

Master Thesis
Software Engineering
Thesis no: MCS-2003:07 / MSE-2003:10
06 2003



Locating transmembrane domains in protein sequences

Johan Hagelbäck
Kenny Svensson

Department of
Software Engineering and Computer Science
Blekinge Institute of Technology
Box 520
SE – 372 25 Ronneby
Sweden

This thesis is submitted to the Department of Software Engineering and Computer Science at Blekinge Institute of Technology in partial fulfillment of the requirements for the degree of Master of Science in Software Engineering. The thesis is equivalent to 20 weeks of full time studies.

Contact Information:

Authors:

Johan Hagelbäck

Address: Karlskronavägen 11, 372 37 Ronneby.

E-mail: johan.hagelback@home.se

Kenny Svensson

Address: Folkparksvägen 21, 37240 Ronneby

E-mail: kenny.svensson@bth.se

University advisor:

Paul Davidsson

Department of Software Engineering and Computer Science

Department of
Software Engineering and Computer Science
Blekinge Institute of Technology
Box 520
SE – 372 25 Ronneby
Sweden

Internet : www.bth.se/ipd
Phone : +46 457 38 50 00
Fax : + 46 457 271 25

ABSTRACT

We have developed a new approach for locating transmembrane domains in protein sequences based on hydrophobicity analysis and backpropagation neural network or k-nearest-neighbor as classifiers. Our system was able to locate over 98% of the transmembrane domains and the total accuracy including overpredictions was above 95%.

Keywords: transmembranes, membrane proteins, prediction

CONTENTS

ABSTRACT	I
CONTENTS	II
1 INTRODUCTION	1
1.1 BACKGROUND.....	1
1.2 RESEARCH QUESTIONS.....	1
1.3 METHODOLOGY	1
2 BIOLOGICAL BACKGROUND	2
2.1 AMINO ACIDS AND PROTEINS	2
2.2 BIOLOGICAL MEMBRANES AND TRANSMEMBRANE PROTEINS	5
3 RELATED WORK	8
3.1 LEARNING VECTOR QUANTIZATION.....	8
3.1.1 <i>Method</i>	8
3.1.2 <i>Experiment</i>	8
3.1.3 <i>Result</i>	8
3.1.4 <i>Analysis</i>	8
3.2 GENETIC PROGRAMMING	8
3.2.1 <i>Method</i>	8
3.2.2 <i>Experiment</i>	9
3.2.3 <i>Result</i>	9
3.2.4 <i>Analysis</i>	9
3.3 ELEMENTARY FORMAL SYSTEM	9
3.3.1 <i>Method</i>	9
3.3.2 <i>Experiment</i>	9
3.3.3 <i>Result</i>	9
3.3.4 <i>Analysis</i>	9
3.4 TRANSMEMBRANE HIDDEN MARKOV MODEL.....	10
3.4.1 <i>Method</i>	10
3.4.2 <i>Experiment</i>	10
3.4.3 <i>Result</i>	10
4 OUR SYSTEM	11
4.1 APPROACH	11
4.2 SYSTEM DESCRIPTION	12
4.2.1 <i>Parsing of training data</i>	12
4.2.2 <i>Classification</i>	13
4.2.3 <i>Post-processing</i>	14
4.2.4 <i>The backpropagation Neural Network</i>	14
4.2.5 <i>The K-Nearest-Neighbor</i>	16
5 EXPERIMENTS	19
5.1 PREDICTION ACCURACY USING DIFFERENT INPUT SCALES.....	19
5.2 PREDICTION ACCURACY COMPARED TO TMHMM.....	20
6 EXPERIMENT RESULTS	21
6.1 PREDICTION ACCURACY USING DIFFERENT INPUT ENCODINGS	21
6.2 PREDICTION ACCURACY COMPARED TO TMHMM.....	22
7 CONCLUSION	24
8 FUTURE WORK	26
APPENDIX A: NEURAL NETWORKS	27

APPENDIX B: K-NEAREST-NEIGHBOR	29
APPENDIX C: SLIDING WINDOW	30
REFERENCES	31

1 INTRODUCTION

1.1 Background

Protein sequence analysis is an important aspect of bioinformatics. Proteins consist of long chains of amino acids which can be seen as a sequence of digital symbols. The digital nature of proteins makes it possible to use machine learning algorithms to locate similarities. The amino acids of a protein organize themselves into certain features (for example secondary structure or transmembrane domains) which decides the behavior and functionality of the protein. To experimentally verify these features is very expensive and time consuming, and for most of the known proteins only the sequence of amino acids is verified. The idea is to use the amount of experimentally verified proteins as training examples of a machine learning algorithm and locate these features in the rest of the proteins with the highest possible accuracy. The biological background is explained in detail in the following chapter.

1.2 Research questions

There exist today several different systems for classifying transmembrane domains based on different input signals and classifiers. The different input signals are the amino acid sequence, hydrophobicity where each amino acid is assigned a value based on its hydrophobic ability or charge bias analysis where each amino acid is assigned a value based on its electrical charge bias. We will investigate the effects of using different input signals to a classification system. We believe that the use of primary sequence alone is not as accurate as the use of hydrophobicity values and will perform an experiment to investigate the differences. We will also test if an instance-based classifier such as k-nearest-neighbor is able to compete with the more common used neural networks and hidden Markov models.

1.3 Methodology

We will develop a system based on two machine learning algorithms (backpropagation neural networks and k-nearest-neighbor) with the possibility of choosing which input signal and learning algorithm to use. We will analyze the accuracy of different input signals and perform experiments to investigate if k-nearest-neighbor can classify transmembrane domains with the same or better accuracy as the more common used neural networks. We will also compare the performance of our system with the state-of-the-art system of today TMHMM developed by Krogh et al (2001).

2 BIOLOGICAL BACKGROUND

The aim of this chapter is to give the reader an insight in the biological background containing amino acids, protein structures, biological membranes and transmembrane proteins. This chapter does not discuss anything that has to do with our work and may therefore be skipped by the reader which has a good insight in the biological background.

2.1 Amino acids and proteins

Proteins have several important functions in living organisms. They are the structure of our muscles, the basis for our immunity system, they generate nerve impulses and they transport particles e.g. for example oxygen in blood. The most important usage of proteins is however to catalyze chemical reactions in the form of enzymes. The catalytic power of an enzyme is so enormous that without the enzyme the reaction would probably not occur.

Proteins consist of long chains of amino acids. An amino acid consists of a backbone and an R-group. The backbone is a regular structure that is the same in all amino acids. The R-group is a side chain which differ between the different amino acids. Figure 2.1 shows an example of the two amino acids aspartic acid and histidine (Metzenberg, 1996).

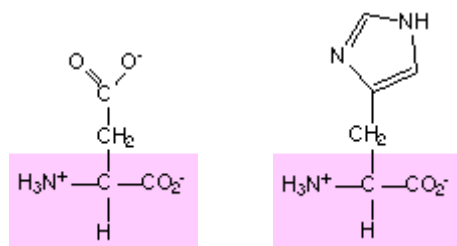


Figure 2.1. The amino acids aspartic acid (left) and histidine (right). The colored area is the backbone and the above side chains the R-groups. Note that the backbone is the same in all amino acids.

Proteins are created in the cells and are encoded by nucleotide (DNA) sequences. DNA consists of four different nucleotides with the bases A, T, C and G. The transcription of a protein is performed using RNA, which is a copy of a DNA sequence. The difference between DNA and RNA is that RNA has a U (uracil) base instead of T (thymine). A unit of three RNA bases (called a codon) encodes a specific amino acid. The codon to amino acid encoding is known as the standard genetic code which is presented in table 2.1. The creation of a protein is initialized with a start codon and continues until a stop codon is found. The codons between start and end encode a sequence of amino acids which is bound together in a linear chain to form a specific protein (Sengbush, 2002). This process is known as translation and is illustrated in figure 2.2.

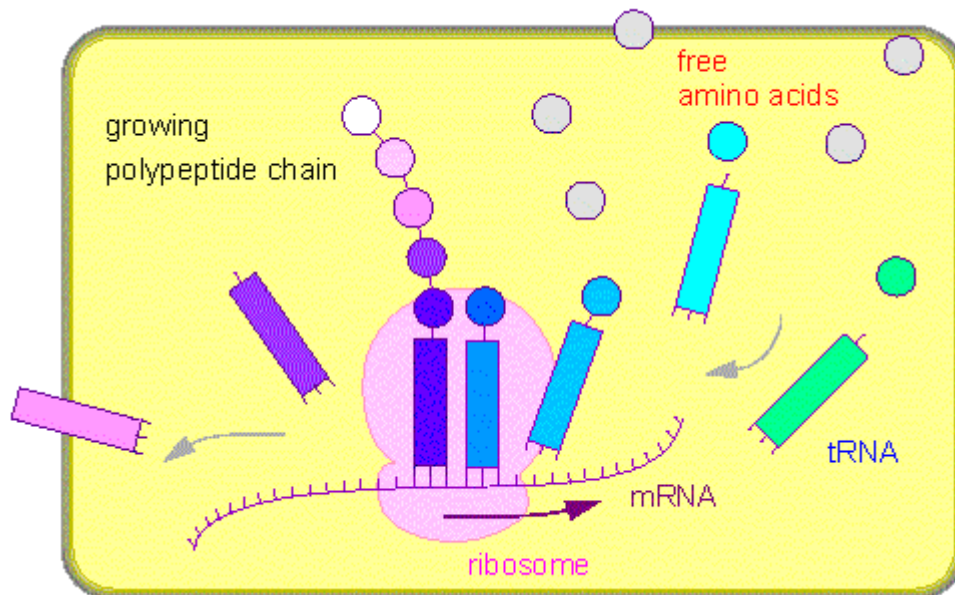


Figure 2.2. The translation process. The mRNA contains the genetic information to form a protein. Free amino acids are bound to a corresponding tRNA molecule which is bound to a specific codon in the mRNA string and the polypeptide chain grows.

Amino acid	Three-letter code	One-letter code	Genetic code
Alanine	Ala	A	GC*
Cysteine	Cys	C	UGU, UGC
Aspartic Acid	Asp	D	GAU, GAC
Glutamic Acid	Glu	E	GAA, GAG
Phenylalanine	Phe	F	UUU, UUC
Glycine	Gly	G	GG*
Histidine	His	H	CAU, CAC
Isoleucine	Ile	I	AUU, AUC, AUA
Lysine	Lys	K	AAA, AAG
Leucine	Leu	L	UUA, UUG, CU*
Methionine	Met	M	AUG
Asparagine	Asn	N	AAU, AAC
Proline	Pro	P	CC*
Glutamine	Gln	Q	CAA, CAG
Arginine	Arg	R	CG*, AGA, AGG
Serine	Ser	S	UC*, AGU, AGC
Threonine	Thr	T	AC*
Valine	Val	V	GU*
Tryptophan	Trp	W	UGG
Tyrosine	Tyr	Y	UAU, UAC

Table 2.1. The standard genetic code. A * could be one of the four bases. Proline is for example encoded by CCA, CCG, CCC and CCU.

Two amino acids are bound together with a peptide bond. A chain of amino acids is therefore called a polypeptide. Each amino acid in a polypeptide is referred to as a residue. An example of a peptide bond between two residues is found in figure 2.3 (Wampler, 1996).

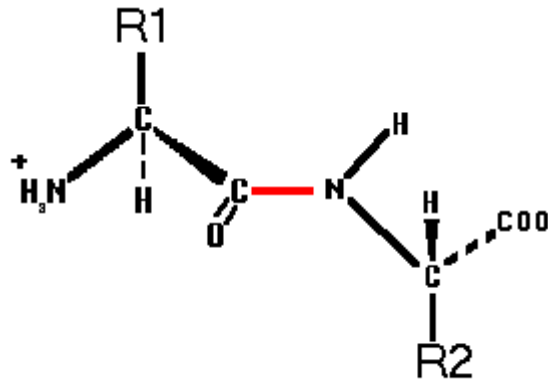


Figure 2.3. A peptide bond (red) between two residues R1 and R2.

A protein is organized in four different structures. The sequence of amino acids is called the primary structure of the protein. The primary structure is easy to verify experimentally and it is known for a lot of proteins.

Proteins are locally arranged into certain structures. This arrangement is called the secondary structure of a protein. The most common structures are the α -helices and β -sheets (Wampler, 1996). The α -helix is a spiral while the β -sheet is a zigzag structure which consists of several β -strands (figure 2.4).

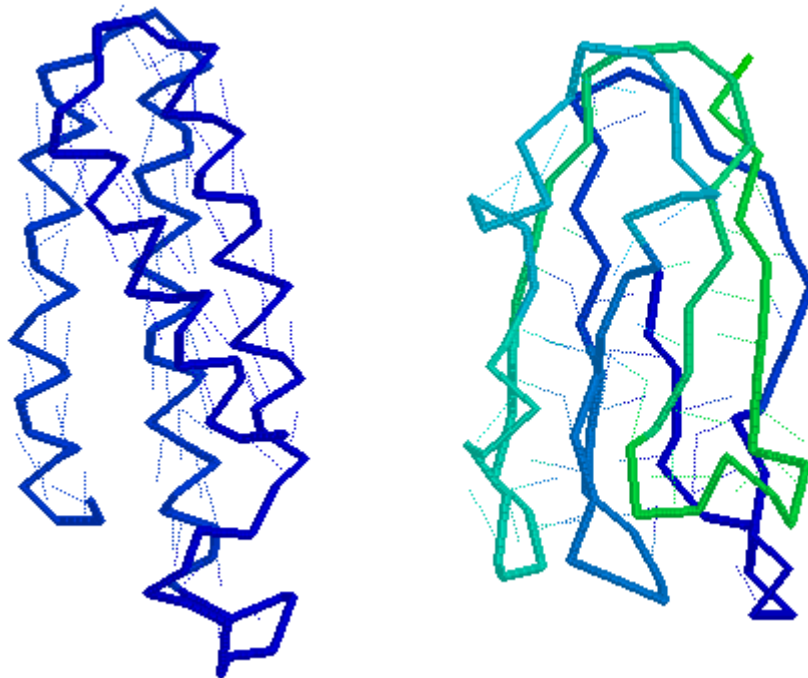


Figure 2.4. The α -helix spiral structure (left) and the zigzag β -sheet structure (right).

The three-dimensional arrangement of amino acids in a protein is called the tertiary structure. To determine the three-dimensional structure of a protein the coordinate for every atom in the protein must be known. It is very time consuming and expensive to determine tertiary structures of proteins and very few structures has been experimentally verified (Koza, 1998).

Some proteins contain more than one amino acid chain (called subunit). The three-dimensional arrangement of the subunits is the quaternary structure of proteins.

2.2 Biological membranes and transmembrane proteins

All cells are surrounded by a membrane. The membrane separates the cell from the outside world. Since membranes are the only interaction between the cell and the outside world they are necessary for life. Biological membranes have several important functions in a living organism (Lipson, 2003):

- Barrier. Membranes keep the contents of a cell together and prevent harmful substances from entering the cell.
- Signaling. Membranes relay signals from outside to a cell and vice versa.
- Factory site. Membranes provide places where enzymes can be arranged in an assembly line fashion.
- Energy conversion. Membranes allow conversion of light and chemical energy to more usable forms.
- Subdividing the cell. In most cells, membranes separate parts of the cell which perform different functions.
- Recognition. Different cell membranes have different surfaces and will interact differently with different other cells. The cell is therefore able to recognize which cells to interact with.

A biological membrane consists of a lipid bilayer (30 – 80% of the mass), proteins (20 – 60%) and sometimes carbohydrate (0 – 10%) (see figure 2.5). Proteins located in a membrane are called transmembrane proteins. Therefore transmembrane proteins are often referred to as integral proteins. One part of an integral protein is located on one side of the membrane, one part inside the lipid bilayer and the rest on the opposite side. The parts of the protein located inside the membrane are called transmembrane domain(s). There exists both single-pass and multi-pass transmembrane proteins. A multi-pass protein crosses the membrane back and forth several times while a single-pass only crosses the membrane once. Figure 2.5 shows a single-pass and a multi-pass transmembrane protein crossing a biological membrane (Childs, 2001).

Transmembrane proteins are necessary for the functionality of a biological membrane, for example to transmit signals. It has been estimated that 20 - 30% of the proteins in a living organism is transmembrane proteins and the understanding of these proteins is an important aspect of bioinformatics (Krogh et al., 2001).

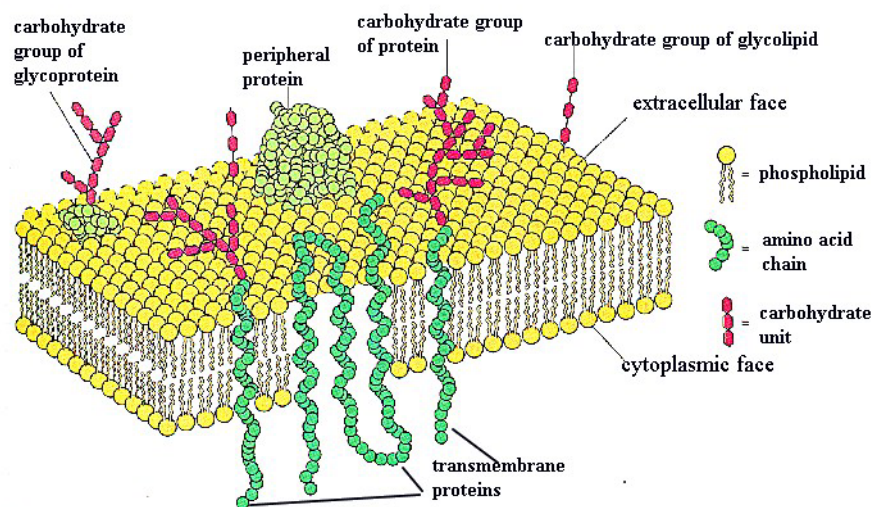


Figure 2.5. Transmembrane proteins crossing a lipid bilayer.

Transmembrane proteins are amphipathic, which means that they have both hydrophobic (mixes poorly with water) and hydrophilic (attracted to water) regions (Childs, 2001). Biological membranes are of hydrophobic nature and the residues in

the transmembrane domains are therefore mostly hydrophobic. The residues located on the in- or outside of a transmembrane domain are mostly hydrophilic. Kyte and Doolittle (1982) developed a scale where each amino acid where assigned a value based on its hydrophobic capabilities. Table 2.2 shows the Kyte-Doolittle hydrophobicity scale. Today there exist about 160 different hydrophobicity scales in the literature (Sandri, 2000).

Amino acid	One-letter code	Kyte-Doolittle value	Category
Isoleucine	I	+4.5	Hydrophobic
Valine	V	+4.2	Hydrophobic
Leucine	L	+3.8	Hydrophobic
Phenylalanine	F	+2.8	Hydrophobic
Cysteine	C	+2.5	Hydrophobic
Methionine	M	+1.9	Hydrophobic
Alanine	A	+1.8	Hydrophobic
Glycine	G	-0.4	Neutral
Threonine	T	-0.7	Neutral
Serine	S	-0.8	Neutral
Tryptophan	W	-0.9	Neutral
Tyrosine	Y	-1.3	Neutral
Proline	P	-1.6	Neutral
Histidine	H	-3.2	Hydrophilic
Glutamine	Q	-3.5	Hydrophilic
Asparagine	N	-3.5	Hydrophilic
Glutamic Acid	E	-3.5	Hydrophilic
Aspartic Acid	D	-3.5	Hydrophilic
Lysine	K	-3.9	Hydrophilic
Arginine	R	-4.0	Hydrophilic

Table 2.2. The Kyte-Doolittle hydrophobicity scale.

A hydrophobicity scale can be used to plot a hydrophobicity moving sum graph of a protein. The moving sum value for a residue r is the sum of the hydrophobicity values for r and the n residues located before r . The moving sum graph can be used to locate transmembrane domains. The hydrophobic regions which are likely to be transmembrane domains are shown as distinctive peaks in the graph. An example of a hydrophobicity plot for a part of the protein GAR2_HUMAN is found in figure 2.6.

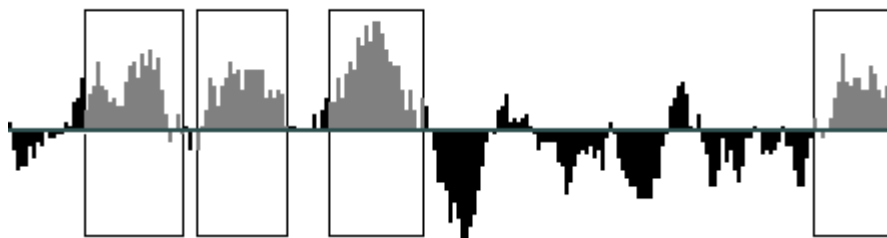


Figure 2.6. Part of the moving sum plot for GAR2_HUMAN. The boxes show where the transmembrane domains are located. The residues colored gray are inside a membrane and the black are located on one side of the biological membrane. The graph shows that transmembrane domains are located where there are peaks in the graph.

The problem is that there exist distinctive peaks in the graph of some proteins which are not transmembrane domains. The plot for the protein 1CDE_TRANSFERASE(FORMYL) (figure 2.7) is one example.

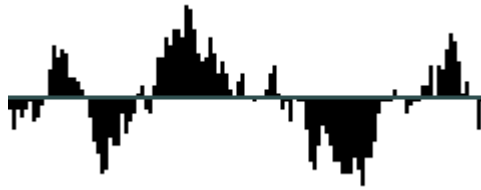


Figure 2.7. Part of the moving sum plot for 1CDE_TRANSFERASE(FORMYL). The graph has one distinctive peak which is not a transmembrane domain. It is interesting to notice that this is not a membrane protein, but could be predicted as one by a transmembrane prediction algorithm.

Hydrophobicity analysis is the most widely used technique in transmembrane prediction (Krogh et al., 2001). It is however not enough to rely on hydrophobicity if it shall be possible to locate all kinds of transmembrane domains. The other important signal is the positive inside rule (von Heijne, 1997). It means that the residues between two transmembrane domains located on the inside of the lipid bilayer are mainly composed of positively charged amino acids. There exists however systems that are based on primary sequence analysis alone.

3 RELATED WORK

Several approaches for transmembrane prediction have been developed. We have chosen to discuss four systems in detail. The reason why these systems were chosen is because they are based on very different learning methods and input data. We want to give the reader an insight in the broad area of solutions to the transmembrane prediction problem.

3.1 Learning vector quantization

3.1.1 Method

This approach uses a supervised learning vector quantization neural network based on hydrophobicity analysis and is developed by Chen et al. (2001). Learning Vector Quantization (LVQ) was developed by Kohonen in 1991. The basic 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 hope is that vectors of the same class 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 their experiment a LVQ-network with two output classes were used. The two classes was one for the inside a transmembrane domain and the other for outside transmembrane domains.

3.1.2 Experiment

547 transmembrane segments belonging to 92 proteins from the MPtopo database were used. Half of the proteins were used for training. Of the other half was 50% used for validation and 50% was used for performance measures.

3.1.3 Result

The system was able to correctly predict 94-96% of the actual transmembrane domains. The final result using the prediction accuracy value Q^1 was 91%.

3.1.4 Analysis

The authors has however pointed out that the dataset contained a very small number of records. Another interesting fact is that the developers have tested the system using several hydrophobicity scales, but Kyte-Doolittle produced the most accurate predictions.

3.2 Genetic programming

3.2.1 Method

This approach is a genetic programming algorithm based on primary sequence alone developed by Koza (1998). Genetic programming is a form of evolutionary computation where a population consisting of computer programs evolves through random mutations and crossover, which are patterned after processes in biological evolution (Mitchell, 1997).

$$^1 Q = 100 \sqrt{\frac{N_{correct}^2}{N_{known} \cdot N_{predicted}}}$$

3.2.2 Experiment

248 mouse transmembrane proteins from the SWISS-PROT database were used. 123 of the 248 available proteins were used for training. One transmembrane and one non-transmembrane domain were selected from each protein. As validation were 125 positive and 125 negative domains from the rest of the proteins used.

3.2.3 Result

The final result was 96.8 correctly predicted transmembrane domains.

3.2.4 Analysis

This approach does not predict complete protein sequences (where the number of domains can vary from 0 to 18 and the number of residues from 200 to above 3000). To predict the locations of transmembrane domains is a more complex learning problem than to decide of a sequence of a fixed length is a transmembrane domain or not. It should have been interesting to see if the genetic programming approach produced an equal result if whole protein sequences should be predicted.

3.3 Elementary formal system

3.3.1 Method

EFS is a elementary formal system based on hydrophobicity analysis developed by Arikawa et al. (1992) An elementary formal system is a kind of logic program consisting of if-then rules (ibid.). The developers used a simplified hydrophobicity scale where each amino acid is categorized into hydrophobic, neutral or hydrophilic based on the Kyte-Doolittle scale.

3.3.2 Experiment

37 positive and 100 negative training examples were used in the training of the system and 689 positive and 634 negative examples from the PIR database were selected for evaluation.

3.3.3 Result

Over 90% of the positive examples were correctly predicted and 82-84% of the negative.

3.3.4 Analysis

The developers has used the same approach as the Genetic Programming system described above, i.e. they have cut out positive and negative sequences from proteins and the output is either a membrane or not. It should have been interesting to see if this approach too could predict complete protein sequences with an equal result.

3.4 Transmembrane hidden Markov model

3.4.1 Method

TMHMM is a hidden Markov model based on hydrophobicity and charge bias analysis developed by Krogh et al. (2001). A hidden Markov model is a stochastic generative model for time series defined by a finite set of states, a discrete alphabet of symbols, a probability transmission matrix and a probability emission matrix (Baldi and Brunak, 2001). Figure 3.1 presents the state architecture for the hidden Markov model in the TMHMM system. The system classifies residues as globular, cytoplasmic loops, helix caps, helix and non-cytoplasmic loops. The parameters of the hidden Markov model has been optimized using a test set containing 160 proteins with known transmembrane domains. The authors states that the boundaries of the located helices are often inaccurate and therefore they use an estimation procedure where the boundaries are redefined to be more accurate. A classification is considered to be correct if the predicted domain overlaps the real domain with at least five residues. The system also keeps track of shifted predictions (overlaps with one to four residues) which are also considered correctly predicted domains.

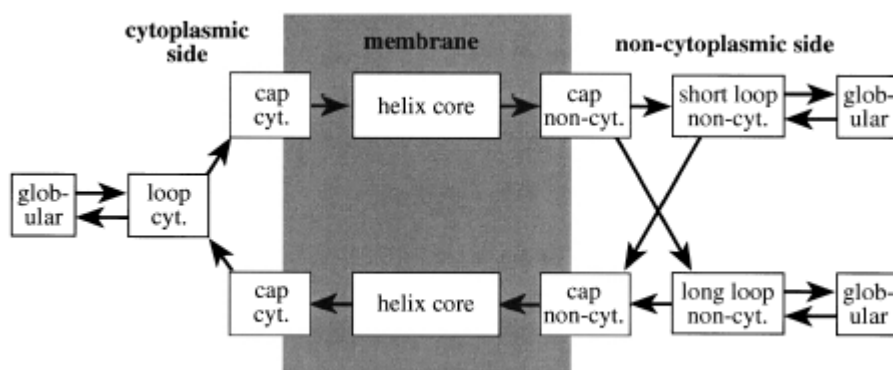


Figure 3.1. The TMHMM architecture.

TMHMM not only predicts the transmembrane domains, but also the full topology with inside and outside regions. It has a post-processing of the result where the expected number of domains in a protein is estimated to reduce the amount of overpredictions. The post-processing also incorporates expectations of the minimum and maximum length of transmembrane domains.

3.4.2 Experiment

For training and classification 160 proteins from the SWISS-PROT database were cross-validated using ten-fold cross-validation. The data set is available at the TMHMM website (www.cbs.dtu.dk/services/TMHMM/) and we will use it in our experiments to be able to compare the result.

3.4.3 Result

The final result was above 96% correctly predicted transmembrane domains and 77.5% correctly predicted topologies. The authors claim that TMHMM is the currently most accurate transmembrane prediction system available.

4 OUR SYSTEM

4.1 Approach

As mentioned earlier there are several approaches that have been tried in transmembrane prediction. The most common algorithms are neural networks and hidden Markov models. They are often based on hydrophobicity analysis. There are also approaches based on primary structure analysis and simplified hydrophobicity analysis. However, the results from these experiments cannot be compared with for example the results from TMHMM because they are not predicting full protein sequences.

One problem with hydrophobicity analysis is the fact that hydrophobicity is not exactly defined and several different scales appear in the literature (Koza, 1998). In some of these scales the differences is large enough to affect the rank order of the amino acids. The primary structure of a protein is exactly defined, and it is therefore interesting to see if an algorithm based on primary structure analysis can compete with hydrophobicity analysis. We will conduct an experiment to investigate the differences in accuracy using primary sequence analysis and hydrophobicity analysis.

Experiments made by Chen et al. (2001) have shown that the Kyte-Doolittle scale produces the most accurate result using their algorithm. It is however no guarantee that this scale is the most accurate using our approach and we will therefore try out three different hydrophobicity scales besides Kyte-Doolittle. These scales are:

- kPROT (Pilpel et al, 1999). kPROT is a knowledge-based scale where the values are derived from the frequency of each amino acid in transmembrane domains.
- GCS (Sandri, 2000). GCS is a consensus scale derived from the 160 hydrophobicity scales found in the literature.
- ProtScale (Guy, 1985). ProtScale is a scale based on the free energy of transfer.

The amino acid values for these scales are presented in table 4.1

We will develop a system based on two different machine learning algorithms and different input scales to compare the performance with the state-of-the-art algorithm TMHMM. We will use the same cross-validation data used by the developers of TMHMM to reproduce the experiment made by Krogh et al (2001). Our algorithm is described in detail below.

Amino acid code	kPROT	ProtScale	GCS
Ala	0.02 ± 0.02	0.100	-0.28
Cys	0.27 ± 0.03	-1.420	3.21
Asp	-0.87 ± 0.08	0.780	-10.00
Glu	-0.86 ± 0.08	0.830	-9.18
Phe	-0.11 ± 0.02	-2.120	9.30
Gly	-0.12 ± 0.02	0.330	-2.69
His	-0.34 ± 0.06	-0.500	-2.88
Ile	0.12 ± 0.01	-1.130	10.00
Lys	0.25 ± 0.04	1.400	-9.88
Leu	0.19 ± 0.01	-1.180	8.94
Met	-0.31 ± 0.03	-1.590	7.04
Asn	-0.68 ± 0.05	0.480	-7.17
Pro	-0.51 ± 0.03	0.730	-2.71
Gln	-0.54 ± 0.05	0.950	-7.43
Arg	0.18 ± 0.04	1.910	-9.84
Ser	-0.21 ± 0.02	0.520	-5.87
Thr	-0.02 ± 0.02	0.070	-4.29
Val	0.23 ± 0.01	-1.270	6.16
Trp	-0.12 ± 0.03	-0.510	6.76
Tyr	-0.12 ± 0.03	-0.210	2.14

Table 4.1. Hydrophobicity values for kPROT, ProtScale and GCS.

4.2 System description

Our system is divided in three parts. The first is the parsing of the training data, the second is classification using a machine learning algorithm and the third is the post-processing of the result. We will test our system with two different machine learning algorithms, k-nearest-neighbor and backpropagation neural network.

4.2.1 Parsing of training data

The training data consists of a number of proteins with their amino acid sequences and start/end points of each transmembrane domain. A sliding window is used to divide the sequence into sub sequences with a fixed length (see appendix C for a more detailed description of sliding window). Each sub sequence is stored as an array of hydrophobicity values corresponding to the amino acid sequence. The training instances are divided in four classes; helix, helix cap, near membrane and globular. The limit values for each class are as follows (see also figure 4.1):

- Helix: The residue is located after five residues from a transmembrane start and before five residues from a transmembrane end.
- Helix cap: The residue is located within ± 5 residues from a transmembrane start or end.
- Near membrane: The residue is located within 10 residues before or after a helix cap on the non-transmembrane side.
- Globular: All residues not belonging to one of the above-mentioned classes.

An index between two transmembrane domains located very close to each other can belong to a helix cap relative the first membrane and near the second membrane. If an index belongs to two or more classes the assignment follows a rank order presented in table 4.2.

Class	Rank
Helix	1 (highest)
Helix cap	2
Near	3
Globular	4 (lowest)

Table 4.2. The rank order of the four classes.

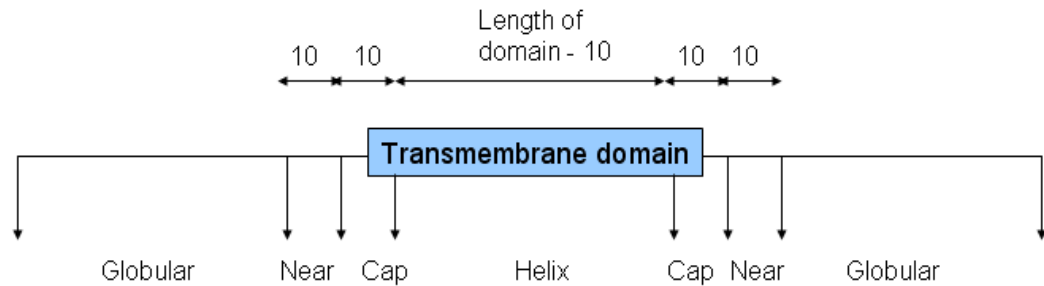


Figure 4.1. The four classes.

4.2.2 Classification

A query instance consists of an amino acid sequence of variable length. To divide the sequence into sub sequences of fixed length a sliding window approach is used. The window size is the same as the size used in the parsing of the training data. Each sub sequence is classified using a machine learning algorithm. In our experiment we will use backpropagation neural networks and a slightly different variant of k-nearest-neighbor. The reason why these two algorithms have been chosen is:

- Neural networks are known to perform well on noisy data sets (Haykin, 1999). As mentioned earlier there exist hydrophobic regions in some proteins which are not transmembrane domains. These regions can be seen as noise in the data set.
- K-Nearest-Neighbor is an instance-based learning algorithm. Instance-based learning algorithms have an advantage in that they does not have to converge to a single function hypotheses that has to cover the entire instance space. They use many different local linear functions constructed at classification time (Mitchell, 1997). Therefore it is interesting to see if nearest-neighbor can compete with the more common used neural networks or hidden Markov models.

A more detailed description of the learning algorithms and the parameters used in our system is found in chapter 4.2.4 and 4.2.5.

The classification result is a sequence of the four different classes' helix, helix cap, near and globular. A result could look like

...NNNCCHHHHHCCNN.....CHC.....NCCHHHHHHHCCNN...

where globular is written as dots to make it easier to see the possible transmembrane regions. This example illustrates that there is often noise in the output. The short region containing two helix caps and one helix is not likely to be a transmembrane domain while the two longer regions are probably domains. A post-processing to decide which regions are likely to be transmembranes and which are not is therefore needed.

4.2.3 Post-processing

As discussed in the previous sub-chapter there is a need for post-processing of the classification result. The post-processing is performed in four different steps:

1. To reduce the noise in the classification result the system calculates a probability (described later in this sub chapter) that each residue is located in a transmembrane domain. If the probability for a specific residue is above 50% the residue is a start of a domain. The membrane continues until the probability for a residue is below 50% and an end point is found.
2. The length of transmembrane domains often varies between 18 and 30 residues (Krogh et al, 2001). Therefore all domains with a length of 40 or above are split into two domains. All domains with a size of 8 or less are not likely to be transmembranes and are therefore removed. The reason why domains with the size of 8 and not 17 are removed is because 8 – 17 residues classified as helices or helix caps are located after one another and is therefore probably a part of a transmembrane domain. If they should be removed there is a high probability that a correctly predicted domain is removed.
3. All predicted domains with a start point between 0 and 6 and a length below 19 residues are removed. The reason is that this part of the membrane protein is more likely to be a part of the out- or inside loops than a transmembrane domain but is often predicted as being a part of a membrane. The reason is that parts of the sliding window are outside the sequence and are therefore treated as hydrophobicity value 0 (which is slightly hydrophobic). Predictions with a length of 20 or more residues are more likely to be transmembrane domains and are therefore not removed.
4. All domains with a size below 18 are extended to 18 residues (since membrane length often varies between 18 and 30 residues). If for example a domain between residue 20 and 30 is found, it is replaced by a new domain from 16 to 34.

The probability that a specific residue is located in a transmembrane domain is calculated as follows:

- Each class has a different probability. Helix is 1.0, helix cap 0.8, near 0.2 and globular 0.
- Calculate the sum of the class probabilities for the residue r and each residue located -5 and $+4$ residues from r .
- Divide the sum with the window size and multiply with 100. The result is a percent value between 0 and 100.

4.2.4 The backpropagation Neural Network

The purpose of this subchapter is to explain the settings used in our backpropagation neural network. The reader not familiar with neural networks should first read the general description in appendix A.

We use a standard backpropagation neural network consisting of 32 hidden nodes, 19 input nodes and 2 output nodes. Several tests were made to find the optimal relation between learning rate (0.1 and 0.2 in the graph) and the number of hidden nodes (see figure 4.2). During all optimization tests cross-validation was used but the neural network was not trained for as many epochs, 32 epochs, as the real tests. The reason is that it requires a lot of computation time to train the net to perform its best. The optimization test is not complete and more parameters need to be tested, but the lack of time made this impossible, e.g., different learning rates, learning methods and even different neural networks should be tested for a more optimal result. The output nodes uses a binary signal for the four different instance classes (see table 4.3). Experiments

with four output nodes were also done but the result (93.97%) was significantly worse than with two output nodes and are therefore not used during any experiments.

During the real experiments the neural network was trained for 128 epochs. As seen in Figure 4.3 the error rate around 100 epochs are quite stable and there are therefore no gain in training for more than around 100 epochs.

As input a sliding window is used with the size of 19 residues. For each residue in the sliding window one scale or sequence data are used. Other window sizes have been tested during initial experiments but the size 19 has been the most successful and this size is commonly used for hydrophobicity analysis (Chen et al. 2001).

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

Table 4.3. Binary class encoding

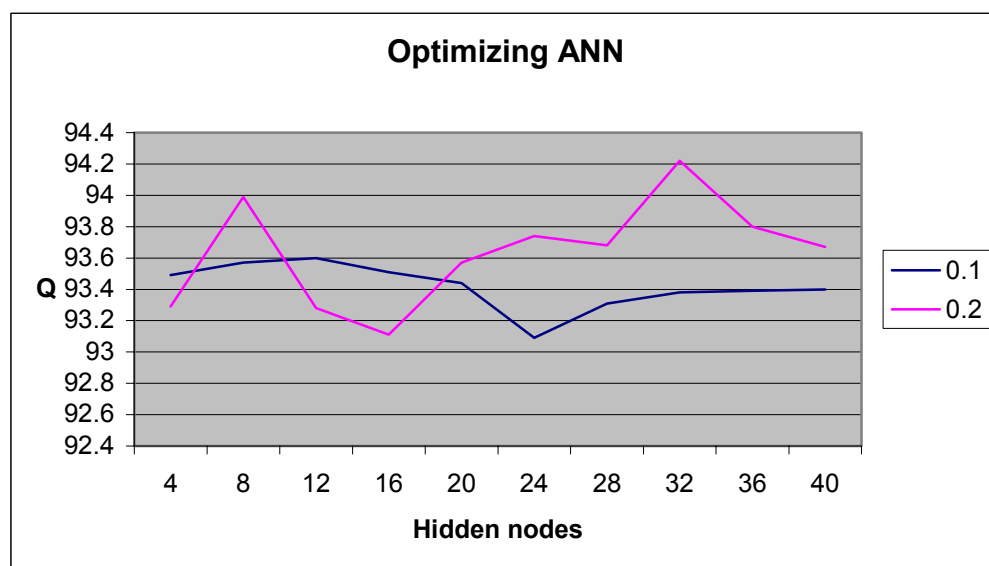


Figure 4.2. Hidden nodes and learning rate. 20 experiments were made with hidden nodes between 4 and 40. All experiments used either 0.1 or 0.2 as learning rate. The result with 32 hidden and 0.2 as learning rate performed best of the experiments.

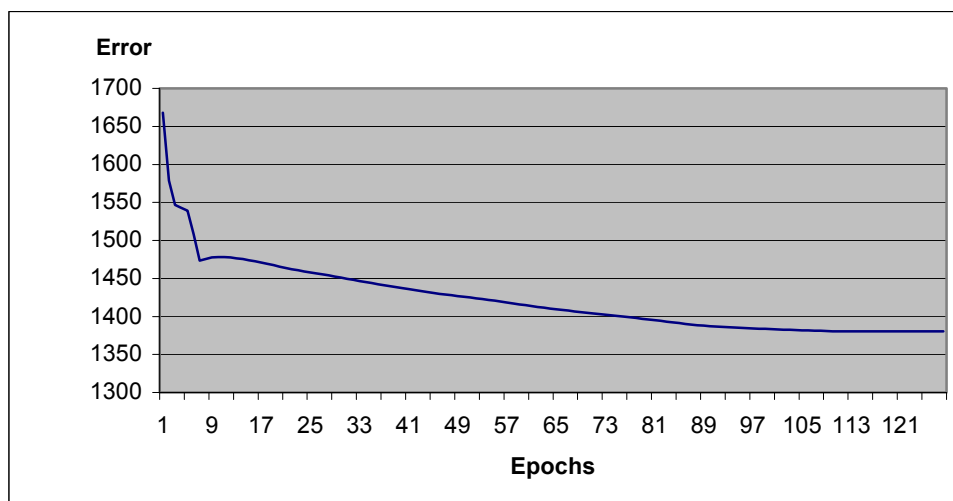


Figure 4.3, An example of a typical error rate. As seen the error rate is stable around the 100-110th epoch. A low error rate means that the neural network handles the current training set well.

4.2.5 The K-Nearest-Neighbor

The purpose of this subchapter is to explain the settings used in our k-nearest-neighbor algorithm. The reader not familiar with nearest neighbor should first read the general description in appendix B.

There is a difference between k-nearest-neighbor and our approach in how the distance between two instances is calculated when using primary sequence analysis. In this case the hamming distance is used. The hamming distance is the number of characters that differ between two text strings (Black, 2002). The reason is that it does not require a numerical encoding of each amino acid. A numerical encoding would result in that the differences between A and B compared to A and H would not be the same. The result of the classification would therefore be dependent on the encoding, and different encodings would result in different classifications.

In k-nearest-neighbor there are two ways of calculating a target function from the k nearest training instances. If the target function is discrete-valued the result is the most common class value among the k nearest instances. If the target function is real-valued the result is the mean value of the k nearest neighbors. In our case we have a discrete-valued target function with class values helix = 1, helix cap = 2, near = 3 and globular = 4. The rank order between the classes is 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. In our case we have chosen a linear relationship between the classes. We have also tested alternative relationships, for example when helix cap is closer to helix than to near and near is closer to globular than to helix cap. The result from this experiment is shown in table 4.4. The experiment shows that the linear scale is the most accurate. We have however due to time limits not optimized the k value for each class relationship. There might be a configuration that is slightly more accurate than the one we use, but we have not found it yet. The k value has been set to 21 in all experiments because it was the most accurate value when using the linear relationship (see experiment later in this chapter).

Helix	Helix cap	Near	Globular	Accuracy
1	2	3	4	92,89
1	1.3	3.7	4	90,94
1	1.8	3.2	4	92,37
1	2.2	2.8	4	92,55
1	2	3	6	87,45
1	2	3	5	89,38
1	4	5	6	91,24
1	3	4	5	91,94

Table 4.4. Accuracy with different class relationships.

We found out that even if we have a discrete-valued target function the real-valued function produced a more accurate result. Consider the following example which could be a result from a classification when using 10-nearest-neighbor:

Globular: 2 Near: 3 Helix cap: 4 Helix: 1

When using the discrete-valued function the result is helix cap. The result when using the real-valued function is near. The example illustrates that if the classes of the k nearest instances is scattered the result is insecure and a mean value is therefore a better representation of the instances.

A refinement of k-nearest-neighbor is to weight each instance based on the distance between the training instance and the query instance (Mitchell, 1997). There exists a distance-weighted function for estimating both discrete- and real valued target functions. We have performed an experiment to see the differences in accuracy when using the four different function estimations and when the k value is set to 1. The result from this experiment is presented in table 4.5. It shows that the most accurate function estimation is the weighted real valued function, which is used in the rest of our experiments.

	1-N-N	Discrete-valued	Real-valued	Weighted discrete-valued	Weighted real-valued
K value	1	21	21	21	21
Known domains	696	696	696	696	696
Correct predicted	623	642	638	635	640
Underpredicted	73	54	58	61	56
Overpredicted	82	53	43	50	42
Accuracy (Q)	88.94	92.31	92.67	91.97	92.89

Table 4.5. Accuracy when using the different k-nearest-neighbor variants.

In the parsing of the training examples a sliding window is used as explained in chapter 4.2.1. When using the k-nearest-neighbor algorithm every second window sequence is stored as a training instance instead of all windows. The effect is a classification that requires less computation time and experiments have shown that the accuracy is not reduced. An optimization of the k value has been performed (figure 4.4). We used the weighted real-valued variant as described above and tested different k values ranging from 14 to 32. The accuracy was obtained using ten-fold cross-validation on a 160 protein dataset. We have not due to time limits optimized the k value for the different variants in table 4.5. A better optimization should include k values below 14 and some above 32, and should be performed using the different variants in table 4.5 as well as different class relationships as described in table 4.4. The result graph shows that the values 21 and 24 produced the most accurate result.

We have chosen to use 21 in the rest of our experiments. The sliding window size used in training and classifying instances is set to 11. Larger window sizes would make the algorithm very time consuming and our tests has shown that this size is large enough to produce accurate results.

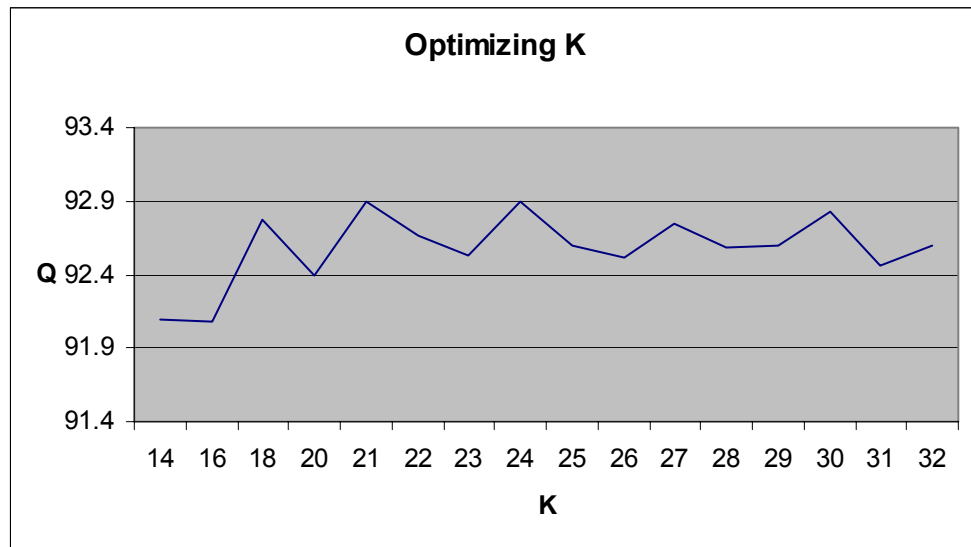


Figure 4.4. The differences in accuracy using different K values.

5 EXPERIMENTS

We will perform two experiments regarding transmembrane prediction. Both experiments will be performed and evaluated with both the backpropagation neural network and the k-nearest-neighbor algorithm. The idea behind the experiments is to evaluate different analysis methods and to compare our system with the state-of-the-art algorithm TMHMM. A complete experiment description is found in the following subchapters.

The accuracy of a prediction will be calculated using the accuracy formula Q (Chen et al, 2001) where

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

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 amount of predicted domains

A prediction is considered correct if the predicted domain overlaps the real domain with at least five residues (Krogh et al, 2001). The reason why the domains only needs to overlap with five residues are that the transmembrane proteins are bouncing a little bit up and down in the biological membrane and the boundaries are therefore not exactly defined.

Each experiment will be repeated five times since eager learning algorithms (for example backpropagation neural networks) have to converge to a single hypothes and there is no guarantee that the algorithm converges to the same hypothes in different runs (Mitchell, 1997). The result may therefore be slightly different in two tests. We will keep records of:

- Correct: The number of correctly predicted domains.
- Known: The total number of known domains in the classification set.
- Underpredicted: Number of known domains minus number of correctly predicted domains.
- Overpredicted: Number of false predictions.

Prediction accuracy, mean value and standard deviation will be calculated.

5.1 Prediction accuracy using different input scales

The purpose of this experiment is to evaluate different input transformations. The input always consists of the amino acid sequence and this sequence is translated into different hydrophobicity values used as input to the classifier. We will try to use the primary sequence (converted to a numerical scale where A is first and Z is last) as input as well as translating the sequence to the following hydrophobicity scales:

- Kyte-Doolittle
- kPROT
- ProtScale
- GCS

The purpose is to see if there are differences in prediction accuracy using different input value transformations. We will investigate, using our algorithms, if primary sequence analysis alone can compete with hydrophobicity in prediction accuracy.

5.2 Prediction accuracy compared to TMHMM

The hidden Markov model based prediction server TMHMM developed by Krogh et al. (2001) is the state-of-the-art system in full protein sequence predictions. We will therefore compare the accuracy of our own algorithms with TMHMM. To be able to compare the results we will use the same ten-fold cross validation set used in the development of TMHMM. The data set is available at the TMHMM website.

6 EXPERIMENT RESULTS

6.1 Prediction accuracy using different input encodings

The dataset for this experiment contains 160 proteins and a total of 696 known transmembrane domains. The result has been obtained using ten-fold cross validation. An overview of the result is shown in figure 6.1. The complete experiment data can be found in table 6.1. The diagram shows that Kyte-Doolittle was the most accurate hydrophobicity scale using our system. It also shows that the backpropagation neural network (ANN) was in most cases slightly more accurate than the k-nearest-neighbor (KNN). The greatest difference between the classifiers is when using primary sequence as input. The accuracy using k-nearest-neighbor was in this case more than 40% greater than the backpropagation neural network. This is due to the extremely low amount of overpredictions probably because the hamming distance was used as distance function (see the k-nearest-neighbor chapter).

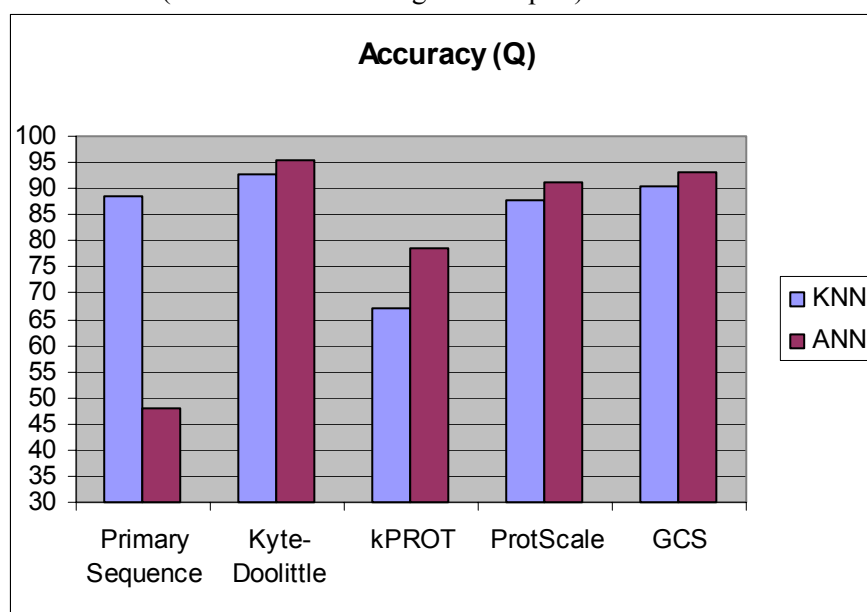


Figure 6.1. Result diagram from the different input scales experiment

	k-Nearest-Neighbor			Backpropagation ANN		
Primary sequence	Best result	Mean value	Standard deviation	Best result	Mean value	Standard deviation
Correct predicted	575	575	0	608	606.4	2.07
Underpredicted	121	121	0	88	89.6	2.07
Overpredicted	29	29	0	1679	1684.6	11.50
Accuracy	88.68	88.68	0	48.19	48.02	0.10
Kyte-Doolittle	Best result	Mean value	Standard deviation	Best result	Mean value	Standard deviation
Correct predicted	640	640	0	667	667	2.74
Underpredicted	56	56	0	29	29	2.74
Overpredicted	42	42	0	33	36.6	2.70
Accuracy	92.89	92.89	0	95.56	95.31	0.17
kPROT	Best result	Mean value	Standard deviation	Best result	Mean value	Standard deviation
Correct predicted	407	407	0	554	553.4	2.07
Underpredicted	289	289	0	142	142.6	2.07
Overpredicted	122	122	0	158	159.6	2.51
Accuracy	67.08	67.08	0	78.70	78.56	0.11
ProtScale	Best result	Mean value	Standard deviation	Best result	Mean value	Standard deviation
Correct predicted	579	579	0	643	645	2.92
Underpredicted	117	117	0	53	51	2.92
Overpredicted	47	47	0	69	76	6.93
Accuracy	87.72	87.72	0	91.34	91.05	0.25
GCS	Best result	Mean value	Standard deviation	Best result	Mean value	Standard deviation
Correct predicted	604	604	0	651	655	3.08
Underpredicted	92	92	0	45	41	3.08
Overpredicted	38	38	0	44	54.2	6.65
Accuracy	90.36	90.36	0	93.60	93.23	0.25

Table 6.1. Results from the different input scales experiment.

6.2 Prediction accuracy compared to TMHMM

The data set used in the experiment contained 160 proteins with a total of 696 transmembrane domains. The input scale using k-nearest-neighbor and backpropagation neural network was Kyte-Doolittle because the previous experiment showed that this was the most accurate hydrophobicity scale. The experiment result is shown in the diagram below. It shows that we were unable to locate transmembrane domains with the same accuracy as TMHMM. The best result was however only two percent lower. It is important to notice that TMHMM is based on both hydrophobicity and charge bias as input signals and uses a post-processing where the expected number of domains in a protein is estimated. This estimation is very complex and is therefore not implemented in our system at the moment. The complete result from the experiment is found in table 6.2. Table 6.2 also shows the time needed to perform a ten fold cross-validation on the data set. K-nearest-neighbor is a lazy learner and is very fast at training but requires lot of computation time in classifying instances. Backpropagation neural network is the opposite (eager learner) and is therefore fast at classifying instances but slow in training the algorithm. Therefore the time aspect should always be considered when a specific algorithm is chosen for protein sequence classification.

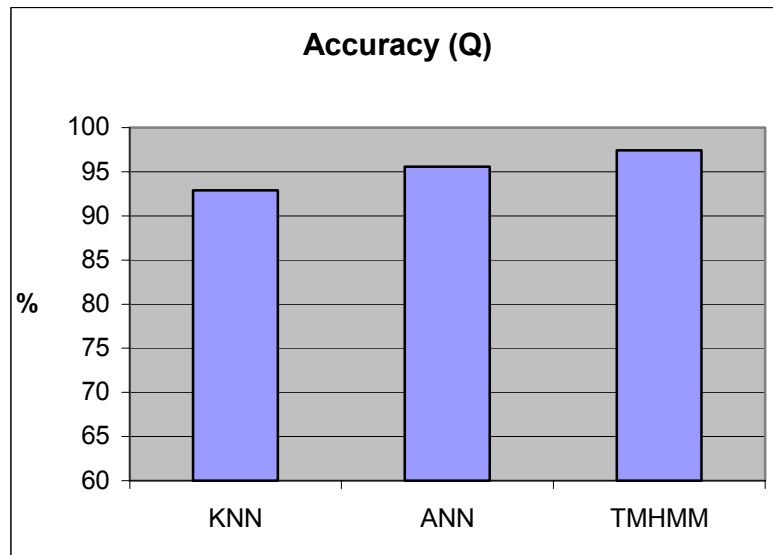


Figure 6.1. Result from the prediction accuracy experiment.

k-Nearest-Neighbor			
Input scale	Kyte-Doolittle		
Training time	0 minutes 3 seconds		
Classification time	31 minutes 16 seconds		
	Best result	Mean value	Standard deviation
Correct predicted	640	640	0
Underpredicted	56	56	0
Overpredicted	42	42	0
Accuracy	92.89	92.89	0
Backpropagation Neural Network			
Input scale	Kyte-Doolittle		
Training time	18 minutes 22 seconds		
Classification time	4 minutes 32 seconds		
	Best result	Mean value	Standard deviation
Correct predicted	667	667	2.74
Underpredicted	29	29	2.74
Overpredicted	33	36.6	2.70
Accuracy	95.56	95.31	0.2
TMHMM			
	Best result	Mean value	Standard deviation
Correct predicted	677	-	-
Underpredicted	19	21,7	1,8
Overpredicted	17	20,1	0,6
Accuracy	97,41	-	-

Table 6.2. Results from the prediction accuracy experiment.

7 CONCLUSION

In the experiment where different input scales were compared the Kyte-Doolittle produced the most accurate result using both k-nearest-neighbor and the backpropagation neural network algorithms. Experiments made by Chen et al (2001) have shown the same result. There is however no information in their article about which hydrophobicity scales they used besides Kyte-Doolittle. Sandri (2000) points out that there exists about 160 different hydrophobicity scales in the literature and since we have not tried out more than four scales we cannot draw any conclusion that Kyte-Doolittle is the most accurate. There is however a reason to believe, based on the results from our experiments as well as the experiments made by Chet et al (2001), that Kyte-Doolittle is a hydrophobicity scale that produces very accurate classifications and should be tried out in every prediction system. It is however possible that there exist a hydrophobicity scale that is more accurate using our data set and approach.

The experiment has also shown that classifications based on primary sequence analysis produces less accurate results than using hydrophobicity analysis. Our system was, using primary sequence as input, able to locate less transmembrane domains than using the hydrophobicity based input scale. A hydrophobicity scale can be seen as a relation between the amino acids. If we used a larger data set for training it is possible that the machine learning algorithms was able to find a relation between the amino acids that is more accurate than the hydrophobicity scales.

Our experiments showed that we were unable to locate transmembrane domains with the same accuracy as the state-of-the-art system TMHMM. The best result using the backpropagation neural network as machine learning algorithm was about 2% less accurate than TMHMM. One reason that we failed to reach the accuracy of TMHMM is that our system is based on hydrophobicity analysis alone while TMHMM is based on both hydrophobicity and charge bias analysis. Krogh et al (2001) has shown that some transmembrane domains are impossible to locate using hydrophobicity analysis alone. An improvement to our system could therefore be to implement charge bias analysis functionality. We have found very little information about how to use charge bias as input signal and it is therefore not implemented in our system at the moment. TMHMM uses an estimation of the expected number of domains in a protein. If a protein is for example expected to have four domains and a classification resulted in five predicted domains, the region least likely to be a transmembrane domain could be removed. This feature could therefore be used to reduce the amount of overpredicted domains.

We have also found out that the k-nearest-neighbor algorithm were unable to locate transmembrane domains with the same accuracy as the more common used neural networks. One reason is that neural networks are known to well handle noisy data sets (Mitchell, 1997). An interesting result is that the k-nearest-neighbor algorithm with primary sequence analysis had an extremely low amount of overpredictions due to the use of hamming distance. When hamming distance is used the nearest neighbors is almost exactly the same as the query instance since the distance is the number of characters that differ between the sub sequences. When using a hydrophobicity scale the amino acid sequence between two instances could be very different even though they have a short Euclidean distance between each other. When hydrophobicity scales are used the system is able to locate more transmembrane domains but the disadvantage is more overpredictions. In some cases of protein sequence classifications where the relation between the output and the primary sequence as input are stronger this algorithm might be very accurate.

K-nearest-neighbor and backpropagation neural networks have different advantages. A system using both k-nearest-neighbor with primary sequence analysis and hamming distance as distance function and backpropagation with hydrophobicity analysis combined with a third classifier with the purpose of choosing the most likely topology based on the results from the first two classifiers could maybe improve the accuracy even more.

8 FUTURE WORK

We could not compare how well our approach of dividing the instances in the four classes globular, near, helix cap and helix is compared to the TMHMM system. To do this we need two improvements:

1. Implement a combination of charge bias and hydrophobicity analysis as input signal.
2. Implement and use a hidden Markow model as classifier. This has not been performed due to time limits.
3. Implement an estimation of the expected number of domains in a protein which is a part of the TMHMM system. There exists no rules for this estimation and it is very complex to implement this functionality. We have not found any information in the TMHMM article about how their estimation is performed.

Our system does not predict the full topology of the transmembrane proteins. The full topology of a membrane protein is the transmembrane domains as well as the in- and outside regions. This functionality requires charge bias analysis to support the positive inside rule (von Heijne, 1997). The approach has to be changed as well. We divide the instances in the four classes globular, near, helix cap and helix where globular and near can be located in both out- and inside regions. If we were to predict the full topology we need different classes depending on if the instance is located on the out- or inside of the biological membrane.

It is possible that we could improve the accuracy further by optimizing the parameters for the machine learning algorithms. This could be performed using an optimization algorithm such as genetic algorithms or hill climbing. It is however extremely time consuming and therefore we have not at the moment optimized our system using this approach.

As mentioned in the conclusion chapter a combination of k-nearest-neighbor and backpropagation neural network with a third classifier responsible for choosing the most accurate topology from the results of the first two algorithms could improve the accuracy of the system.

APPENDIX A: NEURAL NETWORKS

The first article on neural networks was written in 1943 by McCulloch and Pitt (Russell et. al., 1995). Neural network simulates the working of neurons, axons and synapses in a human brain. Even if the computer never could simulate all neurons in a human brain we can still use this technology in learning. Neural networks are quite unpredictable and it is hard to understand what the program really learns. In most learning algorithms it is quite easy to look at the training set and then predict what the outcome will be. With neural networks this can be really hard to do (Rojas, 1996).

A neural network consists of nodes that are bound together by weighted connections. The processing is done in parallel through the whole network without any central control. There are several different types of neural networks. The simplest type is the single-layer network. It is called a perceptron network, illustrated in Figure A.1, and is actually only input nodes and one output node.

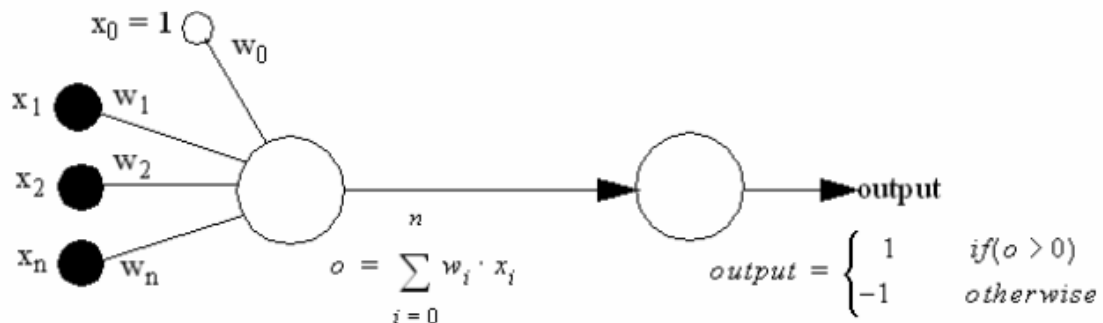


Figure A.1 Illustration of a perceptron

The output of perceptron is always 1 or -1, i.e. a boolean output. A perceptron network can only represent linear-separable functions, see figure A.2.

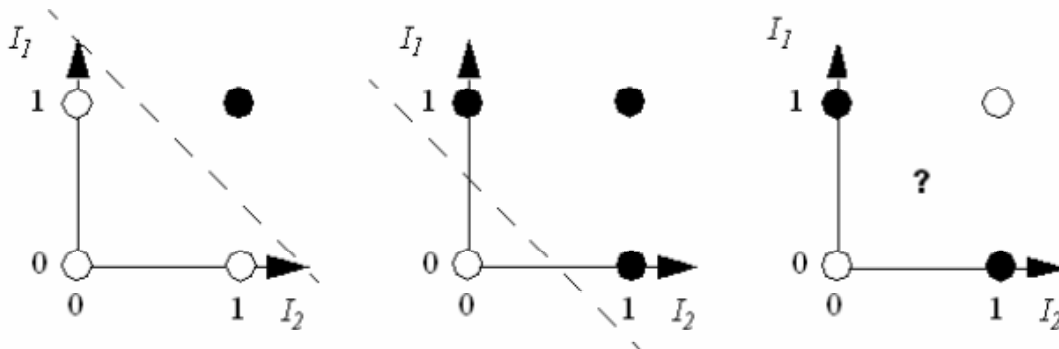


Figure A.2 Linear-separable problems; I1 AND I2 (left), I1 OR I2 (middle) and I1 XOR I2 (right). The XOR problem is not linear-separable.

Example of linear-separable functions is boolean functions as OR and AND. A XOR function can be solved by a two layer perceptron network which is learnt by back propagation.

In multi layered networks where we have hidden nodes between the input and the output nodes. A multi layered network can contain perceptron nodes or sigmoid nodes which is quite different from the perceptron node that only can output a boolean value. A sigmoid node, illustrated in Figure A.3, uses a different activation function often based on tanh or ex. The output of a sigmoid node is always between 0 and 1. The

biggest problem with a multi layered net is that it is not as easy to learn as a single layered network with only one perceptron.

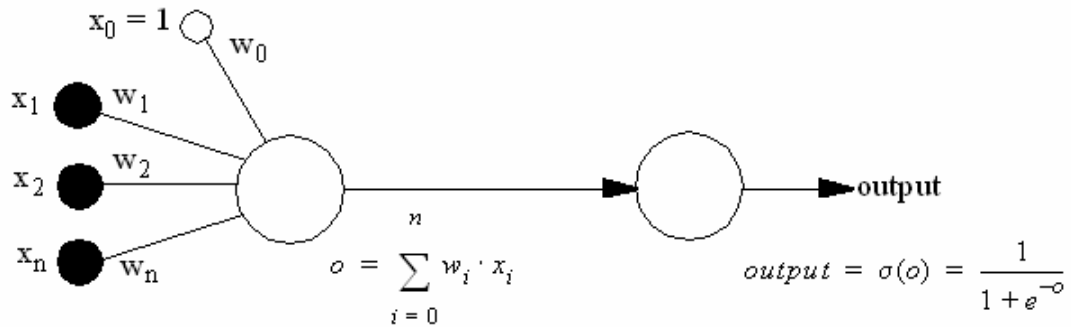


Figure A.3 Illustration of a sigmoid node

To solve the problem with learning multi layered networks Bryson and Ho came up with the back propagation algorithm in 1969. But it became ignored until the mid 80s. One of the reasons might be that this method is quite computing intensive.

There are other neural network variants, e.g. the Hopfield network and the Kohonen network. The Hopfield goes one step closer to nature by taking the energy of a network in count and a Kohonen network is described as a self-organizing map. Both algorithms are recurrent and differ from the usual feed forward networks by the fact that the graph is not one-way (Haykin, 1999).

The neural network has several advantages to other machine learning algorithms, e.g., it can handle noisy data well (Jain et al., 1996). One problem is that they can be quite hard to learn and it may take some time to go through a training set. And it will often take several iterations of the training set to achieve a low error rate. Another problem is to stop the training before the network is over fitted, i.e., a little too good at the training set which can lead to worse results on a validation set or real classification data (Rojas, 1996).

APPENDIX B: K-NEAREST-NEIGHBOR

Instance-based learning methods are often called lazy learners. The reason is that they do not construct a description of the target function during training time. Instead a generalization of the training data is postponed until a new instance shall be classified (Mitchell, 1997). The opposite is eager learners, for example the backpropagation algorithm. The advantage of instance-based learning is that different approximations to the target function is constructed each time an instance shall be classified. The disadvantage is that classification requires long computation time compared to eager learners.

The most basic instance-based method is k-nearest-neighbor. In this algorithm all training instances are regarded as distinct points in an n-dimensional space. The nearest neighbors of a query instance are the instances with for example the shortest Euclidean distance to the query point. The distance is calculated using the formula

$$d(x_i, x_j) = \sqrt{\sum (a_r(x_i) - a_r(x_j))^2}$$
 where $a_r(x_i)$ denotes the value of the r th attribute of instance x_i .

The instances located near a query instance are used to approximate the target function. In 1-nearest-neighbor the output value of a query instance is the same as the output value of the nearest training example. In k-nearest-neighbor the output value is the most common value among the k nearest instances (if the target function is discrete-valued) or the mean value of the k nearest neighbors (if the target function is real-valued).

One difficulty with k-nearest-neighbor is the *curse of dimensionality* (ibid.). This occurs when there are a lot of irrelevant attributes in each instance. If, for example, an instance consists of 20 attributes but only two of them are relevant in the approximation of a target function two instances with very different values of the relevant attributes can be located very close to each other when the values of the other 18 attributes are similar. The output would then be very misleading. This is however not a problem when classifying protein sequences because every residue in the sequence is relevant.

As mentioned earlier lazy learners have an advantage in that the generalization of the target function is postponed until classification time. An interesting question is if this difference affects the inductive bias of the learners when using the same instance space. The answer is yes because

- Lazy learners can consider the query instance when approximating the target function
- Eager learners must approximate a target function before the query instance is known

An eager learner must therefore commit to a single target function that has to cover the whole instance space while a lazy learner can use different local approximations to form the global function hypothesis (ibid.).

APPENDIX C: SLIDING WINDOW

Since protein sequence and transmembrane domains is of variable length a sliding window is used to divide the sequence into sub sequences of fixed length. For every residue r in a protein sequence (in training as well as classification) a sub sequence located around r is used as a training or query instance. Figure C.1 illustrates a sliding window with the size of 19.

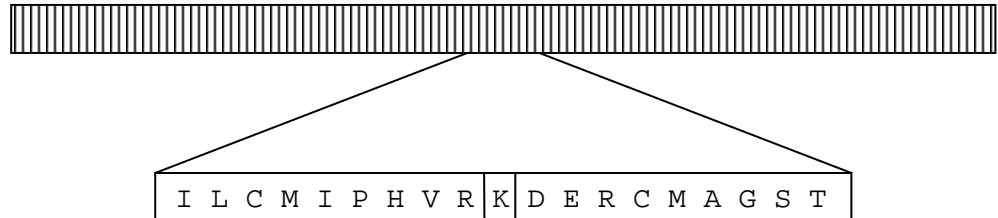


Figure C.1 Illustration of a sliding window. The upper rectangle represents a protein sequence. Each white space represents a residue. The lower rectangle represents the 19 residues that are part of the sub sequence for the residue K used as input to the system.

REFERENCES

- Arikawa, S., Kuhara, S., Miyano, S., Shinohara, A., Shinohara, T. (1992). "A Learning Algorithm for Elementary Formal Systems and its Experiments on Identification of Transmembrane Domains". IEEE
- Baldi, P., Brunak, S. (2001), "Bioinformatics – The Machine Learning Approach, second edition". MIT Press
- Black, P. E. (2002). "Dictionary of Algorithms and Data Structures".
<http://www.nist.gov/dads>
- Chen, Z., Liu, Q., Zhu, Y., Yixue, Li, Xu, Y. (2001). "A hydrophobicity based neural network method for predicting transmembrane segments in protein sequences". IEEE
- Childs, G. V. (2001). "Membrane Structure and Function".
http://cellbio.utmb.edu/cellbio/membrane_intro.htm
- Guy, H. R. (1985). "ProtScale: Hydrophobicity scale based on free energy of transfer (kcal/mole)". Biophys J. 47:61-70
- Haykin, S. (1999). "Neural Networks". Prentice-Hall. New Jersey. US
- Jain, A., Mao, J., Mohiuddin, K. (1996). "Artificial Neural Networks: A tutorial". IEEE Computer Special Issue on Neural Computing
- Koza, J. R. (1998). "Genetic Programming II". MIT Press. 430-492
- Kyte J., Doolittle, R.F. (1982). "A simple method for displaying the hydrophobic character of a protein". J. Mol. Biol. 157, 105-132.
- Krogh, A., Larsson, B., von Heijne, G., Sonnhammer, E.L.L. (2001). "Predicting Transmembrane Protein Topology with a Hidden Markov Model: Application to Complete Genomes". Academic Press
- Lipson, E., Catterall, S., Vidali, G. (2003). "SimScience - A web site devoted to areas of science where computer simulations are at the forefront of discovery".
<http://www.simscience.org>
- Metzenberg, S. (1996). "Amino acids and protein structure".
<http://www.escience.ws/b572/L9/L9.htm>
- Mitchell, T.M. (1997). "Machine Learning". McGraw-Hill
- Pilpel, Y., Nir, B-T., Lancet, D. (1999). "kPROT: A Knowledge-based Scale for the Propensity of Residue Orientation in Transmembrane Segments. Application to Membrane Protein Structure Prediction". Academic Press.
- Rojas, R. (1996). "Neural networks". Springer Verlag. Berlin
- Russell, S., Norvig, P. (1995). "Artificial Intelligence - A Modern approach". Prentice-Hall. New Jersey. US
- Sandri, L. (2000). "GCS". <http://www.bbcm.univ.trieste.it/~tossi/HydroPlot/HydroPlot.html>
- Sengbush, P. V. (2002). "Molecular Genetics - Genetic Information: The Genetic Code, Transcription, Translation (Protein Biosynthesis) and Replication". <http://www.biologie.uni-hamburg.de/b-online/e21/21.htm>
- von Heijne, G. (1997). "Principles of membrane protein assembly and structure". Prog. Biophys. molec. Biol., Vol. 66. 113-139.
- Wampler, J.E. (1996). "Tutorial on peptide and protein structure"
<http://bmbiris.uga.edu/wampler/tutorial/prot0.html>