not the propositional approaches statistically significantly outperform the basic one-class approach. It uses the notation from Table 5.3. The table reveals that Random Forests more often lead to competitive AUC results compared to the basic one-class PHMM approach than linear Support Vector Machines do. Indeed, when propositionalising after the first iteration, Random Forests boost AUC to a greater extent than linear Support Vector Machines. For the latter, there are even three cases (*enzyme_8*, *enzyme_11* and *enzyme_14*) for which the AUC decreases significantly compared to the one-class case. In these cases, Random Forests perform equally well as the basic one-class PHMM. For *enzyme_8* and *enzyme_1* they increase the AUC though only marginally.

Table 5.4. *Comparison of AUCs for linear Support Vector Machines trained on the combined logarithmic representation, Random Forests trained on the exponential representation compared to the AUC of a fully converged one-class PHMM. The table indicates statistical significance at the* 0.05 *level.*

dataset	AUC	AUC and significance compared to one-class PHMM			
	one-class PHMM	linear SVM		Random Forests	
	last	first	last	first	last
enzyme_1	0.819	0.869 +	0.840 +	0.921 +	0.932 +
enzyme_2	0.864	0.851 =	0.873 =	0.934 +	0.912 +
enzyme_3	0.679	0.707 =	0.742 +	0.915 +	0.903 +
enzyme_4	0.802	0.853 +	0.885 +	0.932 +	0.907 +
enzyme_5	0.834	0.840 =	0.810 =	0.843 =	0.851 =
enzyme_6	0.838	0.859 +	0.856 +	0.955 +	0.935 +
enzyme_7	0.759	0.821 =	0.794 =	0.880 +	0.820 +
enzyme_8	0.876	0.816 -	0.841 =	0.910 =	0.920 =
enzyme_9	0.909	0.951 +	0.961 +	0.971 +	0.973 +
enzyme_10	0.724	0.865 +	0.831 =	0.952 +	0.880 +
enzyme_11	0.941	0.888 -	0.945 =	0.936 =	0.951 =
enzyme_12	0.679	0.725 =	0.768 +	0.877 +	0.860 +
enzyme_13	0.834	0.865 +	0.862 +	0.949 +	0.930 +
enzyme_14	0.993	0.991 -	0.980 -	0.992 =	0.983 =
enzyme_15	0.949	0.973 +	0.914 =	0.958 =	0.962 =
enzyme_16	0.849	0.888 +	0.912 +	0.957 +	0.938 +
euk_0	0.600	0.734 +	0.690 +	0.817 +	0.807 +
euk_1	0.878	0.922 +	0.937 +	0.946 +	0.946 +
euk_2	0.769	0.813 +	0.850 +	0.826 +	0.852 +
euk_3	0.773	0.801 +	0.834 +	0.863 +	0.871 +
pro_0	0.598	0.796 +	0.862 +	0.840 +	0.868 +
pro_1	0.862	0.863 =	0.880 =	0.854 =	0.915 +
pro_2	0.800	0.805 =	0.839 +	0.875 +	0.877 +

When looking at propositionalising after the final iteration of the PHMM training algorithm, Random Forests still consistently outperform the linear Support Vector Machines' predictive power, even though linear Support Vector Machines only decrease AUC in one dataset instead of three. An interesting observation with Random Forests is that if they achieve a significantly higher AUC after the first iteration, they do so as well after the last iteration. This is different to the Fisher score approach, where often the highest AUC is achieved after the first iteration.

The real power of the Fisher score approach after the first iteration becomes apparent when its AUC is compared to the ones gained from linear Support Vector Machines and Random Forests. Table 5.5 presents the results of this comparison. Propositionalising a one-class PHMM after one training iteration by using the Fisher score method and subsequent normalisation leads consistently to significantly higher AUC than a linear Support Vector Machine trained on a combined logarithmic representation independent of whether training of the underlying PHMM is stopped after the first iteration or completed. There is only one dataset, namely *enzyme_14*, for which the normalised Fisher score constructed after one iteration of PHMM training used in a Support Vector Machine with RBF kernel performs equally well in terms of AUC as the combined logarithmic representation with a linear kernel. The situation looks slightly better for Random Forests. At least for *enzyme_6* the AUC for propositionalising after one iteration is significantly higher to the one that is based on normalised Fisher score vectors. There is no significant difference in AUC for seven datasets. However, for the majority of the 15 datasets the normalised Fisher score approach has a significantly greater predictive power. The situation is even more advantageous to normalised Fisher scores, when their results based on the first iteration are compared to the resulting AUC of Random Forests based on a combined exponential representation from the last iteration. Apart from six datasets for which both approaches perform equally well, the performance of Random Forests is significantly worse in terms of AUC.

Fisher score vectors with Random Forests

The last set of experiments tests Fisher score vectors and their normalised versions both in combination with Random Forests. First, we will present the re-

Table 5.5. *Comparison of AUCs for linear Support Vector Machines trained on the combined logarithmic representation, Random Forests trained on the exponential representation compared to Support Vector Machine classification with normalised Fisher score vectors including statistical significance. The table indicates statistical significance at the* 0.05 *level.*

dataset	AUC	AUC and significance compared to normalised Fisher scores after the first iteration			
	normalised Fisher score	linear SVM		Random Forest	
	first	first	last	first	last
enzyme_1	0.932	0.869 (−)	0.840 (−)	0.921 (=)	0.932 (=)
enzyme_2	0.923	0.851 (−)	0.873 (−)	0.934 (=)	0.912 (=)
enzyme_3	0.945	0.707 (−)	0.742 (−)	0.915 (−)	0.903 (−)
enzyme_4	0.946	0.853 (−)	0.885 (−)	0.932 (=)	0.907 (−)
enzyme_5	0.896	0.840 (−)	0.810 (−)	0.843 (=)	0.851 (=)
enzyme_6	0.936	0.859 (−)	0.856 (−)	0.955 (+)	0.935 (=)
enzyme_7	0.906	0.821 (−)	0.794 (−)	0.880 (−)	0.820 (−)
enzyme_8	0.955	0.816 (−)	0.841 (−)	0.910 (−)	0.920 (=)
enzyme_9	0.996	0.951 (−)	0.961 (−)	0.971 (−)	0.973 (−)
enzyme_10	0.982	0.865 (−)	0.831 (−)	0.952 (−)	0.880 (−)
enzyme_11	0.983	0.888 (−)	0.945 (−)	0.936 (−)	0.951 (−)
enzyme_12	0.945	0.725 (−)	0.768 (−)	0.877 (−)	0.860 (−)
enzyme_13	0.951	0.865 (−)	0.862 (−)	0.949 (=)	0.930 (−)
enzyme_14	1.00	0.991 (=)	0.980 (−)	0.992 (=)	0.983 (−)
enzyme_15	1.00	0.973 (−)	0.914 (−)	0.958 (−)	0.962 (−)
enzyme_16	0.962	0.888 (−)	0.912 (−)	0.957 (=)	0.938 (=)
euk_0	0.929	0.690 (−)	0.734 (−)	0.807 (−)	0.817 (−)
euk_1	0.983	0.922 (−)	0.937 (−)	0.946 (−)	0.946 (−)
euk_2	0.960	0.813 (−)	0.850 (−)	0.826 (−)	0.852 (−)
euk_3	0.961	0.801 (−)	0.834 (−)	0.863 (−)	0.871 (−)
pro_0	0.989	0.796 (−)	0.862 (−)	0.840 (−)	0.868 (−)
pro_1	0.968	0.863 (−)	0.880 (−)	0.854 (−)	0.915 (−)
pro_2	0.951	0.805 (−)	0.839 (−)	0.875 (−)	0.877 (−)

sults for some characteristic, individual datasets. The comparison with the basic one-class PHMMs will be summarised in a table. Finally, the overall results are compared to the one from normalised Fisher score vectors used in a Support

Vector Machine with RBF kernel.

Figure 5.16 shows the results for *enzyme_1* and *enzyme_3*. The first observa-

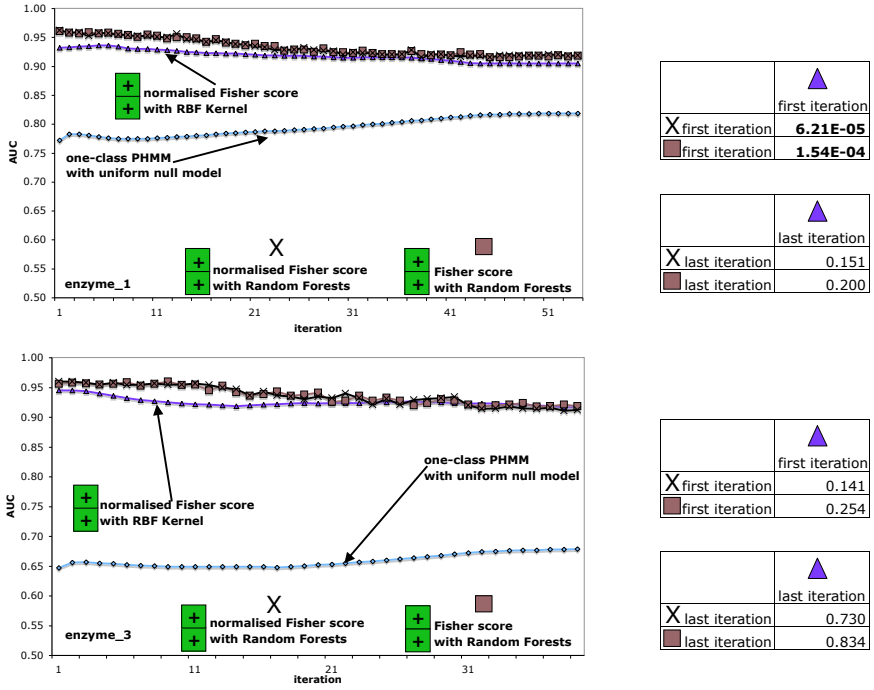


Figure 5.16. *AUC values for enzyme_1 and enzyme_3 in one-class PHMM classification and propositional classification using Fisher score vectors with Random Forests. The AUC is calculated in each iteration of the Baum-Welch training algorithm. The table on the right contains the p-values. P-values printed in bold indicate a statistically significant difference in AUC at the* 0.05 *level. The meaning of the squares in the left-hand graphs is defined in Figure 5.7.*

tion is that the Fisher score vector and their normalised version perform almost identically when used in combination with Random Forests. This fact is in contrast to the situation presented earlier when the different Fisher score vectors are used with a Support Vector Machine. In the latter case, normalisation improves performance. Random Forests do not suffer from lack of normalisation. For *enzyme_1* the AUC achieved with Random Forests is higher than the one achieved by a Support Vector Machine. However, the differences are only statistically significant after the first iteration. All approaches independent from propositionalising after the first or last iteration significantly outperform a fully trained one-class PHMM.

The situation is only slightly different for *enzyme_3*. Again, Random Forests perform better than Support Vector Machines, however the differences are not significant. In addition, when propositionalising after the last iteration, it is only the Fisher score without normalisation in combination with Random Forests that better discriminate the two classes than the Support Vector Machine approach based on normalised Fisher score vectors. Similar conclusions can be drawn from datasets *euk_0* and *euk_1*. Their results can be found in Appendix C.3.

When looking at *enzyme_6*, as presented in Figure 5.17, Random Forests boost AUC in the same way they do for *enzyme_1* and *enzyme_3*. However, in this case

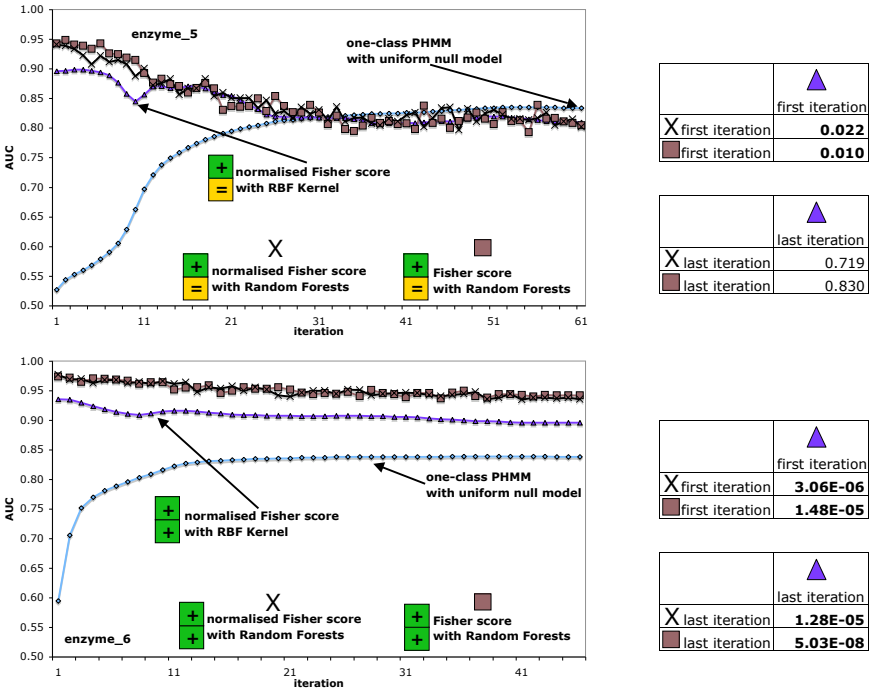


Figure 5.17. *AUC values for enzyme_5 and enzyme_6 in one-class PHMM classification and propositional classification using Fisher score vectors with Random Forests following the layout of Figure 5.16.*

the AUC from Random Forests is consistently better than the one gained from Support Vector Machines and the differences in AUC after the first and last iteration are significant. Again, when Random Forests are used, normalisation does not influence the resulting AUC greatly. In fact, when the Fisher score vectors are built after the first iteration there is no dataset reporting a statistically significant difference in AUC between normalised Fisher scores and those which are

unnormalised. The situation is different when Support Vector Machines with an RBF kernel are used. In that case, in 17 of the 23 datasets, normalisation improves the results significantly. In the other 5 cases, there is no statistically significant difference.

For *enzyme_5*, we observe again a statistically significant improvement in AUC after the first iteration when Random Forests are used instead of a Support Vector Machine. Propositionalisation after the last iteration and subsequent use of Support Vector Machines and Random Forests on Fisher scores show no significant differences. This is the case if the Support Vector Machine uses a normalised Fisher score representation. As previous experiments after the last iteration show (see Figure 5.11) the AUC decreases significantly when normalisation is omitted before Support Vector Machine classification. Compared to one-class classification on a fully trained PHMM, propositionalisation after the first iteration leads to a significant increase in AUC, whereas a propositional representation built from a fully trained one-class PHMM performs marginally worse.

The only dataset where Support Vector Machines trained with normalised Fisher scores from the first iteration significantly outperform the Random Forests' approaches is *pro_0* as Figure 5.18 shows. The AUC increases from 0.983 to 0.989. This is a small but significant gain in predictive power. After the last iteration there is no statistically significant difference in performance of the propositional approaches. All of them improve the AUC of the fully converged one-class PHMM significantly in both settings – after propositionalising past one iteration or past the final iteration of PHMM training.

This last observation also holds for *pro_1*. However, the relationship between the performance of the two propositional learners is different. All three approaches perform equally well if applied after the first iteration of PHMM training. However, *pro_1*, *enzyme_15* and *euk_3* are the only three datasets for which the Support Vector Machine approach leads to significantly higher AUC when propositionalisation is performed on a fully trained one-class PHMM. The results for the latter two datasets can be found again in Appendix C.3.

In Figure 5.19 *pro_2* shows the opposite behaviour. The performance of Random Forests as propositional learner on Fisher scores from the last one-class PHMM training iteration is significantly better than the one from a Support Vec-

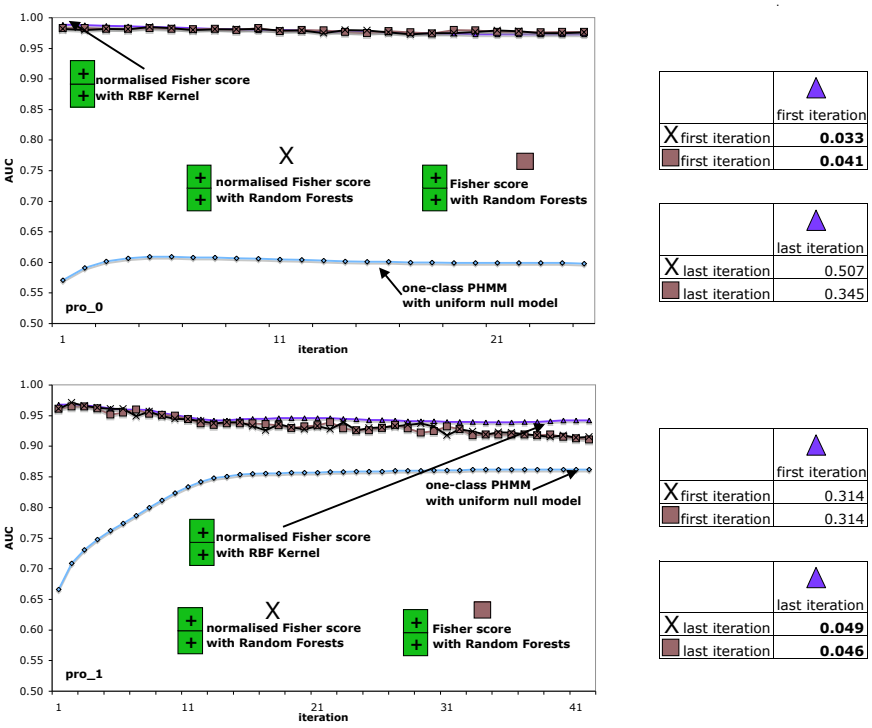


Figure 5.18. *AUC values for pro_0 and pro_1 in one-class PHMM classification and propositional classification using Fisher score vectors with Random Forests following the layout of Figure 5.16.*

tor Machine trained on normalised Fisher scores. Additionally, there is no statistically significant difference in predictive power between the three approaches when applied after the first PHMM training iteration. Concerning these two aspects, *enzyme_9* produces the same results. When comparing Fisher scores with Random Forests and one-class PHMM classification, there is a significant improvement for the first and the last iteration for *pro_2*. Support Vector Machines achieve this sort of improvement only after the first iteration.

For *enzyme_11* all three propositional approaches perform without significant difference. Classification with a fully trained one-class PHMM has equally good predictive power to the propositional learners used on top of a representation from the last PHMM training iteration. However, propositionalisation after the first iteration boosts AUC significantly in all three cases compared to the basic PHMM approach.

Appendix C.3 presents the results for all other datasets. They follow closely

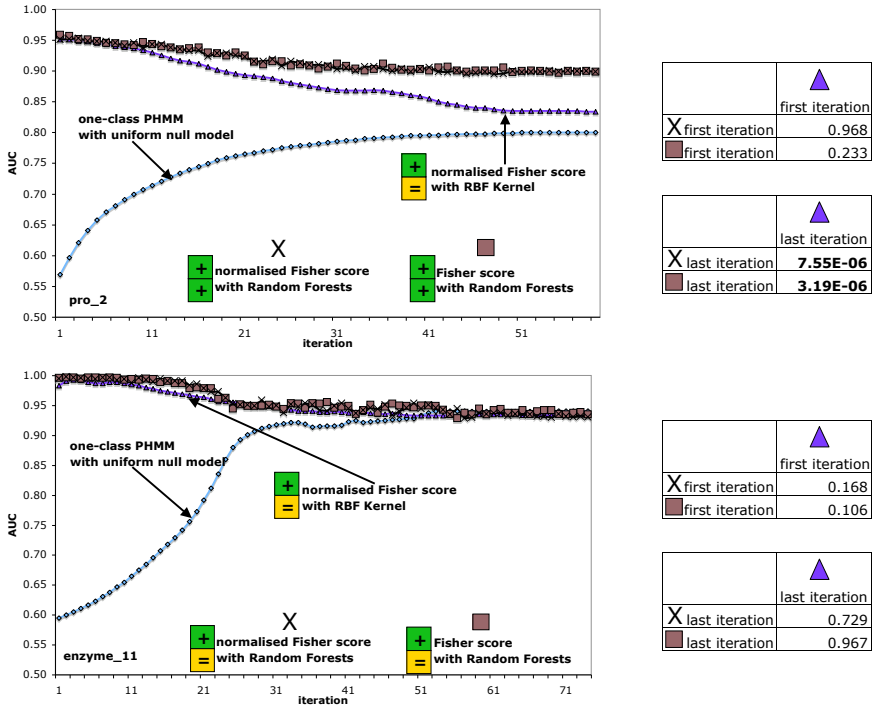


Figure 5.19. *AUC values for enzyme_11 and pro_2 in one-class PHMM classification and propositional classification using Fisher score vectors with Random Forests following the layout of Figure 5.16.*

one of the characteristic results from this section.

Table 5.6 summarises statistically significant performance differences. It compares the Fisher score based approaches to one-class PHMM classification. The table indicates whether or not the normalised Fisher score representation used with a Support Vector Machine with an RBF kernel and Random Forests trained on the Fisher score vector without normalisation perform statistically significantly better(+), equally well(=) or worse(-) in terms of AUC than a fully converged one-class PHMM. In combination with Random Forests, experiments have shown that normalisation of Fisher score does not decisively influence the results. In 16 of 23 datasets there is a decrease in AUC after normalisation of vectors based on the first iteration albeit not a significant one in most cases. This is why this table and the following presents the results without normalisation when Random Forests are the propositional learners.

For all but three datasets both approaches show the exact same behaviour. If

Table 5.6. *Summary of statistically significant differences of Fisher scores used with Support Vector Machines and Random Forests. The Support Vector Machine approach uses normalised Fisher score vectors as opposed to Random Forests that omit normalisation prior to classification.*

dataset	normalised Fisher score with RBF kernel		Fisher score with Random Forests	
	first	last	**first**	last
enzyme_1	+	+	+	+
enzyme_2	+	=	+	+
enzyme_3	+	+	+	+
enzyme_4	+	+	+	+
enzyme_5	+	=	+	=
enzyme_6	+	+	+	+
enzyme_7	+	=	+	=
enzyme_8	+	=	+	=
enzyme_9	+	+	+	+
enzyme_10	+	+	+	+
enzyme_11	+	=	+	=
enzyme_12	+	+	+	+
enzyme_13	+	+	+	+
enzyme_14	=	=	=	=
enzyme_15	+	+	+	=
enzyme_16	+	+	+	+
euk_0	+	+	+	+
euk_1	+	+	+	+
euk_2	+	+	+	+
euk_3	+	+	+	+
pro_0	+	+	+	+
pro_1	+	+	+	+
pro_2	+	=	+	+

propositionalisation is applied after just one iteration of the Baum-Welch training algorithm, both methods significantly boost AUC compared to a fully converged one-class PHMM for all datasets but *enzyme_14*. And even in that case the results are not statistically significantly worse. When a fully trained PHMM is propositionalised, we observe a favourable behaviour of Random Forests for *enzyme_2* and *pro_2*. However, the kernel based approach leads to better results for *enzyme_15*.

Table 5.7 compares the actual AUCs of the two propositional approaches and tests whether or not the reported difference is statistically significant. Both approaches have been applied after just one iteration of PHMM training. There is only one dataset, namely *pro_0*, for which a Support Vector Machine trained on normalised Fisher score vectors significantly outperforms Random Forests. This is the only dataset where the positive class is actually the majority class. In ten other datasets, Random Forests trained with Fisher score vectors boost AUC to a considerably greater extent than Support Vector Machines. For the other twelve datasets there is no significant difference, even though in seven of them the use of Random Forests leads to a slight increase in AUC.

The results show the power of propositionalisation. It is especially remarkable that the best results in terms of AUC are achieved by stopping PHMM training after just one iteration and subsequent propositionalisation. Fisher score vectors in combination with Random Forests as propositional learners are highly competitive. Returning to alignment quality from Section 2.6, this chapter shows that using a low quality alignment as the basis for propositionalisation boosts predictive power significantly not only compared to basic PHMM based classification but also compared to propositionalising a high quality alignment. As pointed out in Section 2.6, alignment quality is regarded as crucial for PHMM based approaches. However, as this chapter shows, propositionalisation does not require a high quality alignment. In fact, it can be to the detriment of AUC.

Liao and Noble (2002) significantly increase AUC compared to unnormalised Fisher score vectors gained from a one-class PHMM after the final iteration of training used in combination with a Support Vector Machine. This thesis achieves a significant boost of predictive performance as well. However, their approach differs in two ways. First, their PHMMs are more prone to overfit as explained in Section 2.5.2. Secondly and more importantly, their methods lead

Table 5.7. *Comparison of AUCs for Support Vector Machine classification with normalised Fisher score vectors and classification with Random Forests using Fisher score vectors including statistical significance. The table indicates statistical significance at the* 0.05 *level.*

dataset	AUC after first iteration	
	normalised Fisher score with RBF kernel	Fisher score with Random Forests
enzyme_1	0.932	0.961 (+)
enzyme_2	0.923	0.962 (+)
enzyme_3	0.945	0.956 (=)
enzyme_4	0.946	0.979 (+)
enzyme_5	0.896	0.943 (+)
enzyme_6	0.936	0.974 (+)
enzyme_7	0.906	0.970 (+)
enzyme_8	0.955	0.986 (=)
enzyme_9	0.996	0.995 (=)
enzyme_10	0.982	0.992 (=)
enzyme_11	0.983	0.996 (=)
enzyme_12	0.945	0.972 (+)
enzyme_13	0.951	0.983 (+)
enzyme_14	1.00	0.995 (=)
enzyme_15	1.00	0.989 (=)
enzyme_16	0.962	0.988 (+)
euk_0	0.929	0.936 (=)
euk_1	0.983	0.990 (=)
euk_2	0.960	0.973 (+)
euk_3	0.961	0.956 (=)
pro_0	0.989	0.983 (−)
pro_1	0.968	0.961 (=)
pro_2	0.951	0.959 (=)

to a significant increase in runtime whereas our approach reduces runtime.

These findings have impacts on the runtime of the classification process and the following section will discuss them.

5.2.2 Runtime

As pointed out in Section 5.1.4 it is asymptotically as expensive to train one sequence in one iteration of the Baum-Welch algorithm than to propositionalise

the sequence. Therefore, propositionalisations on top of learning a one-class PHMM are especially competitive in terms of runtime if either the PHMM already exists or, as the results suggest, if the PHMM learning can be stopped after one iteration.

However, as Figure 5.20 shows, even propositionalising after one-class PHMM training is fully completed does not increase runtime in a drastic way. As op-

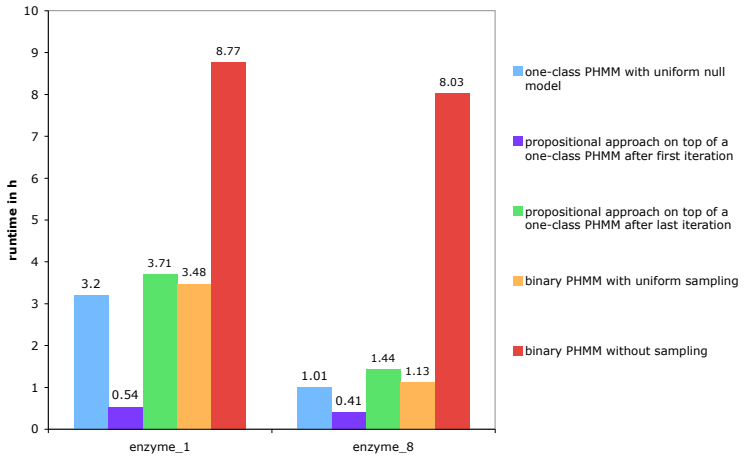


Figure 5.20. *Comparison of runtime for PHMM based and propositional approaches for enzyme_1 and enzyme_8. The proposed propositional approach is classification with Random Forests using the Fisher score representation. The propositionalisation takes place after the first and the final iteration of the Baum-Welch training algorithm. As opposed to the previous results, all approaches are only evaluated once.*

posed to the runtimes presented in Section 4.5.2, all approaches are only evaluated once and not after each iteration of the Baum-Welch training algorithm. This way the comparison is fair towards propositionalising after just one iteration of the Baum-Welch training algorithm. Random Forests with Fisher score vectors are used as the propositional approach in this figure as they lead to the most competitive results in terms of AUC. The figure compares the runtime of basic one-class classification to those of propositional approaches after the first and the final iteration of PHMM training. Additionally, it displays the runtimes of binary PHMM classification with and without sampling. These approaches and the two propositional ones perform binary classification, whereas the one-class PHMM is a pure one-class classifier.

Stopping one-class PHMM training after one iteration, propositionalising and using Random Forests on the propositional representation leads to the fastest runtime. This is not surprising as PHMM training is time intense. Additionally, as the experimental evaluation in terms of AUCs has shown, this approach is overall most beneficial to predictive power as well.

Propositionalising after the final iteration of one-class PHMM training increases runtime compared to the pure one-class approach. In relative terms, the increase is more pronounced for *enzyme_8* as it is the smaller of the two datasets in terms of positive instances and thus, there are fewer instances to train for the one-class PHMM. However, all propositional approaches are binary. Consequently, the increase in absolute terms is similar for *enzyme_8* and *enzyme_1*. For the former, propositionalisation, training and evaluating a propositional binary classifier takes approximately an additional half hour, for the latter 24 minutes. There is a small time difference. However as pointed out earlier, the timing experiments were run on servers without exclusive access.

The binary PHMM approach with sampling is faster in runtime than propositionalisation based on a one-class PHMM after it is fully converged.

For both datasets, propositional classification after one iteration of PHMM training leads to the fastest runtime. In combination with its competitive AUC, propositionalisation is an approach worth pursuing.

5.3 Summary

In this chapter we introduced propositional binary classification based on one-class PHMMs. In terms of AUC propositional binary classification is able to outperform the basic one-class approach. This is especially true after the first iteration. Further investigations are needed to determine the cause of this surprising but advantageous behaviour. Not only does stopping the PHMM training after one iteration and propositionalising lead to higher AUCs, it is also beneficial in terms of runtime compared to the one-class PHMM approach after full convergence or the binary PHMM with or without sampling.

The best performing propositional representation is the Fisher score vector combined with Random Forests. As opposed to Support Vector Machines and RBF kernel, they do not rely upon the normalisation of the Fisher score vector.

Thus, apart from revealing the power of propositionalisation after the first iteration of one-class PHMM training, this chapter showed two additional ways of improving the work of Jaakkola et al. (1999): Either use Random Forests instead of Support Vector Machines or normalise the Fisher score vectors before training the Support Vector Machine.

Chapter 6

Binary classification using propositionalisations of binary models

Experiments in the two previous chapters investigated the benefit of adding negative information in two fundamentally different ways:

- First, we extended the generative one-class PHMM directly to a binary classifier, the so called binary PHMM.
- Second, the one-class PHMM was the input for a feature generating process. The resulting binary, propositional representation was used to train standard Machine Learning classifiers.

The idea of the former approach is to train two separate one-class PHMMs – one on the positive data and the other one on the negative training instances. At prediction time, a sequence is classified according to the class of the PHMM that leads to a higher score. We showed in Chapter 4 that the one-class PHMM trained on the negative instances acts like a null model for the positive class. Thus, adding negative information in the form of a one-class PHMM trained on negative instances leads to a new scoring function. This new way of scoring a sequence is superior in terms of AUC compared to the basic one-class case.

The latter approach, presented in Chapter 5, treats a one-class PHMM trained on positive instances as structured input data. Propositionalisation allows creating fixed length feature vector representations out of these PHMMs and thus makes the information in a one-class PHMM available for standard Machine Learning classifiers. Furthermore, the propositionalisation step transforms the classification problem from a one-class to a binary one by creating fixed length

feature vectors for both positive and negative instances using the one-class PHMM. Experiments showed that the predictive power of these binary, propositional classifiers outperforms the basic one-class PHMMs.

This chapter brings together the two previous approaches as Figure 6.1 shows. A PHMM is treated as structured data as in Chapter 5. However, as opposed to

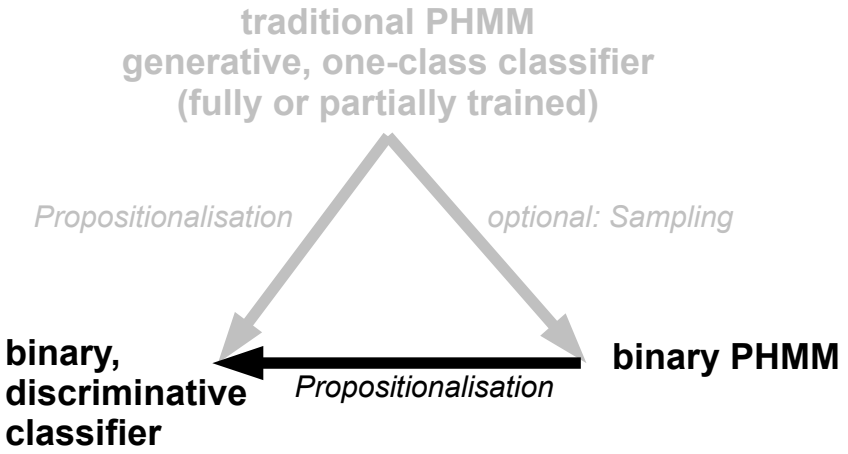


Figure 6.1. *The relevant part of the thesis' scheme*

the previous approach, the input PHMM is binary and not one-class. We will investigate how to propositionalise a binary PHMM and evaluate the benefit of the proposed methods in terms of runtime and predictive power. This chapter is the only chapter that does not use traditional one-class PHMMs as input into the classification process. The proposed methods are built on the basis of binary PHMMs that have been introduced in Chapter 4.

The chapter contributes in three different ways to the thesis as a whole. It introduces a simple approach to propositionalise binary PHMMs. A binary PHMM consists of two one-class PHMMs. Both one-class PHMMs are propositionalised on their own and the resulting feature vectors are concatenated.

Second, in accordance with the results of the previous chapter on one-class PHMMs, propositionalisation of binary PHMMs leads to competitive results compared to basic one-class or binary PHMM classification. Additionally, experiments in both chapters reveal that propositionalising after the first iteration of the Baum-Welch training algorithm is competitive in terms of AUC compared to propositionalisations of fully converged PHMMs and the performance of purely

PHMM based approaches. This is a remarkable property which decreases the runtime of the classification process. Consequently, this chapter and the previous chapter offer the following new strategy for sequence classification as an alternative to a one-class PHMM:

- train a one-class or binary PHMM for one iteration,
- propositionalise the PHMM and consequently
- use a standard discriminative learner on the resulting propositional representation.

This approach is advantageous in both directions of optimisation that this thesis addresses. It leads to a higher AUC and a faster runtime compared to standard one-class PHMM or binary PHMM training to full convergence. Even if the final, fully converged PHMM is needed for further studies, propositionalisation can be done after one iteration before continuing the Baum-Welch training process. This way, even though the final PHMM model is required, the predictive power of purely discriminative tasks benefit from propositionalisation.

The third contribution is two-fold and applies to specific forms of propositionalisation. First, experiments will show that for a specific simple propositional representation, the generated feature vectors from binary PHMMs lead overall to a better discrimination compared to the respective feature vectors built from one-class models. This conclusion is drawn when Random Forests are used as propositional learners. However, as a second part of the contribution, this thesis demonstrates that the aforementioned fact does not necessarily hold when the one-class and the binary PHMM are propositionalised with different methods. This chapter reveals that the overall best performing methods in terms of AUC and runtime is to train a one-class PHMM for one iteration only, create Fisher score vectors and subsequently use Random Forests. Future work will shed more light into the relationship of propositionalisations between one-class and binary PHMMs. Additionally, future investigations will extend the propositionalisations of binary PHMMs to different forms of propositionalisation.

The remainder of the chapter is organised as follows. The next section will introduce the propositionalisation of binary PHMMs. Experimental results will be presented in Section 6.2, and the final section will succinctly summarise our findings.

6.1 Propositionalisations of binary Profile Hidden Markov Models

A key feature of the approaches on one-class PHMMs presented in Chapter 5 is the combination of different propositional representations. This section extends propositionalisation to binary PHMMs by utilising the simple technique of combining representations.

6.1.1 The simple idea: concatenation

A binary PHMM consists of two one-class PHMMs. One is trained solely on the instances of the positive class whereas the other represents the negative class. Both of these PHMMs are propositionalised separately into a fixed-length feature vector following the methods described in detail in Section 5.1. The two resulting feature vectors are concatenated into one. This fixed length vector is the input for all propositional learners.

As the graphical structure of the positive and negative one-class PHMMs are identical and they use the same set of amino acids, the propositionalisations based on binary PHMMs have twice as many attributes as the one-class ones from the previous chapter.

The next section explains specific characteristics of the propositionalisation process for binary PHMM that contrast to the one-class case. It introduces a normalisation strategy that is not available in the one-class setting.

6.1.2 Normalisations of feature vectors

The representation based on the Viterbi path and all representations based on logarithmic scores built on top of binary PHMMs are not post-processed further.

Concerning the representation based on posterior states scores, we normalise the exponential scores separately for each class into real probabilities. Thus, in the final representation the probabilities derived from the positive one-class PHMM sum to one. The same holds for the probabilities from the negative one-class PHMM.

The only difference to the one-class case occurs for the exponential representation of the overall log-odds ratio and the log-odds score for the best path. In

the one-class case, as explained in Section 5.1.2, we do not transform the exponential values into real probabilities due to the fact that a lot of them would evaluate to 0 and 1 respectively. However, in the binary case normalisation is possible. The reason for this is that we have a score for the best path from the positive one-class PHMM and one from the negative PHMM. Let l_p be the logarithmic score of the best path of the positive one-class PHMM and l_n be the respective score of the negative one-class PHMM. In a first step, the scores are used as input in the exponential function resulting in the exponential score b_p and b_n where

$$b_p = e^{l_p}$$

and

$$b_n = e^{l_n}$$

These are the scores used in the one-class case. In the binary case, probabilities are calculated according to the following two formulae for normalisation:

$$p(b_p) = \frac{b_p}{b_p + b_n}$$

and

$$p(b_n) = 1 - p(b_p)$$

The same holds for the overall score. We transform each of these pairs of scores into real probabilities. Table 6.1 illustrates the process using the first instance of the *pro_2* dataset. Values in the table are rounded. This example extends the one presented for the one-class case in Table 5.1. Note that with our normalisation strategy, the probability of the best path can be higher

Table 6.1. *Example illustrating the normalisation of the exponential representations based on path scores for binary PHMMs. The scores are from the first instance of pro_2. Values are rounded*

	positive class		negative class	
	best path	all paths	best paths	all paths
logarithmic	-77.2	9.9	-81.3	8.2
exponential	2.9E-34	19206.6	4.9E-36	3641.0
normalised	**0.98**	**0.85**	0.02	0.15

than the overall probability as indicated in bold in the table. This can happen, because the scores for the best path are normalised independently from the overall scores. A possible theoretic solution would be to normalise all four scores together. However, in practice this would lead to the problem illustrated in Table 5.1 for the one-class case. The scores of many best paths would be normalised to a probability of 0. In the experiments we use the logarithmic and the normalised exponential representations both combined with the representation based on the states.

6.2 Experimental results

This section presents the experimental evaluation for propositionalisations based on binary PHMMs. They show that Random Forests using propositional representations from binary PHMMs gained from stopping their training after one iteration increase the discriminative power compared to a fully converged binary PHMM based classification for most datasets. If there is a decrease in AUC, then it is not significant.

The experiments from Chapter 4 have revealed that binary PHMM classification significantly improves the predictive power of the basic one-class PHMM approach in all but four datasets. For these four datasets, the resulting AUCs from the binary PHMMs are marginally higher than their one-class counterparts. Thus, this section compares results in terms of AUC only to the more powerful binary PHMMs independent of whether the propositionalisation is based on a one-class PHMM or a binary one.

For this comparison we use a similar notation to that of Figure 5.7. However, to indicate that binary PHMMs are now the basis of comparison, it is slightly altered as Figure 6.2 shows. Instead of squares for the one-class case, this section visualises the results with an octagon for binary PHMM based comparisons.

In the following, we will present results for bagging unpruned decision trees as propositional learner. After this, the experiments with Random Forests and linear Support Vector Machines will be shown. Finally, runtime will be discussed.

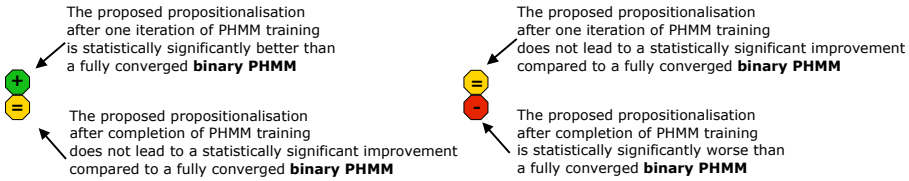


Figure 6.2. *Notation to compare statistical significance.*

6.2.1 Predictive power

Bagging

The experiments for bagging compare the propositionalisations from one-class PHMMs and binary PHMMs to standard binary PHMM classification. In the one-class case the combined exponential representation is used for all datasets. When binary PHMMs are subjected to propositionalisation, the corresponding combined normalised exponential representation forms the input for the bagged decision trees.

Figure 6.3 shows the results for *enzyme_5* and *enzyme_6*. For both datasets, both propositionalisations perform similarly in terms of AUC and indeed there is no statistically significant difference in terms of AUC after the first and the last iteration as the figure indicates. Additionally, when compared to basic binary PHMM classification, there is no difference in performance of both propositionalisations for each dataset on its own. For *enzyme_5* the propositional approaches do not have a significant advantage in terms of AUC over the binary PHMM approach. The one-class approach leads to a marginal increase in AUC when used after the final iteration of one-class PHMM training. The binary counterpart decreases AUC marginally.

The situation is different for *enzyme_6*. Both propositionalisations boost AUC significantly after the first and final iteration of their respective PHMM training.

Figure 6.4 introduces a different possible outcome. In accordance with the results from the previous figure, there is again no statistically significant difference in predictive power between propositional approaches on the basis of one-class and binary PHMMs for *enzyme_8* and *enzyme_14*. Note, the y-axis, representing the AUC values, is scaled differently for *enzyme_14* as all of its AUCs are very close to one.

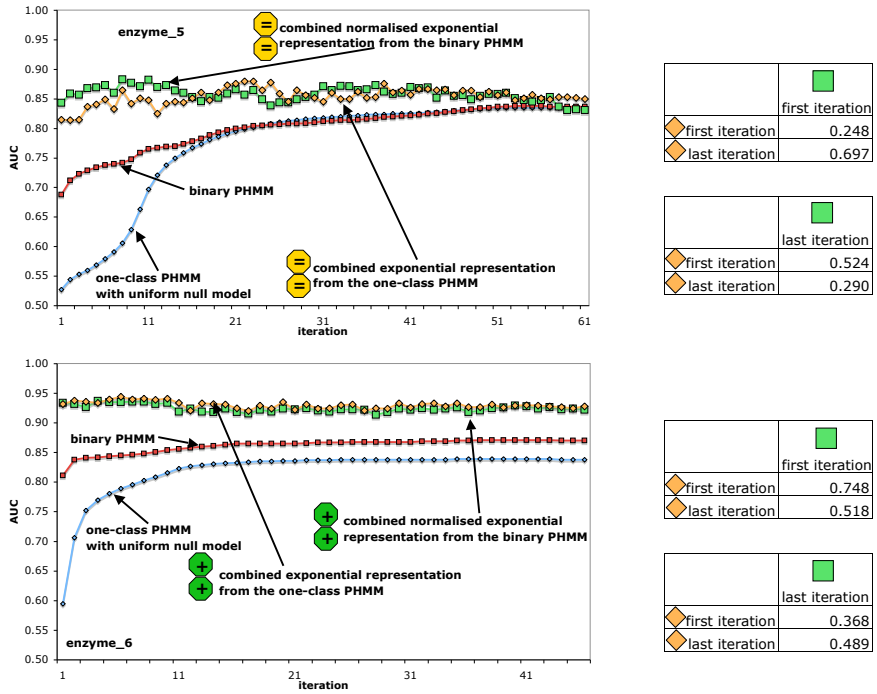


Figure 6.3. *AUC values for enzyme_5 and enzyme_6 in one-class and binary PHMM classification as well as bagging using the combined exponential and the combined normalised exponential representation. The AUC is calculated in each iteration of the Baum-Welch training algorithm. The table on the right contains the p-values. P-values printed in bold indicate a statistically significant difference in AUC at the* 0.05 *level. The meaning of the squares in the left-hand graphs is defined in Figure 6.2.*

The only dataset for which both propositionalisations decrease the predictive power significantly after the first and final training iteration compared to a binary PHMM is *enzyme_8*. For *enzyme_14* propositionalising after the first iteration of the Baum-Welch training algorithm leads to an equal performance in terms of AUC compared to fully training a binary PHMM. However, the AUC is slightly lower.

There are datasets, however, for which propositionalisations based on binary PHMMs work significantly better in terms of AUC than their one-class counterparts. Figure 6.5 introduces two of them, namely *euk_2* and *pro_2*. In these cases propositionalisations based on binary PHMMs are superior over the one-class based ones through the entire Baum-Welch training process. For *euk_2*, there is a statistically significant difference in AUC for the binary PHMM based

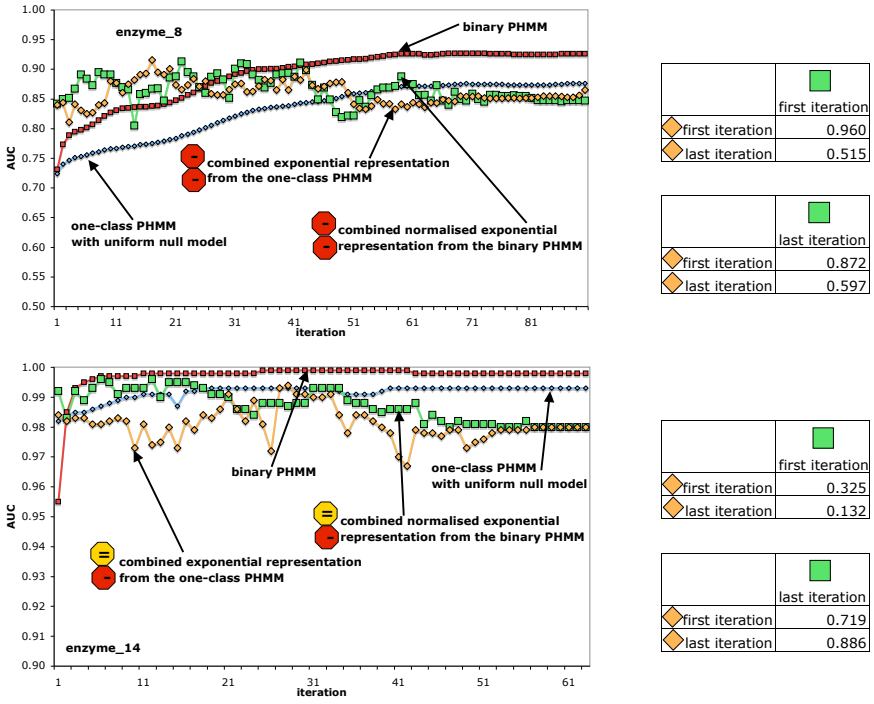


Figure 6.4. *AUC values for enzyme_8 and enzyme_14 in one-class and binary PHMM classification as well as bagging using the combined exponential and the combined normalised exponential representation following the layout of Figure 6.3. The y-axis is scaled differently for enzyme_14.*

propositionalisation compared to the one-class PHMM based one after the first and the final iteration. Additionally, the binary PHMM based propositionalisations perform equally well in terms of AUC as a fully trained binary PHMM. On the contrary, the propositionalisation built from a one-class PHMM decreases AUC significantly after the first and final training iteration compared to a binary PHMM after full convergence.

When compared to pure binary PHMM classification after full convergence, then the situation is slightly more favourable towards propositional approaches for *pro_2*. When a binary PHMM is propositionalised after full convergence and the combined normalised exponential representation is used as input for bagging unpruned decision trees then the AUC increases significantly compared to the pure binary PHMM approach. The propositional classifier trained on a representation from the first iteration performs equally as well as the binary PHMM after training is completed. In the one-class case, the corresponding proposi-

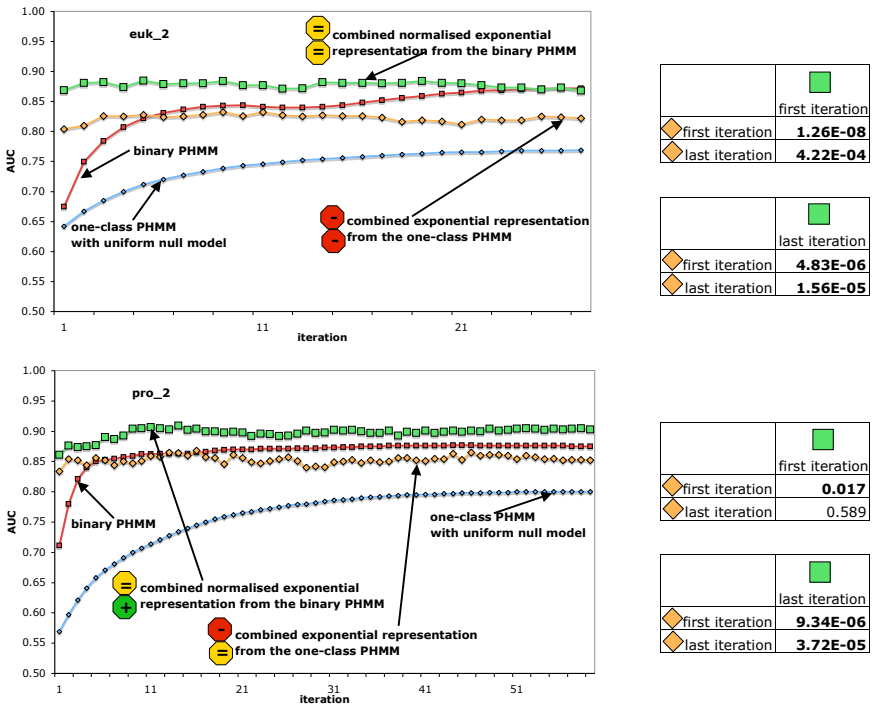


Figure 6.5. *AUC values for euk_2 and pro_2 in one-class and binary PHMM classification as well as bagging using the combined exponential and the combined normalised exponential representation following the layout of Figure 6.3.*

tionalisation after the first iteration decreases AUC significantly compared to a binary PHMM. When the propositional representation is constructed from a fully trained one-class PHMM, its predictive power is competitive with the one from a fully trained binary PHMM.

The graphs for all other datasets are located in Appendix D.

Table 6.2 summarises the results for bagging based on representations from one-class and binary PHMMs in terms of the relationship of their resulting AUC compared to a fully converged binary PHMM. It shows whether or not four different propositional representations in conjunction with bagging have a significant advantage in terms of the AUC compared to binary PHMM classification after full convergence. Both, a binary and a one-class PHMM are propositionalised after the first and the final iteration of the Baum-Welch training algorithm. Subsequently, bagging is used on all four different propositional, binary datasets. The combined exponential representation is used in the one-class case and its

Table 6.2. *Summary of statistically significant differences of two propositional approaches after the first and last iteration compared to a fully converged binary PHMM as baseline classifier. The propositional approaches are bagging with a combined exponential representation built from a one-class PHMM and bagging with a combined normalised exponential representation built from a binary PHMM.*

dataset	propositionalisation based on			
	binary PHMM		one-class PHMM	
	iteration		iteration	
	first	last	**first**	last
enzyme_1	+	+	+	+
enzyme_2	=	+	=	=
enzyme_3	+	+	+	+
enzyme_4	+	+	+	+
enzyme_5	=	=	=	=
enzyme_6	+	+	+	+
enzyme_7	=	=	=	-
enzyme_8	-	-	-	-
enzyme_9	=	+	=	=
enzyme_10	=	-	=	=
enzyme_11	=	=	=	=
enzyme_12	=	=	=	=
enzyme_13	+	+	+	+
enzyme_14	=	-	=	-
enzyme_15	=	=	-	-
enzyme_16	+	+	=	-
euk_0	+	+	-	-
euk_1	-	=	-	-
euk_2	=	=	-	-
euk_3	+	+	-	-
pro_0	=	=	-	-
pro_1	=	=	-	-
pro_2	=	+	-	=

binary counterpart – the combined normalised exponential representation – in the case of a binary PHMM.

The table clearly shows that propositionalising a binary PHMM leads to superior results than when using a one-class model. Propositionalisation of a binary PHMM after the first iteration only decreases AUC significantly for two datasets, but on the other hand boosts predictive performance in a significant way for eight of the 23 datasets. If propositionalisation is executed after the binary PHMM has converged, the AUC significantly increases for eleven datasets. However, there is also a significant loss for three other datasets.

One-class PHMM, independent of whether propositionalisation takes place after the first or the last iteration, improves the discriminative power significantly on the same five datasets. However, propositionalisations built upon a fully converged one-class PHMM are significantly inferior in terms of AUC for eleven datasets, whereas the propositional models on the basis of a one-class PHMM after one iteration decrease AUC significantly in two fewer cases.

In general, the table shows that propositionalisations based on binary PHMMs perform better than their one-class counterparts in terms of significantly increasing the discriminative power of PHMM based classification. This holds when bagging is used as a propositional classifier.

The next section will investigate Random Forests and linear Support Vector Machines as propositional learners.

Random Forests and linear Support Vector Machines

All experiments for Random Forests and linear Support Vector Machines only include training and evaluating propositional representations built after the first and final iteration of the respective PHMM training.

For linear Support Vector Machines, when propositionalisations are built on top of a one-class PHMM, the combined logarithmic representation performs the best. For binary PHMM based propositionalisation its binary counterpart also leads to the highest AUCs. For Random Forests the combined exponential representation in the one-class case and its binary counterpart the combined normalised exponential representation performs best. However, in the case of linear Support Vector Machines the dominance of the particular representation in terms of AUC is much more pronounced. Therefore, the results presented

for linear Support Vector Machines and Random Forests work with the above mentioned representations.

Table 6.3 summarises the findings. All propositionalisations in the table are based on binary PHMMs. The PHMM models are propositionalised after the first and the final iteration of binary PHMM training. The table also indicates significant changes in AUCs compared to standard binary PHMM classification.

One of the most important observations is that for both propositional learners when a binary PHMM is propositionalised after the first iteration, the AUC is at least as good as the one from a fully trained binary PHMM. All losses in AUC are marginal. Additionally, both propositional learners when used on a representation derived from a binary PHMM after one iteration, increase AUC significantly for twelve datasets. For the other eleven datasets, there is no significant change in AUC. As in the one-class case from the previous chapter, propositionalising after the final iteration of binary PHMM training is not as competitive compared to the performance of the binary PHMM on its own. For Random Forests there are again twelve significant wins. However, there are also two significant losses compared to standard binary PHMM classification. The situation is slightly worse for linear Support Vector Machines. A linear Support Vector Machine on a representation from a fully converged binary PHMM outperforms the binary PHMM itself in terms of AUC on only five datasets. There are five cases where AUC decreases from propositionalising.

Table 6.3 clearly shows that propositionalisation after the first iteration of the Baum-Welch training algorithm on binary PHMMs is preferable for linear Support Vector Machines and Random Forests. Table 6.4 uncovers which of the two propositional learners is favourable in terms of predictive power in this setting. In seven cases Random Forests lead to a significantly higher AUC than linear Support Vector Machines and only for four datasets is the predictive performance significantly inferior to that of a linear Support Vector Machine. For the other twelve datasets there is no statistically significant difference in terms of AUC between the two propositional learners. However, in nine of the twelve cases Random Forests achieve a marginally higher AUC. Therefore, for the remainder of this section, other approaches will be compared to Random Forests trained with a combined normalised exponential representation built from a binary PHMM which stopped its training after one iteration.

Table 6.3. *Comparison of AUCs for linear Support Vector Machines trained on the combined logarithmic representation, Random Forests trained on the combined normalised exponential representation to the AUC of a fully converged binary PHMM. All propositional representations are based on binary PHMMs and retrieved either after the first or the final iteration of binary PHMM training. The table indicates statistical significance at the* 0.05 *level.*

dataset	AUC	AUC and significance compared to binary PHMM			
	binary PHMM	linear SVM		Random Forests	
	last	first	last	first	last
enzyme_1	0.835	0.886 +	0.871 +	0.930 +	0.934 +
enzyme_2	0.884	0.905 +	0.879 =	0.935 +	0.918 +
enzyme_3	0.783	0.840 +	0.831 +	0.917 +	0.905 +
enzyme_4	0.872	0.912 +	0.897 +	0.955 +	0.944 +
enzyme_5	0.837	0.855 =	0.828 =	0.864 =	0.855 =
enzyme_6	0.870	0.909 +	0.905 +	0.957 +	0.938 +
enzyme_7	0.847	0.873 =	0.752 -	0.861 =	0.837 =
enzyme_8	0.926	0.943 =	0.876 -	0.931 =	0.898 =
enzyme_9	0.962	0.972 =	0.968 =	0.979 +	0.978 +
enzyme_10	0.922	0.955 +	0.841 -	0.967 +	0.890 -
enzyme_11	0.955	0.965 =	0.940 =	0.970 =	0.960 =
enzyme_12	0.847	0.874 =	0.856 =	0.911 +	0.867 =
enzyme_13	0.896	0.922 +	0.917 +	0.957 +	0.947 +
enzyme_14	0.998	0.992 =	0.991 =	0.994 =	0.986 =
enzyme_15	0.989	0.997 +	0.931 -	0.968 =	0.966 -
enzyme_16	0.930	0.940 =	0.941 =	0.974 +	0.957 +
euk_0	0.831	0.875 +	0.849 +	0.882 +	0.880 +
euk_1	0.971	0.963 =	0.968 =	0.966 =	0.969 =
euk_2	0.872	0.909 +	0.892 +	0.878 =	0.890 +
euk_3	0.888	0.921 +	0.913 +	0.926 +	0.922 +
pro_0	0.956	0.963 =	0.948 =	0.947 =	0.966 =
pro_1	0.931	0.937 =	0.918 =	0.910 =	0.940 =
pro_2	0.875	0.898 +	0.849 -	0.893 =	0.907 +

Table 6.4. *Comparison of AUCs for linear Support Vector Machine classification with combined logarithmic representation and classification with Random Forests using the combined normalised exponential representation including statistical significance. The propositional representations are built from binary PHMM after stopping their training after the first iteration. The table indicates statistical significance at the* 0.05 *level.*

dataset	AUC after first iteration	
	linear Support Vector Machine	Random Forests
enzyme_1	0.886	0.930 (+)
enzyme_2	0.905	0.935 (+)
enzyme_3	0.840	0.917 (+)
enzyme_4	0.912	0.955 (+)
enzyme_5	0.855	0.864 (=)
enzyme_6	0.909	0.957 (+)
enzyme_7	0.873	0.861 (=)
enzyme_8	0.943	0.931 (=)
enzyme_9	0.971	0.979 (=)
enzyme_10	0.955	0.967 (=)
enzyme_11	0.965	0.970 (=)
enzyme_12	0.874	0.911 (=)
enzyme_13	0.922	0.957 (+)
enzyme_14	0.992	0.994 (=)
enzyme_15	0.997	0.968 (−)
enzyme_16	0.940	0.974 (+)
euk_0	0.875	0.882 (=)
euk_1	0.963	0.966 (=)
euk_2	0.909	0.878 (−)
euk_3	0.921	0.926 (=)
pro_0	0.963	0.947 (−)
pro_1	0.937	0.910 (−)
pro_2	0.898	0.893 (=)

This section presents three more comparisons. First, the results of the approach with Random Forests is compared to bagging.

Table 6.5 gives an overview of the AUCs and the significance of their differences when Random Forests and bagging are both used on a combined normalised exponential representation gained after one iteration of binary PHMM training. Random Forests clearly dominate the bagging approach in twelve of the 23 datasets. For the eleven remaining datasets Random Forests achieve a marginally higher AUC in ten of the eleven cases. However, in some cases like *euk_3* and *pro_1*, the difference in AUC of 0.01 is extremely small. Bagging only leads to a significant advantage in predictive performance for *enzyme_7* compared to Random Forests.

Thus far the experimental results show that Random Forests with a combined normalised exponential representation which originates from a binary PHMM after one training iteration not only outperforms the standard binary PHMM approach, but also linear Support Vector Machines and bagging.

As a next step, Table 6.6 compares the AUC results for Random Forests trained on a propositional representation from a binary PHMM after one iteration to its one-class PHMM counterpart. Both propositional approaches are compared to a fully trained binary PHMM.

It is clear that the Random Forests approach which works on propositionalisations from binary PHMM leads to better overall AUC results. When propositionalised after the first iteration, this approach leads to twelve significant wins and no losses, whereas the one-class based approach only significantly outperforms standard binary PHMM based classification on eight datasets, but significantly loses in terms of AUC on six datasets. Propositionalisations of a one-class PHMM after the last iteration in combination with Random Forests reduces the number of significant wins to seven and increases the number of losses to seven. Random Forests trained on a representation from the last iteration of binary PHMMs do not perform overall as well as when propositionalisation takes place after the first iteration. However, the situation is not as bad as in the one-class case. In the binary case, there are only two significant losses and the number of significant wins is the same as when propositionalisation is executed after the first training iteration of a binary PHMM.

Table 6.5. *Comparison of AUCs for the combined normalised exponential representation derived from binary PHMMs after one iteration used with bagging and Random Forests including statistical significance. The table indicates statistical significance at the* 0.05 *level.*

dataset	AUC after first iteration	
	Bagging	Random Forests
enzyme_1	0.894	0.930 (+)
enzyme_2	0.899	0.935 (+)
enzyme_3	0.884	0.917 (+)
enzyme_4	0.949	0.955 (=)
enzyme_5	0.843	0.864 (=)
enzyme_6	0.934	0.957 (+)
enzyme_7	0.865	0.861 (=)
enzyme_8	0.842	0.931 (+)
enzyme_9	0.970	0.979 (=)
enzyme_10	0.944	0.967 (+)
enzyme_11	0.960	0.970 (=)
enzyme_12	0.865	0.911 (+)
enzyme_13	0.942	0.957 (+)
enzyme_14	0.992	0.994 (=)
enzyme_15	0.966	0.968 (=)
enzyme_16	0.956	0.974 (+)
euk_0	0.872	0.882 (+)
euk_1	0.951	0.966 (+)
euk_2	0.869	0.878 (=)
euk_3	0.925	0.926 (=)
pro_0	0.944	0.947 (=)
pro_1	0.909	0.910 (=)
pro_2	0.861	0.893 (+)

Table 6.6. *Summary of statistically significant differences of two propositional approaches after the first and last iteration compared to a fully converged binary PHMM as baseline classifier. The propositional approaches are Random Forests with a combined normalised exponential representation built from a binary PHMM and Random Forests with a combined exponential representation built from a one-class PHMM.*

dataset	propositionalisation based on			
	binary PHMM		one-class PHMM	
	iteration		iteration	
	first	last	**first**	last
enzyme_1	+	+	+	+
enzyme_2	+	+	+	+
enzyme_3	+	+	+	+
enzyme_4	+	+	+	+
enzyme_5	=	=	=	=
enzyme_6	+	+	+	+
enzyme_7	=	=	=	=
enzyme_8	=	=	=	=
enzyme_9	+	+	=	+
enzyme_10	+	-	+	-
enzyme_11	=	=	=	=
enzyme_12	+	=	=	=
enzyme_13	+	+	+	+
enzyme_14	=	=	=	-
enzyme_15	=	-	-	-
enzyme_16	+	+	+	=
euk_0	+	+	=	-
euk_1	=	=	-	-
euk_2	=	+	-	=
euk_3	+	+	-	-
pro_0	=	=	-	-
pro_1	=	=	-	=
pro_2	=	+	=	=

Therefore, Random Forests built upon a propositional representation from a binary PHMM trained only for one iteration not only outperform the other propositional learners on datasets built from binary PHMMs, but they are also favourable to their one-class counterpart.

The final comparison involves the most successful approach in propositionalisation of binary PHMMs, i.e. Random Forests trained on a combined normalised exponential representation gained from a binary PHMM after one training iteration, and the most successful one-class PHMM based approach, which is to propositionalise a one-class PHMM after one training iteration and use the derived Fisher score vectors with Random Forests. Table 6.7 presents the results of this comparison. The table clearly indicates that the one-class PHMM based Fisher score vectors combined with Random Forests are superior in a significant way compared to the most successful approach based on binary PHMMs. In all but one dataset, the Fisher score vectors achieve a significantly higher AUC. For *enzyme_14* the two approaches perform almost the same as the AUC for the Fisher score vector method is only increased marginally by 0.01.

This chapter compared all propositional approaches to fully trained binary PHMMs independent of whether they have been derived from a one-class or binary PHMM, because the binary PHMM clearly outperforms its one-class counterpart.

To summarise, propositionalisations based on binary PHMMs after one training iteration combined with Random Forests can boost predictive performance. There is either a statistically significant increase in AUC or the propositional and the binary PHMM approach perform equally well. These propositionalisations have an advantage compared to when the corresponding propositional representation from a one-class PHMM is used with Random Forests. In this case, there are six overall losses in comparison to fully converged binary PHMMs. But recall that pure one-class PHMMs have 20 significant losses over pure binary PHMMs.

However, the situation is heterogeneous as the simple propositionalisations based on binary PHMMs decrease AUC significantly in all but one case compared to Random Forests trained on Fisher score vectors from one-class PHMMs that have only been trained for one iteration. It is therefore a promising direction of future research to investigate the difference in performance of Fisher score

Table 6.7. *Comparison of AUCs for the Fisher score vector representation built from a one-class PHMM and the combined normalised exponential representation built from a binary PHMM. Both propositionalisations are carried out after one iteration of PHMM training. Random Forests are used as propositional learner. The table indicates statistical significance at the* 0.05 *level.*

dataset	AUC after first iteration for Random Forests	
	one-class PHMM	binary PHMM
	Fisher score	combined normalised exponential representation
enzyme_1	0.961	0.930 (−)
enzyme_2	0.962	0.935 (−)
enzyme_3	0.956	0.917 (−)
enzyme_4	0.979	0.955 (−)
enzyme_5	0.943	0.864 (−)
enzyme_6	0.974	0.957 (−)
enzyme_7	0.970	0.861 (−)
enzyme_8	0.986	0.931 (−)
enzyme_9	0.995	0.979 (−)
enzyme_10	0.992	0.967 (−)
enzyme_11	0.996	0.970 (−)
enzyme_12	0.972	0.911 (−)
enzyme_13	0.983	0.957 (−)
enzyme_14	0.995	0.994 (=)
enzyme_15	0.989	0.968 (−)
enzyme_16	0.988	0.974 (−)
euk_0	0.936	0.882 (−)
euk_1	0.990	0.966 (−)
euk_2	0.973	0.878 (−)
euk_3	0.956	0.926 (−)
pro_0	0.983	0.947 (−)
pro_1	0.961	0.910 (−)
pro_2	0.959	0.893 (−)

vectors derived from binary PHMMs to those from one-class PHMMs.

6.2.2 Runtime

This section compares the runtimes of propositional approaches built upon binary and one-class PHMMs after one training iteration with basic one-class PHMM and binary PHMM classification with and without sampling. As for all previous experiments with runtime, there has been no exclusive access to the experiment server and thus, results only indicate the true relationship in terms of runtime for different approaches.

Figure 6.6 shows the runtime for the *enzyme_1* and *enzyme_8*; the datasets with the largest and smallest number of positive instances. All approaches pre-

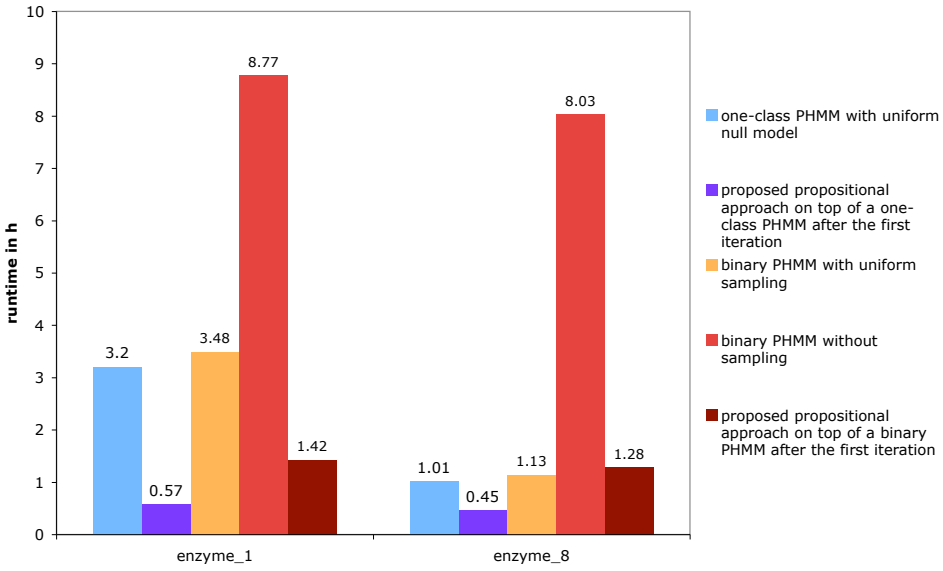


Figure 6.6. *Comparison of runtime for PHMM based and propositional approaches for enzyme_1 and enzyme_8. The proposed propositional approach is classification with Random Forests using the combined normalised exponential representation when a binary PHMM is propositionalised. In the one-class the corresponding combined exponential representation is used. The propositionalisation takes place after the first iteration of the Baum-Welch training algorithm. As opposed to the previous results, all approaches are only evaluated once.*

sented in the figure are only evaluated once. The propositional learners were bagged decision trees on the combined normalised exponential representation

for binary PHMMs and the combined exponential representation for one-class PHMMs.

As expected, propositionalising a one-class PHMM after one iteration leads to the fastest runtime. Additionally, binary PHMM based classification without sampling requires the longest runtime. For the other approaches the runtime depends on the size of the positive class. First of all, propositionalisations from binary PHMMs after one iteration need more runtime than their one-class PHMM counterparts. However, if the positive class is very small, as for example in *enzyme_8*, the fully trained one-class PHMM and even the binary PHMM with uniform sampling of the negative class are faster than propositionalisation of a binary PHMM after one iteration and subsequent bagging.

The situation is different for a larger positive class as the results for *enzyme_1* show. In this setting, the second fastest approach is the propositionalisation of a binary PHMM after one iteration. It is clearly faster than one-class PHMM based classification. The difference between one-class PHMM classification and classification using a binary PHMM with uniform sampling is around twelve minutes. Thus, sampling and training an additional PHMM is not much slower.

The results show that propositionalising a one-class PHMM after one training iteration is particularly advantageous in terms of runtime. Depending on the size of the positive class, the runtime of propositional approaches based on binary PHMMs trained for one iteration can be slightly higher than the one from a one-class PHMM. However, overall, the propositional runtime is still very competitive.

6.3 Summary

This chapter introduced propositionalisations of binary PHMMs and compared these approaches to binary PHMM classification from Chapter 4 and classification based on propositional models built from one-class PHMMs defined in Chapter 5. Therefore, it brings together the ideas and concepts presented in the last two chapters.

We introduced how to propositionalise binary PHMMs. The main idea is to create a fixed length feature vector for both PHMMs of the binary PHMM separately by applying the methods from the previous chapter. After this, the feature

vectors are concatenated.

The propositionalisation of a binary PHMM after its first training iteration in combination with Random Forests outperforms pure binary PHMM classification and its propositional one-class counterpart. This propositional approach uses the combined normalised exponential representation and achieves at least an equally good AUC value or statistically significantly improves on predictive performance. As in the one-class case, the propositional models built from binary PHMMs already predict competitively after the first iteration of the Baum-Welch training algorithm compared to the ones generated after PHMM training is completed. This is an important feature as it leads to a drastic decrease in runtime with a competitive AUC. The experimental results from this chapter and the previous chapter both support this finding. This is a key advantage of propositionalisation.

Additionally, this chapter reveals that propositionalisation based on the first iteration of one-class models in combination with Fisher score vectors and Random Forests lead consistently to significantly higher AUCs than the simple propositionalisations on binary PHMMs.

Chapter 7

Conclusions

The purpose of this work is to further advance the discrimination between and the detection of similar proteins, notably by changing from a one-class classification environment to a binary one. This thesis has shown how the information from the vast amount of negative data can be used to improve predictive performance significantly. Most surprisingly, this does not necessarily lead to an increase in runtime.

As Chapter 2 has determined, Hidden Markov Models are well suited to represent and detect similarity of proteins. Krogh et al. (1994) designed a model specialised to represent proteins. They are called Profile Hidden Markov Models (PHMMs) and form the basis of our investigations.

The following section discusses our contributions in detail. Section 7.2 identifies areas of future work that have been opened up in the course of the thesis. The final section of this thesis summarises the insights gained, lessons learnt and scientific positions validated through this research project.

7.1 Contributions

The introduction to this work has pointed out that the way any research question is asked and the way answers are found to these questions influences the scope of the research. This thesis has investigated its research questions slightly differently compared to previous work. As Section 2.6 reported, it is a common assumption that for sequence-based approaches the quality of a sequence alignment is crucial for the detection and classification of similarity in proteins. Baum-Welch training of a PHMM leads to a (locally) optimal alignment. How-

ever, our results allow us to identify ways to improve the discriminative power with sub-optimal alignments. The discovery of this surprising result has only been possible, because we altered the way to conduct experiments with PHMMs. Instead of working with fully trained PHMMs only, this thesis reports AUC results for all iterations of Baum-Welch training. It is this simple change in the evaluation that allows new contributions to the field of study.

This thesis explicitly recognises that a PHMM as known in the current literature is a one-class classifier trained on instances from the positive target class only. We pointed out that researchers like Jaakkola et al. (1999) have used this fact implicitly. Our work (Mutter et al., 2009) and independently the recent work of Bánhalmi et al. (2009) conclude that a PHMM is a one-class classifier. The validation that a PHMM is a one-class classifier is fundamental to the thesis as our efforts to improve on its predictive performance are based on transforming a one-class classifier to a binary one. The thesis exemplifies that a sound understanding of Machine Learning is essential to advance predictive techniques in Bioinformatics such as PHMM based homology detection and protein classification.

Additionally, we report statistical significance of our results. One of the criteria used to compare previous work in Chapter 2 has been their evaluation measure. Often the chosen evaluation measure is not necessarily well-suited to protein classification. The thesis computes the AUC of different approaches. Notwithstanding its deficiencies highlighted for example by Hand (2009), AUC is a prominent evaluation measure on imbalanced datasets. However, the question arises whether a difference in AUC is meaningful. Certainly, in protein classification and homology detection a meaningful difference in AUC should reflect a meaningful difference in the underlying biological reality of the problem. It is not in the scope of this thesis to investigate this notion of meaningfulness. However, statistical significance provides a necessary guidance to discuss the meaningfulness and the quality of the results. Previous work neglects statistical significance. Thus, it is a serious issue how to differentiate between real advances in this area of research from a Machine Learning point of view and improvements made by chance. To the best of our knowledge, there is only one other approach, namely Liao and Noble (2002), which uses AUC and reports statistical significance.

The other contributions of this thesis draw upon two important paradigm changes:

- From one-class to binary classification by using the vast amount of negative information
- From generative models to discriminative ones by means of propositionalisation.

These changes are also reflected in the thesis' structure as Figure 7.1 illustrates. The top of the triangle represents a one-class approach, whereas all other approaches are binary. The methods summarised in the left bottom corner of the triangle are of a discriminative nature. The two remaining vertices correspond to generative techniques. Chapter 4 implements the first paradigm change and

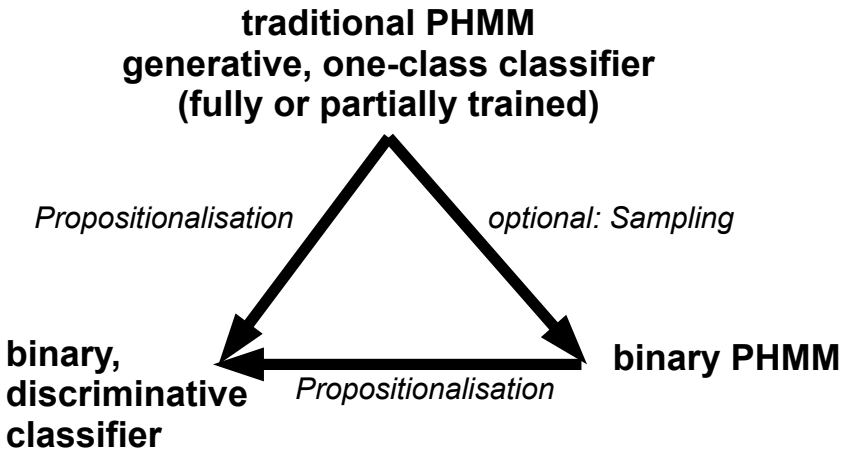


Figure 7.1. *The thesis' scheme revisited*

compares results from one-class and binary PHMMs. The second paradigm change forms the basis of the research and the contributions presented in Chapter 5 and 6. However, we take advantage of both paradigm changes in these chapters. Propositionalising a one-class PHMM does not only allow the use of discriminative classifiers, it also allows negative information to be included in the training of these binary, propositional classifiers. The idea to propositionalise one-class PHMMs has first been presented by Jaakkola et al. (1999). They motivate their approach solely discussing the advantages of a discriminative ap-

proach in a classification problem. They neglect the fact that the problem representation also changed from one-class to binary. By propositionalising a binary PHMM as presented in Chapter 6, its information becomes available to discriminative, binary, propositional classifiers. Liao and Noble (2002) acknowledge and exploit this fact as well.

7.1.1 One-class and binary Profile Hidden Markov Models

The main finding of Chapter 4 is based on the transformation from a one-class PHMM to a binary one. One-class PHMMs are trained on a small set of positive instances. Usually one-class classifiers are designed for problems characterised by an absence or insufficient amount of representative negative information at training time. However, in this area of research, there is a vast amount of negative information compared to a small amount of positively labelled sequences. Nonetheless, a one-class PHMM trained on the positive class only has been established as a standard approach. Liao and Noble (2002) motivated in their work the use of an alternative to PHMMs with the PHMM's inability to deal with negative data during training. This thesis addresses this problem by training a separate one-class PHMM on the negative instances. This includes the information from the negative data in a way that is also sensitive to the imbalanced nature of the dataset. The discriminative power based on AUC of these so-called binary PHMMs is statistically significantly better than the one from the standard one-class approach. Thus, this thesis has introduced binary PHMMs and shown that their ability to discriminate is superior to standard one-class PHMMs. This finding validates the common assumption that when negative information is available that forms a characteristic representation of future negatively labelled instances, binary approaches outperform solely one-class based ones.

The straightforward application of the proposed transformation from a one-class PHMM to a binary one leads to an increase in runtime. Instead of training one PHMM on a small number of instances, our approach additionally trains a second PHMM on a large set of negatively labelled sequences. Consequently, in the presence of a vast amount of negative information, this thesis implements undersampling of the negative class. Chapter 4 introduces four different strategies for undersampling. All of them sample as many negative instances as there are positives ones in the dataset. Three of the four strategies first score

all negative sequences using the positive one-class PHMM and base their choice of sample on these scores. In one setting only the highest scoring negative sequences, i.e. the ones that are most similar to the positive ones, are used to train the negative one-class PHMM. On the contrary, the second strategy incorporates only information from the most dissimilar negative sequences. The third strategy ensures that negative sequences are chosen stratified from the complete range of scores. The fourth and simplest strategy does not score the negative sequences at all and chooses them completely at random. This strategy is the fastest of the four due to the lack of scoring. It is worth noting, however, that for all four strategies the negative instances are re-sampled in each fold in each iteration of the 10-fold cross validation. As our experiments reveal, the simple and the stratified strategies significantly outperform the other two. Notably, the simple strategy performs as well as the stratified one and is considerably faster. Having said this, the thesis shows empirically that the simple sampling strategy achieves a competitive AUC compared to training a binary PHMM on complete data. And compared to other sampling strategies and using the complete negative information, it is much faster.

The final contribution of Chapter 4 concerns research on null models. Null models are a crucial part in PHMM based modelling as they allow the introduction of length independent odds scores. There has been research to show that good null model choice is important for the performance of PHMMs (Barrett et al., 1997; Karplus et al., 1998, 2005). This thesis has experimented with different null models for one-class PHMMs. The two most successful in terms of AUC have been the uniform null model and the reverse sequence null model introduced by Karplus et al. (1998, 2005). Again, the uniform null models follows a simple strategy by treating each occurrence of an amino acid equally likely. In most of our datasets, it outperforms the reverse sequence null model.

There are actually different ways to define a null model. Barrett et al. (1997) define a null model from a Machine Learning point of view as a simpler model that represents the null hypothesis and therefore, the universe of sequences. Gotelli (2001), however, sees the null model as a model that deliberately excludes the mechanism being tested. His definition comes from a biological, ecological point of view. Both definitions look at null models from a slightly different angle. Our work has proven that a PHMM trained on the negative in-

stances acts like a null model for the positive one-class PHMM. From a Machine Learning point of view, the negative one-class PHMM is strictly speaking not a null model even though it acts exactly like one because it is a learnt model that does not represent the universe of sequences. However, following Gotelli's point of view it clearly excludes the mechanism being tested. This exclusion might not be perfect depending on incorrect class labels. Thus, care has to be taken whether or not to actually call a negative one-class PHMM a null model for its positive counterpart depending on the underlying null model definition. However, it clearly acts as one.

To summarise, our work proves that a negative one-class PHMM acts like a null model for the model trained on the positive instances only. Additionally, we empirically show that the combination of a positive and a negative one-class PHMM outperforms commonly used null models combined with positive one-class PHMMs. This provides an alternative interpretation of the main finding of this chapter in terms of null models.

7.1.2 Propositionalisation

Chapter 5 presents the propositionalisation of one-class PHMMs and the subsequent use of the derived propositional representation within a binary, discriminative classifier. It compares the generative, probabilistic, graphical one-class PHMM to a propositional, binary, discriminative classifier. The idea to propositionalise a PHMM is not new. Jaakkola et al. (1999) have introduced the Fisher score vector as a fixed length feature vector representation derived from PHMMs. The basic assumption of their work is that a propositional representation allows the use of discriminative techniques on top of a powerful generative model. They validate the hypothesis that discriminative techniques are favourable over generative ones in a discriminative task for the areas of protein classification and homology detection. The experiments presented in the thesis fully support their findings. Aside from that, our thesis adds two other important reasons to propositionalise. Research in Machine Learning has focussed on propositional learners and consequently, they are highly optimised. Furthermore, propositionalisation transforms the problem not only into a discriminative one, but also into a binary one. The generative PHMM is a one-class classifier whereas all propositional learners are binary.

Our work improves on Jaakkola et al. in different ways. By normalising Jaakkola et al.'s Fisher score vector, we report better AUC results for Support Vector Machines. Additionally, Jaakkola et al. test their approach on Support Vector Machines with RBF kernel. We extend the set of discriminative learners to bagged decision trees and Random Forests.

Our thesis validates the notion that propositionalisation outperforms a pure one-class PHMM based approach. The most surprising and fundamental finding is that the AUCs of the propositional approaches after the first iteration of PHMM training are highly competitive and in some datasets even better than propositionalising after full convergence. This result is very important as it allows implementing a strategy that is faster in runtime compared to a standard one-class PHMM approach but significantly better in terms of AUC. Consequently, this chapter suggests to train a one-class PHMM for one iteration and to propositionalise it subsequently. Additionally, this also constitutes an improvement compared to Jaakkola et al. as they propositionalise fully converged one-class PHMMs.

The extension of our propositionalisations for binary PHMMs is shown in Chapter 6. The proposed idea is to simply concatenate the propositionalisations from both one-class PHMMs into one feature vector. This idea takes advantage of the flexibility of a feature vector approach. It is motivated from the previous chapter where we put together propositional representations derived from different novel methods into one feature vector. The binary PHMM based propositionalisations are competitive in terms of AUC especially at the start of training compared to binary PHMM classification. They share this highly favourable behaviour with the propositionalisations for one-class PHMMs. Additionally, the simple propositionalisations are also better than propositionalisations originating from the one-class case. However, compared to the observed boost in AUC when going from a one-class PHMM to a binary one, the AUC difference between one-class based propositionalisations and binary ones is much smaller. In addition, the best performing method overall is derived from a propositionalisation based on a one-class PHMM. It uses Random Forests on a Fisher score vector representation derived from a one-class PHMM when training is stopped after one iteration.

7.2 Future work

The idea to concatenate fixed-length feature vectors of different propositionalisations can be taken a step further. We can put together the resulting vectors from all possible methods to propositionalise and submit the single resulting vector to feature selection. Not only might the final feature set lead to a statistically significant improvement in terms of AUC, it might also allow us to draw conclusions about which features are important. Thus, the results might allow interesting insights into which parts of the underlying PHMM and the proteins are important to decide about class membership. Given that this thesis has shown empirically that propositionalising PHMMs after the first iteration is highly favourable, the next step in future work will be to investigate the reasons for this behaviour. This is of particular interest as it is counter-intuitive and contrary to the common assumption that a good model of similarity is required. In the area of one-step learning Hinton (2002) successfully trained a Markov Chain for a few steps only instead of full convergence. However, this has been achieved by changing the training algorithm itself to approximately minimise a function that is called *contrastive divergence*. The technique has been successfully applied to Conditional Random Fields in the area of image labelling (He et al., 2004) and products of Hidden Markov Models (Brown and Hinton, 2000). This thesis trains PHMMs for one step by means of standard Baum-Welch training. The resulting models are propositionalised. Thus, it is a different form of one-step learning.

Additionally, as far as propositionalisation is concerned, there are at least two more interesting and potentially promising avenues for future research. First, the Fisher score vector representation can be used on binary PHMMs by following the concatenation idea we used for our own propositionalisation methods. Second, researchers might want to experiment with a larger set of discriminative classifiers.

Certainly, future work is not restricted to the area of propositionalisation and binary, discriminative, propositional classification. Other areas of future work include one-class classification and the change of the problem representation itself.

This thesis has established propositionalisation as a tool to change from a generative to a discriminative and from a one-class to a binary setting. However,

it is possible to propositionalise the positive instances only and therefore, train a one-class, discriminative classifier. Initial experiments with a one-class Support Vector Machine (Chang and Lin, 2001) have been conducted but did not lead to an improvement in AUC. This is consistent with the common assumption mentioned earlier in this chapter about the relationship between one-class and binary classification. However, a broad and detailed study into one-class classifiers will shed more light into the relationship between discriminative one-class and binary classification in the area of protein classification and homology detection.

Instead of using a discriminative one-class classifier, the existing generative one-class PHMM can be optimised in future work. This thesis restricts the number of columns in a PHMM to 35. The advantages are that the PHMM's calculations are sufficiently fast and that the risk of overfitting is significantly low. However, other measures to avoid overfitting like simulated annealing or Dirichlet mixture distributions or substitution matrices do exist. If implemented, they might allow testing of PHMMs with considerably more columns. However, an increase in the number of columns leads to an increase in runtime. Thus, the combination of an increase in columns and the strategy to stop training after one iteration of the Baum-Welch algorithm and to propositionalise subsequently is an avenue of research worth exploring.

In addition, PHMMs with more columns will be able to generate new, artificial sequences that are closer length-wise to the original ones. We have tested the generation of artificial sequences from a positive one-class PHMM and trained the additional PHMM on these artificial sequences instead of the negative ones. This constitutes a pure one-class approach that takes advantage of the generative nature of a PHMM. However, the major problem encountered was the fact that small PHMMs create most likely small sequences that are not characteristic in their length compared to the positive (and negative) sequences. It is no surprise that these approaches performed extremely poorly and were considered not worth exploring further. Succinctly put, a one-class PHMM with more columns might be able to generate more characteristic sequences. Future work will have to test this hypothesis and verify whether a one-class approach with artificial sequences is worth pursuing.

Our work stresses the fact that a well-suited problem representation is neces-

sary. This thesis has looked at binary problems and transformed multi-class classification sets to binary ones. We have argued that this representation suits the problem better. However, it is possible to extend our approach to a multi-class setting. We learn a one-class PHMM for each class. A binary PHMM combines one of these models with a model trained on all other instances. In a multi-class setting, we can combine all the one-class models and thus construct a multi-class PHMM. The advantage is that we do not need to learn an additional PHMM on the instances that do not belong to the class under consideration. Thus, our idea to combine one-class classifiers can be extended to the multi-class case.

7.3 Final conclusion

This thesis has shown that a binary transformation of a one-class PHMM improves its discriminative power significantly. For applications that are sensitive in terms of runtime, a simple and fast strategy to undersample the vast amount of newly added negative information has been presented. Our work has empirically established that this sampling method is competitive in terms of AUC. Foremost, this thesis reveals the power of propositionalisation. Compared to standard one-class PHMM classification, the best way to improve significantly on AUC and runtime is to train PHMMs for one iteration only, to propositionalise them subsequently and use the derived representation with binary, discriminative, propositional classifiers. These are standard, binary, Machine Learning classifiers.

Our work reinforces the potential of simple ideas, extensions and transformations to solve practical problems and in our case, advance the classification based on sequence similarity in proteins.

There is one principle of utmost importance that resonates throughout the entire thesis. We need a sound understanding of both Bioinformatics and Machine Learning to further advance research in this area. It starts with understanding the problem and expands to problem representation and its consequences, experimental design and evaluation. This thesis first and foremost validates this fact and builds a sound and solid bridge for the gap between Bioinformatics and Machine Learning.

Appendix A

UniProt identifiers for the *enzyme* dataset

Table A.1. *UniProt identifiers for dataset enzyme_1*

O00097	O45687	P00325	P00328	P00331	P00334	P06525
Q6LCE4	P07161	P07246	P08319	P09370	P10847	P12854
P14139	P14674	P17648	Q2R8Z5	P21518	P22246	P23237
P23361	P25139	P25406	P25988	P27581	P28469	P33010
P38113	P40394	P41682	P42328	P48585	P48814	P49383
P49645	P51550	P54202	P80338	Q00669	Q00672	Q05114
Q09009	Q17334	Q64413	O07399	P0A2C9	P33207	P50941
P70720	P73826	O34268	P50169	Q27979	P50842	P11759
P29781	P55463	P27867	Q00796	Q59787	P15428	P16232
P50172	P51975	Q29608	P35270	P45856	P77851	O51544
B0B960	P0CD77	P37417	P95837	P19337	O28578	O67619
O65992	P42957	Q45421	O75828	P47844	Q29529	O26337
P95872	O24562	P30360	P31657	P42734	Q02971	Q40976
O57380	P25984	P50578	P28475	P08793	P91711	O50316
P0ADG9	P20839	P24547	P39567	P47996	P50095	P50098
Q07152	P65167	P07943	P21300	P45377	P14720	P51103
P51106	P51110	O05973	O60701	O86422	P76373	P0C0F5
P0C0F4	Q57346	O26327	O34651	P10370	P28736	Q02136
P50163	P25415	P49365	P15770	P46240	Q44607	P50166
Q12634	P0A3N0	P00337	P00340	P00343	P04034	P06151
P10655	P13715	P16115	P19858	P20619	P29038	P42119

P42123	P50934	P78007	Q07251	Q27888	Q60009	Q95028
P26298	P51011	P13443	P08499	P31116	P46806	P56429
P29147	P29266	O26662	P04035	P12684	P16237	P29057
P48020	P51639	Q01559	Q12577	Q58116	O08756	P34439
Q61425	P23238	P50204	O08349	O59028	P61889	P11708
P17783	P25077	P37228	P44427	P48364	B3QPY9	P0C890
Q04820	Q42972	Q59202	P26616	P45868	O30807	P37225
P12628	P22178	P36444	P43279	P51615	O43837	P28834
P50213	Q93353	O14254	O75874	P16100	P39126	P41562
P50214	P50217	P54071	P80046	Q58991	O13287	P00349
P14062	P37754	P41570	P41576	P52208	P70718	P96789
P12310	P39483	P40288	O00091	O24357	O83491	P11410
P11413	P0AC53	P37986	P44311	P48848	P54996	P77809
Q27464	Q43727	Q04520	P19871	P39160	P32816	P38945
P14061	P51656	P51659	P70385	Q62904	P50199	O57656
P08507	P21695	P0A6T0	P52425	Q00055	Q27567	Q44472
P21528	P52426	O27441	O59930	O94114	P05644	P08791
P18869	P24098	P29696	P34733	P41019	P43860	P54354
P87186	P94631	P96197	Q02143	Q56268	O27491	O33114
O67289	P05989	P37253	P78827	Q02138	Q57179	Q59818
P39849	O34296	P76251	O67555	O08651	O33116	P0A9T3
P40510	P87228	P09437	P32891	Q00922	P47195	P81156
P22637	Q01745	P54223	P65419	P33940	P13650	Q07982
P0A9C2	P18158	P43304	P52111	P90795	O05542	P15279
P28036	Q44002					

Table A.2. *UniProt identifiers for dataset enzyme_2*

P45382	P78870	P77580	O26890	P0A1F8	O67716	P10539
P23247	P30903	P41399	P41404	P0A542	Q51344	Q55512
Q56734	Q59291	P64178	O13507	O27090	O34425	O43026
O59494	O67161	P04406	P00356	P00358	P00360	P00362
P04796	P04970	P07486	P08439	P09124	P09316	P10097

P12858	P12860	P16858	P17329	P17331	P17729	P17819
P19089	P19315	P61880	P20445	P22513	P25856	P25858
P26517	P26519	P26521	P27726	P29272	P30724	P32636
P32638	P32810	P34783	P34917	P34919	P34921	P34923
P35143	P44304	P46713	P47543	P49433	P50321	P50362
P51009	P52987	P53430	P54118	P54270	P56649	P78958
P87197	Q00584	Q01077	Q01597	Q01982	Q07234	Q12552
Q27890	Q41595	Q0J8A4	B0B879	P0CE13	Q58546	Q59800
Q64467	Q92243	O32507	P38947	Q55585	P06131	P24183
P33160	P46448	Q07103	Q50570	Q60316	P37685	P51650
P42412	Q02252	Q07536	P00352	P08157	P12693	P13601
P20000	P0C6D7	P24549	P30837	P30840	P33008	P40047
P41751	P46329	P46368	P47738	P48644	P51648	P54115
Q25417	Q28399	P07702	P50113	O08318	P23715	P54895
P54899	P96136	Q59279	O67166	P07004	P39821	P0C1E0
P0C1E1	P54885	P54903	P74935	P96489	P11883	P43353
P47771	P48448	P08639	P29236	P80505	O24174	P17445
P42757	P56533	P77674	P81406	Q43272	P07003	Q06278
Q54970	P45851	O66112	P0AFG8	P11177	P21873	P21881
P26267	P26284	P32473	P35487	P45119	P47516	P51266
P52899	P52902	P52904	P75391	Q09171	Q10504	Q59097
P0AFG3	P20967	P45303	Q02218	P09060	P11178	P21839
P37940	P50136	O05651	O27772	O58415	O73986	P56815
Q51803	Q51805	Q56317	Q57715	Q57717	O27113	O29779
O29782	Q57956	O27743	P26693	P27989	Q49161	Q49163
Q50538	Q57617	O27002	O31112	P61938	Q58571	

Table A.3. *UniProt identifiers for dataset enzyme_3*

P43901	P20049	P08088	Q12882	Q18164	Q28007	Q28943
P42330	Q04828	P46844	P53004	O26891	O29353	O67061
O84369	O86836	P24703	P38103	P40110	P42976	P45153
P46829	P72024	P72642	Q52419	Q57865	P15047	P39071

Q56632	O30847	O84992	O87612	P27137	P94135	Q45072
P13653	P17652	P21218	P26156	P26163	P26180	P26237
P26238	P28372	P29683	P36208	P36437	P37846	P48099
P48100	P51188	P51278	P54208	P56302	P56303	Q00864
Q04607	Q95666	P42593	Q16698	Q64591	O27083	P21920
P72711	Q10680	Q53139	P07772	P23102	O07400	O24990
O67505	P16657	P42829	P44432	P0A5Y6	P54616	P73016
P80030	Q05069	P65908	O27281	O29513	O66461	P0A7E1
P25468	P25996	P28272	P28294	P32747	P32748	P45477
P46539	P46727	A2RJT9	P54322	P74782	Q47741	Q58070
Q63707	O75845	O88822	P11353	P33771	P35055	P36552
P36553	P43898	P72848	Q42840	Q42946	O24163	O24164
P0A5A7	P0ACB4	P32397	P40012	P50336	P51175	P55826
P56601	Q12737	P05335	P06598	P07872	P08790	P11356
P13711	P34355	Q15067	O42772	P21801	P21911	P21912
P21914	P31039	P31040	P47052	P48932	P48933	P80477
P80480	Q00711	Q09508	Q09545	O06913	O06914	P00363
P0AC50	P07014	P08065	P08066	P0AC41	P17596	P20921
P20922	P31038	P44893	P44894	P51053	P51054	P64174
Q10761	Q59661	Q59662	P12007	P26440	P34275	P07670
P15650	P28330	P51174	P79274	Q51697	Q51698	P15651
P16219	Q06319	Q07417	P08503	P11310	P41367	P45952
Q22347	Q04616	P18405	P24008	P31213	P31214	Q28891
Q28892	P31210	P51857	Q60759	Q92947		

Table A.4. *UniProt identifiers for dataset enzyme_4*

P17557	P30234	Q08352	P09831	P09832	P39812	Q05755
Q05756	Q03460	P04964	P24295	P27346	P28997	P33327
P39633	P41755	P0C934	P22823	P23307	O52310	O59650
O74024	P00366	P00367	P10860	P26443	P49448	P52596
P54385	P80053	P80319	P96110	Q47951	Q53199	Q56304
P00369	P00370	P07262	P14657	P15111	P28724	P29051

P29507	P31026	P39708	P43793	P54386	P54387	P55990
P94316	P94598	P95544	P13154	P54531	P31228	Q99489
P16636	P28300	P28301	Q05063	P65499	P10902	P38032
P74562	Q51363	O93364	P23623	O35078	P00371	P14920
P18894	P22942	P80324	Q19564	Q99042	P19643	P21396
P21397	P21398	P27338	P49253	P65682	O33065	P21159
P0AFI8	P38075	P44909	P74211	O46406	O70423	O75106
P12807	P19801	P36633	P46881	P46883	Q07121	Q07123
Q12556	Q16853	Q29437	Q43077	Q59118	O49850	O49954
P15505	P23378	P26969	P33195	P49095	P49361	P49362
P54377	Q09785	Q50601	P23225	P51375	P55037	P55038
Q06434	P0A6J5	P00372	P22619	P22641	P23006	P29894
Q49124	Q50420	Q59542	Q59543			

Table A.5. *UniProt identifiers for dataset enzyme_5*

P07275	P30038	P39634	P78568	P55818	Q02046	P00386
Q44524	O25773	O66553	P0A9L9	P17817	P22008	P22350
P27771	P32263	P43869	P0C1E4	P0C1E5	P46725	P52053
P54893	P54904	P74572	Q04708	Q12641	Q12740	Q20848
O74927	O80585	P42898	P46151	P53128	Q17693	P15244
Q44297	O62583	P00374	P00375	P00376	P00377	P0ABQ6
P00380	P00381	P00382	P00383	P00384	P04174	P04382
P05794	P07807	P09503	P0A017	P11045	P11731	P12833
P13955	P16184	P17719	P22573	P22906	P27421	P27422
P27498	P28019	P31074	P31500	P36591	P43791	P47470
P78028	P78218	P95524	Q07801	Q54277	Q54801	P0ABQ8
Q59397	Q59408	Q59487	P0C0P1	Q93341	O75891	P28037
P38997	P38998	P43065	Q09694	O87386	O87388	P23342
P40854	P40859	P40873	P40874	P40875	Q46336	Q46338
O64411	P08159	P30986	P87111	P94132	Q08822	Q11190
Q48303	Q63342	P16099	O29544	P55300	P94951	Q50501
Q58441						

Table A.6. *UniProt identifiers for dataset enzyme_6*

P07001	P41077	P51995	P00387	P36060	P00389	P16603
P37040	P50126	P00390	P27456	P42770	P48638	P48641
Q43621	O62768	O84101	P38816	P43788	P50971	P52214
P75531	P94284	Q17745	Q92375	P39040	O03060	O03172
O03175	O03203	O03206	O03850	P69235	O21070	O21233
O21325	O21333	O21336	O21405	O21408	O21514	O21798
O47430	O47492	O47498	O53307	O63850	O66842	O68853
O78680	O78688	O78694	O78697	O78701	O78704	O78707
O78710	O78714	O78748	O78755	O79408	O79411	O79421
O79427	O79435	O79438	O79677	O79874	O79881	O84970
O85274	O99823	O99826	O99978	P03888	P03891	P03894
P03897	P03900	P03903	P03909	P03912	P03916	P03919
P03922	P03925	P04540	P05507	P05510	P06253	P0CC99
P06259	P06262	P06265	P06410	P07706	P07709	P08740
P09045	P11628	P11631	P11658	P11991	P0C386	P0C322
P0C328	P0C337	P12199	P12771	P12774	P12777	P15550
P15553	P15577	P15580	P15584	P15956	P15959	P16673
P60160	P18931	P18934	P18938	P18941	P19044	P19050
P20113	P20686	P21301	P24873	P24877	P24884	P24887
P24895	P24969	P24972	P24975	P24978	P24982	P24997
P25707	P26289	P26522	P26525	P26847	P26850	P27572
P29801	P29915	P29918	P29921	P29924	P30826	P0AFE8
P32421	P33509	P33512	P0AFD0	P33602	P0AFE3	P0AFF0
P0A1Y8	P34192	P34195	P34847	P34850	P34853	P34856
P34859	P38599	P38602	P41296	P41299	P41304	P41307
P41315	P42032	P43191	P43194	P43197	P43200	P43203
P43206	P0CD59	P46722	P48176	P48653	P48656	P48897
P48900	P48903	P48907	P48910	P48913	P48916	P48919
P48922	P48925	P48928	P48931	P50367	P50940	P50975
P51097	P51100	P52765	P55780	P55783	P56752	P56755
P56896	P56908	P56911	P56914	P92475	P92483	P92486

P92659	P92667	P92670	P92697	P92700	P93401	P65561
P65573	P65575	Q00236	Q00244	Q00540	Q00543	Q00570
Q01562	Q04050	Q31849	Q32238	Q33635	Q33821	Q34050
Q34573	Q34947	Q34950	Q35100	Q35535	Q35542	P61796
Q35813	Q36346	Q36424	Q36428	Q36457	Q36460	Q36836
Q37312	Q37371	Q37375	Q37381	Q37546	Q37603	Q37626
Q37680	Q37710	Q37714	Q37809	Q44241	Q56218	Q56221
Q56224	Q56227	Q60010	Q95704	Q95710	Q95891	Q95915
Q95918	Q96007	Q96067	Q96070	Q96186	P28304	P43903
Q28452	P11605	P17569	P27967	P39865	P39868	P43101
P27783	P22945	P39863	P49050	P22944	P42435	P60560
O87948	O85762	P41816	Q03558	P05982	Q64669	P37061
P75389	Q60049	P24232	Q03331			

Table A.7. *UniProt identifiers for dataset enzyme_7*

O04420	O32141	O74409	P04670	P09118	P11645	P16163
P16164	P22673	P23194	P25689	P33282	P34798	P34799
P53763	P78609	Q00511	Q45697	Q50925	P05314	P39661
Q51879	P25006	P38501	Q01537	Q06006	Q53239	Q60214
P09152	P11349	P0AF26	P0AF32	P19317	P19318	P19319
P33937	P39185	P39458	P42175	P42176	P42178	P42434
P73448	P81186	Q06457	Q53176	Q56350	O54235	O67422
P0AEZ2	P45208	P71319	P19573	P94127	Q01710	Q51705
Q59105	Q59746	O06844	O50651	Q51662	Q52527	Q59646
Q59647						

Table A.8. *UniProt identifiers for dataset enzyme_8*

P17846	P38038	P38039	P39692	P52673	P52674	Q09878
O00087	O08749	O18480	O50286	O50311	O84561	P0A9P3
P09063	P09622	P09623	P09624	P11959	P14218	P21880

P31023	P31046	P31052	P43784	P47513	P49819	P50970
P52992	P54533	P75393	P90597	P95596	Q04829	Q04933
P0A0E7	P07850	P51687	Q07116	P30008	O33998	P45573
P45574	P45575	Q59109	Q59110	O05927	O06737	P17853
P17854	P52672	P56859	P56860	P56891	P65668	P72794
P94498	Q10270	Q55309				

Table A.9. *UniProt identifiers for dataset enzyme_9*

O03167	O03198	O03848	O13082	O21327	O21399	O21403
P69215	O47491	O47667	O47669	O47671	O47673	O47675
O47677	O47679	O47681	O47686	O47688	O47690	O47692
O47694	O47696	O47698	O47700	O47702	O47705	O47708
O47710	O48316	O48374	O54069	O74471	O78682	O78750
O79404	O79417	O79433	O79673	O79876	O99255	O99819
P00395	P00397	P00399	P00401	P00403	P00405	P00407
P00409	P00411	P00413	P00415	P00417	P00419	P00421
P00423	P00425	P00427	P00429	P03945	P04038	P04371
P04373	P05490	P05502	P05505	P06030	P07255	P07471
P07657	P08306	P68539	P08743	P08745	P08749	P09669
P10175	P10606	P10888	P11947	P11950	P12074	P12700
P12702	P12787	P13182	P13184	P14058	P14544	P14546
P14574	P14578	P14852	P14854	P15545	P15952	P15954
P16262	P18943	P18945	P19536	P20374	P20386	P20609
P20674	P20682	P20684	P24010	P24012	P24310	P24794
P24881	P24891	P24894	P24985	P24987	P24989	P25002
P25312	P26455	P26457	P26857	P27168	P29505	P29856
P29860	P67799	P29870	P29872	P29874	P29876	P29878
P29880	P30815	P32799	P33504	P33508	P33518	P34189
P34838	P34842	P35171	P38596	P41293	P41295	P41311
P41775	P43024	P43370	P43372	P43374	P43376	P47918
P48171	P48659	P48661	P48772	P48867	P48869	P48871
P48873	P48887	P48889	P48891	P50253	P50268	P50666

P50672	P68294	P50676	P50678	P50680	P50684	P50686
P50688	P50690	P50692	P55777	P56392	P79010	P80439
P80441	P92478	P92514	P92662	P92692	P92696	P98000
P98002	P98012	P98020	P98023	P98025	P98027	P98031
P98033	P98035	P98037	P98039	P98042	P98044	P98047
P98049	P98053	P98055	P98057	Q00527	Q00529	Q01555
Q02211	Q02221	Q02766	Q03227	Q03439	Q03736	Q04441
Q04452	Q05572	Q06474	P60621	Q08855	P63854	Q20779
Q33824	Q34941	Q35101	Q35539	Q36309	Q36452	Q36675
Q36837	Q36952	Q37369	Q37374	Q37416	Q37430	Q37472
Q37548	Q37604	Q37620	Q37677	Q37684	Q37705	Q37718
Q42841	Q94514	Q95840	Q95914	Q96065	Q96133	Q96190
P24474	Q51700					

Table A.10. *UniProt identifiers for dataset enzyme_10*

O01374	O14949	O14957	O31214	O60044	P00126	P00127
P00128	P00130	P05417	P07056	P07256	P07257	P07552
P07919	P08067	P08525	P13271	P13272	P14927	P16536
P22289	P22695	P23004	P31800	P31930	P32551	P37299
P37841	P43264	P43265	P43266	P46269	P47985	P48502
P48503	P48504	P48505	P49345	P49346	P50523	P51130
P51132	P51133	P51134	P51135	P78761	P81380	Q02762
Q09154	P43309	P43310	P43311	Q00024	Q06215	Q08296
Q08304	Q08305	Q08306	Q08307	P06811	P17489	P56193
Q01679	Q02075	Q02079	Q02081	Q02497	Q03966	Q12541
Q12542	Q12718	Q12719	Q12729	Q12739	Q99044	Q99046
Q99049	Q99055	P14133	P24792	P37064	Q00624	Q40588
P08980	P14698	P26290	P26292	P30361	P49728	P70758
Q02585	Q46136					

Table A.11. *UniProt identifiers for dataset enzyme_11*

O31158	O31168	P04963	P25026	P49053	P49323	Q55921
P48534	P19136	P00431	P14532	P37197	O13289	O52762
O61235	O68146	P04040	P04762	P06115	P07145	P07820
P11934	P12365	P13029	P15202	P17336	P17598	P17750
P18122	P18123	P21179	P24168	P25819	P25890	P26901
P29422	Q0E4K1	P29756	P30263	P30264	P30567	P32290
P37743	P42234	P42321	P44390	P45737	P45739	P46817
P0A323	P48350	P48351	P48352	P49284	P49316	P49317
P49319	P50979	P55303	P55304	P55305	P55306	P55307
P55308	P55310	P55311	P55312	P55313	P77872	P78574
P78619	P81138	P95539	P95631	Q01297	Q04657	Q08129
Q27710	Q42547	Q43206	Q59296	Q59337	Q59602	Q59635
Q59714	Q64405	Q92405	Q96528	O35244	O77834	P00433
P00434	P05164	P11247	P11678	P11965	P15004	P15232
P17179	P17180	P22079	P22195	P22196	P24101	P27337
P28313	P28314	P30041	P37834	A2YPX3	Q0D3N0	P49290
P80025	Q01603	Q02200	Q05855	P07202	P09933	P14650
P35419	O02621	O18994	O23968	O23970	O32770	O46607
O59858	O62327	O75715	P04041	P07203	P11352	P11909
P18283	P21765	P22352	P28714	P30708	P30710	P67878
P35666	P36014	P36968	P36969	P37141	P38143	P40581
P46412	P52032	P52033	P74250	Q00277	Q64625	Q95003

Table A.12. *UniProt identifiers for dataset enzyme_12*

O33950	O67987	P0A396	P07773	P11451	P27098	P31019
P20351	P48775	P48776	Q09474	P08170	P09186	P09439
P09918	P14856	P27480	P29114	P29250	P37831	P38414
P38415	P38416	P38417	P38419	Q06327	Q05353	P06622
P08127	P17262	P17295	P17296	P31003	Q04285	Q53034
P21816	Q16878	O23920	O42764	O48604	P49429	P80064
P93836	Q00415	Q02110	Q22633	Q27203	P69053	Q5ZT84

P00436	P00437	P15109	P15110	P20371	P20372	P16469
P18054	P39654	P39655	P55249	Q02759	Q01284	Q12723
P12530	P16050	P12527	P48999	P51399	P08695	P11122
P17297	P47228	P47231	P47233	P14902	P28776	O09173
Q00667	Q93099	P46952	P46953	P22635	P22636	P11295
P0A3V2	P0A3V3	P06617	P25017	Q04564	Q09109	P21795
P27652	P17554	P08659	P13129	Q01158		

Table A.13. *UniProt identifiers for dataset enzyme_13*

O75936	Q19000	Q12797	P13674	P54001	Q60716	O60568
P24802	Q20679	P28038	Q05963	Q06942	P07770	P0A111
Q51494	P0C618	Q07944	Q05182	P23262	O24312	P37114
P48522	Q04468	Q43033	Q43240	P17549	P18125	P46634
Q64505	P20586	P27138	P11987	P22868	P27353	P27355
Q08477	O54705	O61309	P29473	P29475	P29477	P70313
Q26240	Q28969	P42535	P19729	P19731	P19733	P31020
P16549	P17636	P31513	P36365	P36367	P49326	P97501
Q01740	Q28505	O09158	O16805	O18809	O18992	O35293
O42231	O42457	O46420	O54749	O55071	O62671	O73686
O93299	P00178	P00181	P00184	P00186	P04167	P04799
P05176	P05178	P05180	P05182	P08684	P08683	P10610
P10613	P10615	P10633	P10635	P11509	P11511	P11711
P11712	P12789	P12791	P12939	P13108	P13584	P14762
P15128	P15149	P16141	P17666	P19225	P20812	P20814
P20852	P21595	P24453	P24455	P24457	P24460	P24462
P24464	P24903	P29981	P30608	P30610	P33260	P33262
P33264	P33268	P33270	P33274	P43083	P49602	P51538
P51589	P51869	P51871	P56590	P56592	P56654	P56656
P70091	P79304	P79401	P79690	P79760	P93846	Q00557
Q05047	Q05555	Q06367	Q09736	Q12586	Q12589	Q16678
Q16850	Q27593	Q27606	Q27712	Q27902	Q29488	Q29510
Q29624	Q64391	Q64406	Q64417	Q64458	Q64462	Q64481

Q64654	Q64680	Q92087	Q92090	Q92100	Q92110	Q92112
Q92148	P07739	P09140	P12744	P19839	P19907	P23146
P24113	P29239	P08516	P14579	P14581	P20817	P15150
P15538	P19099	P30100	P97720	Q29552	Q64658	P00189
P10612	P79153	Q07217	P04176	P30967	P90925	O42091
P07101	P17289	P24529	P09810	P17532	P70080	P09172
Q05754	P08478	P12890	P19021	O42713	P06845	P11344
P33180	P55023	P55025	Q04604	O02768	O62698	P05979
P23219	P35354	P70682	Q05769	P00191	P08686	P15540
O19998	O70453	O78497	P09601	P14901	P30519	P43242
P71119	P74133	P07308	P13516	Q64420	P22243	P28645
P32061	P32063	Q01753	Q40731	Q42770	Q43593	O48651
O65403	O65726	P32476	P52020	Q92206	O73853	P05185
P12394	P27786	P70085	Q29497	Q92113		

Table A.14. *UniProt identifiers for dataset enzyme_14*

O04997	O09164	O12933	O13401	O15905	O22373	O30563
O30826	O42724	O46412	O49044	O49066	O51917	P0A4J5
O59924	O67470	O86165	O93724	P00441	P00442	P00443
P61852	P00445	P00446	P00448	P00449	P03946	P04178
P04179	P07505	P07509	P07632	P08228	P08294	P0AGD6
P09212	P09213	P61503	P09223	P09224	P09670	P09671
P09678	P09737	P09738	P10791	P10792	P11418	P11796
P11964	P13367	P13926	P14830	P14831	P15107	P15453
P17550	P17670	P18655	P18868	P19665	P19666	P19685
P20379	P23345	P23346	P23744	P24669	P24702	P24704
P24705	P24706	P25842	P27082	P27084	P28755	Q0DRV6
P28757	P28758	P28759	P28763	P28764	P31108	P31161
P33431	P34107	P34461	P34697	P36214	P37369	P41962
P41963	P41973	P41974	P41975	P41976	P41978	P41979
P41980	P41981	P43019	P43312	P43725	P47201	P50059
P50061	P0A4J2	P51547	P0AGD1	P53636	P69049	P53638

A1VXQ2	P53641	P53642	P53649	P53651	P53652	P53653
P0C0F8	P54375	P77929	P77968	P80174	P80293	P80566
P80734	P80857	P81926	P93258	P93407	Q00637	Q01137
Q02610	Q03301	Q03302	Q07182	Q07449	Q07796	Q08420
Q08713	Q42684	Q43779	Q59094	Q59448	Q59452	Q59519
Q59623	Q59679	Q60036	Q92429	Q92450		

Table A.15. *UniProt identifiers for dataset enzyme_15*

O15910	O46310	O61065	O66503	O83092	O83972	O84834
P00452	P69521	P0CAP6	P03190	P06474	P07201	P07742
P08543	P09247	P09853	P09938	P10224	P11156	P11157
P11158	P12848	P16782	P20493	P20503	P21524	P21672
P23921	P26685	P26713	P28846	P29883	P31350	P32209
P32282	P32984	P33799	P36602	P36603	P37426	P37427
P39452	P42170	P42491	P42492	P42521	P43754	P47471
P47473	P48591	P48592	P49723	P49730	P50620	P50621
P50641	P50642	P50643	P50644	P50645	P50646	P50647
P50648	P50650	P50651	P52343	P55982	P55983	P74240
P75461	P78027	P79733	Q01037	Q01038	Q01319	Q03604
Q08698	Q10840	Q60561	P07071	P28903	P43752	Q05262

Table A.16. *UniProt identifiers for dataset enzyme_16*

P42454	O04397	O04977	O23877	P00454	P00455	P08165
P10933	P22570	P28861	P31973	P41343	P41344	P41345
P41346	P53991	Q00598	P65528	Q41014	Q44532	Q44549
Q55318	Q61578	P07771	A5W4E9	P0C621	P21394	P23101
P37337	P77650	Q03304	Q07946	Q52126	O07643	O26739
O27605	O27606	O68940	O68943	O68946	O68951	P00457
P00458	P00459	P00460	P00461	P00462	P00463	P00464
P00467	P00468	P06117	P06118	P06119	P06120	P06121

P06122 P06662 P06769 P07328 P07329 P08624 P08625
P08717 P08718 P09552 P09553 P09554 P09555 P09772
P11347 P15052 P15332 P15334 P15335 P16266 P16267
P16268 P16269 P16855 P16856 P17303 P19066 P19067
P19068 P20620 P20621 P22548 P22921 P25314 P25767
P26248 P26250 P26251 P26252 P33178 P46034 P51754
P54799 P54800 P55170 P71526 P71527 P77874 P95296
Q00240 Q02452 Q07933 Q07934 Q07935 Q07942 Q44044
Q44045 Q46083 Q46244 Q50218 Q50785 Q50788 Q55029
Q55030 Q57118 Q58289 Q59270 P07598 P07603 P12635
P12636 P12943 P12944 P13063 P13065 P13628 P13629
P15283 P15284 P17632 P17633 P18188 P18190 P18191
P18636 P18637 P0ACD9 P69739 P21852 P21949 P21950
P29166 P31891 P31892 P33374 P33375 P0ACE2 Q46046
P69741

Appendix B

Additional results for Chapter 4

B.1 One-class and binary classification without sampling

The figures in this section depict AUC values in one-class and binary PHMM classification. In the one-class case, a uniform and a reverse null model is used. The AUC is calculated in each iteration of the Baum-Welch training algorithm. The tables show the p-value for each pair of approaches after the last iteration of the Baum-Welch algorithm. Statistical significance is tested at the 0.05 level. A significant difference is printed in bold.

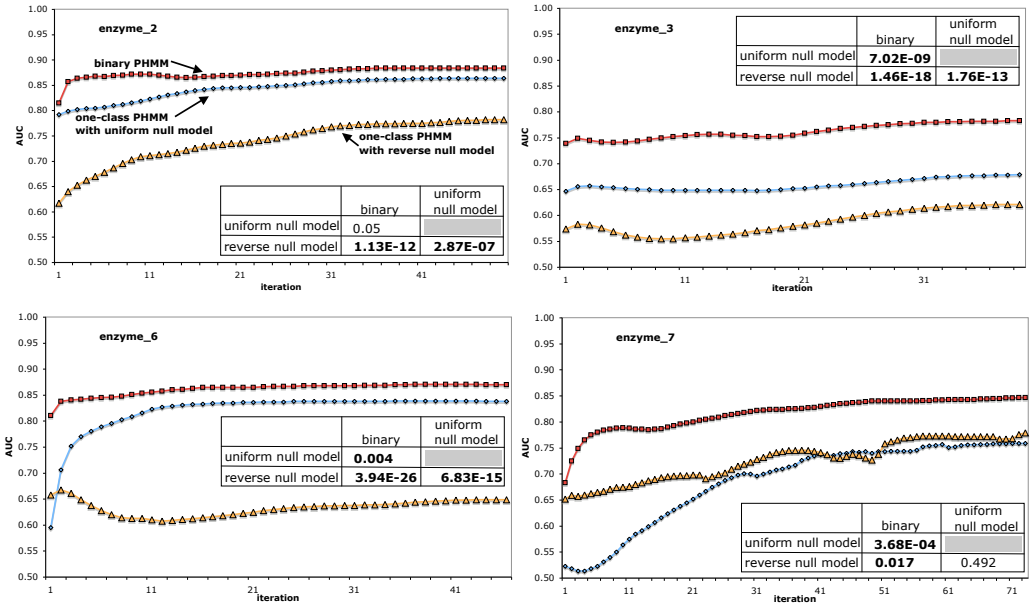


Figure B.1. *AUC values for enzyme_2, enzyme_3, enzyme_6 and enzyme_7 following the description from the beginning of Section B.1.*

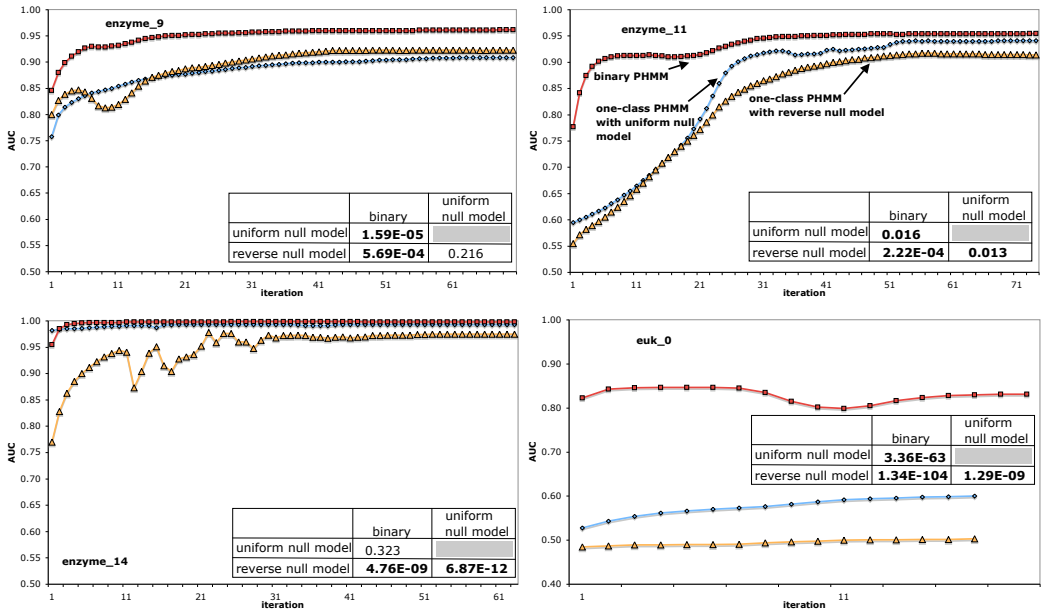


Figure B.2. *AUC values for enzyme_9, enzyme_11, enzyme_14 and euk_0 following the description from the beginning of Section B.1.*

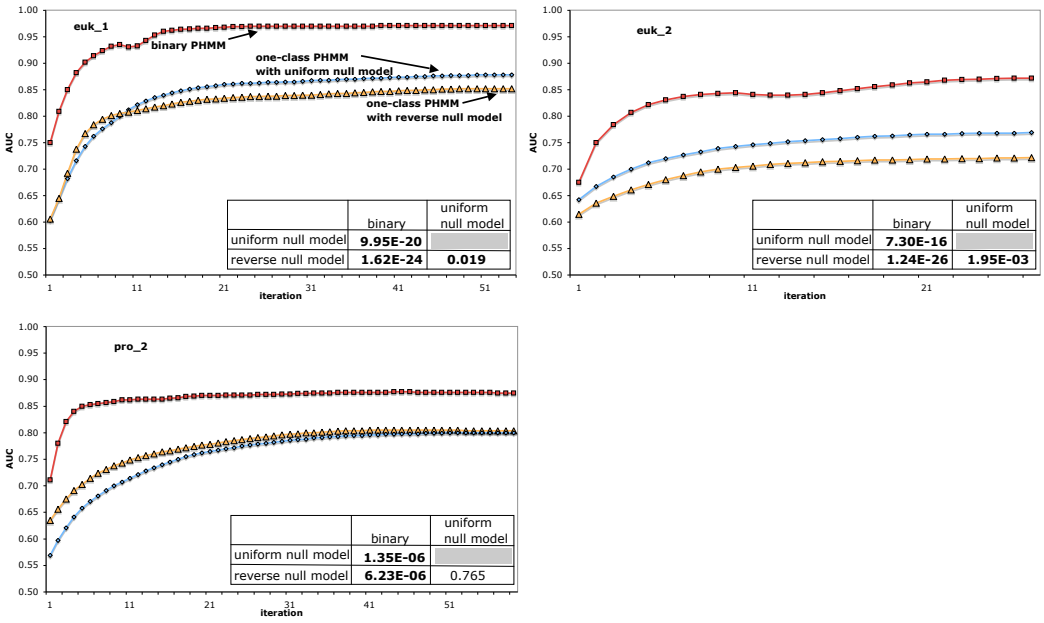


Figure B.3. *AUC values for euk_1, euk_2 and pro_2 following the description from the beginning of Section B.1.*

B.2 AUC under other null models

The figures in this section depict AUC values in one-class PHMM classification for the uniform null model and the null models built from the background distribution of the positive, the negative and all training instances.

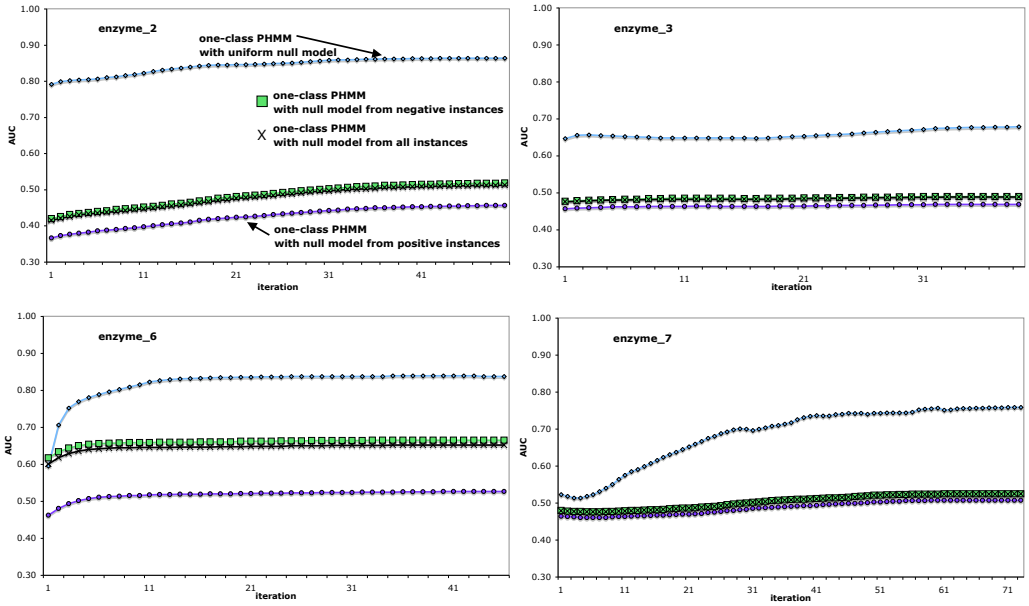


Figure B.4. *AUC values for enzyme_2, enzyme_3, enzyme_6 and enzyme_7 following the description from the beginning of Section B.2.*

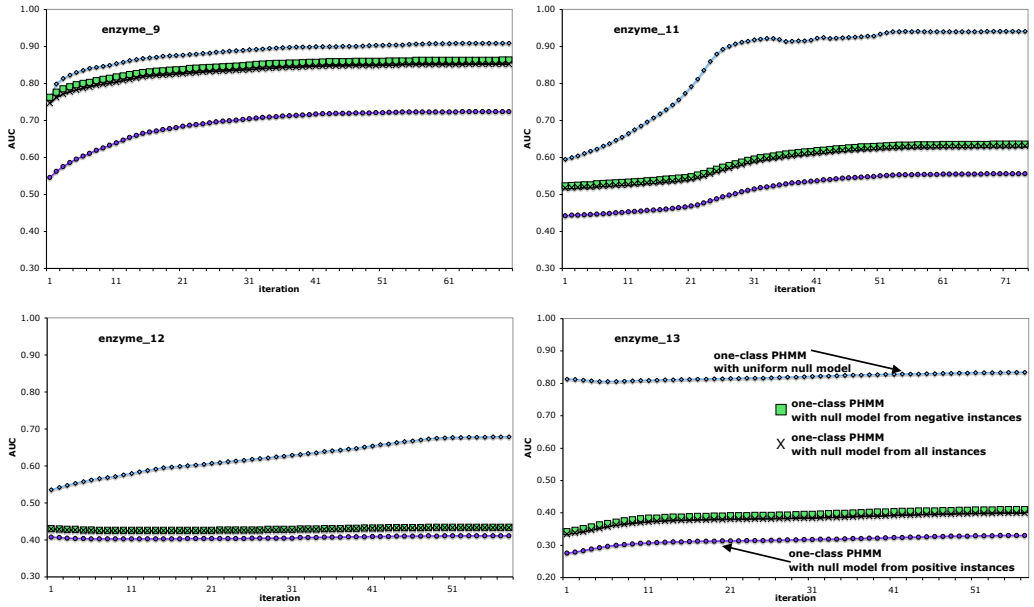


Figure B.5. *AUC values for enzyme_9, enzyme_11, enzyme_12 and enzyme_13 following the description from the beginning of Section B.2.*

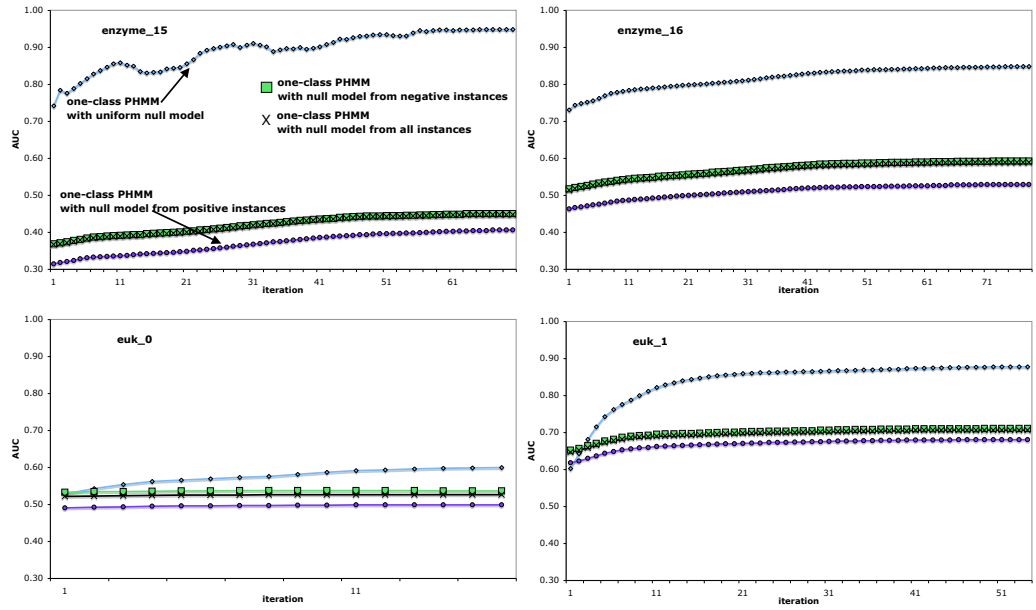


Figure B.6. *AUC values for enzyme_15, enzyme_16, euk_0 and euk_1 following the description from the beginning of Section B.2.*

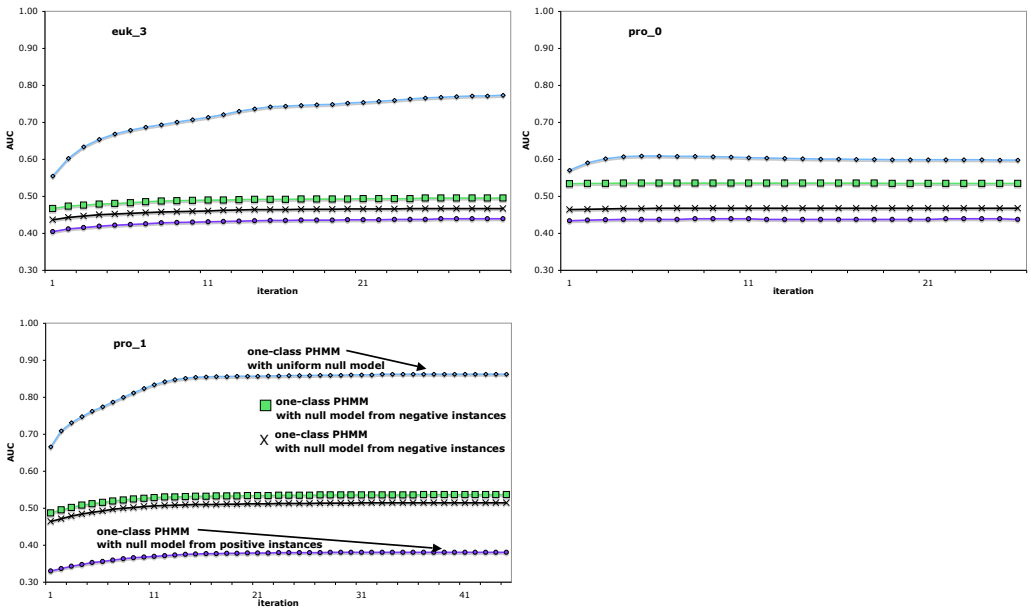
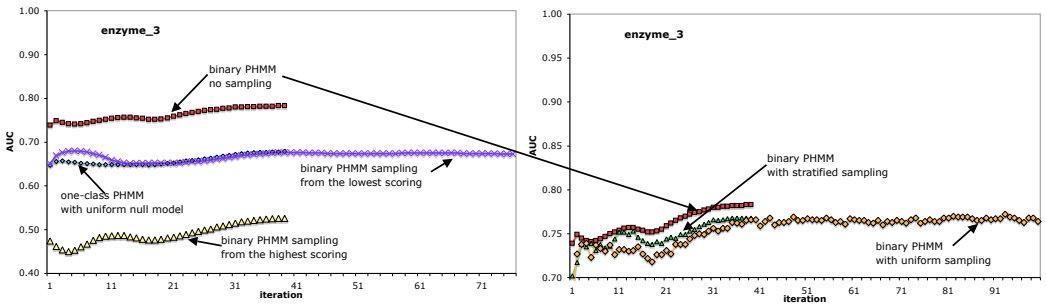


Figure B.7. *AUC values for euk_3, pro_0, and pro_1 following the description from the beginning of Section B.2.*

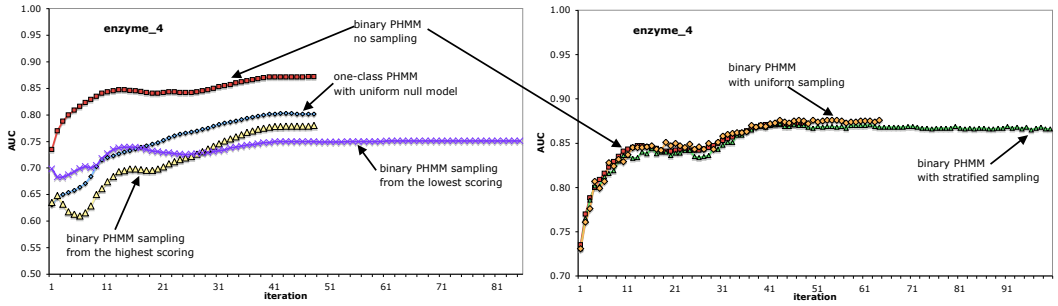
B.3 Binary classification with sampling

All figures in this section are composed identically. The graph on the left compares a one-class PHMM with a uniform null model, a binary PHMM trained without sampling, a binary PHMM sampling from the highest scoring negative instances and one that samples from the lowest scoring negatives instances. On the right, the binary PHMM without sampling is compared to uniform and stratified sampling combined with a binary PHMM. Note the y-axes are scaled differently. The table below contains the p-values for all pairs of approaches after convergence. Bold print indicates a statistically significant difference at the 0.05 level. A green background colour indicates that there is no statistically significant difference between uniform sampling and no sampling. The absence of green colour shows that the binary PHMM with full information statistically significantly outperforms the uniform sampling approach.



	binary no sampling	one-class uniform null model	binary uniform sampling	binary stratified sampling	binary sampling from the lowest scoring
one-class, uniform null model	**7.02E-09**				
binary, uniform sampling	**0.044**	**4.00E-06**			
binary, stratified sampling	**0.049**	**7.65E-07**	0.680		
binary, sampling from the lowest scoring	**7.21E-11**	**8.73E-08**	**1.76E-13**	**1.24E-07**	
binary, sampling from the highest scoring	**1.45E-29**	**2.66E-25**	**2.02E-30**	**1.32E-27**	**2.65E-06**

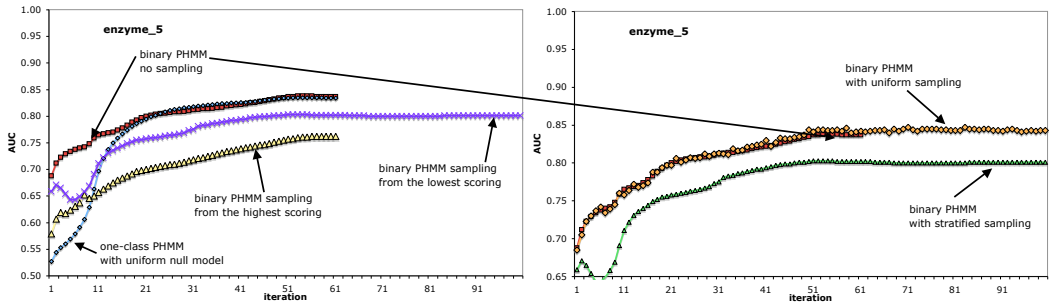
Figure B.8. *Comparison of sampling approaches for enzyme_3 following the description from the beginning of Section B.3.*



	binary no sampling	one-class uniform null model	binary uniform sampling	binary stratified sampling	binary sampling from the lowest scoring
one-class, uniform null model	**2.17E-06**				
binary, uniform sampling	0.403	**5.99E-07**			
binary, stratified sampling	0.384	**1.33E-05**	0.194		
binary, sampling from the lowest scoring	**4.34E-15**	**5.54E-04**	**1.42E-15**	**1.39E-15**	
binary, sampling from the highest scoring	**2.49E-08**	0.374	**1.06E-08**	**3.60E-06**	0.239

uniform sampling statistically as good as a binary PHMM trained on a full dataset

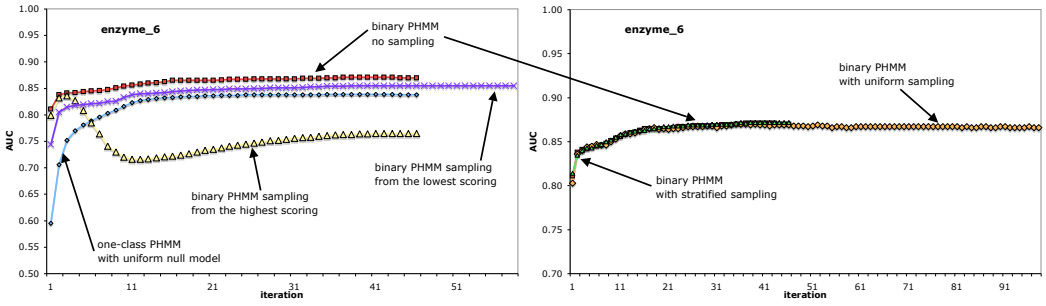
Figure B.9. *Comparison of sampling approaches for enzyme_4 following the description from the beginning of Section B.3.*



	binary no sampling	one-class uniform null model	binary uniform sampling	binary stratified sampling	binary sampling from the lowest scoring
one-class, uniform null model	0.894				
binary, uniform sampling	0.620	0.983			
binary, stratified sampling	0.198	0.594	0.201		
binary, sampling from the lowest scoring	**0.046**	**0.015**	0.063	**0.018**	
binary, sampling from the highest scoring	**0.003**	**0.032**	**0.004**	**0.002**	0.276

uniform sampling statistically as good as a binary PHMM trained on a full dataset

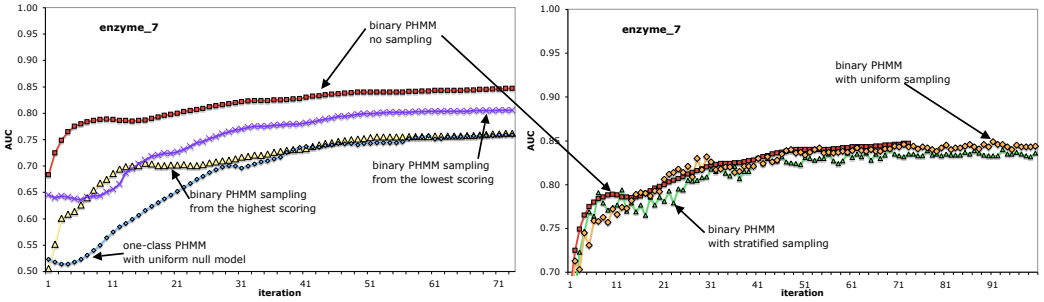
Figure B.10. *Comparison of sampling approaches for enzyme_5 following the description from the beginning of Section B.3.*



	binary no sampling	one-class uniform null model	binary uniform sampling	binary stratified sampling	binary sampling from the lowest scoring
one-class, uniform null model	**0.004**				
binary, uniform sampling	0.122	**0.014**			
binary, stratified sampling	0.730	**0.005**	0.217		
binary, sampling from the lowest scoring	**0.026**	**0.009**	0.146	**0.030**	
binary, sampling from the highest scoring	**4.91E-11**	**8.13E-04**	**8.02E-10**	**3.13E-12**	**2.82E-06**

uniform sampling statistically as good as a binary PHMM trained on a full dataset

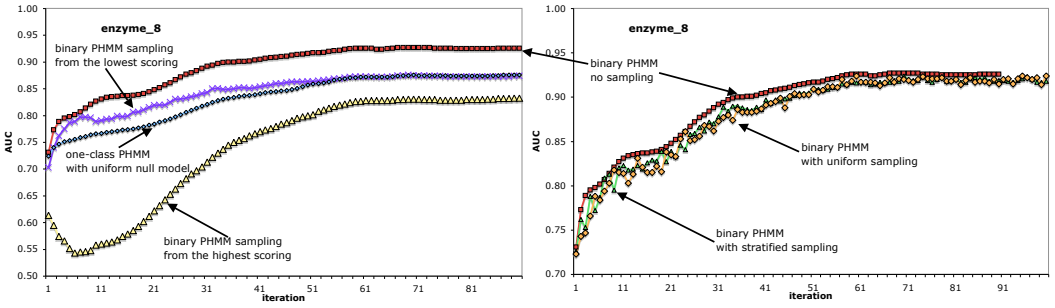
Figure B.11. *Comparison of sampling approaches for enzyme_6 following the description from the beginning of Section B.3.*



	binary no sampling	one-class uniform null model	binary uniform sampling	binary stratified sampling	binary sampling from the lowest scoring
one-class, uniform null model	**3.68E-04**				
binary, uniform sampling	0.684	**3.18E-04**			
binary, stratified sampling	0.145	**0.001**	0.442		
binary, sampling from the lowest scoring	**0.017**	0.056	0.052	**0.120**	
binary, sampling from the highest scoring	**0.002**	0.910	**0.003**	**0.004**	0.204

uniform sampling statistically as good as a binary PHMM trained on a full dataset

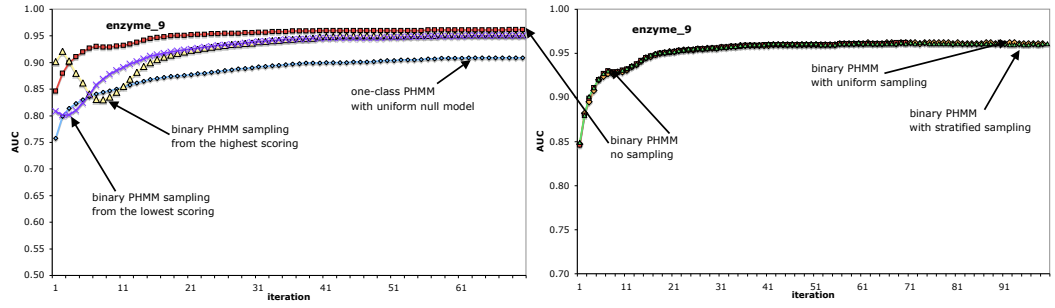
Figure B.12. *Comparison of sampling approaches for enzyme_7 following the description from the beginning of Section B.3.*



	binary no sampling	one-class uniform null model	binary uniform sampling	binary stratified sampling	binary sampling from the lowest scoring
one-class, uniform null model	**9.57E-04**				
binary, uniform sampling	0.850	**0.003**			
binary, stratified sampling	0.189	**0.012**	0.375		
binary, sampling from the lowest scoring	**0.003**	0.888	**0.007**	**0.010**	
binary, sampling from the highest scoring	**2.42E-04**	0.171	**4.98E-04**	**0.002**	0.224

uniform sampling statistically as good as a binary PHMM trained on a full dataset

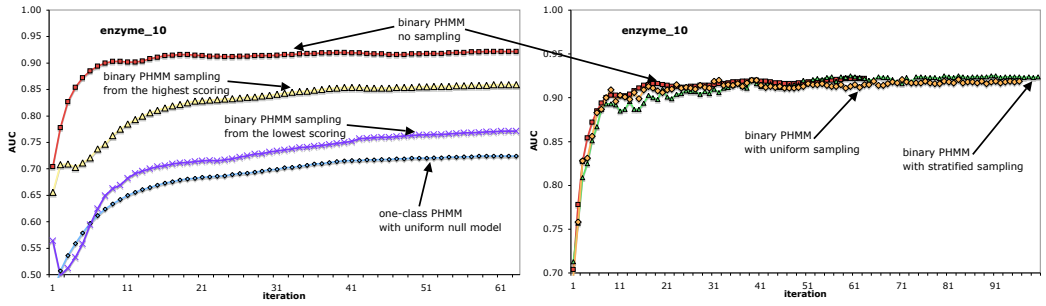
Figure B.13. *Comparison of sampling approaches for enzyme_8 following the description from the beginning of Section B.3.*



	binary no sampling	one-class uniform null model	binary uniform sampling	binary stratified sampling	binary sampling from the lowest scoring
one-class, uniform null model	**1.59E-05**				
binary, uniform sampling	0.877	**1.52E-05**			
binary, stratified sampling	0.407	**2.19E-05**	0.625		
binary, sampling from the lowest scoring	**0.008**	**3.03E-05**	**0.008**	**0.022**	
binary, sampling from the highest scoring	**0.016**	**0.001**	**0.031**	0.052	0.664

uniform sampling statistically as good as a binary PHMM trained on a full dataset

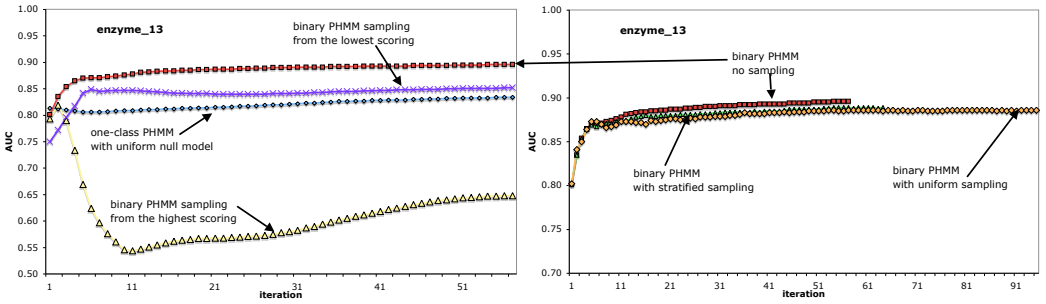
Figure B.14. *Comparison of sampling approaches for enzyme_9 following the description from the beginning of Section B.3.*



	binary no sampling	one-class uniform null model	binary uniform sampling	binary stratified sampling	binary sampling from the lowest scoring
one-class, uniform null model	**1.28E-09**				
binary, uniform sampling	0.605	**2.93E-08**			
binary, stratified sampling	0.794	**2.36E-09**	0.548		
binary, sampling from the lowest scoring	**1.12E-11**	**0.003**	**2.64E-09**	**1.75E-11**	
binary, sampling from the highest scoring	**5.19E-04**	**0.003**	**8.05E-05**	**3.21E-04**	**0.010**

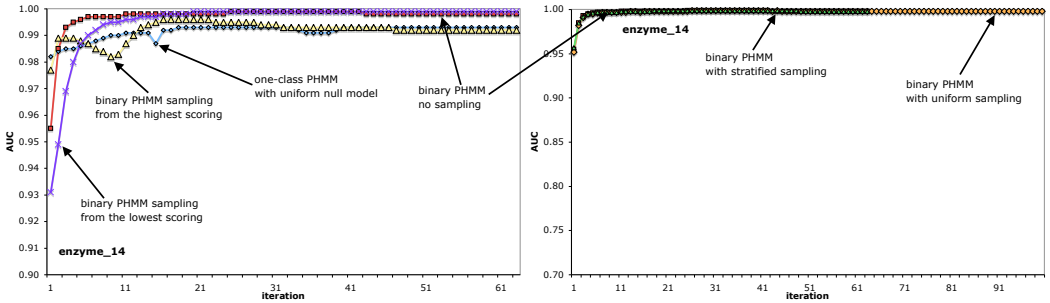
uniform sampling statistically as good as a binary PHMM trained on a full dataset

Figure B.15. *Comparison of sampling approaches for enzyme_10 following the description from the beginning of Section B.3.*



	binary no sampling	one-class uniform null model	binary uniform sampling	binary stratified sampling	binary sampling from the lowest scoring
one-class, uniform null model	**1.81E-15**				
binary, uniform sampling	**3.63E-04**	**8.78E-12**			
binary, stratified sampling	**0.008**	**7.45E-12**	0.429		
binary, sampling from the lowest scoring	**1.55E-10**	**0.004**	**5.24E-07**	**3.45E-08**	
binary, sampling from the highest scoring	**7.29E-52**	**1.40E-21**	**2.52E-49**	**9.92E-50**	**1.01E-29**

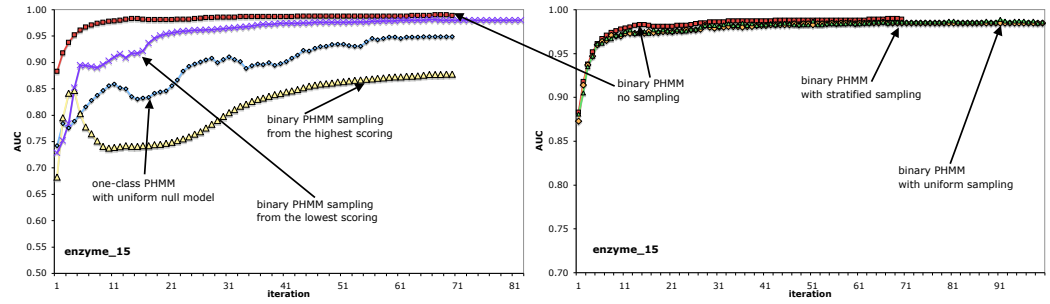
Figure B.16. *Comparison of sampling approaches for enzyme_13 following the description from the beginning of Section B.3.*



	binary no sampling	one-class uniform null model	binary uniform sampling	binary stratified sampling	binary sampling from the lowest scoring
one-class, uniform null model	0.323				
binary, uniform sampling	0.393	0.332			
binary, stratified sampling	0.278	0.314	0.298		
binary, sampling from the lowest scoring	0.222	0.243	0.220	0.285	
binary, sampling from the highest scoring	**0.001**	0.885	**0.001**	**8.25E-04**	**8.71E-04**

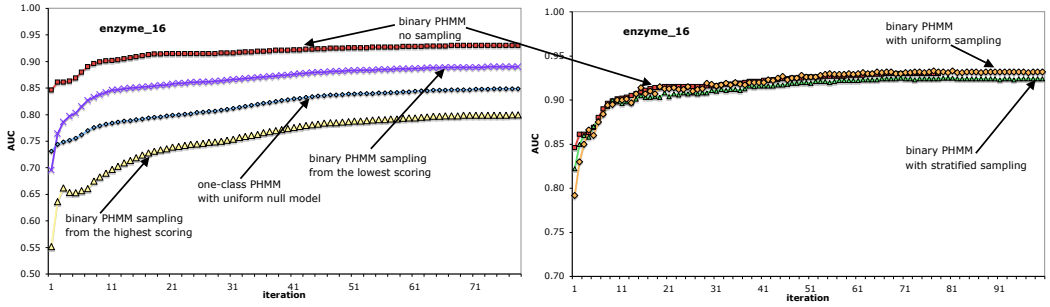
uniform sampling statistically as good as a binary PHMM trained on a full dataset

Figure B.17. *Comparison of sampling approaches for enzyme_14 following the description from the beginning of Section B.3.*



	binary no sampling	one-class uniform null model	binary uniform sampling	binary stratified sampling	binary sampling from the lowest scoring
one-class, uniform null model	**4.70E-12**				
binary, uniform sampling	**0.032**	**5.62E-10**			
binary, stratified sampling	0.053	**1.10E-10**	0.524		
binary, sampling from the lowest scoring	**0.021**	**2.67E-07**	0.228	**0.097**	
binary, sampling from the highest scoring	**1.23E-11**	**3.53E-05**	**3.60E-11**	**4.17E-11**	**3.84E-10**

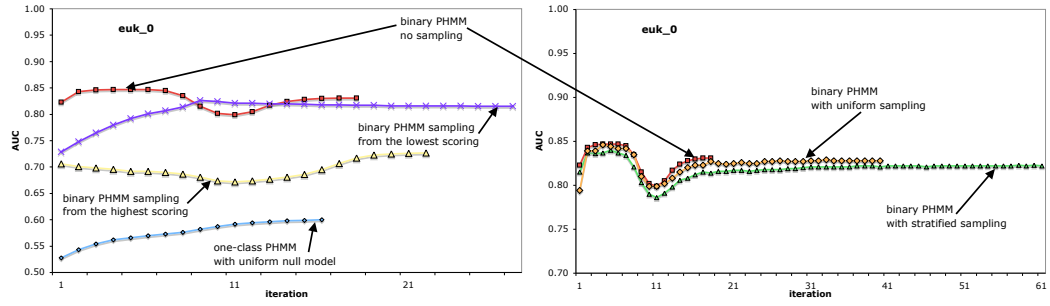
Figure B.18. *Comparison of sampling approaches for enzyme_15 following the description from the beginning of Section B.3.*



	binary no sampling	one-class uniform null model	binary uniform sampling	binary stratified sampling	binary sampling from the lowest scoring
one-class, uniform null model	**3.30E-08**				
binary, uniform sampling	0.489	**1.08E-08**			
binary, stratified sampling	**0.005**	**3.15E-07**	**0.020**		
binary, sampling from the lowest scoring	**1.28E-06**	**2.00E-04**	**5.12E-07**	**3.07E-05**	
binary, sampling from the highest scoring	**1.11E-24**	**0.027**	**1.18E-24**	**2.10E-23**	**1.96E-07**

uniform sampling statistically as good as a binary PHMM trained on a full dataset

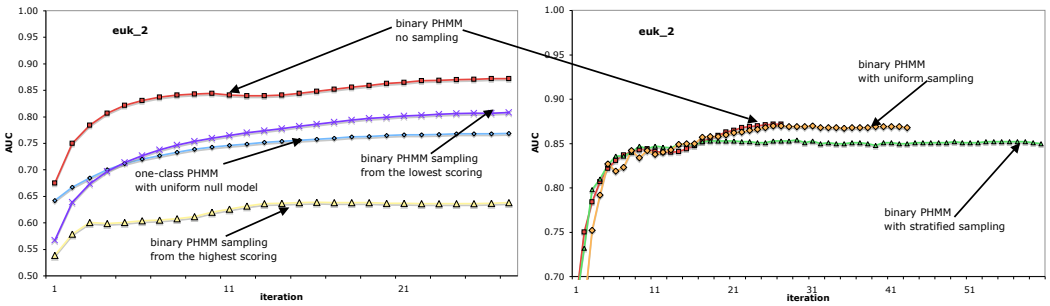
Figure B.19. *Comparison of sampling approaches for enzyme_16 following the description from the beginning of Section B.3.*



	binary no sampling	one-class uniform null model	binary uniform sampling	binary stratified sampling	binary sampling from the lowest scoring
one-class, uniform null model	**3.36E-63**				
binary, uniform sampling	0.361	**1.16E-60**			
binary, stratified sampling	**0.010**	**1.85E-53**	0.173		
binary, sampling from the lowest scoring	**0.005**	**5.85E-78**	**0.027**	0.299	
binary, sampling from the highest scoring	**9.60E-28**	**1.61E-11**	**4.50E-28**	**2.16E-25**	**4.29E-12**

uniform sampling statistically as good as a binary PHMM trained on a full dataset

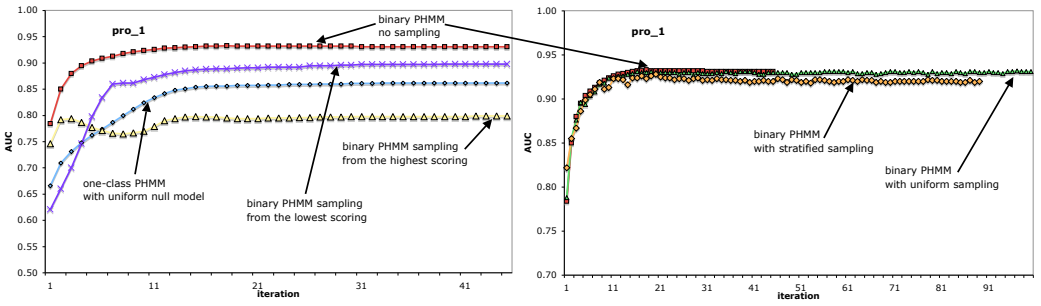
Figure B.20. *Comparison of sampling approaches for euk_0 following the description from the beginning of Section B.3.*



	binary no sampling	one-class uniform null model	binary uniform sampling	binary stratified sampling	binary sampling from the lowest scoring
one-class, uniform null model	**7.30E-16**				
binary, uniform sampling	0.362	**8.17E-04**			
binary, stratified sampling	**1.46E-04**	**3.29E-09**	**0.005**		
binary, sampling from the lowest scoring	**7.04E-11**	**9.27E-07**	**8.85E-09**	**1.65E-04**	
binary, sampling from the highest scoring	**2.46E-54**	**4.61E-09**	**6.53E-54**	**1.69E-46**	**1.16E-16**

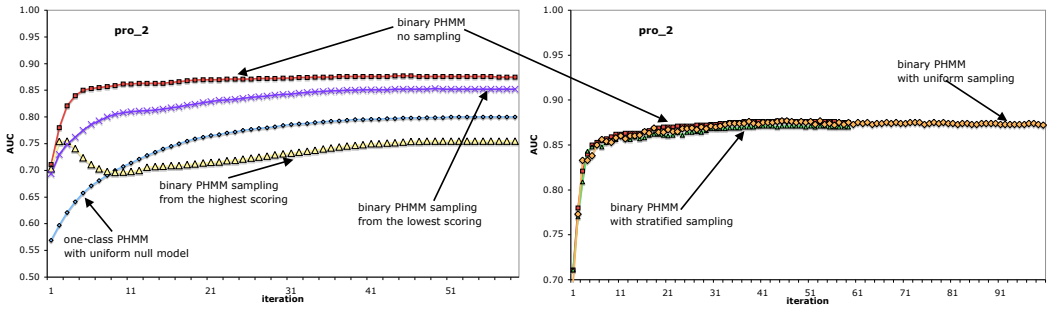
uniform sampling statistically as good as a binary PHMM trained on a full dataset

Figure B.21. *Comparison of sampling approaches for euk_2 following the description from the beginning of Section B.3.*



	binary no sampling	one-class uniform null model	binary uniform sampling	binary stratified sampling	binary sampling from the lowest scoring
one-class, uniform null model	**3.42E-07**				
binary, uniform sampling	**0.022**	**1.54E-05**			
binary, stratified sampling	0.945	**9.31E-08**	0.067		
binary, sampling from the lowest scoring	**5.17E-05**	**5.27E-04**	**0.006**	**3.50E-05**	
binary, sampling from the highest scoring	**1.59E-12**	**0.011**	**2.73E-10**	**2.28E-12**	**8.26E-06**

Figure B.22. *Comparison of sampling approaches for pro_1 following the description from the beginning of Section B.3.*



	binary no sampling	one-class uniform null model	binary uniform sampling	binary stratified sampling	binary sampling from the lowest scoring
one-class, uniform null model	**1.35E-06**				
binary, uniform sampling	0.266	**2.72E-06**			
binary, stratified sampling	0.148	**1.20E-05**	0.660		
binary, sampling from the lowest scoring	**0.009**	**4.21E-06**	**0.027**	0.059	
binary, sampling from the highest scoring	**1.14E-07**	0.068	**2.52E-15**	**1.95E-17**	**5.05E-07**

uniform sampling statistically as good as a binary PHMM trained on a full dataset

Figure B.23. *Comparison of sampling approaches for pro_2 following the description from the beginning of Section B.3.*

B.4 Sampling from all negative data

The figures depict AUC values for binary PHMM classification with no sampling and when taking all instances of the *enzyme*, *pro* and *euk* datasets as negative examples and subsequent use of uniform sampling.

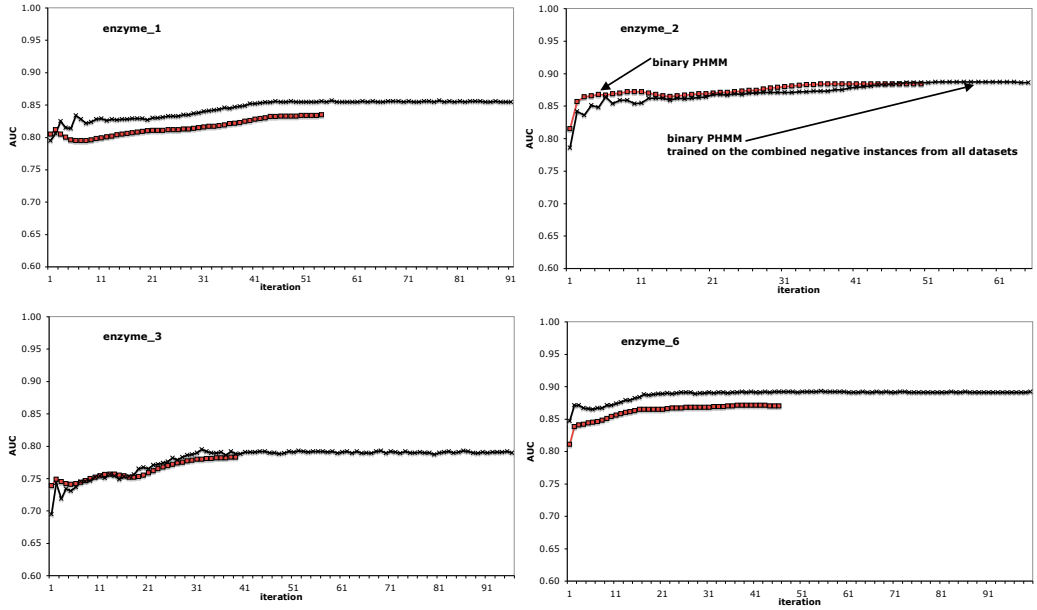


Figure B.24. *AUC values for enzyme_1, enzyme_2, enzyme_3 and enzyme_6 following the description from the beginning of Section B.4.*

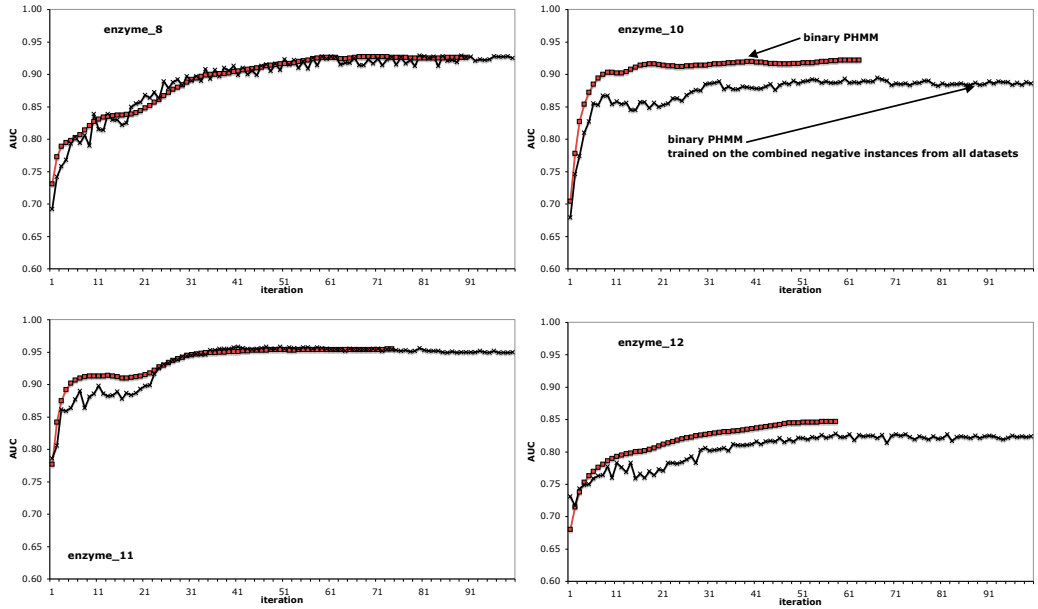


Figure B.25. *AUC values for enzyme_8, enzyme_10, enzyme_11 and enzyme_12 following the description from the beginning of Section B.4.*

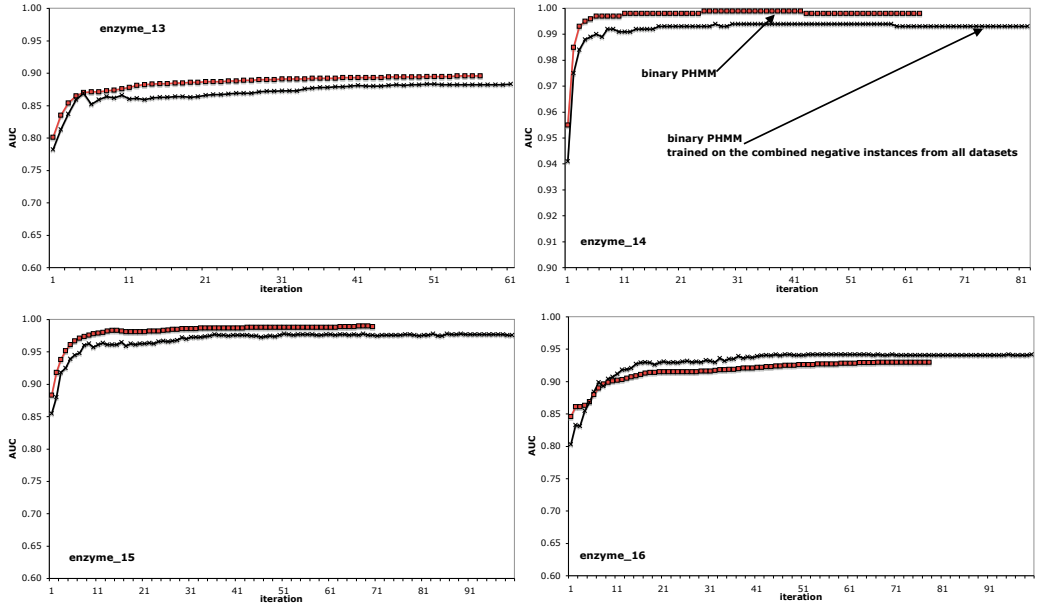


Figure B.26. *AUC values for enzyme_13, enzyme_14, enzyme_15 and enzyme_16 following the description from the beginning of Section B.4.*

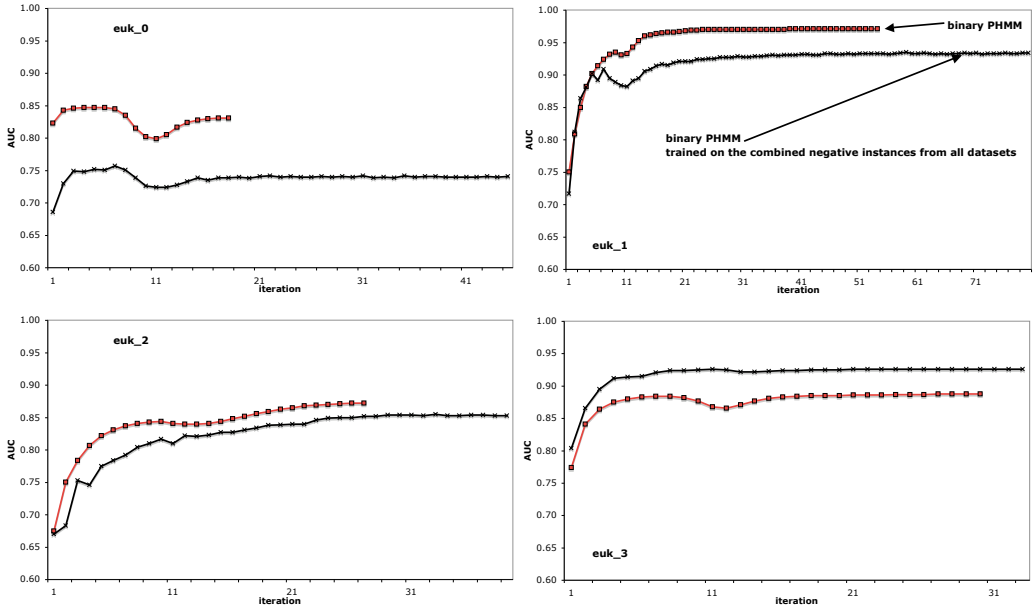


Figure B.27. *AUC values for euk_0, euk_1, euk_2 and euk_3 following the description from the beginning of Section B.4.*

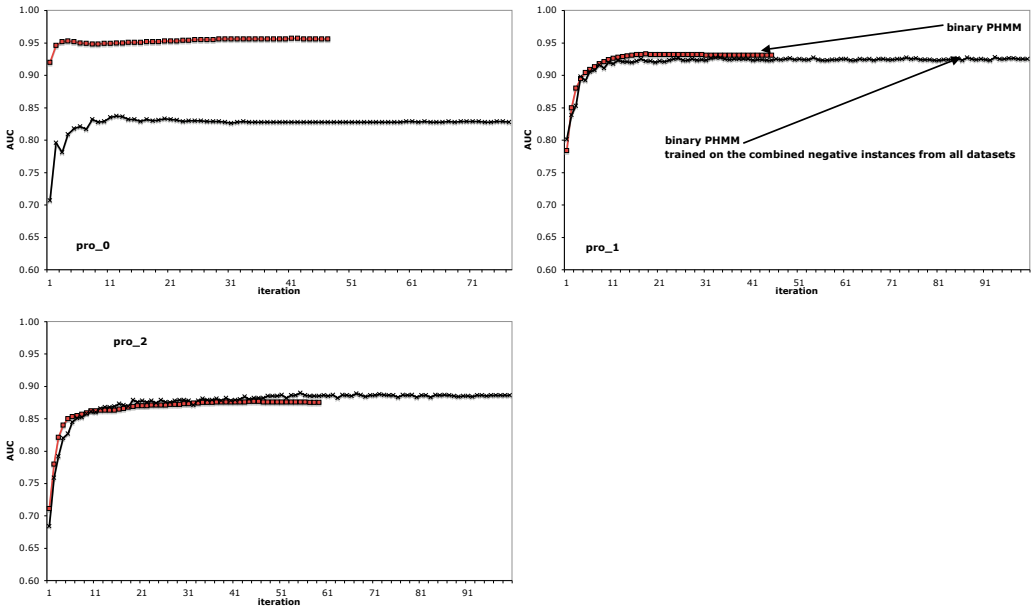


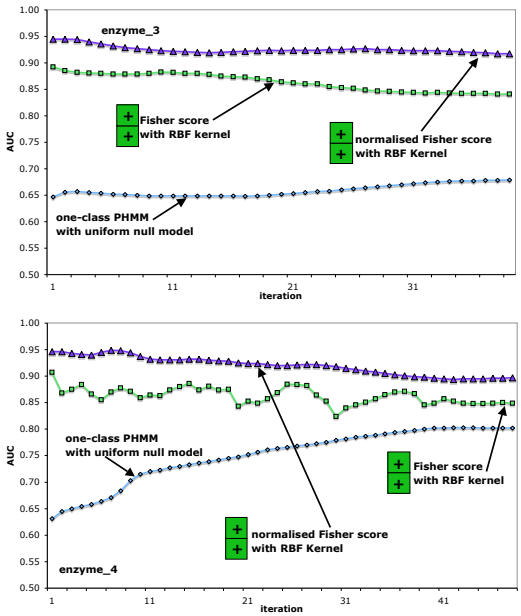
Figure B.28. *AUC values for pro_0, pro_1 and pro_2 following the description from the beginning of Section B.4.*

Appendix C

Additional results for Chapter 5

C.1 Fisher score vectors with Support Vector Machines

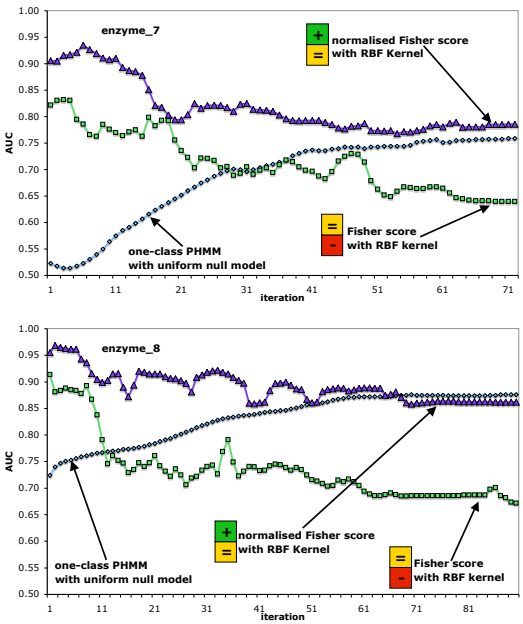
The figures in this section depict the AUC values in one-class PHMM classification and propositional classification using Fisher score vectors. The AUC is calculated in each iteration of the Baum-Welch training algorithm. The tables on the right contain the p-values. P-values printed in bold indicate a statistically significant difference in AUC at the 0.05 level. The meaning of the squares in the left-hand graphs is defined in Section 5.2.



	first Iteration	last Iteration	first Iteration
last iteration	**0.009**		
first iteration	**4.83E-05**	**0.044**	
last iteration	**1.20E-10**	**1.72E-07**	**0.001**

	first Iteration	last Iteration	first Iteration
last iteration	**0.003**		
first iteration	**0.004**	0.562	
last iteration	**2.81E-05**	**0.010**	**0.014**

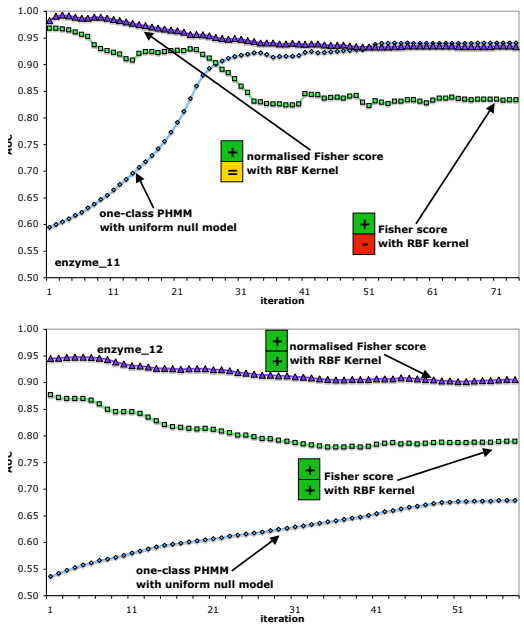
Figure C.1. *AUC values for enzyme_3 and enzyme_4 following the description from the beginning of Section C.1.*



	first Iteration	last Iteration	first Iteration
last iteration	**1.12E-05**		
first iteration	**0.003**	0.216	
last iteration	**1.44E-11**	**0.002**	**1.07E-04**

	first Iteration	last Iteration	first Iteration
last iteration	**0.005**		
first iteration	**0.049**	0.191	
last iteration	**7.96E-11**	**8.05E-06**	**1.15E-07**

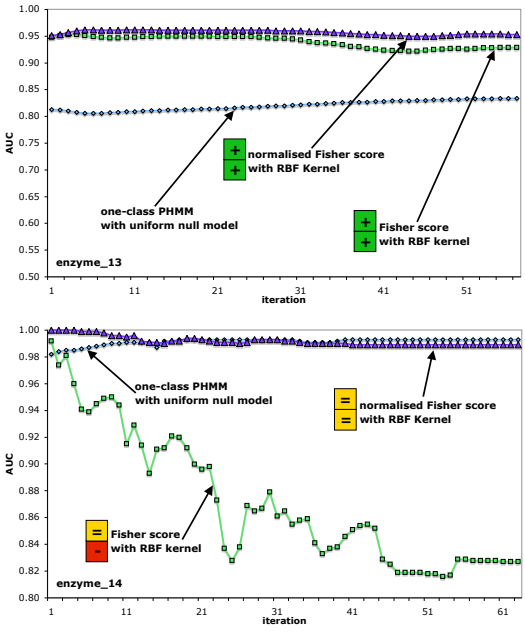
Figure C.2. *AUC values for enzyme_7 and enzyme_8 following the description from the beginning of Section C.1.*



	first Iteration	last Iteration	first Iteration
last iteration	**2.74E-04**		
first iteration	0.136	**0.018**	
last iteration	**6.55E-11**	**9.66E-06**	**2.80E-10**

	first Iteration	last Iteration	first Iteration
last iteration	**0.039**		
first iteration	**0.001**	0.260	
last iteration	**7.87E-09**	**4.72E-06**	**0.003**

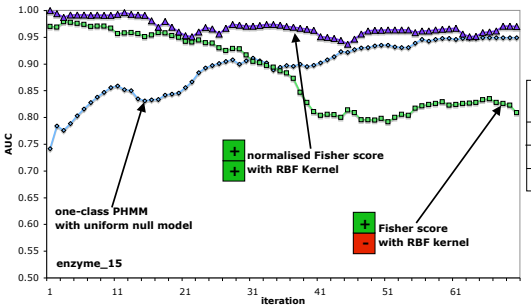
Figure C.3. *AUC values for enzyme_11 and enzyme_12 following the description from the beginning of Section C.1.*



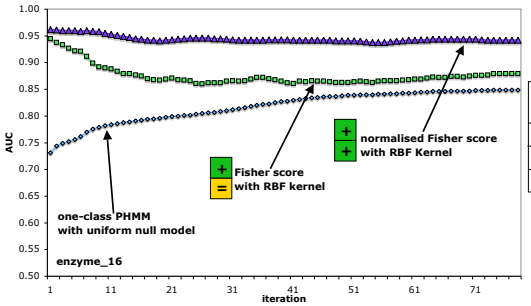
	first Iteration	last Iteration	first Iteration
last iteration	0.721		
first iteration	0.586	0.465	
last iteration	**0.024**	**0.004**	**0.061**

	first Iteration	last Iteration	first Iteration
last iteration	0.117		
first iteration	0.259	0.624	
last iteration	**2.05E-13**	**1.10E-12**	**1.16E-12**

Figure C.4. *AUC values for enzyme_13 and enzyme_14 following the description from the beginning of Section C.1. The x-axis is scaled differently for enzyme_14.*

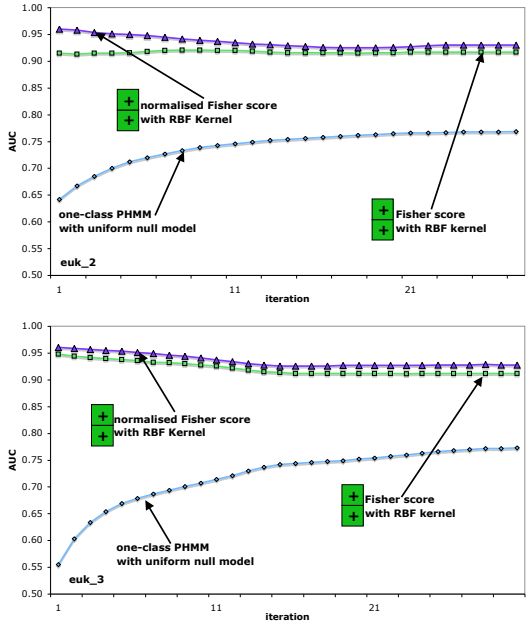


	first Iteration	last Iteration	first Iteration
last iteration	**9.72E-04**		
first iteration	**0.018**	0.945	
last iteration	**2.82E-08**	**1.09E-06**	**1.93E-06**



	first Iteration	last Iteration	first Iteration
last iteration	**0.011**		
first iteration	0.095	0.795	
last iteration	**4.98E-07**	**1.95E-04**	**5.17E-05**

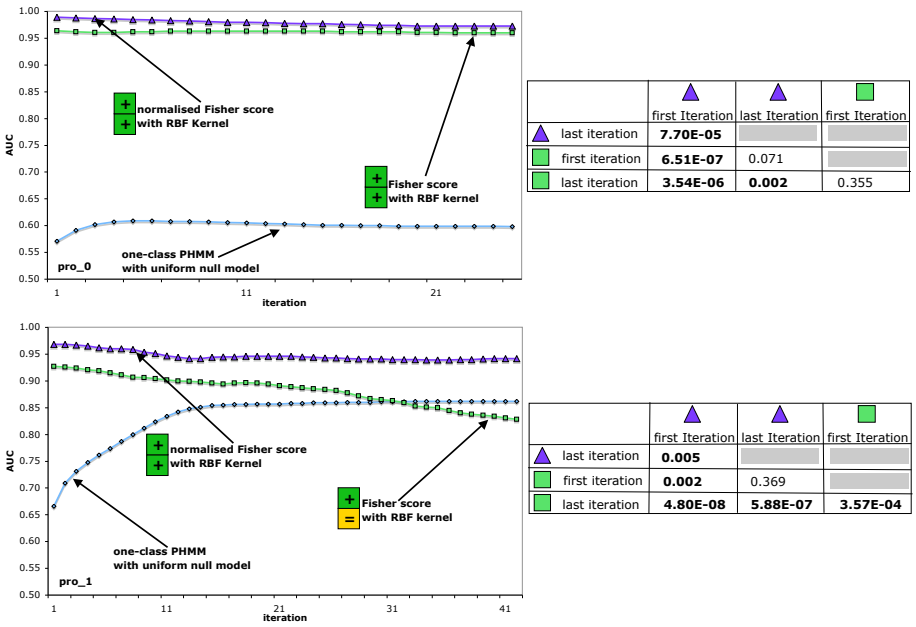
Figure C.5. *AUC values for enzyme_15 and enzyme_16 following the description from the beginning of Section C.1.*



	first Iteration	last Iteration	first Iteration
last iteration	**1.76E-07**		
first iteration	**2.51E-08**	0.102	
last iteration	**7.27E-07**	0.144	0.796

	first Iteration	last Iteration	first Iteration
last iteration	**9.98E-16**		
first iteration	**4.70E-07**	**7.07E-06**	
last iteration	**1.31E-24**	**2.48E-05**	**1.04E-14**

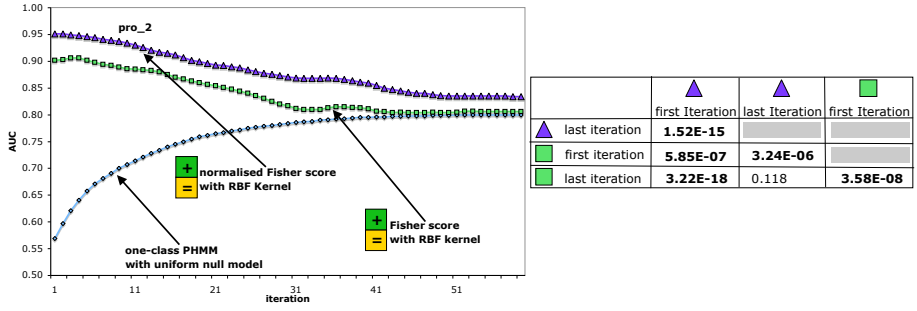
Figure C.6. *AUC values for euk_2 and euk_3 following the description from the beginning of Section C.1.*



	first Iteration	last Iteration	first Iteration
last iteration	**7.70E-05**		
first iteration	**6.51E-07**	0.071	
last iteration	**3.54E-06**	**0.002**	0.355

	first Iteration	last Iteration	first Iteration
last iteration	**0.005**		
first iteration	**0.002**	0.369	
last iteration	**4.80E-08**	**5.88E-07**	**3.57E-04**

Figure C.7. *AUC values for pro_0 and pro_1 following the description from the beginning of Section C.1.*

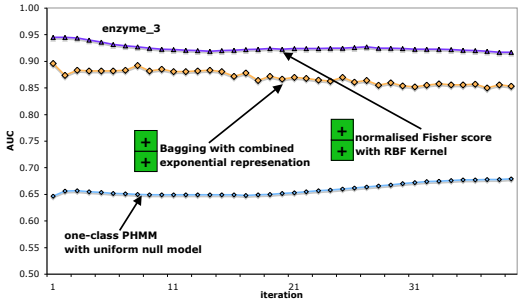


	first Iteration	last Iteration	first Iteration
last iteration	**1.52E-15**		
first iteration	**5.85E-07**	**3.24E-06**	
last iteration	**3.22E-18**	0.118	**3.58E-08**

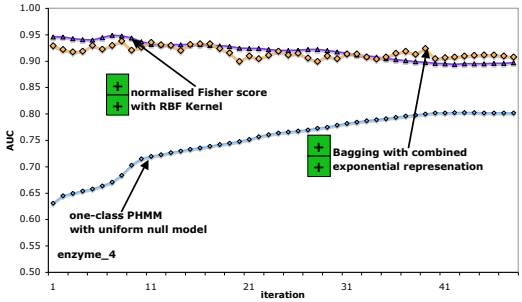
Figure C.8. *AUC values for pro_2 following the description from the beginning of Section C.1.*

C.2 Bagging

The figures in this section depict AUC values in one-class PHMM classification and bagging using the combined exponential representation. The AUC is calculated in each iteration of the Baum-Welch training algorithm. The tables on the right contain the p-values. P-values printed in bold indicate a statistically significant difference in AUC at the 0.05 level. The meaning of the squares in the left-hand graphs is defined in Section 5.2.

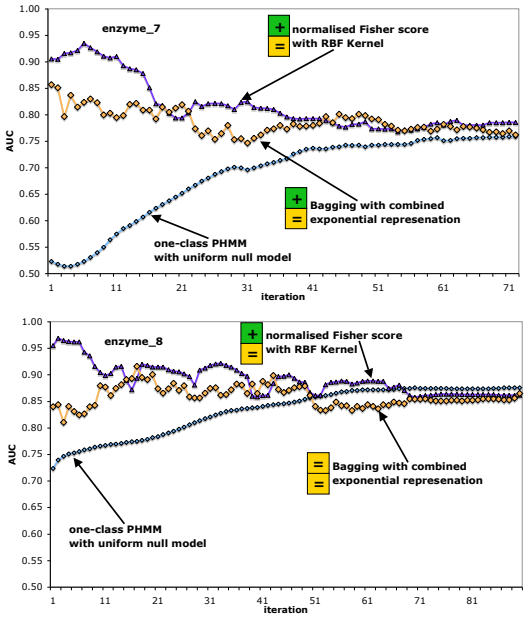


	first Iteration	last Iteration	first Iteration
last iteration	**0.009**		
first iteration	**2.22E-04**	**0.018**	
last iteration	**1.65E-09**	**2.38E-07**	**4.911E-04**



	first Iteration	last Iteration	first Iteration
last iteration	**0.003**		
first iteration	0.227	0.084	
last iteration	0.061	0.578	0.275

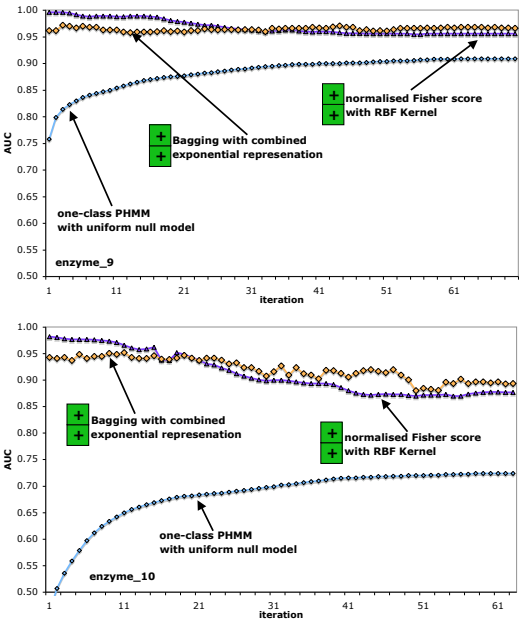
Figure C.9. *AUC values for enzyme_3 and enzyme_4 following the description from the beginning of Section C.2.*



	first Iteration	last Iteration	first Iteration
last iteration	**1.12E-05**		
first iteration	**0.010**	0.063	
last iteration	**8.84E-06**	0.675	**0.041**

	first Iteration	last Iteration	first Iteration
last iteration	**0.005**		
first iteration	**1.48E-04**	0.415	
last iteration	**0.004**	0.979	0.510

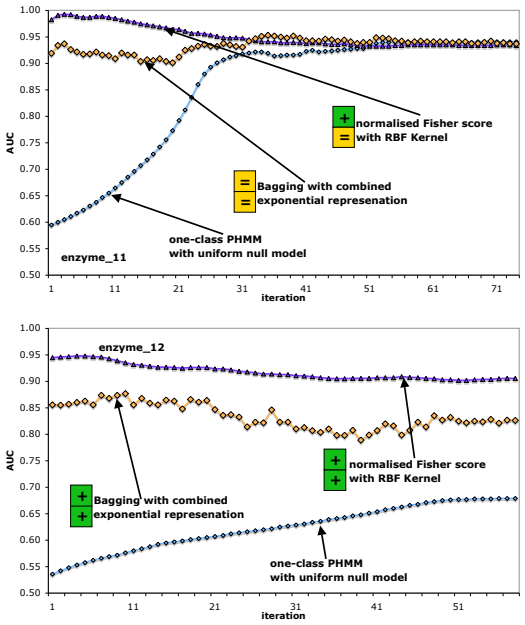
Figure C.10. *AUC values for enzyme_7 and enzyme_8 following the description from the beginning of Section C.2.*



	first Iteration	last Iteration	first Iteration
last iteration	**1.37E-06**		
first iteration	**1.44E-04**	0.507	
last iteration	**9.51E-05**	**0.045**	0.532

	first Iteration	last Iteration	first Iteration
last iteration	**2.37E-07**		
first iteration	**0.004**	**0.004**	
last iteration	**1.98E-06**	0.427	**0.027**

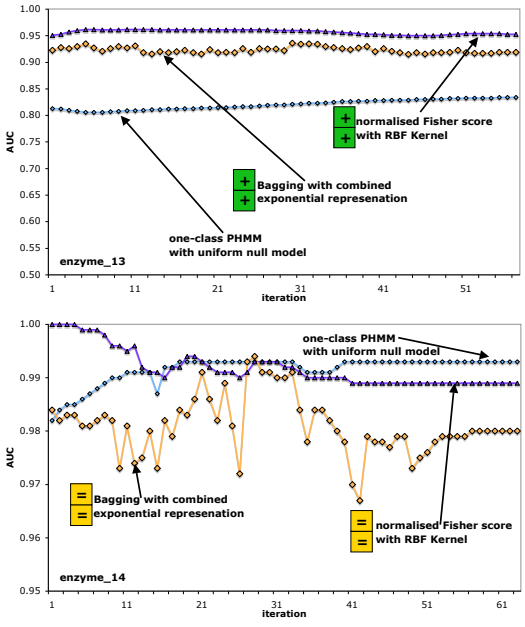
Figure C.11. *AUC values for enzyme_9 and enzyme_10 following the description from the beginning of Section C.2.*



	first Iteration	last Iteration	first Iteration
last iteration	**2.74E-04**		
first iteration	**5.41E-05**	0.325	
last iteration	**1.03E-04**	0.789	0.270

	first Iteration	last Iteration	first Iteration
last iteration	**0.039**		
first iteration	**2.19E-05**	**0.020**	
last iteration	**3.49E-06**	**7.82E-04**	0.228

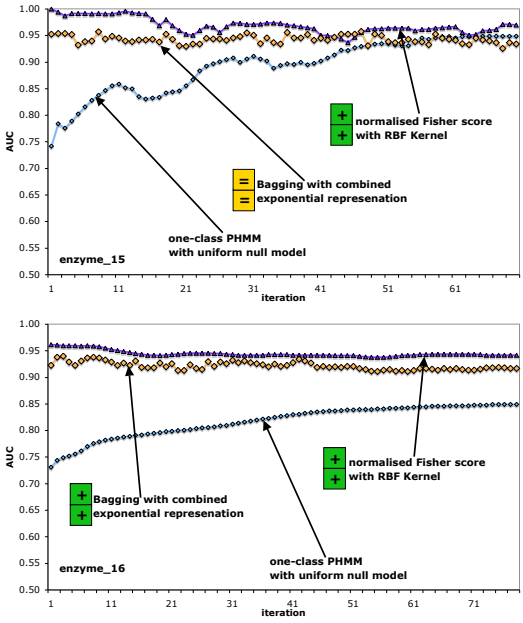
Figure C.12. *AUC values for enzyme_11 and enzyme_12 following the description from the beginning of Section C.2.*



	first Iteration	last Iteration	first Iteration
last iteration	0.721		
first iteration	**0.007**	**0.004**	
last iteration	**0.004**	**0.001**	0.564

	first Iteration	last Iteration	first Iteration
last iteration	0.117		
first iteration	0.072	0.546	
last iteration	**0.046**	0.107	0.714

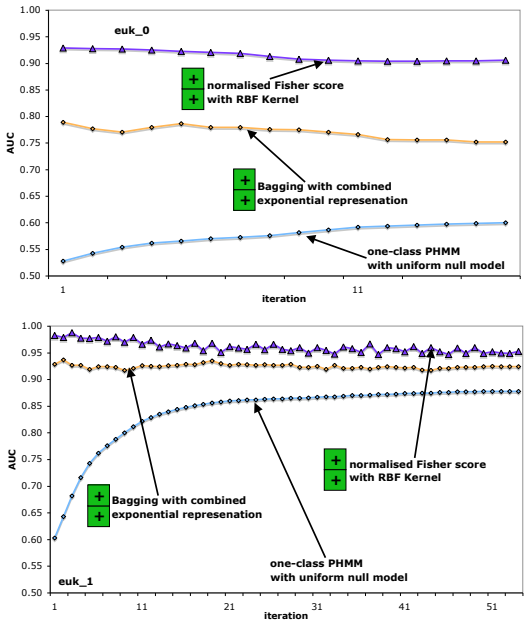
Figure C.13. *AUC values for enzyme_13 and enzyme_14 following the description from the beginning of Section C.2.*



	first Iteration	last Iteration	first Iteration
last iteration	**9.72E-04**		
first iteration	**0.004**	0.336	
last iteration	**0.002**	**0.033**	0.455

	first Iteration	last Iteration	first Iteration
last iteration	**0.011**		
first iteration	**0.016**	0.261	
last iteration	**0.002**	0.085	0.689

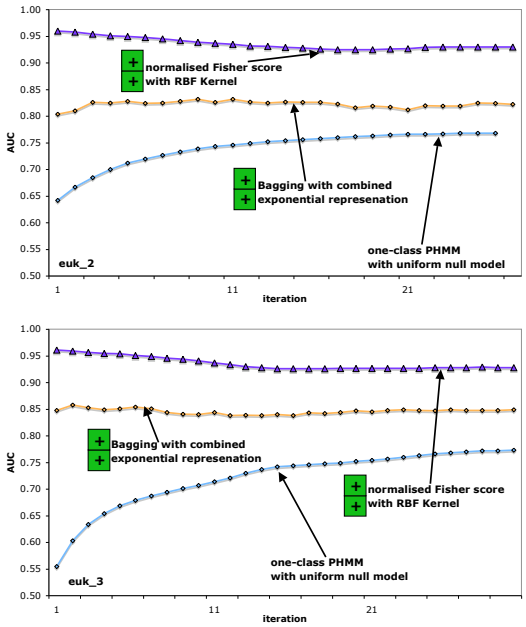
Figure C.14. *AUC values for enzyme_15 and enzyme_16 following the description from the beginning of Section C.2.*



	first Iteration	last Iteration	first Iteration
last iteration	**3.49E-07**		
first iteration	**4.82E-38**	**2.41E-24**	
last iteration	**5.45E-60**	**2.25E-42**	**0.002**

	first Iteration	last Iteration	first Iteration
last iteration	**4.37E-05**		
first iteration	**5.22E-09**	0.051	
last iteration	**5.99E-09**	**0.030**	0.609

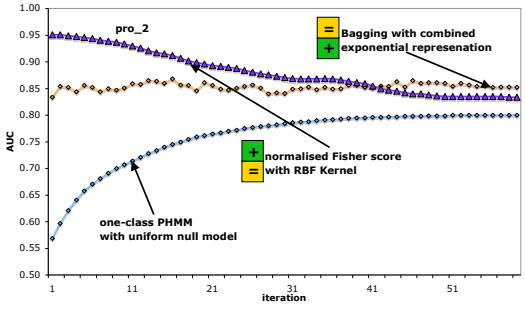
Figure C.15. *AUC values for euk_0 and euk_1 following the description from the beginning of Section C.2.*



	first Iteration	last Iteration	first Iteration
last iteration	**1.76E-07**		
first iteration	**4.32E-31**	**1.57E-19**	
last iteration	**1.10E-33**	**8.17E-21**	0.167

	first Iteration	last Iteration	first Iteration
last iteration	**9.98E-16**		
first iteration	**1.03E-50**	**1.87E-24**	
last iteration	**6.44E-53**	**5.41E-29**	0.883

Figure C.16. *AUC values for euk_2 and euk_3 following the description from the beginning of Section C.2.*



	first Iteration	last Iteration	first Iteration
last iteration	**1.52E-15**		
first iteration	**5.24E-14**	0.992	
last iteration	**8.64E-11**	0.292	0.264

Figure C.17. *AUC values for pro_2 following the description from the beginning of Section C.2.*

C.3 Fisher score vectors with Random Forests

The figures in this section depict AUC values in one-class PHMM classification and propositional classification with Random Forests based on Fisher score vectors. The AUC is calculated in each iteration of the Baum-Welch training algorithm. The tables on the right contain the p-values. P-values printed in bold indicate a statistically significant difference in AUC at the 0.05 level. The meaning of the squares in the left-hand graphs is defined in Section 5.2.

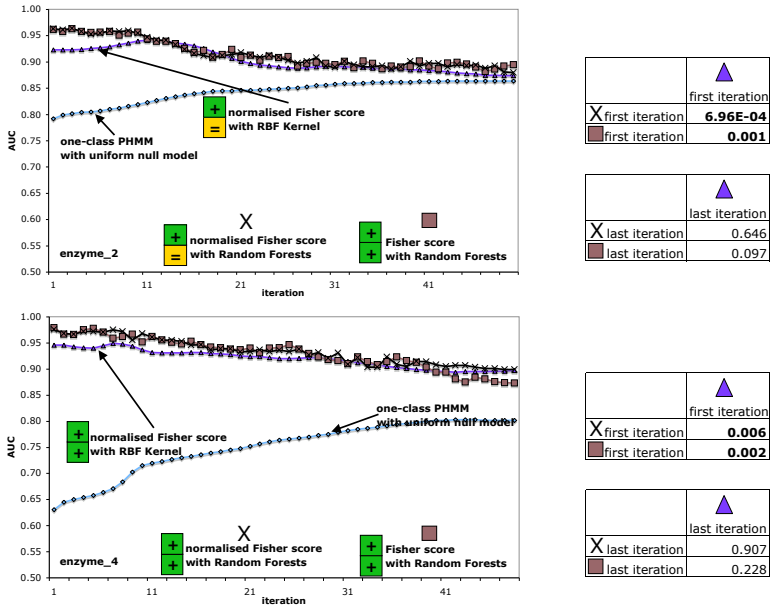


Figure C.18. *AUC values for enzyme_2 and enzyme_4 following the description from the beginning of Section C.3.*

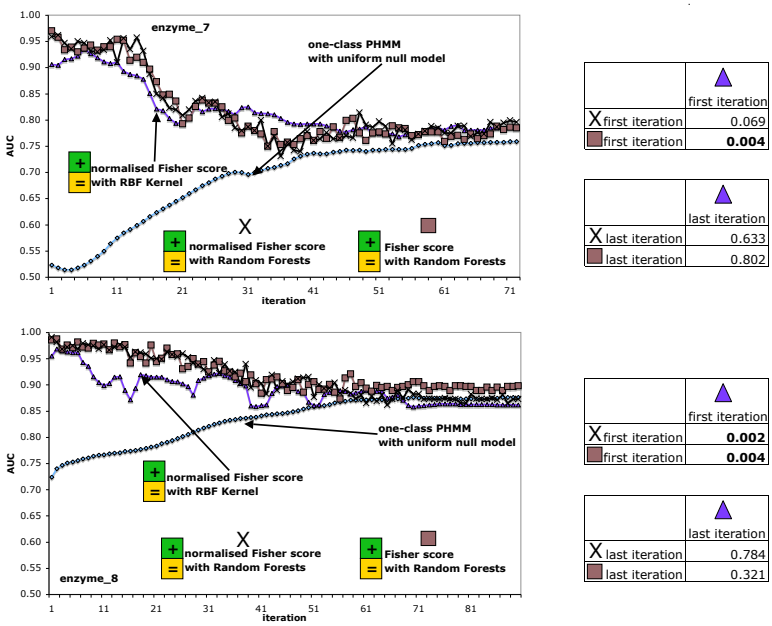


Figure C.19. *AUC values for enzyme_7 and enzyme_8 following the description from the beginning of Section C.3.*

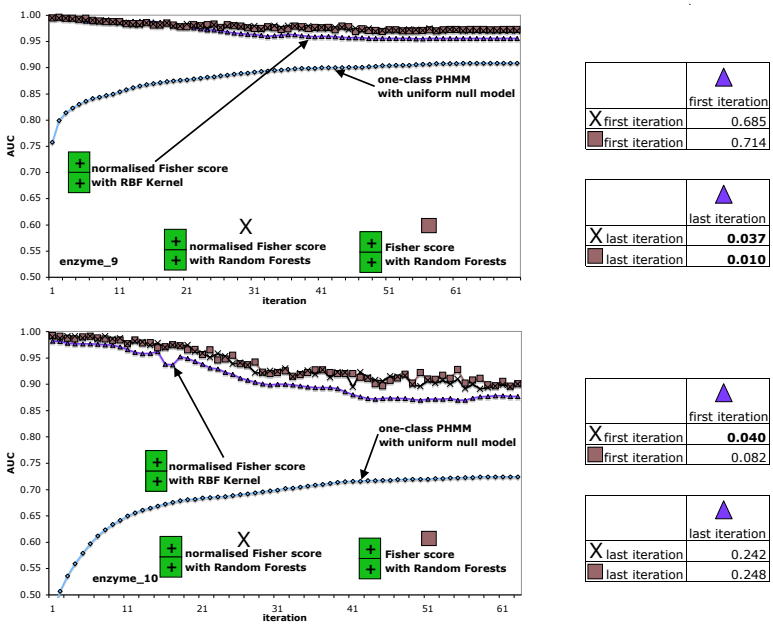


Figure C.20. *AUC values for enzyme_9 and enzyme_10 following the description from the beginning of Section C.3.*

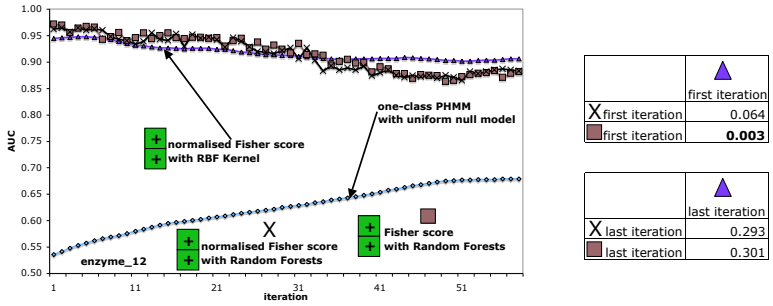


Figure C.21. *AUC values for enzyme_12 following the description from the beginning of Section C.3.*

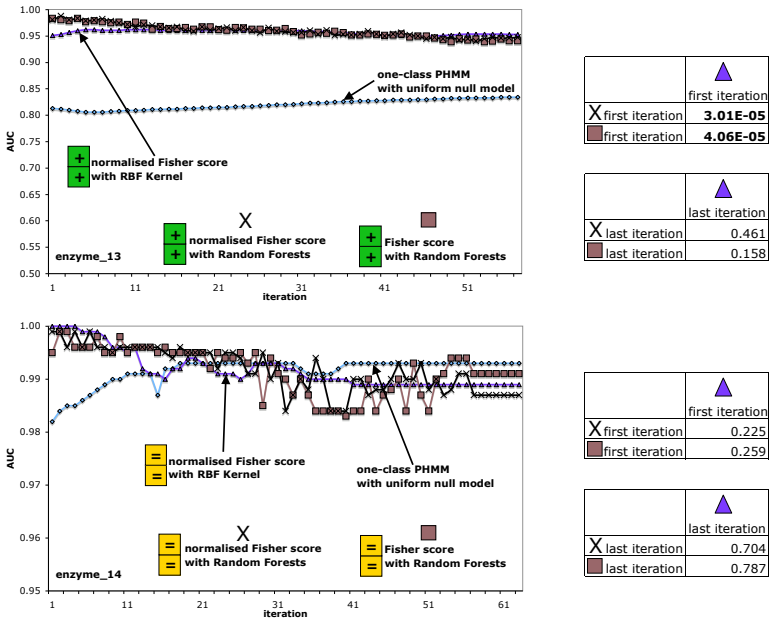


Figure C.22. *AUC values for enzyme_13 and enzyme_14 following the description from the beginning of Section C.3.*

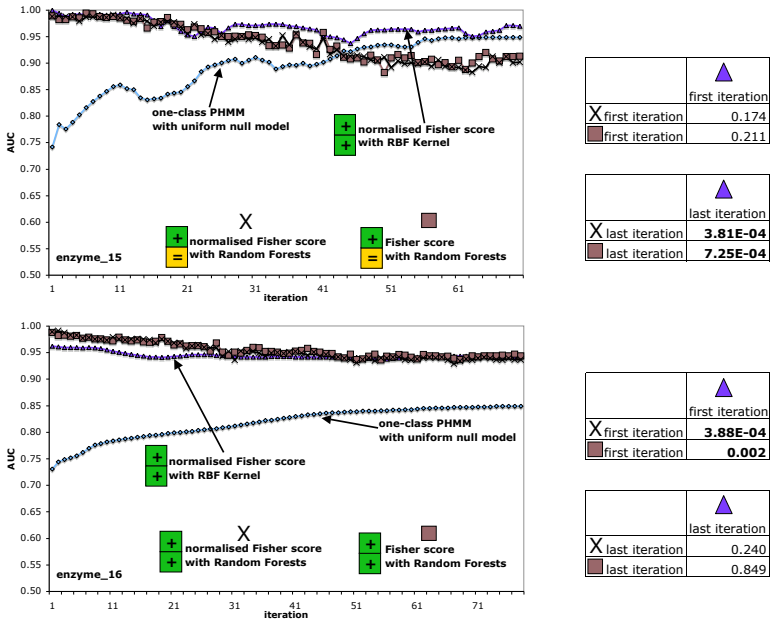


Figure C.23. *AUC values for enzyme_15 and enzyme_16 following the description from the beginning of Section C.3.*

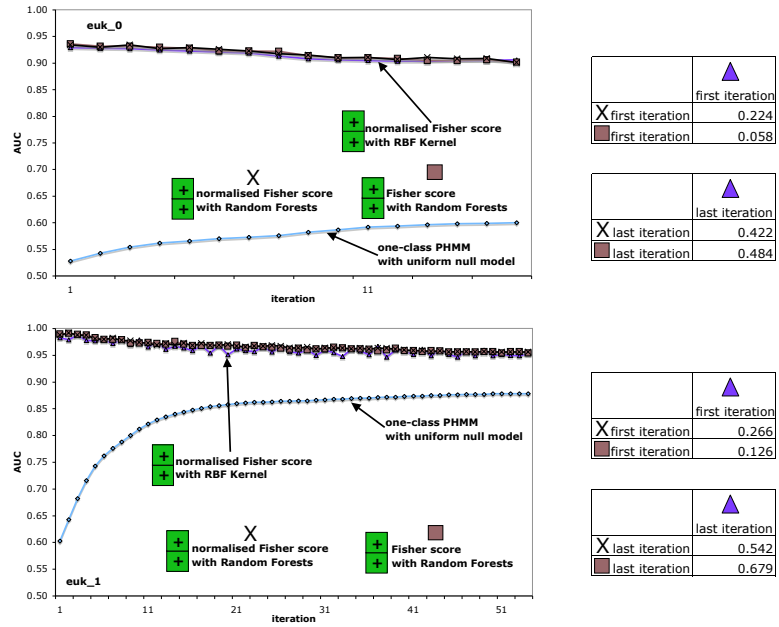


Figure C.24. *AUC values for euk_0 and euk_1 following the description from the beginning of Section C.3.*

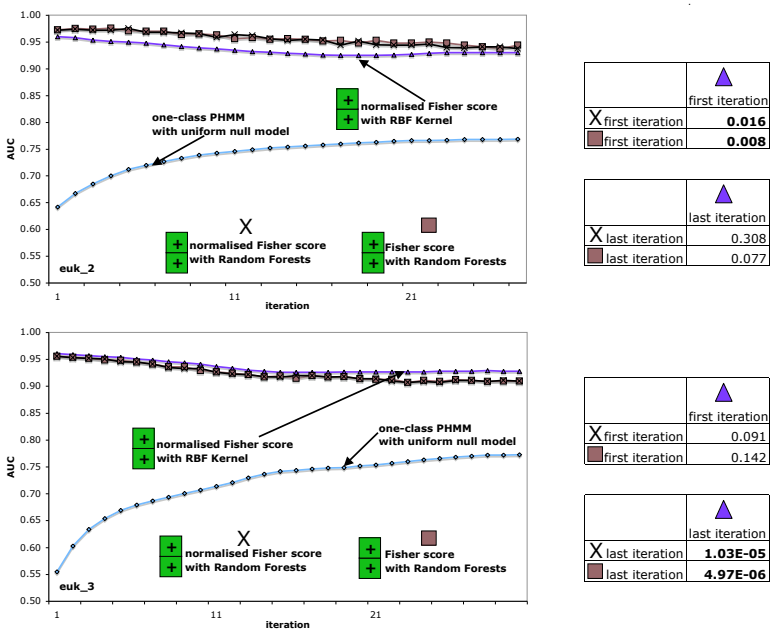
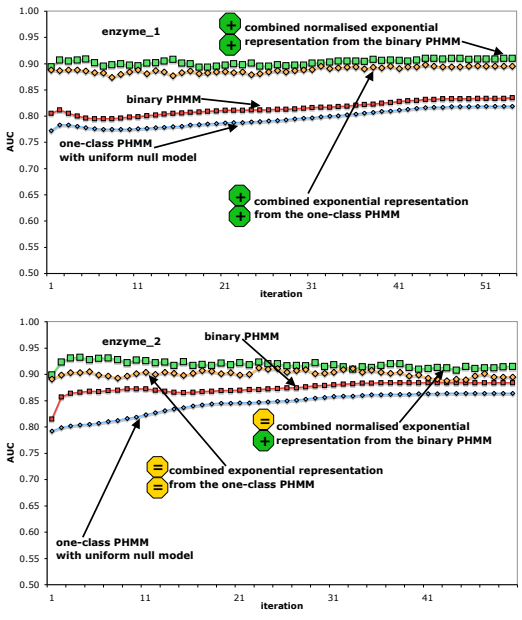


Figure C.25. *AUC values for euk_2 and euk_3 following the description from the beginning of Section C.3.*

Appendix D

Additional results for Chapter 6

The figures depict AUC values in one-class and binary PHMM classification as well as bagging using the combined exponential and the combined normalised exponential representation. The AUC is calculated in each iteration of the Baum-Welch training algorithm. The tables on the right contain the p-values. P-values printed in bold indicate a statistically significant difference in AUC at the 0.05 level. The meaning of the squares in the left-hand graphs is defined in Section 6.2.



	first iteration
first iteration	0.105
last iteration	0.914

	last iteration
first iteration	**0.033**
last iteration	0.079

	first iteration
first iteration	0.427
last iteration	0.740

	last iteration
first iteration	0.059
last iteration	**0.035**

Figure D.1. *AUC values for enzyme_1 and enzyme_2 following the description from the beginning of Appendix D.*

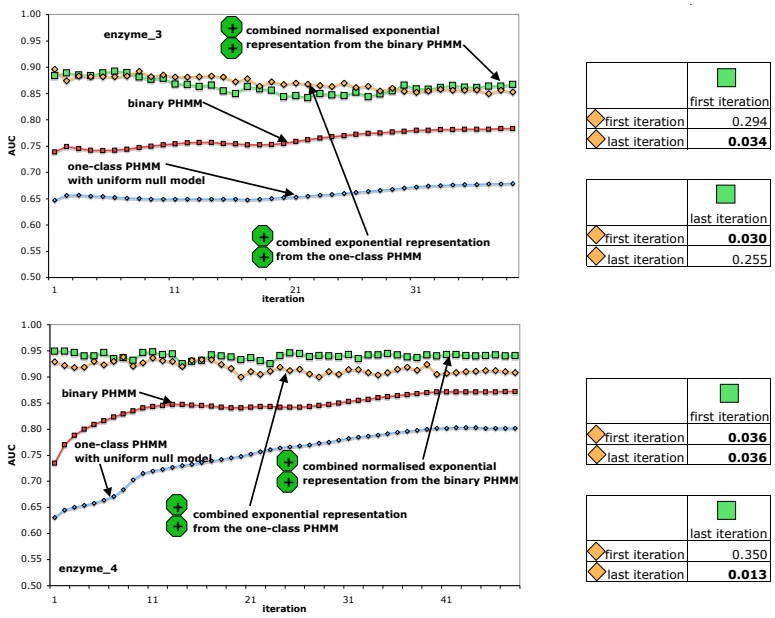


Figure D.2. *AUC values for enzyme_3 and enzyme_4 following the description from the beginning of Appendix D.*

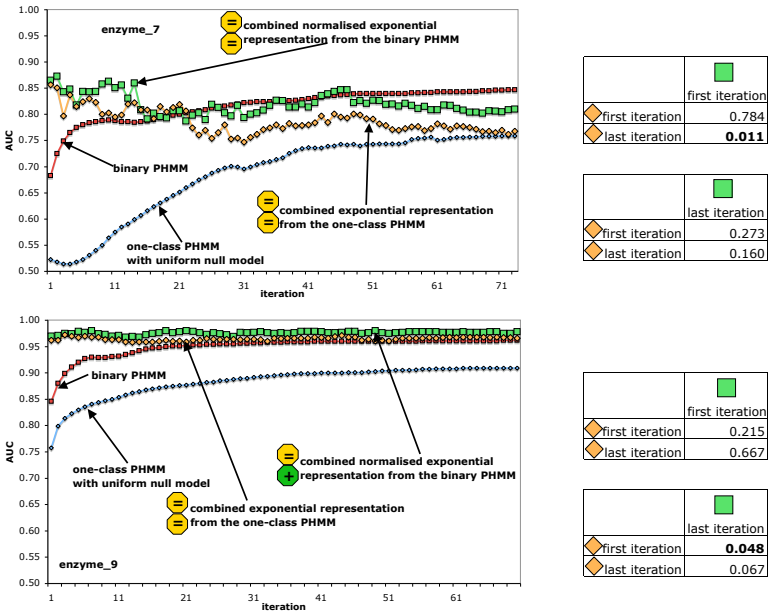


Figure D.3. *AUC values for enzyme_7 and enzyme_9 following the description from the beginning of Appendix D.*

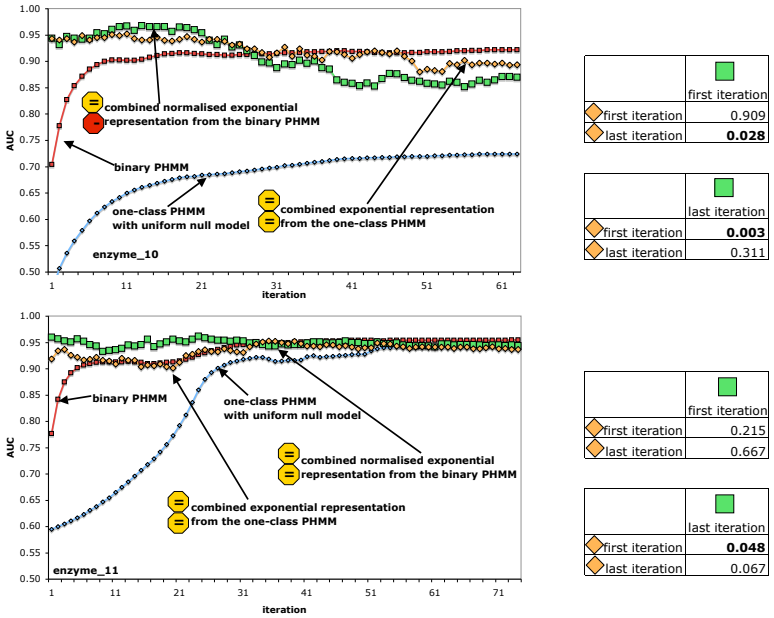


Figure D.4. *AUC values for enzyme_10 and enzyme_11 following the description from the beginning of Appendix D.*

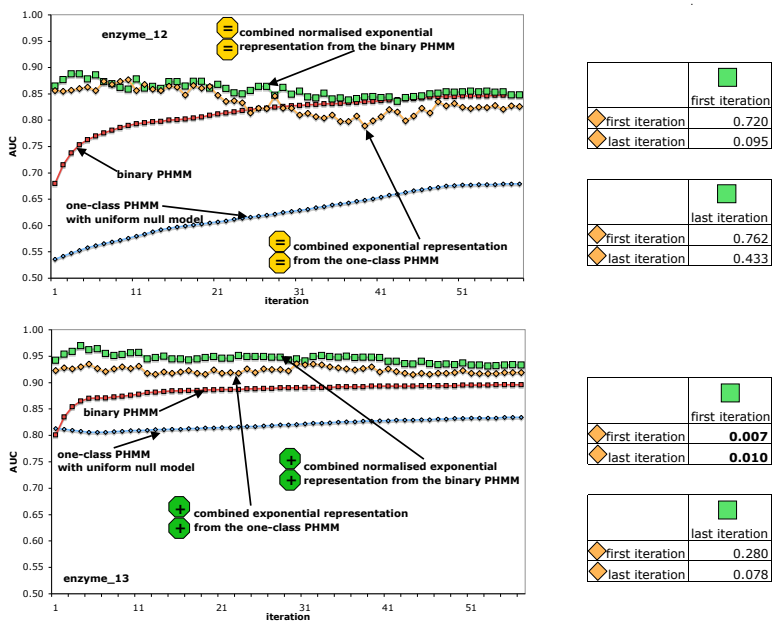


Figure D.5. *AUC values for enzyme_12 and enzyme_13 following the description from the beginning of Appendix D.*

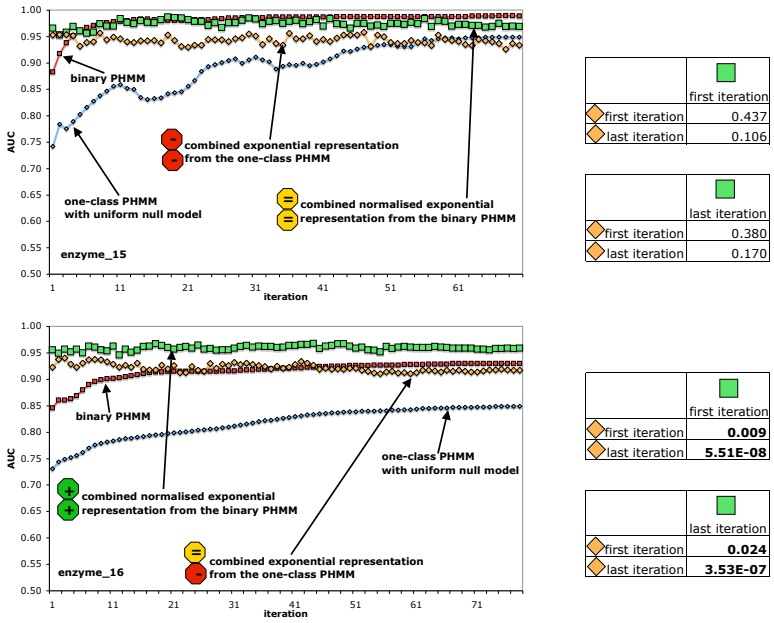


Figure D.6. *AUC values for enzyme_15 and enzyme_16 following the description from the beginning of Appendix D.*

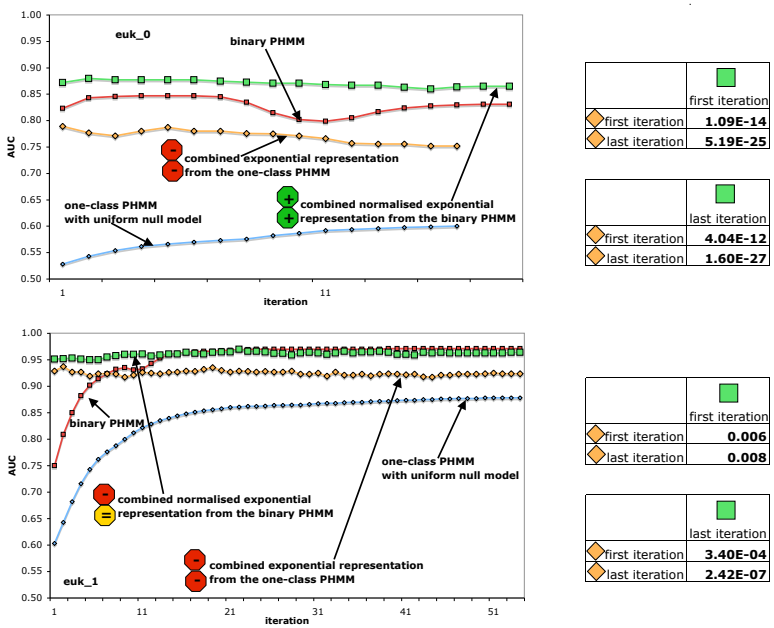


Figure D.7. *AUC values for euk_0 and euk_1 following the description from the beginning of Appendix D.*

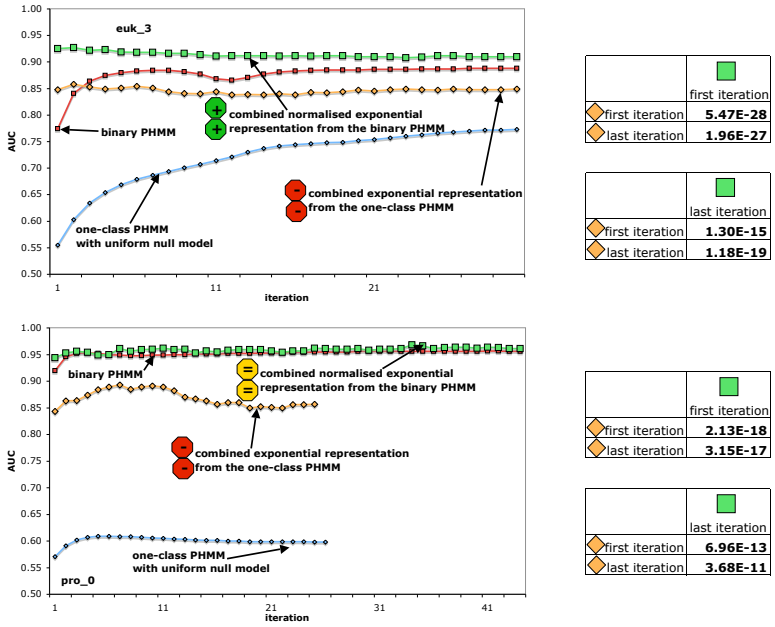


Figure D.8. *AUC values for euk_3 and pro_0 following the description from the beginning of Appendix D.*

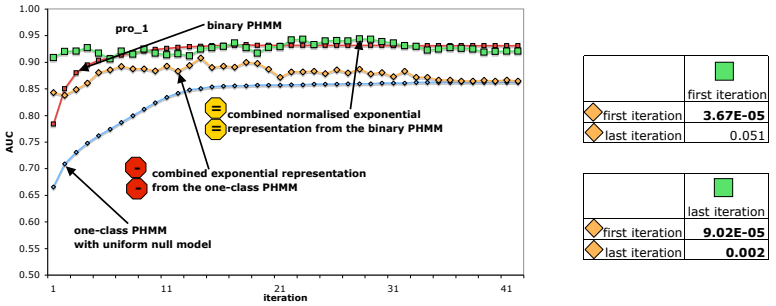


Figure D.9. *AUC values for pro_1 following the description from the beginning of Appendix D.*

Bibliography

D. W. Aha, D. Kibler, and M. K. Albert. Instance-Based Learning Algorithms. *Machine learning*, 6(1):37–66, 1991. doi: 10.1023/A:1022689900470.

S. F. Altschul, W. Gish, W. Miller, E. Myers, and D. Lipman. Basic local alignment search tool. *Journal of Molecular Biology*, 215(3):403–410, 1990. doi: 10.1016/S0022-2836(05)80360-2.

S. F. Altschul, T. L. Madden, A. A. Schäffer, J. Zhang, Z. Zhang, W. Miller, and D. J. Lipman. Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Research*, 25(17):3389–3402, 1997. doi: 10.1093/nar/25.17.3389.

R. Apweiler, A. Bairoch, C. H. Wu, W. C. Barker, B. Boeckmann, S. Ferro, E. Gasteiger, H. Huang, R. Lopez, M. Magrane, M. J. Martin, D. A. Natale, C. O'Donovan, N. Redaschi, and L. L. Yeh. UniProt: the Universal Protein Knowledgebase. *Nucleic Acids Research*, 32(Database-Issue):D115–D119, 2004. doi: 10.1093/nar/gkh131.

A. Bánhalmi, A. Kocsor, and R. Busa-Fekete. Counter-Example Generation-Based One-Class Classification. In *Proceedings of the 18th European Conference on Machine Learning (ECML)*, pages 543–550, 2007. doi: 10.1007/978-3-540-74958-5_51.

A. Bánhalmi, R. Busa-Fekete, and B. Kégl. A One-Class Classification Approach for Protein Sequences and Structures. In *Proceedings of the 5th International Symposium on Bioinformatics Research and Applications (ISBRA)*, pages 310–322, 2009. doi: 10.1007/978-3-642-01551-9_30.

C. Barrett, R. Hughey, and K. Karplus. Scoring hidden Markov models. *Computer applications in the biosciences : CABIOS*, 13(2):191–199, 1997. doi: 10.1093/bioinformatics/13.2.191.

L. Baum. An equality and associated maximization technique in statistical estimation for probabilistic functions of Markov processes. *Inequalities*, 3:1–8, 1972.

A. Biegert and J. Söding. A Boost for Sequence Searching. *BIOforum Europe*, 10:26–27, 2009.

K. M. Borgwardt, C. S. Ong, S. Schönauer, S. V. N. Vishwanathan, A. J. Smola, and H. Kriegel. Protein function prediction via graph kernels. *Bioinformatics*, 21(Suppl. 1):i47–i56, 2005. doi: 10.1093/bioinformatics/bti1007.

A. P. Bradley. The use of the area under the ROC curve in the evaluation of machine learning algorithms. *Pattern Recognition*, 30(7):1145–1159, 1997. doi: 10.1016/S0031-3203(96)00142-2.

U. Brefeld, C. Büscher, and T. Scheffer. Multi-view Discriminative Sequential Learning. In *Proceedings of the 16th European Conference on Machine Learning (ECML)*, pages 60–71, 2005. doi: 10.1007/11564096_11.

L. Breiman. Random Forests. *Machine Learning*, 45(1):5–32, 2001. doi: 10.1023/A:1010933404324.

L. Breiman. Bagging Predictors. *Machine Learning*, 24(2):123–140, 1996. doi: 10.1023/A:1018054314350.

A. Brown and G. E. Hinton. Products of Hidden Markov Models. Technical Report GCNU TR 2000-008, Gatsby Computational Neuroscience Unit, University College London (UCL), 2000. URL `http://www.gatsby.ucl.ac.uk/publications/tr/tr00-008.html`.

M. Brown, R. Hughey, A. Krogh, I. S. Mian, K. Sjölander, and D. Haussler. Using Dirichlet Mixture Priors to Derive Hidden Markov Models for Protein Families. In *Proceedings of the 1st International Conference on Intelligent Systems for Molecular Biology (ISMB)*, pages 47–55, 1993.

C. Bystroff, V. Thorsson, and D. Baker. HMMSTR: a hidden Markov model for local sequence-structure correlations in proteins. *Journal of Molecular Biology*, 301(1):173– 190, 2000. doi: 10.1006/jmbi.2000.3837.

C-C. Chang and C-J. Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

N. V. Chawla, N. Japkowicz, and A. Kołcz. Editorial: Special Issue on Learning from imbalanced data sets. *ACM SIGKDD Explorations Newsletter*, 6(1), 2004. doi: 10.1145/1007730.1007733.

C. Cherry. A General Survey of Hidden Markov Models in Bioinformatics, 2001. URL `http://webdocs.cs.ualberta.ca/~colinc/projects/606project.ps`. [Online; accessed May 2010].

K. C. Chou and D. W. Elrod. Prediction of Enzyme Family Classes. *Journal of Proteome Research*, 2(2):183–189, 2003. doi: 10.1021/pr0255710.

K. C. Chou and D. W. Elrod. Protein subcellular location prediction. *Protein Engineering*, 12(2):107–118, 1999. doi: 10.1093/protein/12.2.107.

D. A. Clifton, P. R. Bannister, and L. Tarassenko. A Framework for Novelty Detection in Jet Engine Vibration Data. *Key Engineering Materials*, 347:305–310, 2007. doi: 10.4028/www.scientific.net/KEM.347.305.

The UniProt Consortium. The Universal Protein Resource (UniProt) in 2010. *Nucleic Acids Research*, 38(Database-Issue):D142–D148, 2010. doi: 10.1093/nar/gkp846.

E. R. DeLong, D. M. DeLong, and D. L. Clarke-Pearson. Comparing the Areas under Two or More Correlated Receiver Operating Characteristic Curves: A Nonparametric Approach. *Biometrics*, 44(3):837–845, 1988.

J. Demšar. Statistical Comparisons of Classifiers over Multiple Data Sets. *The Journal of Machine Learning Research*, 7:1–30, 2006.

M. des Jardins, P. D. Karp, M. Krummenacker, T. J. Lee, and C. A. Ouzounis. Prediction of Enzyme Classification from Protein Sequence without the use of Sequence Similarity. In *Proceedings of the 5th International Conference on Intelligent Systems for Molecular Biology (ISMB)*, pages 92–99, 1997.

U. Dick and K. Kersting. Fisher Kernels for Relational Data. *Proceedings of the 17th European Conference on Machine Learning (ECML)*, pages 114–125, 2006. doi: 10.1007/11871842_15.

M. Doderer, K. Yoon, J. Salinas, and S. Kwek. Protein subcellular localization prediction using a hybrid of similarity search and error-correcting output code techniques that produces interpretable results. *In Silico Biology*, 6(5):419–33, 2006.

R. F. Doolittle. Similar amino acid sequences: chance or common ancestry? *Science*, 214(4517):149–159, 1981. doi: 10.1126/science.7280687.

C. Drummond. Discriminative vs. Generative Classifiers for Cost Sensitive Learning. In *Proceedings of 19th Conference of the Canadian Society for Computational Studies of Intelligence (Canadian AI)*, pages 479–490, 2006. doi: 10.1007/11766247_41.

C. Drummond and R. C. Holte. C4.5, Class Imbalance, and Cost Sensitivity: Why Under-sampling beats Over-sampling. In *Proceedings of the 20th International Conference on Machine Learning (ICML): Workshop on Learning from Imbalanced Data Sets (II)*, pages 1–8, 2003.

R. Durbin, S. Eddy, A. Krogh, and G. Mitchison. *Biological sequence analysis: probabilistic models of proteins and nucleic acids*. Cambridge University Press, Cambridge, England, 1998.

S. R. Eddy. Profile hidden Markov models. *Bioinformatics*, 14(9):755–763, 1998. doi: 10.1093/bioinformatics/14.9.755.

O. Emanuelsson. Predicting protein subcellular localisation from amino acid sequence information. *Briefings in Bioinformatics*, 3(4):361–376, 2002. doi: 10.1093/bib/3.4.361.

R. D. Finn, J. Mistry, J.Tate, P. Coggill, A. Heger, J. E. Pollington, O. L. Gavin, P. Gunasekaran, G. Ceric, K. Forslund, L. Holm, E. L. L. Sonnhammer, S. R. Eddy, and A. Bateman. The Pfam protein families database. *Nucleic Acids Research*, 38(Database-Issue):D211–D222, 2010. doi: 10.1093/nar/gkp985.

E. Frank, M. Hall, L. Trigg, G. Holmes, and I. H. Witten. Data mining in bioinformatics using Weka. *Bioinformatics*, 20(15):2479–2481, 2004. doi: 10.1093/bioinformatics/bth261.

E. Frank, M. Hall, G. Holmes, R. Kirkby, B. Pfahringer, and I. H. Witten. Weka: A machine learning workbench for data mining. In O. Maimon and L. Rokach, editors, *Data Mining and Knowledge Discovery Handbook: A Complete Guide for Practitioners and Researchers*, pages 1305–1314. Springer, Berlin, Germany, 2005.

Y. Freund and R. E. Schapire. Experiments with a new boosting algorithm. In *Proceedings of the 13th International Conference on Machine Learning*, pages 148–156, 1996.

T. Gärtner. A survey of kernels for structured data. *ACM SIGKDD Explorations Newsletter*, 5(1):49–58, 2003. doi: 10.1145/959242.959248.

N. J. Gotelli. Research frontiers in null model analysis. *Global Ecology & Biogeography*, 10(4):337–343, 2001. doi: 10.1046/j.1466-822X.2001.00249.x.

J. Guo, Y. Lin, and Z. Sun. A novel method for protein subcellular localization based on boosting and probabilistic neural network. In *Proceedings of the 2nd Conference on Asia-Pacific bioinformatics (APBC)*, pages 21–27, 2004.

M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten. The WEKA data mining software: an update. *ACM SIGKDD Explorations Newsletter*, 11(1):10–18, 2009. doi: 10.1145/1656274.1656278.

D. J. Hand. Measuring classifier performance: a coherent alternative to the area under the ROC curve. *Machine Learning*, 77(1):103–123, 2009. doi: 10.1007/s10994-009-5119-5.

J. A. Hanley and B. J. McNeil. A method of comparing the areas under receiver operating characteristic curves derived from same cases. *Radiology*, 148:839–843, 1983.

D. Haussler, A. Krogh, S. Mian, and K. Sjölander. Protein modeling using hidden Markov models: analysis of globins. In *Proceedings of the 26th Annual Hawaii International Conference on System Sciences*, pages 792–802, 1993. doi: 10.1109/HICSS.1993.270611.

X. He, R. S. Zemel, and M. Á. Carreira-Perpiñán. Multiscale Conditional Random Fields for Image Labeling. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 695–702, 2004. doi: 10.1109/CVPR.2004.1315232.

K. Hempstalk. *Continuous Typist Verification using Machine Learning*. PhD thesis, The University of Waikato, 2009. URL `http://hdl.handle.net/10289/3282`.

G. E. Hinton. Training Products of Experts by Minimizing Contrastive Divergence. *Neural Computation*, 14(8):1771–1800, 2002. doi: 10.1162/089976602760128018.

S. Hua and Z. Sun. Support vector machine approach for protein subcellular localization prediction. *Bioinformatics*, 17(8):721–728, 2001. doi: 10.1093/bioinformatics/17.8.721.

Y. Huang and C. Bystroff. Improved pairwise alignments of proteins in the Twilight Zone using local structure predictions. *Bioinformatics*, 22(4):413–422, 2006. doi: 10.1093/bioinformatics/bti828.

R. Hughey, K. Karplus, and A. Krogh. Sam: Sequence alignment and modeling software system. Technical Report UCSC-CRL-99-11, Baskin Center for Computer Engineering and Science, University of California, July 2003. URL `http://compbio.soe.ucsc.edu/papers/sam_doc.ps.gz`.

T. Jaakkola and D. Haussler. Exploiting generative models in discriminative classifiers. *Proceedings of the 1998 conference on Advances in neural information processing systems II*, pages 487–493, 1999.

T. Jaakkola, M. Diekhans, and D. Haussler. Using the Fisher Kernel Method to Detect Remote Protein Homologies. In *Proceedings of the 7th International Conference on Intelligent Systems for Molecular Biology (ISMB)*, pages 149–158, 1999.

T. Jaakkola, M. Diekhans, and D. Haussler. A Discriminative Framework for Detecting Remote Protein Homologies. *Journal of Computational Biology*, 7 (1-2):95–114, 2000. doi: 10.1089/10665270050081405.

F. Jacob. Evolution and Tinkering. *Science*, 196(4295):1161–1166, 1977. doi: 10.1126/science.860134.

L. Jaroszewski, W. Li, and A. Godzik. In search for more accurate alignments in the twilight zone. *Protein Science*, 11(7):1702–1713, 2002. doi: 10.1110/ps.4820102.

G. H. John and P. Langley. Estimating continuous distributions in Bayesian classifiers. In *Proceedings of the 11th Conference on Uncertainty in Artificial Intelligence (UAI)*, volume 1, pages 338–345, 1995.

R. Karchin and R. Hughey. Weighting hidden Markov models for maximum discrimination. *Bioinformatics*, 14(9):772–782, 1998. doi: 10.1093/bioinformatics/14.9.772.

K. Karplus, C. Barrett, and R. Hughey. Hidden Markov models for detecting remote protein homologies. *Bioinformatics*, 14(10):846–856, 1998. doi: 10.1093/bioinformatics/14.10.846.

K. Karplus, R. Karchin, G. Shackelford, and R. Hughey. Calibrating E-values for hidden Markov models using reverse-sequence null models. *Bioinformatics*, 21(22):4107–4115, 2005. doi: 10.1093/bioinformatics/bti629.

E. Keogh, J. Lin, and A. Fu. HOT SAX: efficiently finding the most unusual time series subsequence. In *Proceedings of the 5th IEEE International Conference on Data Mining (ICDM)*, pages 226–233, 2005. doi: 10.1109/ICDM.2005.79.

S. Kirkpatrick, D. Gelatt Jr., and M. P. Vecchi. Optimization by Simmulated Annealing. *Science*, 220(4598):671–680, 1983. doi: 10.1126/science.220.4598.671.

S. Kramer. Relational learning vs. propositionalization: Investigations in inductive logic programming and propositional machine learning. *AI communications*, 13(4):275–276, 2000.

S. Kramer, N. Lavrač, and P. Flach. Propositionalization approaches to relational data mining. In S. Džeroski and N. Lavrač, editors, *Relational Data Mining*, pages 262–286. Springer, New York, NY, 2001.

A. Krogh, M. Brown, I. S. Mian, K. Sjölander, and D. Haussler. Hidden Markov Models in Computational Biology: Applications to Protein Modeling. *Journal of Molecular Biology*, 235(5):1501–1531, 1994. doi: 10.1006/jmbi.1994.1104.

P. Larrañaga, B. Calvo, R. Santana, C. Bielza, J. Galdiano, I. Inza, J. A. Lozano, R. Armañanzas, G. Santafé, A. Pérez, and V. Robles. Machine learning in bioinformatics. *Briefings in Bioinformatics*, 7(1):86–112, 2006. doi: 10.1093/bib/bbk007.

L. Liao and W. S. Noble. Combining pairwise sequence similarity and support vector machines for remote protein homology detection. In *Proceedings of the 6th Annual International Conference on Computational Biology (RECOMB)*, pages 225–232, 2002. doi: 10.1145/565196.565225.

M. Madera and J. Gough. A comparison of profile Hidden Markov Model procedures for remote homology detection. *Nucleic Acids Research*, 30(19):4321–4328, 2002. doi: 10.1093/nar/gkf544.

C. E. Metz. Basic principles of ROC analysis. *Seminars in Nuclear Medicine*, 8 (4):283–298, 1978. doi: 10.1016/S0001-2998(78)80014-2.

T. Minka. Discriminative models, not discriminative training. Technical Report TR-2005-144, Microsoft Research, Cambridge, UK, 2005.

S. Mutter, B. Pfahringer, and G. Holmes. The Positive Effects of Negative Information: Extending One-Class Classification Models in Binary Proteomic Sequence Classification. In *Proceedings of the 22nd Australasian Joint Conference on Artificial Intelligence (AI)*, pages 260–269, 2009. doi: 10.1007/978-3-642-10439-8_27.

K. Nakai. Protein sorting signals and prediction of subcellular localization. *Advances in Protein Chemistry*, 54:277–343, 2000. doi: 10.1016/S0065-3233(00)54009-1.

A. Y. Ng and M. I. Jordan. On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. In *Advances in Neural Information Processing Systems (NIPS) 14*, 2002.

C. Notredame. Recent progresses in multiple sequence alignment: a survey. *Pharmacogenomics*, 3(1):131–144, 2002. doi: 10.1517/14622416.3.1.131.

J. Park, K. Karplus, C. Barrett, R. Hughey, D. Haussler, T. Hubbard, and C. Chothia. Sequence comparisons using multiple sequences detect three times as many remote homologues as pairwise methods. *Journal of Molecular Biology*, 284(4):1201–1210, 1998. doi: 10.1006/jmbi.1998.2221.

R. K. Pearson. *Mining Imperfect Data: Dealing with Contamination and Incomplete Records*. SIAM: Society for Industrial and Applied Mathematics, Philadelphia, PA, 2005.

J. Platt. Fast training of support vector machines using sequential minimal optimization. In B. Schhölkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*. MIT Press, Cambridge, MA, 1998.

L. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989. doi: 10.1109/5.18626.

A. Reinhardt and T. Hubbard. Using neural networks for prediction of the subcellular location of proteins. *Nucleic Acids Research*, 26(9):2230–2236, 1998. doi: 10.1093/nar/26.9.2230.

S. Rosset. Model Selection via the AUC. In *Proceedings of the 21st International Conference on Machine Learning (ICML)*, page 89, 2004. doi: 10.1145/1015330.1015400.

H. Saigo, J-P. Vert, N. Ueda, and T. Akutsu. Protein homology detection using string alignment kernels. *Bioinformatics*, 20(11):1682–1689, 2004. doi: 10.1093/bioinformatics/bth141.

B. Schölkopf, R. C. Williamson, A. J. Smola, J. Shawe-Taylor, and J. Platt. Support vector method for novelty detection. *Advances in Neural Information Processing Systems (NIPS)*, 12:582–588, 2000.

M. Sewell. Fisher kernel, 2008. URL `http://www.svms.org/kernels/fisher/fisher-kernel.pdf`. [Online; accessed May 2010].

C. J. A. Sigrist, L. Cerutti, N. Hulo, A. Gattiker, L. Falquet, M. Pagni, A. Bairoch, and P. Bucher. PROSITE: A documented database using patterns and profiles as motif descriptors. *Briefings in Bioinformatics*, 3(3):265–274, 2002. doi: 10.1093/bib/3.3.265.

J. Söding. Protein homology detection by HMM-HMM comparison. *Bioinformatics*, 21(7):951–960, 2005. doi: 10.1093/bioinformatics/bti125.

L. Tarassenko, P. Hayton, N. Cerneaz, and M. Brady. Novelty detection for the identification of masses in mammograms. In *Proccedings of the 4th International IEEE Conference on Artificial Neural Networks*, pages 442–447, 1995. doi: 10.1049/cp:19950597.

D. Tax. *One-class classification: Concept-learning in the absence of counter-examples*. PhD thesis, Delft University of Technology, 2001.

V. N. Vapnik. *Statistical Learning Theory*. Wiley, New York, NY, 1998.

I. A. Vergara, T. Norambuena, E. Ferrada, A. W. Slater, and F. Melo. StAR: a simple tool for the statistical comparison of ROC curves. *BMC Bioinformatics*, 9:265, 2008. doi: 10.1186/1471-2105-9-265.

J. T. L. Wang, Q. Ma, D. Shasha, and C. H. Wu. New techniques for extracting features from protein sequences. *IBM Systems Journal*, 40(2):426–441, 2001. doi: 10.1147/sj.402.0426.

Wikipedia. Logistic function — Wikipedia, the free encyclopedia, 2010. URL `http://en.wikipedia.org/wiki/File:Logistic-curve.svg`. [Online; released into the public domain by its author Qef; accessed June 2010].

O. Yakhnenko, A. Silvescu, and V. Honavar. Discriminatively trained Markov model for sequence classification. In *Proceedings of the 5th IEEE International Conference on Data Mining (ICDM)*, pages 498–505, 2005. doi: 10.1109/ICDM.2005.52.

G. Yona and M. Levitt. Within the twilight zone: a sensitive profile-profile comparison tool based on information theory. *Journal of Molecular Biology*, 315 (5):1257–1275, 2002. doi: 10.1006/jmbi.2001.5293.

K. Yoon. One-Class Learning: Bioinformatics & Machine Learning, 2005. URL `http://cse.spsu.edu/clo/teaching/cs7123/Fall2005/KihoonYoon.pdf`. [Online; accessed May 2010].