

# Comparing Approaches to Predict Transmembrane Domains in Protein Sequences

Paul Davidsson  
Blekinge Institute of Technology  
Systems and Software Engineering  
372 25, Ronneby, Sweden  
+46 457 385841  
paul.davidsson@bth.se

Johan Hagelbäck  
Travelstart Nordic  
Bredgatan 19  
25225, Helsingborg, Sweden  
+46 709 584405  
johan@travelstart.se

Kenny Svensson  
Ericsson AB  
Box 518  
371 23 Karlskrona, Sweden  
+46 455 395977  
kenny.svensson@ericsson.com

## ABSTRACT

There are today several systems for predicting transmembrane domains in membrane protein sequences. As they are based on different classifiers as well as different pre- and post-processing techniques, it is very difficult to evaluate the performance of the particular classifier used. We have developed a system called MemMiC for predicting transmembrane domains in protein sequences with the possibility to choose between different approaches to pre- and post-processing as well as different classifiers. Therefore it is possible to compare the performance of each classifier in a certain environment as well as the different approaches to pre- and post-processing. We have demonstrated the usefulness of MemMiC in a set of experiments, which shows, e.g., that the performance of a classifier is very dependent on which pre- and post-processing techniques are used.

## Categories and Subject Descriptors

H.2.8 [Database Management]: Database Applications – *Data mining*.

## General Terms

Algorithms, Measurement, Performance, Experimentation.

## Keywords

Learning, Classifiers, Protein sequences.

## 1. INTRODUCTION

There exist today several systems for locating transmembrane domains in membrane proteins. These systems are based on different classifiers, neural networks and hidden Markov models being the most widely used [7]. They are also using different types of pre-processing, e.g., transformation of the primary sequence into a numerical sequence using different hydrophobicity scales or a combination of charge bias and a hydrophobicity scale. These

systems are using different data sets and different approaches to pre-processing, generalization and post-processing. Because of this it is hard to obtain information about the performance of different techniques used, which make it difficult to compare for example classifiers or approaches to post-processing.

In order to be able to do fair comparisons, we have developed a system, MemMiC, for predicting transmembrane domains where it is possible to choose between some of the most widely used algorithms, pre- and post-processing techniques. MemMiC is available through a web interface at: [www.ipd.bth.se/memmic](http://www.ipd.bth.se/memmic).

In the next chapter we describe some previous approaches to predicting transmembrane domains. This is followed by a description of MemMiC and the experiments performed.

## 2. RELATED WORK

Krogh et al. [7] have developed a system for predicting transmembrane domains and the full topology of membrane proteins called TMHMM. It is using hydrophobicity and charge bias in the pre-processing and a hidden Markov model as classifier (which hydrophobicity scale and learning algorithm used are not mentioned in their article). The authors mention that their post-processing includes an expectation of the most probable number of transmembrane domains in each protein (it is however not clear how this is performed). In the experiments presented in their article, TMHMM was able to predict more than 96% of the actual transmembrane domains in the data set.

Tusnády and Simon [11] have developed a system called HMMTOP. It is based on the difference in amino acid distributions as pre-processing and a hidden Markov model as classifier. The hidden Markov model uses predetermined initial values which are optimized in the training part and the Viterbi algorithm for classifying instances. There are no information about which algorithm is used for optimizing the HMM parameters. In their experiments HMMTOP was able to predict more than 98% of the known transmembrane domains. There is no information in their article about post-processing of the result.

Another system is developed by Chen et al. [3] is using the Kyte-Doolittle scale [8] in the pre-processing and a learning vector quantization network (LVQ) as classifier. It is however not clear how their post-processing is performed. In the experiments presented in their article, the system was able to predict 91% of the actual transmembrane domains in the data set.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

SAC'05, March 13-17, 2005, Santa Fe, New Mexico, USA.  
Copyright 2005 ACM 1-58113-964-0/05/0003...\$5.00.

Zhou and Zhou [12] have developed a system called THUMBUP that is based on a hidden Markov model and mean burial propensity in the pre-processing. It uses a five step post-processing.

1. Segments shorter than a cut-off value are deleted. Segments are also deleted if the pre-processing values of the residues in each segment are below a cut-off value.
2. Two segments are merged if they are separated by a sequence distance lower than a cut-off value and the length of both segments are below 19 residues. Two segments are also merged if one segment is longer than 19 residues and the other has less than 10 residues.
3. A segment is split into two segments if its length is greater than or equals  $2 * 19$  residues plus a cut off value.
4. A segment is deleted if it starts before a cut-off index.
5. Determining the orientation of the amino terminus.

In the experiments presented in their article, the system was able to locate 88% of the known transmembrane domains. The same experiment was performed using TMHMM and HMMTOP which was able to locate 68% and 73% respectively.

Despite that we know the accuracy of these systems it is hard to compare them. For instance, how much depends on the data sets used for training and validation and how much depends on the classifier used? In addition the pre- and post-processing used are different and are often not described in such detail that it is possible to reproduce the experiments. This makes it very difficult to know how much the different components (i.e., pre-processing, learning algorithm/classifier, and post-processing) contributes to the performance of the system. The purpose of MemMiC is to make it possible to easily compare the performance of the different components (in a certain environment).

### 3. SYSTEM DESCRIPTION

MemMiC consists of three major parts which will be described in this chapter: pre-processing, classification and post-processing.

#### 3.1 Pre-processing

There are two types of pre-processing possible in MemMiC, one concerns the coding (transformation) of the primary sequence, and the other concerns the categories used during learning and classification. The primary sequence can be coded using a hydrophobicity scale or used as it is, i.e., no pre-processing of the first type. There are three hydrophobicity scales available; Kyte-Doolittle [8], Eisenberg [4] and a general consensus scale (GCS) based on 160 normalized and filtered scales [10].

As additional input in the learning phase and as output in the classification phase the four classes globular, near membrane, helix cap or helix are used. The idea of using more than two classes was inspired from the work by Krogh et al [7]. A cross-validation test using a backpropagation neural network with Kyte-Doolittle has improved the accuracy from 94.66 % to 94.84% when four classes are used. The same test using k-nearest neighbor has improved the accuracy from 91.16% to 92.11%. The rules for assigning a class to a residue are:

- *Helix*: all residues located inside a transmembrane domain and more than five residues from the border.
- *Helix cap*: all residues located at most five residues before or after a domain border.

- *Near*: residues located 5 to 15 residues after a helix cap on the outside of the transmembrane domain.
- *Globular*: all residues not part of one of the above mentioned classes.

It is possible that a residue belongs to more than one class, e.g., near one membrane and helix cap relative another. If this occurs, the classes are ranked with helix highest and globular lowest.

#### 3.2 Classification

The classification part consists of two phases; the actual training using a machine learning algorithm and the classification using the learned classifier. Four different classifiers are included in our system. We have chosen artificial neural networks, LVQ networks, k-nearest neighbor and hidden Markov models. They were chosen because we wanted to evaluate a wide range of different classifiers and most of them have been used with good results in previous studies. All classifiers except hidden Markov model use a sliding window as input; for an index  $i$  it contains all residues from  $i - (window\ size - 1)/2$  to  $i + (window\ size - 1)/2$ .

In order to make a fair comparison between the different classifiers we have devoted some time optimizing each of them. The parameters and settings for each algorithm are described in the following subsections.

##### 3.2.1 Artificial Neural Network

We have used the backpropagation algorithm to train a neural network consisting of 32 hidden nodes, 19 input nodes and 2 output nodes. The choice of hidden nodes has a great impact on the result. We have tested different amount of hidden nodes, ranging from 20 to 40, in order to find the most optimal value. We encountered that 32 was a good choice. Other experiments using backpropagation-trained neural networks for a related secondary structure prediction problem has shown that networks with 30 to 40 hidden nodes yields the best results [5].

In the case of primary sequence as pre-processing a binary input must be used, otherwise the big difference in input values (1-20) would have a great influence on the result. We have therefore used a 95 nodes input layer where each sliding window index is represented by a 5 nodes binary signal. The sliding window size used is 19 since it has been the most successful during the experiments. The output is a binary signal representing the four different instance classes (see Table 1). We have also tried to use four output nodes but the result was less accurate (93.97% compared to 95.1% with two output nodes) and is therefore not used in our experiments.

**Table 1. Binary class encoding**

Class	Value
Helix	1, 1
Helix cap	1, 0
Near	0, 1
Globular	0, 0

We have tried backpropagation both with and without momentum [9]. The experiments indicated that the momentum version has a higher mean accuracy and has therefore been used in the experiments. During the experiments we have used a learning rate of 0.2, 32 hidden nodes and a momentum of 0.3.

### 3.2.2 K-Nearest Neighbor

K-nearest neighbor is an instance-based classifier, i.e. the generalization of the training data is delayed until classification time. Therefore the classifier is very fast at training but slow when classifying instances. The output of a classification is calculated from the  $k$  training instances with the shortest Euclidean distance from the classification instance. A refinement of  $k$ -nearest neighbor is to weight each instance based on the distance between the training instance and the query instance [9].

There is a difference between standard  $k$ -nearest neighbor and our approach in how the distance between two instances is calculated when using primary sequence as input values. In this case the hamming distance is used, i.e., the number of characters that differ between two patterns [2]. The reason is that hamming distance calculation does not require a numerical encoding of each amino acid. A numerical encoding would result in that the difference between A and B in two input strings compared to A and H may not be the same. The result of a classification would therefore be dependent on the encoding, and different encodings would result in different classifications. There are two ways of calculating the result of a classification:

1. The most common class value among the  $k$  nearest training instances. This is used when the class values are discrete.
2. The mean class value from the  $k$  nearest training instances. This approach is used when the class values are real.

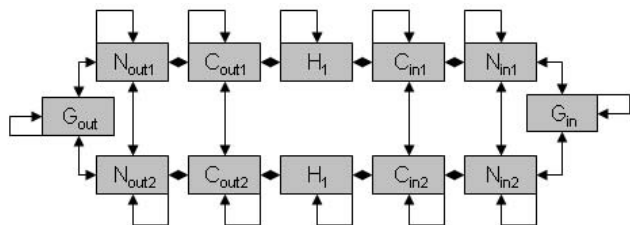
In MemMiC, we have the four discrete classes helix = 1, cap = 2, near = 3 and globular = 4. The rank order between the class values are chosen with respect to their locations in a protein. Helix is always close to helix cap, near is located next to helix cap on one side and globular on the other. The optimization we have performed has shown that even though we have discrete class values the real valued calculation produced more accurate results and is therefore used in our experiments (see Table 2). The  $k$  value has been set to 21 in all experiments because it was the most accurate value when  $k$  values from 1 to 32 were tested.

**Table 2. Accuracy when using different KNN variants**

KNN variant	K	Accuracy
1-NN	1	90.1%
Discrete valued	21	91.8%
Real valued	21	92.0%
Weighted discrete value	21	91.3%
Weighted real valued	21	92.1%

### 3.2.3 Hidden Markov Model

A Hidden Markov Model is a statistical model with states, an alphabet of symbols, and transition/emission matrices [1]. We have used a model consisting of twelve states (see Figure 1).



**Figure 1. The Hidden Markov Model architecture**

The model is trained using a supervised learning algorithm and the Viterbi algorithm is used for classifying instances (ibid.). The emission matrix is constructed by calculating the frequency of each emission symbol in each state. The same approach was used in the THUMBUP system [12]. The transition matrix is constructed with initial weights. The weights are refined by classifying each training instance and if the model misses a transition, the weight from the previous state to the correct state is increased with a learning rate value and the weight from the previous state to the classified state is decreased by the same value. The model that best fits the training data is used to classify future instances. We have used a learning rate of 0.01 in our experiments.

The hidden Markov architecture can be used to model the length distribution of the transmembrane helices by using for example 20 states between the  $N_{out}$  and  $N_{in}$  states. In this case all membranes found will be with a length of 20 residues. The advantage is that no short or long regions will be found, and the disadvantage is that a few hydrophobic residues might be falsely predicted as a 20 residue region. The post-processing used in MemMiC handles short and long domains, and we therefore believe that the advantage of modeling the length distribution will be small if any. It would however be interesting to evaluate such a model, but it has not been performed due to time limits.

### 3.2.4 Learning Vector Quantization

Learning Vector Quantization (LVQ) was developed by Kohonen et al. [6]. The idea of this algorithm is to find a natural grouping in a set of data. Every data vector is associated with a point in an  $n$ -dimensional data space. The vectors of the same class should form a cloud or cluster in data space. The algorithm presupposes that the vectors belonging to the same class are normally distributed with a mean or prototype vector. To classify a feature vector the Euclidean distance to all mean vectors is measured. In our case we have four vectors representing the classes' helix, helix cap, near and globular. The only parameter used is learning rate which have been set to 0.005 in our experiments

## 3.3 Post-processing

The output of a classifier is a sequence of the four classes; helix, helix cap, near and globular. This output sequence is transformed into actual transmembrane domains. A possibility that a specific residue  $R$  is inside a domain is calculated as follows:

1. Each class is assigned a score (helix = 1, helix cap = 0.8, near = 0.2 and globular = 0).
2. The sum of the scores from the residues located from -5 to +4 residues from the  $R$  is calculated.
3. The possibility is the mean value of the sum.

If the possibility is above 0.5, the residue  $R$  is inside a transmembrane domain. The transmembrane domain continues until the score for a residue is below 0.5. To filter out transmembrane domains that probably are false predictions, we have used an optional post-processing method, consisting of two steps:

1. The length of transmembrane domains often varies between 18 and 30 residues [7]. Therefore all domains with a length of 40 or above are split into two domains. In addition, all transmembrane domains shorter than 8 residues are removed.
2. All predicted domains with a start point between residue 1 and 7 and a length below 19 residues are removed. The reason

is that signal peptides has a hydrophobic region in the beginning of the sequence and can therefore be misidentified as a transmembrane protein [11]. Another reason is that sliding window indexes below zero are treated as hydrophobicity value 0, which is slightly hydrophobic in for example the Kyte-Doolittle scale.

## 4. EXPERIMENTS

We have performed four sets of experiments, each corresponding to one of the pre-processing alternatives. For each set all four classifiers has been tested both with and without the optional post-processing method.

All experiments are performed using ten-fold cross validation. The data set contains 160 proteins and a total of 696 known transmembrane domains. It is divided in ten subsets with 16 proteins and therefore no random selection is used in the cross-validation experiments. The same subsets were used by Krogh et al [7]. The training set when performing the cross-validation experiments consists of 55000–57500 training instances. Since most of the learning algorithms are time consuming therefore only every fourth sliding window is used during training.

The accuracy of a prediction will be calculated using the accuracy formula  $Q$  as described by Tusnády and Simon [11]:

$$Q = 100 \sqrt{\frac{N_{correct}^2}{N_{known} \cdot N_{predicted}}}$$

where  $n_{correct}$  is the number of correctly predicted domains,  $n_{known}$  is the total number of known domains in the data set, and  $n_{predicted}$  is the number of predicted domains.

A prediction is considered correct if the predicted domain overlaps the real domain with at least five residues. The reason why domains only need to overlap with five residues is that the transmembrane proteins are floating up and down in the biological membrane and the boundaries are therefore not exactly defined. If a prediction overlaps with 1 to 4 residues it is considered a shifted prediction [7]. Shifted predictions are not considered correct when calculating accuracy in our experiments.

The neural network and LVQ classifiers used in MemMiC are non-deterministic because random initial weights are used. These experiments are therefore repeated five times. Prediction accuracy, mean value and standard deviation are calculated. We keep records of: *Correct* (the amount of correctly predicted domains), *Shifted* (the amount of shifted predictions), *Underpredicted* (false negatives), and *Overpredicted* (false positives).

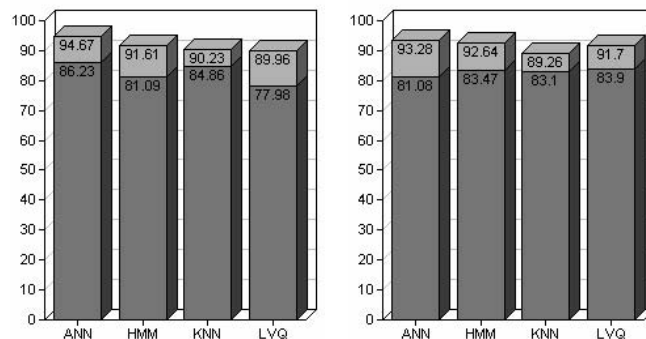
## 5. RESULTS AND ANALYSIS

The results from the experiments using different pre-processing are presented in figure 2 and 3. The complete data from the experiments can be found in the Appendix.

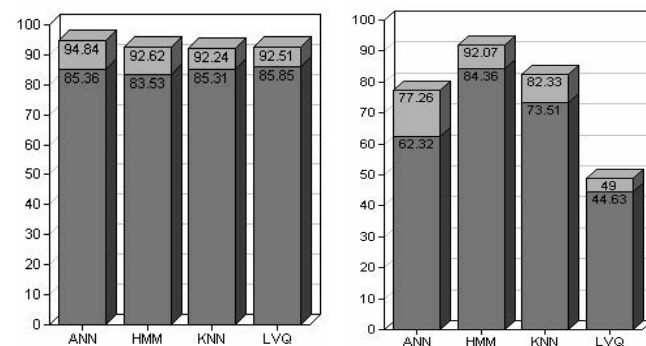
The experiments show that neural networks and hidden Markov models are generally better than k-nearest neighbor and LVQ when using one of the hydrophobicity scales. Neural networks are however not very accurate when using primary sequence as pre-processing. They also show that the difference between classifiers is depending on the pre-processing used. When GCS is used as

pre-processing, k-nearest neighbor is less accurate than LVQ. This is not the case when Eisenberg is used as pre-processing.

The experiments also show that the optional post-processing improves the accuracy with 6 to 16 percentage units depending on which pre-processing and classifier being used. Data from the post-processing part can be found in the Appendix. It shows the amount of domains added or removed in each post-processing step as well as the errors produced, i.e., removed correct domains or added incorrect domains.



**Figure 2. Prediction accuracy using the Eisenberg scale (left) and GCS (right) as pre-processing. The result is both with (light) and without (dark) optional post-processing.**



**Figure 3. Prediction accuracy using the Kyte-Doolittle scale (left) and the primary sequence (right) as pre-processing.**

## 6. CONCLUSIONS

In the literature there are often statements concerning the appropriateness of a particular classifier for predicting transmembrane domains. These are sometimes based on weak evidence or even prejudices. Also, it seems that there is an emerging consensus that hidden Markov models are the most appropriate technique. We have shown that the performance of a specific classifier is very dependent on which pre- and post-processing is used. For instance, there is no guarantee that hidden Markov models, or neural networks, are the most accurate in all configurations. Thus, when developing systems for transmembrane predictions and similar problems, it is important to test and evaluate different combinations of classifiers, pre- and post-processing techniques. In our experiments, the difference in accuracy between the different methods is smaller than in most other studies. A possible reason for this is that we are not biased towards any particular method. Most importantly, a prediction method must be taken as a

whole, but a wide range of combinations should be evaluated when constructing a system. We hope that MemMiC will be a useful tool for doing this.

## 7. REFERENCES

- [1] Baldi, P., Brunak, S. *Bioinformatics – The machine learning approach*. MIT Press, 2001.
- [2] Brookshear, J. G. *Computer Science – An overview*. Addison Wesley, 1997.
- [3] Chen, Z., Liu, Q., Zhu, Y., Yixue, Li, Xu, Y. A hydrophobicity based neural network method for predicting transmembrane segments in protein sequences. In *23rd Annual International Conference of the IEEE*, 2001.
- [4] Eisenberg, D., Schwarz, E., Komaromy, M., Wall, R. Analysis of membrane and surface protein sequences with the hydrophobic moment plot. *Journal of Molecular Biology* (1984) 179:125-142.
- [5] Guimarães, K. S., Melo, J. C. B., Cavalcanti, G. D. C. Combining Few Neural Networks for Effective Secondary Structure Prediction. In *Third IEEE Symposium*, 2003.
- [6] Kohonen, T., Kangas, J., Laakosonen, J., Torkkola, K. LVQ PAK: A program package for the correct application of Learning Vector Quantization algorithms. In *International Joint Conference on Neural Networks*, 725-730, 1992.
- [7] Krogh, A., Larsson, B., von Heijne, G., Sonnhammer, E.L.L. Predicting Transmembrane Protein Topology with a Hidden Markov Model: Application to Complete Genomes. *Journal of Molecular Biology* (2001) 305:567-580.
- [8] Kyte, J., Doolittle, R.F. A simple method for displaying the hydrophatic character of a protein. *Journal of Molecular Biology* (1982) 157:105-132.
- [9] Mitchell, T.M. *Machine Learning*. McGraw-Hill, 1997.
- [10] Tossi, A., Sandri, L., Giangaspero, A. New consensus hydrophobicity scale extended to non-proteinogenic amino acids. In *Proc. of the 27th European Peptide Symposium*, 2002.
- [11] Tusnády, G., Simon, I. Principles governing amino acid composition of integral membrane proteins. *Journal of Molecular Biology* (1998) 283:489-506.
- [12] Zhou, H., Zhou, Y. Predicting the topology of transmembrane helical proteins using mean burial propensity and a hidden Markov-model-based method. *Protein Science* (2003) 12:1547-1555.

## APPENDIX

The table below contains the results from all experiments performed (mean value  $\pm$  standard deviation).

Classifier	Correct	Underpred	Overpred	Shifted	Q1	Q2
<b>Eisenberg</b>						
ANN	611 $\pm$ 2.9	84.8 $\pm$ 2.9	111 $\pm$ 1.1	3.6 $\pm$ 0.9	86.0 $\pm$ 0.2	86.7 $\pm$ 0.3
ANN PP	667 $\pm$ 2.0	29.0 $\pm$ 2.0	46.2 $\pm$ 1.1	0.0 $\pm$ 0.0	94.7 $\pm$ 0.2	94.7 $\pm$ 0.2
HMM	534 $\pm$ 0.0	162 $\pm$ 0.0	89.0 $\pm$ 0.0	1.0 $\pm$ 0.0	81.1 $\pm$ 0.0	81.3 $\pm$ 0.0
HMM PP	644 $\pm$ 0.0	52.0 $\pm$ 0.0	66.0 $\pm$ 0.0	1.0 $\pm$ 0.0	91.6 $\pm$ 0.0	91.8 $\pm$ 0.0
KNN	636 $\pm$ 0.0	60.0 $\pm$ 0.0	171 $\pm$ 0.0	11.0 $\pm$ 0.0	84.9 $\pm$ 0.0	86.3 $\pm$ 0.0
KNN PP	618 $\pm$ 0.0	78.0 $\pm$ 0.0	56.0 $\pm$ 0.0	2.0 $\pm$ 0.0	90.2 $\pm$ 0.0	90.5 $\pm$ 0.0
LVQ	512 $\pm$ 83	184 $\pm$ 83	104 $\pm$ 22	7.4 $\pm$ 2.6	78.0 $\pm$ 6.6	79.1 $\pm$ 6.5
LVQ PP	606 $\pm$ 89	90.2 $\pm$ 89	42.8 $\pm$ 9.4	0.0 $\pm$ 0.0	90.0 $\pm$ 6.9	90.0 $\pm$ 6.9

Classifier	Correct	Underpred	Overpred	Shifted	Q1	Q2
<b>GCS</b>						
ANN	582 $\pm$ 6.0	114 $\pm$ 6.0	158 $\pm$ 10	2.6 $\pm$ 1.1	81.1 $\pm$ 0.7	81.4 $\pm$ 0.8
ANN PP	660 $\pm$ 3.0	36.2 $\pm$ 3.0	59.0 $\pm$ 4.9	0.0 $\pm$ 0.0	93.3 $\pm$ 0.2	93.3 $\pm$ 0.2
HMM	547 $\pm$ 0.0	149 $\pm$ 0.0	70.0 $\pm$ 0.0	7.0 $\pm$ 0.0	83.5 $\pm$ 0.0	84.5 $\pm$ 0.0
HMM PP	659 $\pm$ 0.0	37.0 $\pm$ 0.0	68.0 $\pm$ 0.0	0.0 $\pm$ 0.0	92.6 $\pm$ 0.0	92.6 $\pm$ 0.0
KNN	622 $\pm$ 0.0	74.0 $\pm$ 0.0	183 $\pm$ 0.0	21.0 $\pm$ 0.0	83.1 $\pm$ 0.0	85.9 $\pm$ 0.0
KNN PP	592 $\pm$ 0.0	104 $\pm$ 0.0	40.0 $\pm$ 0.0	4.0 $\pm$ 0.0	89.3 $\pm$ 0.0	89.9 $\pm$ 0.0
LVQ	596 $\pm$ 1.8	100 $\pm$ 1.8	128 $\pm$ 6.0	9.6 $\pm$ 0.9	83.9 $\pm$ 0.3	85.3 $\pm$ 0.3
LVQ PP	635 $\pm$ 5.5	61.4 $\pm$ 5.5	53.4 $\pm$ 2.5	0.6 $\pm$ 0.6	91.7 $\pm$ 0.3	91.8 $\pm$ 0.3
<b>Kyte-Doolittle</b>						
ANN	607 $\pm$ 3.1	89.0 $\pm$ 3.1	120 $\pm$ 4.5	2.4 $\pm$ 1.7	85.4 $\pm$ 0.5	85.7 $\pm$ 0.3
ANN PP	668 $\pm$ 1.0	28.0 $\pm$ 1.0	44.8 $\pm$ 2.2	0.0 $\pm$ 0.0	94.8 $\pm$ 0.2	94.8 $\pm$ 0.2
HMM	588 $\pm$ 0.0	108 $\pm$ 0.0	124 $\pm$ 0.0	5.0 $\pm$ 0.0	83.5 $\pm$ 0.0	84.2 $\pm$ 0.0
HMM PP	657 $\pm$ 0.0	39.0 $\pm$ 0.0	66.0 $\pm$ 0.0	0.0 $\pm$ 0.0	92.6 $\pm$ 0.0	92.6 $\pm$ 0.0
KNN	648 $\pm$ 0.0	48.0 $\pm$ 0.0	181 $\pm$ 0.0	8.0 $\pm$ 0.0	85.3 $\pm$ 0.0	86.4 $\pm$ 0.0
KNN PP	636 $\pm$ 0.0	60.0 $\pm$ 0.0	47.0 $\pm$ 0.0	2.0 $\pm$ 0.0	92.2 $\pm$ 0.0	92.5 $\pm$ 0.0
LVQ	623 $\pm$ 4.2	73.4 $\pm$ 4.2	133 $\pm$ 3.8	10.8 $\pm$ 1.9	85.9 $\pm$ 0.1	87.3 $\pm$ 0.2
<b>Sequence</b>						
ANN	531 $\pm$ 10	165 $\pm$ 10	514 $\pm$ 40	42.6 $\pm$ 5.8	62.3 $\pm$ 1.1	67.3 $\pm$ 1.8
ANN PP	510 $\pm$ 7.0	186 $\pm$ 7.0	116 $\pm$ 14	3.2 $\pm$ 0.8	77.3 $\pm$ 0.6	77.7 $\pm$ 0.5
HMM	588 $\pm$ 0.0	108 $\pm$ 0.0	110 $\pm$ 0.0	6.0 $\pm$ 0.0	84.4 $\pm$ 0.0	85.2 $\pm$ 0.0
HMM PP	654 $\pm$ 0.0	42.0 $\pm$ 0.0	71.0 $\pm$ 0.0	0.0 $\pm$ 0.0	92.1 $\pm$ 0.0	92.1 $\pm$ 0.0
KNN	554 $\pm$ 0.0	142 $\pm$ 0.0	262 $\pm$ 0.0	69.0 $\pm$ 0.0	73.5 $\pm$ 0.0	82.7 $\pm$ 0.0
KNN PP	490 $\pm$ 0.0	206 $\pm$ 0.0	19.0 $\pm$ 0.0	1.0 $\pm$ 0.0	82.3 $\pm$ 0.0	82.5 $\pm$ 0.0
LVQ	248 $\pm$ 42	448 $\pm$ 42	194 $\pm$ 52	28.0 $\pm$ 4.1	44.6 $\pm$ 4.2	49.7 $\pm$ 4.2
LVQ PP	334 $\pm$ 64	362 $\pm$ 64	329 $\pm$ 63	7.4 $\pm$ 4.9	49.0 $\pm$ 5.6	50.1 $\pm$ 5.6

PP denotes that optional post-processing was used.

Correct = Amount of correctly predicted domains.

Underpred = Amount of under predicted (missed) domains.

Overpred = Amount of over predicted domains.

Shifted = Amount of shifted domains.

Q1 = Accuracy when shifted is seen as falsely predicted domains.

Q2 = Accuracy when shifted is seen as correctly predicted domains.

The table below contains the data from the post-processing step in the experiments. It shows the amount of domains that were added and removed in each step as well as the errors (number of correct domains removed or number of incorrect domains added). Each result is presented with mean value  $\pm$  standard deviation.

Classifier	Split	Errors	Rem.Short	Errors	Rem.First	Errors
<b>Eisenberg</b>						
ANN	89.8 $\pm$ 8.1	1.8 $\pm$ 0.5	104 $\pm$ 6.3	4.6 $\pm$ 2.5	23.4 $\pm$ 1.5	3.4 $\pm$ 0.9
HMM	95.0 $\pm$ 0.0	3.0 $\pm$ 0.0	189 $\pm$ 0.0	16.0 $\pm$ 0.0	16.0 $\pm$ 0.0	0.0 $\pm$ 0.0
KNN	11.0 $\pm$ 0.0	0.0 $\pm$ 0.0	129 $\pm$ 0.0	26.0 $\pm$ 0.0	24.0 $\pm$ 0.0	6.0 $\pm$ 0.0
LVQ	64.2 $\pm$ 13	0.6 $\pm$ 0.6	114 $\pm$ 22	9.8 $\pm$ 2.8	22.2 $\pm$ 2.8	6.8 $\pm$ 1.3
<b>GCS</b>						
ANN	116 $\pm$ 8.1	0.8 $\pm$ 0.9	163 $\pm$ 11	3.6 $\pm$ 2.1	25.2 $\pm$ 0.9	3.6 $\pm$ 0.6
HMM	141 $\pm$ 0.0	2.0 $\pm$ 0.0	113 $\pm$ 0.0	3.0 $\pm$ 0.0	15.0 $\pm$ 0.0	1.0 $\pm$ 0.0
KNN	10.0 $\pm$ 0.0	0.0 $\pm$ 0.0	177 $\pm$ 0.0	41.0 $\pm$ 0.0	23.0 $\pm$ 0.0	4.0 $\pm$ 0.0
LVQ	64.2 $\pm$ 11	1.2 $\pm$ 0.5	140 $\pm$ 16	18.0 $\pm$ 3.7	27.6 $\pm$ 1.5	7.4 $\pm$ 0.6
<b>Kyte-Doolittle</b>						
ANN	88.2 $\pm$ 8.1	0.6 $\pm$ 0.6	115 $\pm$ 16	4.2 $\pm$ 1.1	26.2 $\pm$ 1.8	4.4 $\pm$ 1.3
HMM	114 $\pm$ 0.0	0.0 $\pm$ 0.0	121 $\pm$ 0.0	9.0 $\pm$ 0.0	16.0 $\pm$ 0.0	0.0 $\pm$ 0.0
KNN	11.0 $\pm$ 0.0	0.0 $\pm$ 0.0	140 $\pm$ 0.0	20.0 $\pm$ 0.0	23.0 $\pm$ 0.0	5.0 $\pm$ 0.0
LVQ	52.6 $\pm$ 7.8	0.4 $\pm$ 0.6	155 $\pm$ 15	22.0 $\pm$ 2.2	22.6 $\pm$ 0.9	4.4 $\pm$ 0.6
<b>Sequence</b>						
ANN	45.6 $\pm$ 21	5.2 $\pm$ 3.9	676 $\pm$ 55	45.2 $\pm$ 5.4	18.0 $\pm$ 4.2	3.8 $\pm$ 1.5
HMM	131 $\pm$ 0.0	2.0 $\pm$ 0.0	150 $\pm$ 0.0	10.0 $\pm$ 0.0	13.0 $\pm$ 0.0	0.0 $\pm$ 0.0
KNN	2.0 $\pm$ 0.0	0.0 $\pm$ 0.0	361 $\pm$ 0.0	66.0 $\pm$ 0.0	16.0 $\pm$ 0.0	5.0 $\pm$ 0.0
LVQ	289 $\pm$ 152	89.0 $\pm$ 34	252 $\pm$ 120	29.0 $\pm$ 12	13.0 $\pm$ 9.0	4.8 $\pm$ 4.0

Split = amount of added domains when long domains were split into several shorter.

Rem.Short = amount of too short domains that were removed.

Rem.First = amount of domains that were removed from the beginning of each protein sequence.