UPTEC X 02 017
APR 2002

ISSN 1401-2138

ERIK GRANSETH

Predicting nuclear localization signals by using an artificial neural network approach

Master's degree project

Molecular Biotechnology Programme
Uppsala University School of Engineering

UPTEC X 02 017 | **Date of issue 2002-04**

Author

Erik Granseth

Title (English)

Predicting nuclear localization signals by using an artificial neural network approach

Title (Swedish)

Abstract

Nuclear localization is predicted by artificial neural networks, based on the amino acid sequence alone. The network is trained on proteins containing nuclear localization signals. The network had a Mathews' correlation coefficient of 0.46, sensitivity of 0.43 and specificity of 0.69 if incorporated into TargetP and 0.34, 0.45 and 0.49, respectively, alone. The method seems promising and there is plenty of room for improvement, when more is known about nuclear localization.

Keywords

Nuclear localization signals, nuclear localization, artificial neural networks

Supervisors

Gunnar von Heijne
Olof Emanuelsson
Stockholm Bioinformatics Center

Examiner

Arne Elofsson
Stockholm Bioinformatics Center

Project name	Sponsors
Language **English**	Security
ISSN 1401-2138	Classification
Supplementary bibliographical information	Pages **29**

Biology Education Centre Biomedical Center Husargatan 3 Uppsala
Box 592 S-75124 Uppsala Tel +46 (0)18 4710000 Fax +46 (0)18 555217

Predicting nuclear localization signals by using an artificial neural network approach

Erik Granseth

Sammanfattning

För att proteiner ska kunna fungera krävs det att de är på rätt plats vid rätt tidpunkt i cellen. I växt- och djurceller finns det membranskilda organeller med olika uppgifter, som kan liknas vid organ. En av dessa organeller är cellens kärna, som innehåller informationen om alla cellens proteiner; DNAt. I det här examensarbetet har det undersökt om det går att identifiera proteiners nukleära lokalisering utifrån dess aminosyrasekvens. En del nukleära proteiner har så kallade nukleära lokaliseringssignaler i deras aminosyrasekvenser som hjälper till vid transport genom kärnmembranet.

Här har ett artificiellt neuralt nätverk tränats att känna igen dessa signaler. Inspirationen till hur neurala nätverk fungerar kommer ifrån den mänskliga hjärnan och dess förmåga att kategorisera information och lära sig saker. När sedan ett okänt protein presenteras för nätverket, ska det kunna känna igen om det finns en nukleär lokaliseringssignal i aminosyrasekvensen eller ej. Om signalen finns klassificeras proteinet som nukleärt och om den inte finns som icke-nukleärt. Detta nätverk känner igen 45% av proteiner som är nukleära. Av proteiner som inte är nukleära klassificerar nätverket 12% av dem felaktigt som nukleära.

Examensarbete 20 p i Molekylär bioteknikprogrammet

Uppsala Universitet April 2002

Contents

1 Introduction

Now that the human genome is almost fully sequenced, there is no sign that the genomic data will stop its exponential growth. This leads to the necessity of new tools and approaches for understanding of what the available information actually stands for. There is right now a vast amount of different tools available via the World Wide Web that can predict and classify biological information. Computational time is now cheap, but actual lab-experimentation is expensive, so these in silico predictions can reduce the time and effort in the wet labs substantially. One example is TargetP,[1] a program that predicts the subcellular localization of an amino acid string.[2] It can recognize chloroplast transit peptides, mitochondrial targeting peptides and secretory pathway signal peptides. If you have an unknown protein sequence you can then at least get a hint of where to look for it in the lab.

This report describes the development of a tool that predicts nuclear localization signals (NLSs) by using artificial neural networks which later is to be incorporated into TargetP's framework. Artificial neural networks are well suited for the analysis of molecular sequence data.[3] It is a very potent method that is able to solve problems by training on examples with known characteristics. The trained network can then be used to classify unknown examples.

NLSs are not confined to a specific region of the amino acid sequence. They are less well defined than other target peptides and do not have a consensus sequence. This makes them more difficult to predict and especially where in the sequence they are located. The study of nuclear transport is important because communication between the nucleus and the cytoplasm, which involves the transport of proteins through the nuclear envelope, is often a key step in gene regulation.[4] Many viruses also have nuclear localization signals and are therefore interesting to inhibit. Since most nuclear proteins are involved in some way with the DNA, they are key proteins when it comes to gene regulation.

2 Background/Theory

2.1 Biological Background

2.1.1 Proteins

The proteins are the machinery of the cell. They catalyze chemical reaction, they act as transporters, they help other proteins fold and they pack the DNA, their own blueprint. The DNA is a double helix consisting of four different base pairs and is located in the nucleus. It is transcribed into mRNA, similar to DNA, but with a different backbone and one base changed. mRNA is transported out from the nucleus to the cytoplasm and there translated into proteins.

All proteins consist of amino acids joined by peptide bonds. There are 20 different kinds of amino acids and they are varied into proteins that can be up to several thousands amino acids long. The amino acid sequence is called the primary structure of the protein. The different amino acids can either be arranged in a helical, coiled or sheet structure, and this arrangement is the secondary structure. The three-dimensional composition of these elements is their tertiary structure and the quaternary structure is how several polypeptides are bound together.

2.1.2 Targeting Peptides

Most proteins that are transported through a membrane have an amino terminal targeting sequence.[5] This sequence is recognized by a targeting system on the cis side of the membrane. The system aids the transport of the protein through a transmembrane channel.[6] There exists a large amount of these membrane proteins since a protein's function is directly dependent on correct subcellular localization. Various amino terminal targeting sequences direct proteins to the mitochondrion, the chloroplast and the plasma membrane. Proteins that are transported into the nucleus have a different targeting sequence that is explained in 2.1.5.

Up to 25% of randomly generated peptides can aid a protein through a membrane.[6] Therefore it is assumed that that the targeting peptides primary sequences are highly degenerative, and that it is their secondary structure or a similar distribution of charged and apolar residues that matter. But "real" signals are much better than the randomly generated ones in terms of the rate and quantity of proteins transported through the membrane.

2.1.3 Nuclear envelope

Eukaryotic cells differ from prokaryotic in one obvious way; the eukaryotic cell confines its DNA in a compartment, the *nucleus*. It is separated from the cytosol by the nuclear envelope consisting of two membranes separated by a perinuclear space.[7] The inner membrane is in contact with the nuclear lamina and the outer membrane is continuous with the endoplasmic reticulum membrane in the cytosol.

There is a lot of traffic through the envelope: mRNAs and all cytoplasmic RNAs have to be exported from the transcription site in the nucleus. Conversely, the nuclear proteins need to be transported from the place where they are assembled, the cytoplasm, to where they are needed, the nucleus. The magnitude of import into the nucleus can be exemplified by the histones, which are needed during the period of DNA synthesis to associate with a diploid complement of chromosomes. Histones form around half the protein mass of chromatin so around 600 000 chromosomal proteins must be imported per minute during cell division. There is a continuous flow of 3000 mRNA molecules per minute out from the nucleus. To double the amount of rRNA in one cell cycle, 15 000 ribosomal subunits need to be exported.

In order to assemble with the rRNA, ribosomal proteins must first be imported into the nucleus as free proteins and out again as ribosomal subunits. The import of ribosomal proteins is ~80 times larger than the export of ribosomal subunits (~1 200 000 per minute).[8] The most well studied mechanism of active transport through the nuclear envelope is the Ran GTPase cycle,[9] which occurs at the Nuclear Pore Complex.

2.1.4 Nuclear Pore Complex

The Nuclear Pore Complex is a large protein assembly, 125 MDa that form an aqueous channel through the nuclear envelope. It consists of 50-100 distinct polypeptides in vertebrates.[8] Molecules that are smaller than 9 nm in diameter (~60 kDa) can diffuse freely through the pore with a rate that is inversely proportional to their size. It takes a few hours for the levels of an injected protein to equilibrate between the cytoplasm and the nucleus. Proteins that are larger than this need to be actively transported through the envelope and particles as large as 25 nm (~25MDa) can be transported through it.[10] This is larger than the actual radius of the pore, so it is possible that the pore can widen, but some large substrates such as ribonucleoprotein particles may have to change their conformation to pass.

There are approximately 3000 pore complexes in an animal cell and a major question is whether all nuclear pores are identical or whether they have functional differences. The question is raised because there are different known pathways for nuclear transport, each involving carrier proteins that takes the substrate through the pore. There also exist nuclear export signals and nuclear retention signals that hinder proteins from nuclear export but they are not investigated further in this project.

2.1.5 Nuclear Localization Signals

Nuclear localization signals facilitate the transfer of the protein through the nuclear envelope. If the signal is mutated the translocation is disrupted. Other signal sequences such as the ones for the endoplasmic reticulum, the mitochondria, the peroxisome and the bacterial plasma membrane, have some kind of consensus sequence. This is not the case for the nuclear localization signal.

NLSs differ from many other localization signals in that they can be present anywhere in the amino acid sequence, not just the N-terminal part. There are two “classic” NLSs: the monopartite NLS, which is at least four basic residues followed by a helix-breaking one,[11] and the bipartite, which consist of two basic clusters separated by 9-12 variable residues.[12] These sequences can be found in some nuclear proteins, but they can also be found in many non-nuclear proteins.[9]

There is some confusion about the term nuclear localization signal. Some authors only call the classic mono- and bipartite signal NLS, but here the term is defined as *a signal that helps the protein through the nuclear envelope*. If this signal is mutated the transport of the protein should decrease. Other theoretical generalizations for NLSs have been suggested as “NLS cores are hexapeptides with at least four basic residues and neither acidic nor bulky residues”, but this motif matches only few nuclear and many non-nuclear proteins.[13]

2.2 Computational Background

Neural Networks have many advantages that make them useful in molecular sequence analysis. One very important feature is their adaptive nature where learning by example replaces conventional programming in problem solving.[3] This makes them useful when the underlying understanding of the problem is incomplete, but where there exists lots of training data. Neural networks are error-tolerant and can deal with noisy data. They are also capable of capturing and discovering relationships and high-order correlations in input data.

3 Method

3.0.1 Historical Background

The modern era of neural networks began in 1943 when McCulloch and Pitts published a paper about the representation of an event in the nervous system.[14] The paper described a logical calculus of neural networks and was widely read at that time and lead to the construction of a computer (Electronic Discrete Variable Automatic Computer) developed from the first computer ENIAC. The new field of science attracted many scientists and psychologists that developed the field of Artificial Intelligence. In 1958, Rosenblatt came up with a new approach to the pattern recognition problem with his work on the perceptron.[15] It seemed like the perceptron could solve almost anything, but Minsky and Papert (1969) demonstrated that there are fundamental limits to what this one-layered network could compute. They suggested Multi Layer Perceptrons (MLP) to get over this problem. During the 70s the many of the researchers deserted the field, mainly because their theories were too time consuming on that times computers, and the Minsky and Papert paper did not exactly encourage them. In the 1980s, major contributions to the field emerged, and computational time became less expensive. In 1986, the back propagation algorithm was reported by Rumelhart, Hinton and Wilson. This is the most common training algorithm of MLPs. Actually the algorithm was invented in 1974 but no one had noticed. Since the 1980s the concept behind neural networks have been gaining popularity at the expense of rule based Artificial Intelligence. And now neural networks can do as amazing things as predicting nuclear localization.

3.0.2 Basic Concepts

The data available is divided into subsets, the training set and the test set. The training set is used to tune the neural network and practice it to recognize patterns. The test set is used to measure the performance of the model during the training. It is very important not to use test data that have been used to train the model, because then you would use *out-of-sample testing*. The performance is usually much better on the training set than the test set.

When using *supervised* learning, the test and training data must be labelled into classes before the training and testing begins. A *classifying* neural network uses the knowledge from these labelled examples to answer the question: Which class does this unknown example belong to?

Cross validation

Cross validation is used to find the best architecture and its optimal training parameters. Divide the data into n parts with approximately the same number of patterns. Then create n networks with the same architecture and training parameters. Each network is trained with n-1 parts and tested with the remaining one. This avoids misleading results, and mean and standard deviation can be calculated for the performance of the network. When using the neural networks for classifying unknown examples, all previously n networks are used, their outputs summed together and divided by n.

Overtraining

If the network is trained to long on the training data its ability to generalize decreases. This is called overtraining.[16] It is due to that the network also has learned the background noise of the data in the training set and is unable to classify new examples (Figure 1). This can also happen if there are too many free parameters to tune, i.e. if the number of nodes in the network is too large.

In order to not overtrain the network, one needs to have a stop criterion that terminates training before overtraining. Examples are: the root-mean squared error below a certain threshold, after a defined number of epochs or after a certain time.

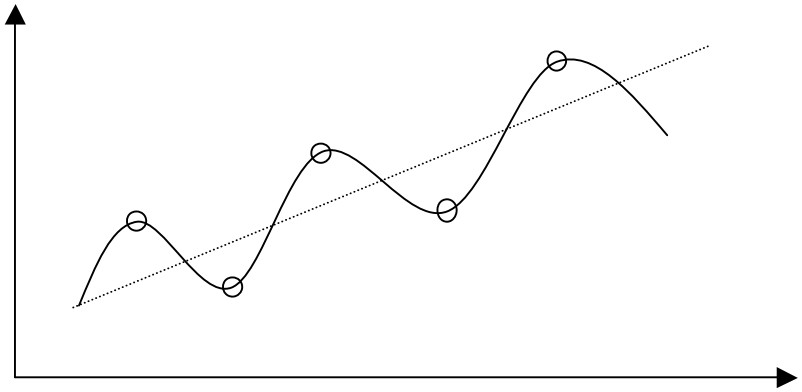

Figure 1. Circles are noisy data, dotted line show good generalization of underlying data. The solid curve is overfitted and thus have bad generalization ability.

3.0.3 The Multilayer Perceptron

Artificial Neural Networks is a broad category of different networks with different algorithms, calculating units and error functions. For this work the Multi Layer Perceptron have been used. The MLP is a very useful architecture for non-parametric modelling.[17] The network consists of interconnected nodes that are arranged into three main parts: the input layer, the hidden layer and the output layer (Figure 2). The hidden layer is optional. Usually one hidden layer is enough for most purposes.

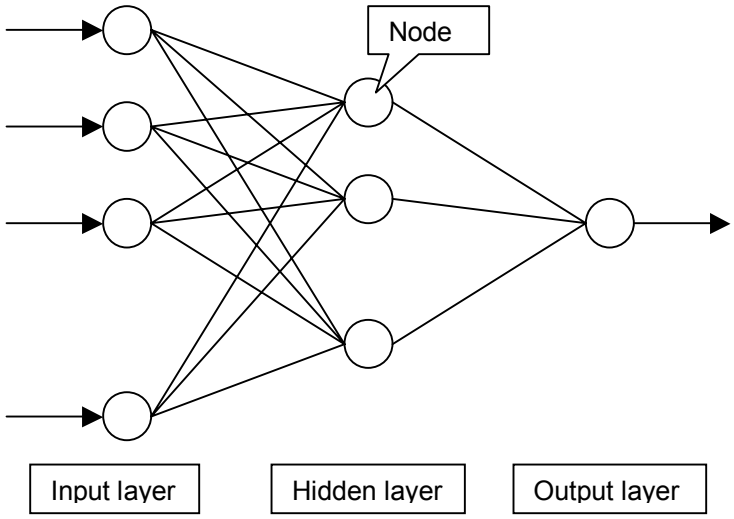

Figure 2. Example of the architectural arrangement of a neural network.

Each node can be thought of as a computational unit (Figure 3). The first part of the network is the input layer where the raw data is entered. Then the data is fed to the next part: one or more hidden layers. Each node in the hidden layer computes a sigmoidal function of a weighted sum of its inputs, and outputs a scalar value to the following layer. The final part is the output layer. The output node is a weighted sum of its inputs if it is an estimating network, or in this case, a sigmoidal function since it is a classifying network.

For each interconnection in the network there is a weight w_{ij}, which means the connection from the previous layers i-th node to the next layer's j-th node. The nodes of the input layer

and the hidden layer also have a bias weight, w_{0j}. This extra weight has a constant input x_0=-1, mainly for simplifying the notation and calculations.

It is these weights that are the free parameters in the network and they are adapting during the training phase. The global parameters are the network topology, learning rate, update frequency and the stop conditions. The learning rate is how much the weights change each epoch and the update frequency is if the weights are updated after every example presented (many times in each epoch) or if they are updated after all examples have been presented (once each epoch).

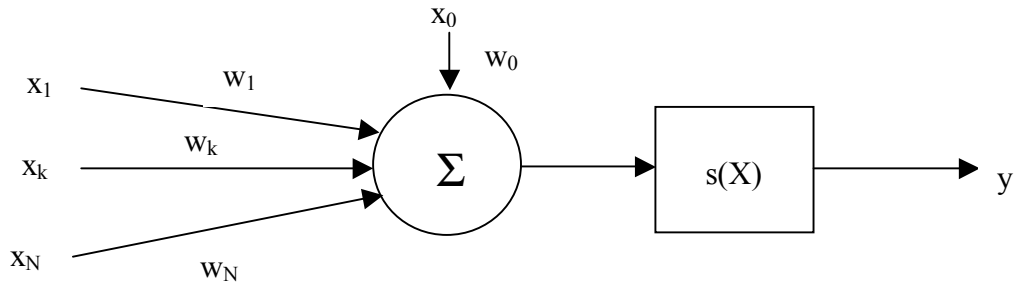

Figure 3. A simple node with sigmoidal function s(X). The output y is fed forward to the next layer. N is the number of nodes of the previous layer.

3.0.4 Redundancy Reduction

Collections of data tend to be biased. This is because research often is conducted where the money lies. Proteins like p53 (a key protein in cancer) or proteins concerning obesity, are well studied and more abundant in the databases. To avoid overtraining the network, from the fact that these proteins occur more frequently in the dataset, the data is redundancy reduced. One method that can be used is the Hobohm algorithm:[18]

Imagine that if your data is represented in space, you probably would get clusters of data points at different places in space (Figure 4). These clusters contain data that have small differences between each other, for instance the p53 protein from various organisms. The algorithm then calculates the number of neighbours each data point has within a certain radius. Then it removes the point with most neighbours, recalculates the neighbours with the point removed and so forth. Eventually no data points have any neighbours within the radius and the algorithm terminates.

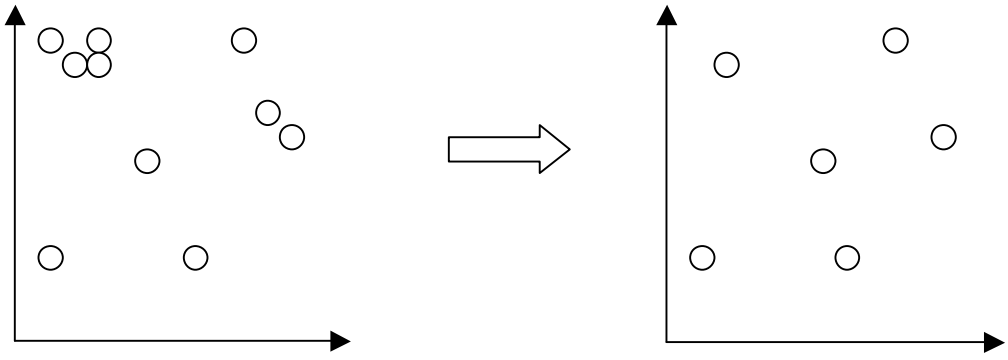

Figure 4. Example of a dataset in 2D before and after redundancy reduction.

The Hobohm algorithm was originally developed for creating non-redundant sets of proteins that are structurally different.

3.2 Phase 1

The first part of the project consisted of finding a well performing neural network that predicted the actual residues that belonged to the nuclear localization signal. The global parameters and the architecture of the network were examined.

3.2.1 The neural network simulator

The program that was used to simulate the artificial neural networks was *Billnet*.[19] It is under GNU public license so you can modify it, add features and use it freely. Its main advantage is that it can simulate many architectures and algorithms for neural networks in an effective and lightweight manner.[20]

3.2.2 Sliding window

Neural networks have a fixed number of input nodes, so the input data have to be of equal size. Since we were using the amino acid sequence as the input, we needed to convert the arbitrarily long sequences into fragments with the same length. This was done using a sliding window that converts the sequence into chunks with an equal size (Figure 5). X:s were inserted at the beginning and the end of the sequences.

```
                    SLIDING WINDOW
                            ==>
XSL SLI LID IDI DIN ING NGX XWI WIN IND NDO DOW OWX
```

Figure 5. Example of a sliding window of size 3.

3.2.3 Input format

Billnet uses a format called Billnet Data Format for understanding the data it uses. The format cannot use one-letter sequence abbreviations directly so the fragments have to be converted into numerical values. This was done using sparse encoding (BIT20) which means that every amino acid is translated into a 20-dimensional vector that consists of nineteen “0” and one “1”. The wild card X is represented by the null-vector consisting of only zeros (Figure 6).

```
X => 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
A => 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Q => 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
L => 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
S => 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
R => 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
E => 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
K => 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0
T => 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0
N => 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0
G => 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0
M => 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0
W => 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0
D => 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0
H => 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0
F => 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0
Y => 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0
C => 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0
I => 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0
P => 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0
V => 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
```

Figure 6. Encoding scheme.

There is no need of normalizing the input since they are all equally long in space. The number of input nodes is thus equal to the sliding window size times 20.

3.2.4 Network training

A fully connected artificial neural network with error back propagation was used. Back propagation is the algorithm for evaluating the derivatives of the error function. In the first stage the derivatives with respect to the weights are evaluated. In the second stage the derivatives are used to compute the adjustments to be made to the weights. The procedure is as follows:

For each node *j* of layer *k*, except the output layer, compute its output starting from the lowest layer.

$$y_{jk} = \frac{1 - e^{-net_{jk}}}{1 + e^{-net_{jk}}} \text{ where } net_{jk} = \sum_{l=1}^{N_{k-1}+1} w_{lj} x_l \quad \textbf{Equation 1}$$

w_{lj} is the weight of the connection between node *l* of layer *k*-1 to node *j* of layer *k*
x_l is the output of the connection from node *l* of layer *k*-1
N_{k-1} is the number of nodes in layer *k*-1 (the bias node is the reason why the number of inputs is plus one in the calculation of net_{jk})

For the output layer:

$$y_{jk} = \frac{1}{1 + e^{-net_{jk}}} \text{ where } net_{jk} = \sum_{l=1}^{N_{k-1}+1} w_{lk} x_l \quad \textbf{Equation 2}$$

Compute the average root-mean square error:

$$E_i = \frac{1}{2}\sum_{j=1}^{J}(d_{ij} - y_{ij})^2$$

Change the weight w_{jl} according to:

$$\Delta w_{jl} = -\eta \frac{\partial E_i}{\partial w_{jl}} + \alpha(\Delta w'_{jl} - \Delta w''_{jl})$$

where $\frac{\partial E_i}{\partial w_{jl}}$ is the partial derivative of the error with respect to the weight w_{jl}

η is the step size of the steepest descent (the learning rate) and α is the momentum term (not used, so α=0 in this case).

$\frac{\partial E_i}{\partial w_{jl}}$ must be computed from the output layer down to the input layer since the computation of successively lower layers depend on the computation of upper layers (back propagation):

$$\frac{\partial E_i}{\partial w_{jl}} = \frac{\partial net_{jk}}{\partial w_{jl}} \cdot \frac{\partial y_{jk}}{\partial net_{jk}} \cdot \frac{\partial E_i}{\partial y_{jk}}$$

where $\frac{\partial net_{jk}}{\partial w_{jl}} = x_l$ and $\frac{\partial y_{jk}}{\partial net_{jk}} = y_{jk}(1 - y_{jk})$

For $y = \frac{1 - e^{-net}}{1 + e^{-net}}$, the $\frac{\partial y_{jk}}{\partial net_{jk}}$ term simplifies to $\frac{1}{2}(1 + y)(1 - y)$ and for $y = \frac{1}{1 + e^{-net}}$ to $y(1 - y)$.

For nodes y_{jk} in the output layers $\frac{\partial E_i}{\partial y_{jk}} = (d_{ij} - y_{ij})$ and for nodes in other layers:

$$\frac{\partial E_i}{\partial y_{jk}} = \sum_{m=1}^{N_{k+1}} \frac{\partial net_{mk+1}}{\partial y_{jk}} \cdot \frac{\partial y_{mk+1}}{\partial net_{mk+1}} \cdot \frac{\partial E_i}{\partial y_{mk+1}} = \sum_{m=1}^{N_{k+1}} w_{jm} y_{mk+1}(1 - y_{mk+1}) \cdot \frac{\partial E_i}{\partial y_{jk+1}}$$

3.3 Phase 2

In phase 2 of the project, the scope was elevated from residue level to protein level. Proteins with known subcellular localization were fed into the previously trained network, each residue in their sequence yielding a value between 0 and 1. These values can be plotted (Figure 7) to an output plot. Ideally, one may see peaks that may be NLSs and classify the protein as nuclear. Non-nuclear proteins should have no clear peak(s) and therefore be classified as non-nuclear. The only thing considered was if the nuclear proteins had any substantial peaks in them or not, no consideration was taken to if the peak contained the actual NLS or not.

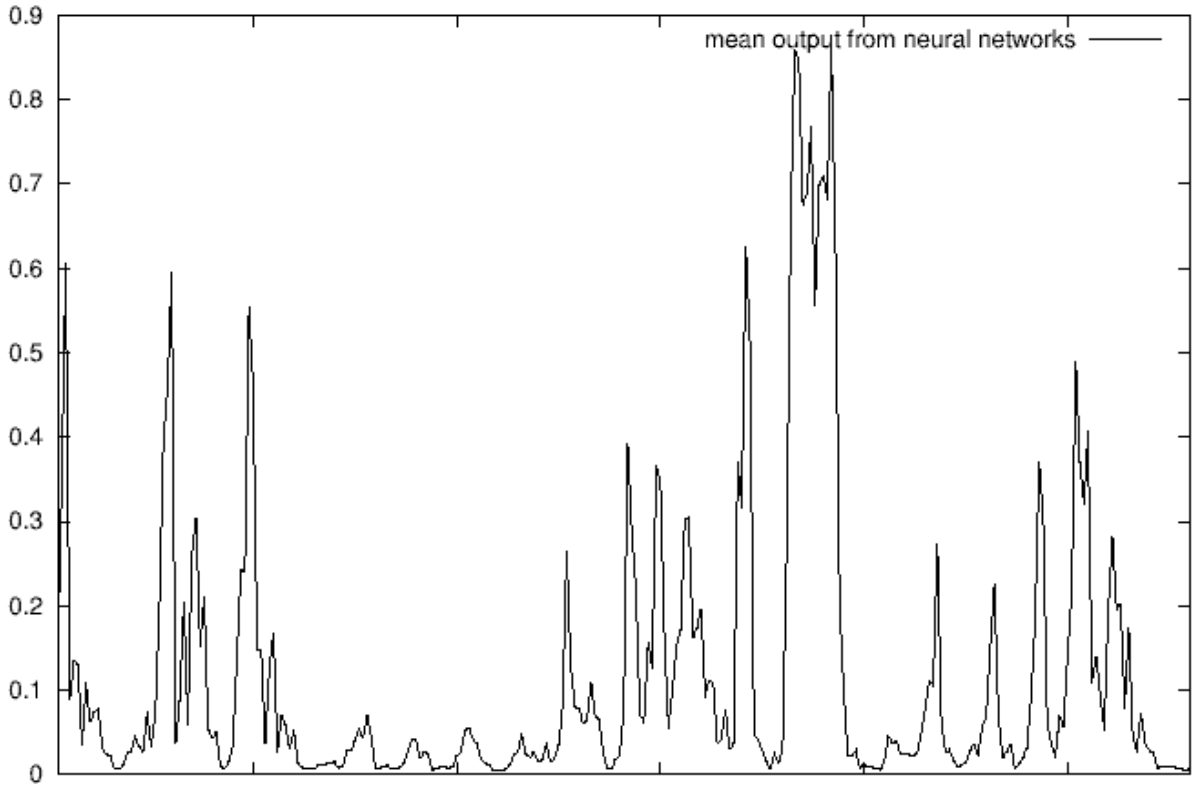

Figure 7. Example of an output plot of P41900, a nuclear protein. The x-axis is the amino acid sequence.

3.3.1 Evaluation of prediction performance

There are several ways to calculate the performance of binary predictors. One way is to use the Mathews correlation coefficient,[21] which is defined as:

$$M_C = \frac{P_t N_t - P_f N_f}{\sqrt{(N_t + N_f)(N_t + P_f)(P_t + N_f)(P_t + P_f)}}$$ where N_t and N_f is the number of true negatives and false negatives respectively and P_t and P_f is the number of true positives and false positives respectively (Figure 8). The result is between –1 and 1 where 1 means a fully perfect prediction and –1 a fully imperfect prediction. A value of 0 means that the prediction is as good as a random guess. Other useful definitions are the *sensitivity* and *specificity*, which are defined as:

$$Sensitivity = \frac{P_t}{P_t + N_f} \qquad Specificity = \frac{P_t}{P_t + P_f}$$

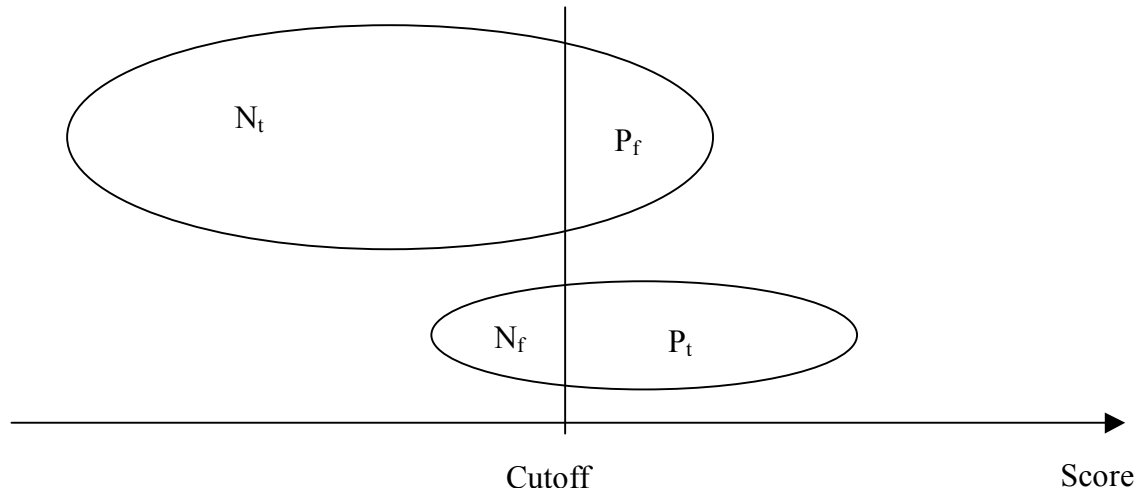

Figure 8. Illustration of the parameters P_t, P_f, N_t and N_f.

4 Results

The reason for partitioning the work into two parts (phase I and II) is that the first goal, to predict where in the amino acid sequence the NLS was situated was impossible at a useful performance level. The predictor always missed amino acids in the beginning and the end of the signal and classified too many false positives, but it seemed that these false positives were more abundant in nuclear proteins. It is probably due to the fact that there is no clear definition of what a nuclear localization signal is. The goal was changed to predict if a protein is nuclear or not without attempting to localize the actual NLS(s).

4.0.1 Dataset

Data was extracted from Swiss-Prot 40,[22] a data bank that contains amino acid sequences that are well defined and to some extent annotated.[23] Sequences that had a feature line (FT) with key name "NUCLEAR LOCALIZATION SIGNAL" was extracted. No consideration of species, length or annotation quality was taken so amino acid sequences with NLSs defined as "PROBABLE", "POTENTIAL" or "BY SIMILARITY" were also included in the data set. This was due to that there were too few sequences whose NLS was experimentally verified. The lengths of the NLSs were often very long (up to 30 residues) but a literature study revealed that just 4 or 5 residues belonging to the signal were mutated to test decreased nuclear transport, leaving the remaining part of them untested. Therefore, only proteins with NLSs shorter or equal to 8 residues were selected for further studies. These proteins were selected for redundancy reduction and aligned against each other by using the Smith-Waterman algorithm with the PAM250 matrix.[i] The whole amino acid sequences were used for redundancy reduction, not just the nuclear localization signal. Since the positive set is the residues belonging to the NLS and the negative set the rest of the amino acids, the negative set is much larger than the positive.

The Hobohm algorithm requires a complete matrix of pair relations (the Smith-Waterman score) among all proteins. Removal of the protein with the largest number of pair relations tends to minimize the total steps needed to remove all pair relations in the matrix. The matrix uses 1:s or 0:s. If the proteins have a sequence similarity above a certain threshold then they are considered as neighbours and the corresponding matrix value is 1, otherwise 0. The protein with most neighbours (1:s) is removed (its values set to 0). This is iterated until all values are 0. The dataset from SwissProt was reduced to 73 proteins with 92 nuclear localization signals (Table 1).

Proteins that have an NLS annotation	543
Proteins used for training and testing	73
Number of NLSs used for training and testing	92
Residues belonging to an NLS	560
Residues not belonging to an NLS	45708

Table 1. Data set used for training and testing the neural networks.

The data was separated into 5 different equally large subsets for cross-validation.

4.1 Phase 1

The first part of the project was to get the best possible architecture of the neural network for recognition of residues in an NLS. This was done by trying many different parameter values

[i] The PAM250 matrix contains the probabilities that one amino acid mutates to another after a particular evolutionary time

and studying the influence on the test set. The redundancy reduced set consisted of 73 proteins with an NLS annotation in SwissProt.

Neural networks are time consuming to train. Therefore, not all possible network architectures were tested. First different numbers of nodes in the hidden layer were tested with two different sizes of the sliding input window (Table 2).

nodes in hidden layer	*win size*	M_C	*Sensitivity*
3	**7**	0.4	34.2
3	**15**	0.42	41.8
5	**7**	0.41	34.4
5	**15**	0.41	40.6
7	**7**	0.42	36
7	**15**	0.42	41
9	**7**	0.42	33.2
9	**15**	0.4	36.6
11	**7**	0.41	34.8
11	**15**	0.4	40

Table 2. Number of nodes in hidden layer. Sensitivity corresponds on the fraction of residues belonging to the NLS.

The Mathews correlation coefficient, M_C is independent on the number of nodes in the hidden layer. However, neither the M_C nor the sensitivity is particularly good. This is because of the amount of negative examples in the training. Since just 560 out of 46268 examples were positive, the signals were hard to detect. In order to overcome that, the amino acids that belonged to the nuclear localization signal were repeated several times in each epoch. This was done only with the training set, not the test set. Hereafter xNLS refers to that the positive set was repeated x times each epoch in the training (Figure 9 and Figure 10).

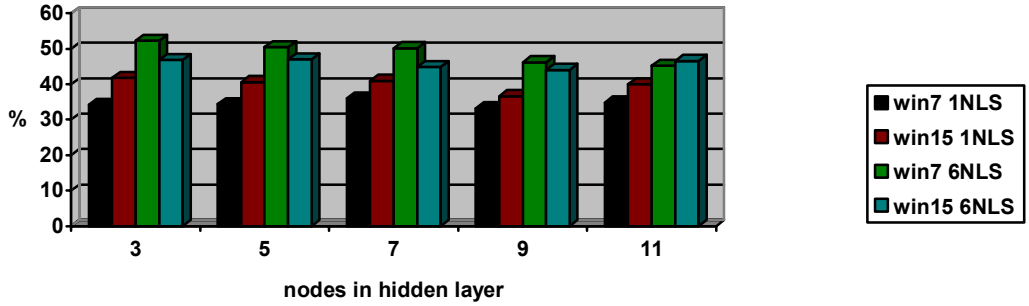

Figure 9. Diagram of the sensitivity for nodes 3-11 with two different window sizes and NLS repetitions.

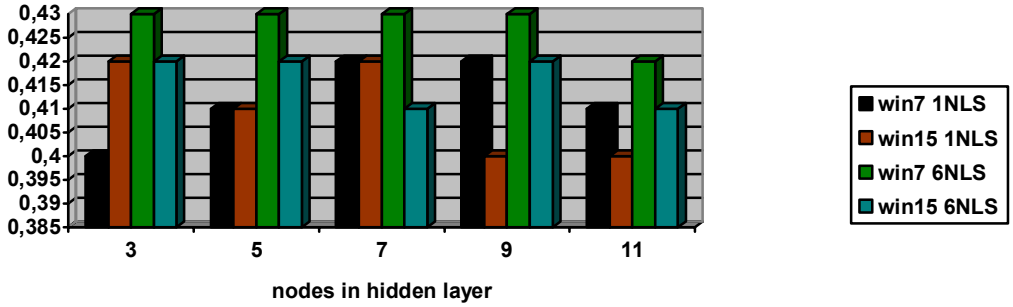

Figure 10. Diagram of Mathews correlation coefficient for nodes 3-11 with two different window sizes and NLS repetitions.

The M_C value is not much affected by increasing the number of nodes, but the amount of found NLSs (the sensitivity) is larger for small networks. Small neural networks are also less time-consuming to train and keeping down the number of free parameters may improve the generalization ability. The decision was made to go on with the network that had 3 nodes in the hidden layer.

The number of nodes in the input layer is directly related to the size of the window taken over the amino acid sequence. Since the actual signals are less than or equal to 8 residues long, a too large window may confuse the neural networks. The performance of various window sizes is discussed below (Table 3).

win size	M_C	*Sensitivity*
3	0.29	0.35
5	0.4	0.46
7	0.43	0.52
9	0.43	0.52
11	0.44	0.50
13	0.42	0.52
15	0.42	0.47

Table 3. Number of nodes in input layer, NLS repeated 6 times.

A window size of 5-11 has the best sensitivity and similar M_C-values. I chose to go on with a window size of 7 even if 11 had a greater M_C value, since it had a greater sensitivity. What the sliding window actually is doing is taking the neighbouring residues into consideration. It looks at the local environment around each residue. A large window size indicates that the network uses global information in its assignment and a small that it does not. The final parameter to investigate was the degree of repetition of the NLSs in each epoch (Figure 11).

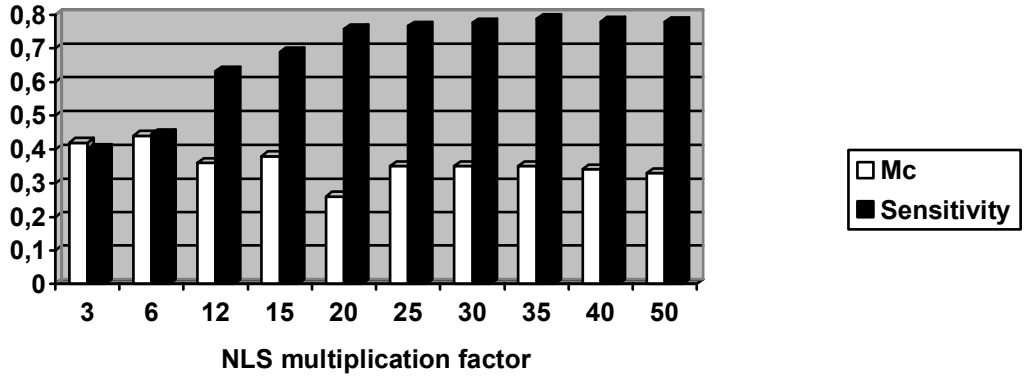

Figure 11. Network with 3 nodes in the hidden layer and a window size of 7.

The sensitivity levels out after an NLS multiplication factor of 20, but the M_C value is better at low multiplication factors. That is because those networks were better at predicting residues that did not belong to the NLS than residues belonging to the NLS. Since the amount of negative examples is so much larger than the positive (Table 1), this leads to higher M_C values and lower sensitivity. Here, a high sensitivity is wanted, so an NLS multiplication factor of 30 was chosen.

The network architecture thus had 140 input nodes (sliding window size of 7), 3 nodes in one hidden layer and 1 output node. The amount of NLS repetition in the training phase was 30. The learning rate for all networks was 0.01 that had the best performance and fastest convergence. The stop-criterion to avoid overtraining was early stopping (epoch 4) which in most cases had the highest M_C value of the different epochs (data not shown).

4.2 Phase 2

The prediction is now elevated from predicting the residues belonging to an NLS to predicting nuclear proteins. The previously trained network's weights were used.

4.2.1 Dataset

The data used for this part were mainly from SwissProt 40, but the thylakoidal proteins were from J-B Peltier and the peroxisomal set from O Emanuelsson[ii] (Table 4).

subcellular location	*before redundancy reduction*	*after redundancy reduction*
nuclear	1584	330
cytoplasmic	1190	365
mitochondrial	1275	266
thylakoidal	259	41
peroxisomal	152	33
signal peptide	1658	579

Table 4. Data set used for testing the performance of the neural network.

The amount of nuclear, cytoplasmic, mitochondrial and secretory proteins in SwissProt were simply too many for redundancy reduction (since the calculation time increases

[ii] personal communication

exponentially), so a simple script that picked proteins in a random way was applied until the set was computationally manageable.

4.2.2 Testing

These data were fed into the previously trained 5 networks and the average value was calculated from their outputs (Figure 7). The mean output was then "smoothed" out using another sliding window that took the mean value of the neighbouring output values (Figure 12). This results in a new output plot (Figure 13).

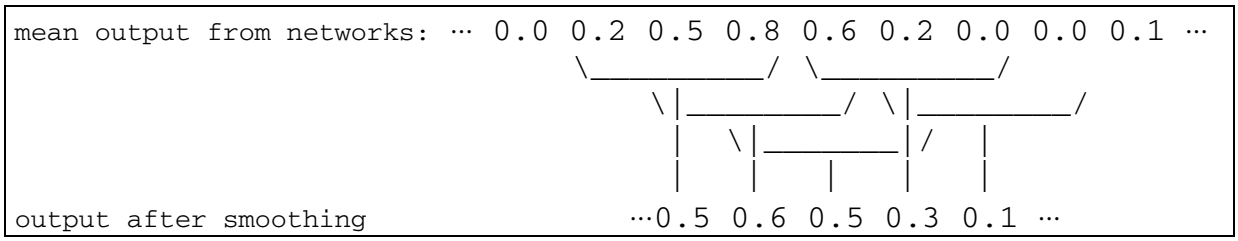

Figure 12. Example of how the sliding window that takes the mean value results in a new "smoothed" output.

A sequence was defined as nuclear if at least three output values in a row were above the cutoff value.

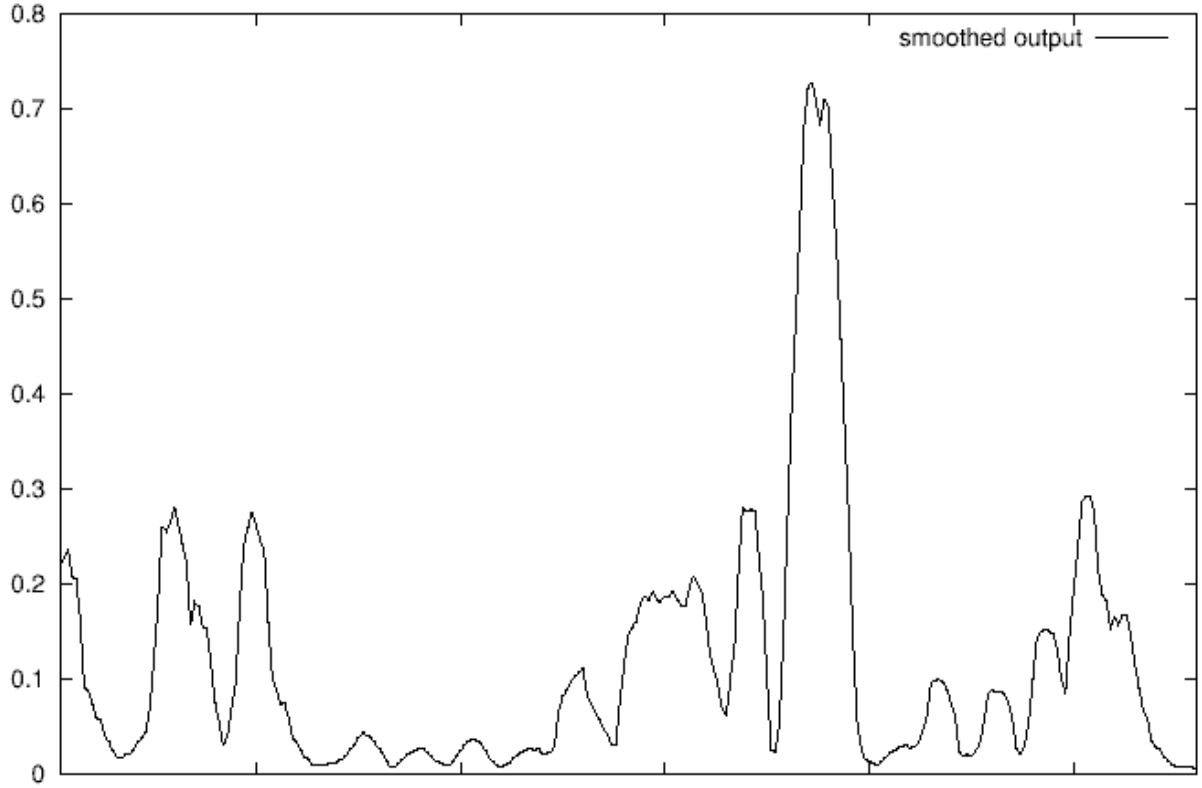

Figure 13. P41900 after being "smoothed" by a sliding window.

4.2.3 Cutoff

One important thing that can alter the overall performance in a dramatic way is to choose the optimal cutoff value. Sequences with an output value greater than the cutoff value are classified as nuclear, otherwise as non-nuclear (Figure 8). This results in a number of false positives (proteins classified as nuclear, but of non-nuclear origin) and false negatives (nuclear proteins classified as non-nuclear). The choice of threshold depends on the purpose

of the prediction. For instance, when classifying tumours as malignant or not, it is very important to reduce the false negatives rather than the false positives. In this study we use the M_C value as a good indicator where to choose the cutoff (Figure 14).

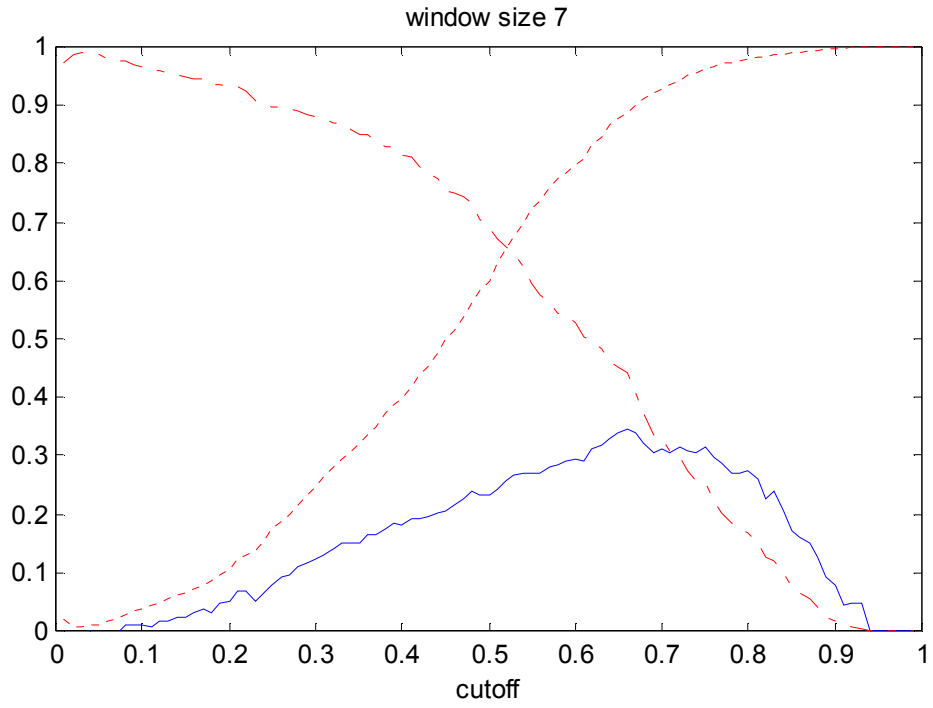

Figure 14. Graph showing the networks performance depending in the choice of cutoff. Dotted red line is sensitivity, semi-dotted red line how big fraction of the negative set that is correct and the blue full line the M_C value.

A value of 0.66 was chosen as cutoff. The resulting sensitivity is 0.45 and the specificity 0.49. The M_C value is discussed below. The highest fraction of false positives was mitochondrial proteins (Table 5). The mitochondrion also contains DNA, so it may be because of DNA binding motifs overlapping with NLSs. The mean amount of false positives is 12%.

location	*category*	*found*	*total*	*fraction*
Nuclear	P_t	149	330	0.45 (=sensitivity)
Cytoplasmic	P_f	46	365	0.13
Mitochondrial	P_f	47	266	0.18
Signal Peptide	P_f	58	579	0.10
Peroxisomal	P_f	3	33	0.09
Thylakoidal	P_f	2	39	0.05

Table 5. Performance of redundancy reduced test set.

The Mathews correlation coefficient for the test set proteins is 0.34, which is better than random, but still not a very good value. In order to improve the results, TargetP was used to initially screen the proteins to find mitochondrial, chloroplastic and signal peptide containing proteins. A previous study had an overall accuracy of 85-90%.[2] Only proteins that TargetP classified as "other" were now considered for potential nuclear localization. This screening reduced the number of false positives more than it reduced the number of true positives (Table 6).

location	*category*	*found*	*total*	*fraction*
Nuclear	P_t	142	330	0.43 (=sensitivity)
Cytoplasmic	P_f	46	365	0.13
Mitochondrial	P_f	7	266	0.03
Signal Peptide	P_f	5	579	0.01
Peroxisomal	P_f	3	33	0.09
Thylakoidal	P_f	2	39	0.05

Table 6. Performance of redundancy reduced set after discrimination of TargetP.

The new M_C is now 0.46, a fairly good increase of performance. The sensitivity decreases to 0.43 but the specificity rises to 0.69 (Table 7).

	M_C	*Sensitivity*	*Specificity*	*mean fraction of P_f*
NN	0.34	0.45	0.49	0.12
NN+TargetP	0.46	0.43	0.69	0.05

Table 7. Resulting performance of Neural Network (NN) and Neural Network after discrimination by TargetP (NN+TargetP).

Out of the 543 proteins with an NLS annotation in SwissProt (from Table 1), 66.3% of the proteins were found. There exists a list of proteins with experimentally verified NLSs (see 4.3.3 PredictNLS). Of the 72 proteins available, 42 were found (58.3%).

4.2.4 Small proteins are preferred

The neural network shows a preference for small proteins (Figure 15, middle), compared with the size distribution of the proteins prior to testing (Figure 15, Left). Only 12 out of 98 of the proteins that are larger than 60 kDa (>~9nm) are found. These are assumed to be transported actively, and thus obliged to contain an NLS (2.1.5 Nuclear Localization Signals). Large proteins may have longer and more intricate signals than the ones this network was trained on. The cytoplasmic proteins classified as nuclear showed a similar distribution of their sizes as the nuclear proteins found (data not shown). Of the proteins used for training the networks, almost 50% were larger than 60 kDa (Figure 15, right). This is surprising as the network mainly finds the small proteins. This might be a result of the sliding window or that large proteins may have longer and specific NLSs and smaller nuclear proteins short and general NLSs. The specific NLSs are then lost as noise during training.

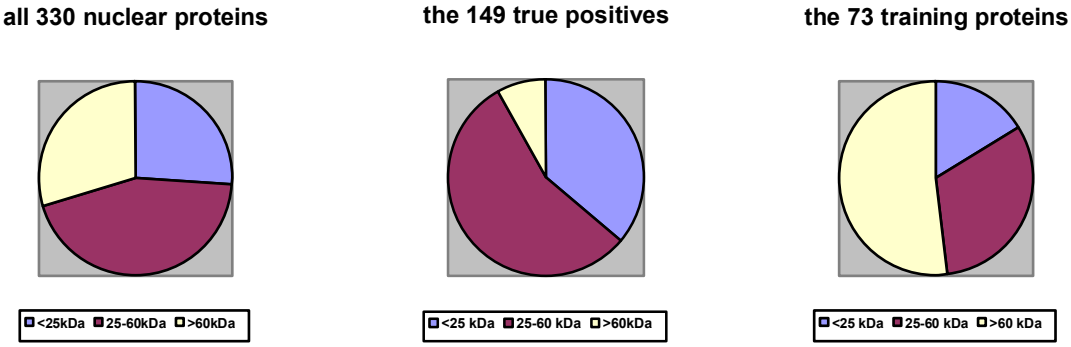

Figure 15. Distribution of sizes of nuclear proteins. Left: the 330 non-redundant proteins used for evaluation. Middle: The 149 proteins that were classified as nuclear by the network, the true positives. Right: The 73 proteins used for training the network.

4.3 Benchmarking

Some of the following methods are for predicting the sorting for various numbers of organelles. The discussion only concerns the nuclear predictor of the methods, but that is generally the worst performing part, especially when it comes to differentiating the cytoplasmic proteins from the nuclear. For the testing of the performance of the following methods, 330 redundancy reduced nuclear proteins were used (Table 4). Some of the methods ignored sequences if they were shorter than some threshold and/or if the sequence had X as an amino acid (which means that that part of the sequence is unknown or absent).

4.3.1 PSORT I and II

PSORT is a rule-based program developed by Kenta Nakai.[24] It detects sorting signals in proteins and predicts the subcellular location. It uses various approaches for the different compartments, e.g. motifs, consensus sequences, hydrophobicity. PSORT I used a dataset of 401 sequences from 17 subcellular locations. Of these, 43 sequences were used for training the nuclear predictor and 19 for the testing. For the NLSs it uses a score that combines different empirical rules on its own data set. It is able to sort 63.2% (12 of a total of 19) nuclear proteins correctly. It misclassifies 16 % of other proteins to be nuclear.[25] PSORT I showed to be too difficult to retrain when the number of additional sequences with known localization sites was increasing, too many manual adjustments of the numerical parameters were required.[26] To overcome this difficulty, PSORT II was developed. It uses the *k*-nearest-neighbour method[iii] together with a set of sequence-derived features such as regions with high hydrophobicity,[27, 28] and it can easily be retrained with different data sets. It uses the yeast genome as the underlying data for predicting the nuclear localization. The dataset contains 1462 sequences divided 10 classes (compartments). It predicts 354 of the proteins to be nuclear (24%), with 216 of 426 to be true positives (50.7%). The largest source of misclassification is the cytoplasmic proteins, with 91 out of 444 proteins (20.4%) predicted to be nuclear.

PSORT I was able to predict 41% of the 330 nuclear proteins to be nuclear and PSORT II 74% of the non-plant nuclear proteins. We used a version of PSORT II that was trained on the yeast genome. The high accuracy of PSORT II leads to the suspicion that the set of nuclear proteins mainly were from yeast. Further investigation revealed that it was 67 out of 330 sequences that originated from yeast so this was not the case. How many cytoplasmic proteins that are classified as nuclear has not been examined.

4.3.2 nnPSL

nnPSL uses neural nets that only look at the amino acid composition.[29] 3420 sequences were divided into 11 localization groups, but since there were too little data for some of the groups this was reduced to 6 classes and 3315 sequences. 1097 of these were nuclear sequences. It uses several neural networks that each discriminate between 2 different classes (compartments). The overall prediction accuracy reached 66.1 % but no results from their set of nuclear proteins were reported.[30]

nnPSL predicted 53% of the 330 nuclear proteins to be nuclear.

4.3.3 PredictNLS

PredictNLS uses something called “in silico mutagenesis” with experimentally verified NLS’s.[9, 31] The verified NLS have some residues changed or removed while monitoring if it is

[iii] The *k*-nearest-neighbour classifyies a pattern by looking at what class the *k* nearest neighbours to the pattern have and choose the class that occurs most frequently of the *k* patterns.

only known nuclear proteins and no non-nuclear that contain this new signal. If this is the case and the new signal is present in at least two distinct protein families, the “in silico mutated” signal is added to the database. The experimentally verified NLSs cover just 10 % of the known nuclear proteins, but with the mutagenesis 43 % of the nuclear proteins were covered. A drawback with this method is that the new motifs may be for DNA-binding motifs instead.

The 330 known nuclear proteins have not been submitted to predictNLS web-service, but the database of experimentally verified NLS proteins were downloaded and used for evaluating our neural network (4.2.3 Cutoff).

5 Discussion

The aim of this study was to predict nuclear localization signals which was later revised to predicting nuclear localization. There already exist nuclear predictors, but none that use neural networks together with the information provided in the amino acid sequence alone. The results show great difficulties in detecting the actual NLS, but a somewhat greater ability to predict nuclear localization. A final Mathews correlation coefficient of 0.46, a sensitivity of 0.43 and a specificity of 0.69 are not overwhelmingly impressing, but the method has, at least, few false positives. There are several explanations to the poor results. The method with sliding windows was not the best one, the results probably could have been better if the negative set came from non-nuclear proteins. Or if the signals were cut out with a certain length, but a problem with that approach is that NLSs have different lengths. But the main reason for the poor results is probably in the data set. The underlying data used for training were of poor quality, and too little is known about the nature of NLSs.

The network does not actually do what it is trained for in phase 1. This can be seen when studying the behaviour of the network. The performance is better on the 543 redundant proteins that had an NLS annotation in SwissProt, but 73 of these proteins were used as the training set so the result is biased. However, on the 72 redundant proteins with experimentally verified NLSs, not used for training, the prediction accuracy is better than for the non-redundant nuclear proteins (58,3% vs. 43%). So the results show that the network has a preference for nuclear proteins with an NLS that is annotated than nuclear proteins that do not. No further investigation has been made about the nature of the nuclear proteins found, except that they are small.

Nuclear proteins are often more basic in their amino acid sequence than other proteins. Most of the NLS patterns that were used for training contained basic residues, so this may be the reason why this network works. It finds clusters of basic residues, and these are more common in nuclear proteins.

Other approaches also have been investigated, such as incorporating the amino acid content into training. The fraction of each amino acid was used as input for a new neural network, but yielded poor results. The nuclear proteins were used as the positive set and a small set of the previously mentioned 5 other subcellular localization categories as a negative set. This network was not able to separate the training data into nuclear/non-nuclear. This may sound surprising as it is the way that nnPSL predicts localization, but it uses many separate networks that discriminates between only two categories at a time. It would be both to time-consuming and out of scope of this project to try a similar approach.

A different representation of the input data was also tested. Then, R and K residues belonged to one vector component, H to another and the rest of the amino acids as the third final vector component. This gave a fairly good M_C since most of the NLSs in SwissProt are classical mono- or bipartite. But too much underlying information was considered to be thrown away, so the approach was not investigated further.

Problems:

This project may have come too early. Many of the poor results in phase 1 can be explained by the nature of the nuclear localization signals. They are not as well defined as other translocation signals such as transit peptides. When the proteins go through the nuclear envelope nothing is cleaved off. The NLS can be anywhere in the amino acid sequence and there can be several of them in one protein. This suggests that it is the 3D structure that is important. Since the network trained on NLS sequences shorter or equal to 8 residues preferred a small window size, 7, the amount of global information taken into account is low. It just examines the local arrangement of the input. To little is known about the folding and

3D structure of nuclear transport to be able to train the network with global 3D information too, but this is something that should be investigated in the future.

Problems with nuclear localization
Not all nuclear proteins contain an NLS. Some are smaller than 9 nm and can diffuse through the nuclear envelope. There are many proteins whose localization sites are not confined to a single space,[25] for instance ribosomal proteins. They are sorted to the nucleus and after assembly transported back to the cytoplasm. Another difficulty with predicting nuclear localization is that some proteins without a signal seem to be transported into the nucleus as part of a protein complex if another subunit of the complex contains a nuclear localization signal.[32]

Most of the proteins used for training were "POTENTIAL", "PROBABLE" or "BY SIMILARITY". This leads to a bias of the training set, since it is unknown how the depositors have come to that conclusion. If they made an alignment with another protein that have an experimentally verified NLS and then transferring over the NLS region, perhaps another redundancy reduction should have been used over the NLS region only, instead of the entire protein. This leads to new problems as the Hobohm algorithm uses alignment score between two amino acid sequences to do its reduction of the data. The score depends on alignment length and since the threshold for sequence similarity is quite low, ~30% similarity, two NLSs that are 6 amino acids long only need to have 2 residues that match to be considered as neighbours. And two identical 11-residue peptides can have different structures.[33] Since most NLSs are basic and short this would lead to a too small and non-representative training set.

Conclusions:
Detecting nuclear localization is perhaps the most difficult task when predicting subcellular location for reasons already mentioned. Predicting nuclear localization is possible by using neural networks for detecting their nuclear localization signals. As experimental data and knowledge about nuclear transport constantly increases, this new knowledge can be used to get better performance. This new predictor shows worse performance than available predictors, but have a quite high specificity. It would be interesting to use all of them for consensus decisions about nuclear localization.

5.1 Future Work

5.1.1 Reduction of cytoplasmic proteins

The largest source of false positive proteins resides from proteins that are cytoplasmic. This is maybe because some proteins shuttle back and forth between the cytosol and nucleus. Proteins that are smaller than 9 nm are also in equilibrium between the two compartments. To decrease the number of false positives either a new neural network that discriminates between the cytoplasm and the nucleus should be trained on the amino acid content alone should be trained or just a simple matrix that performs the discrimination should be added.[34] This is an approach that is known to work.[16, 26] This is the method that PSORT and nnPSL uses.

5.1.2 Trying different network/HMM

Try with a network that gives very few false positives and less true positives in phase 1. Another very interesting thing would be to use a different method, for instance Hidden Markov Models. These are also very common for many predictors available via the WWW and are suitable for biological problems.

5.1.3 Further investigation

Investigate what kind of patterns that yield high outputs and also see what kind of proteins that are found/not found. Do they belong to different classes? What do they have in common?

5.2 Acknowledgements

I wish to thank Olof Emanuelsson for his invaluable assistance and keen support of this project. Arne Elofsson for his hints during the Monday meetings, Scott Melnyc and Greg Downs for setting up the computers so nicely. Gunnar von Heijne for support and project description and finally everyone at SBC for long coffee breaks. The room I was in rocks, but not as much as it used to when Isabelle Westerlund chose the music. Peace.

[1] TargetP: Prediction of subcellular location. (15 Jan 2002) http://www.cbs.dtu.dk/services/TargetP/
[2] Emanuelsson O, Nielsen H, Brunak & von Heijne G. (2000) Predicting subcellular localization of proteins based on their N-terminal amino acid sequence. *J. Mol. Biol.*, 300, 1005-1016.
[3] Wu CH. (1997) Artificial neural networks for molecular sequence analysis. *Computers Chem.,* Vol. 21. No. 4, 237-256.
[4] Dreyfuss G & Struhl K. (1999) Nucleus and gene expression, Multiprotein complexes, mechanistic connections and nuclear organization. *Current Opinion in cell biology*, 11:303-306.
[5] Rusch SL & Kendall DA. (1995) Protein transport via amino-terminal targeting sequences: common themes in diverse systems (Review). *Molecular Membrane Biology*, 12, 295-307.
[6] Schatz G & Dobberstein B. (1996) Common principles of protein translocation across membranes. *Science*, 271, 1519-1525.
[7] Stryer L. (1996) Biochemistry. W. H. Freeman and Company.
[8] Lewin B. (2000) Genes VII. Oxford university press.
[9] Cokol M, Nair R & Rost B. (2000) Finding nuclear localization signals, Cokol et al, *EMBO reports*, 1, 411-415.
[10]Fontes MRM, Teh T & Kobe B. (2000) Structural basis of recognition of monopartite and bipartite nuclear localization sequences by mammalian importin-α. *J. Mol. Biol.* 297, 1183-1194.
[11] Kalderon D, Roberts BL, Richardsson WD & Smith AE. (1984) A short amino acid sequence able to specify nuclear location. *Cell*, 39, 499-509
[12] Robbins J, Dilworth SM, Laskey RA & Dingwall C. (1991) Two interdependent basic domains in nucleoplasmin nuclear targeting sequence: Identification of a class of bipartite nuclear targeting sequence. *Cell,* 64, 615-623
[13] Boulikas T. (1993) Nuclear localization signals (NLS). *Crit. Rev. Eukaryot. Gene Expr.* 3, 193-227.
[14] McCulloch W S & Pitts W. (1945) A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, 5, 115-133.
[15] Rosenblatt F. (1958). The Perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65, 386-408.
[16] Bishop CM. (1995) Neural networks for pattern recognition. Oxford University Press.
[17] Kennedy RL, Lee Y, van Roy B, Reed CD & Lippman RP.(1998) Solving data mining problems through pattern recognition. Prentice Hall.
[18] Hobohm U, Scharf M, Schneider R & Sander C. (1992) *Protein Science*, 1, 409-417.
[19] Billnet: The fast and free neural network simulator. (20 Aug 2001) http://www.iit.demokritos.gr/~vasvir/billnet/
[20] Virvilis V. (20 Aug 2001) *BILLNET user's guide* http://www.iit.demokritos.gr/~vasvir/billnet/doc/userg.ps.gz
[21] Mathews BW. (1975), Comparison of predicted and observed secondary structure of T4 phage lysosyme. *Biochem Biophys Acta*, 405, 442-451.
[22] Swissprot: Protein Knowledgebase. (15 Oct 2001) http://www.expasy.ch/sprot/sprot-top.html
[23] Bairoch A & Apweiler R. (2000) The SWISS-PROT protein sequence database and its supplement TrEMBL in 2000. *Nucleic Acids Res.* 28, 45-48.
[24] PSORT:Prediction of protein sorting signals and localization sites in amino acid sequences. (20 Mar 2002) http://psort.nibb.ac.jp/
[25] Nakai K & Kanehisa M. (1992) Prediction of protein localization sites in cells. *Genomics*, 14, 897-911.
[26] Nakai K & Horton P. (1999) PSORT: a program for detecting sorting signals in proteins and predicting their subcellular localization. TIBS 24, 34-35.
[27] Horton P & Nakai K. (1997) Better prediction of protein cellular localization sites with the *k* nearest neighbours classifier. ISMB-97, 147-152.
[28] Cover T & Hart P. (1967) Nearest neighbour pattern classification. *IEEE Transactions on information theory,* 13, 21-27.
[29] nnPSL: Prediction of the subcellular location of proteins by neural networks. (20 Mar 2002) http://www.doe-mbi.ucla.edu/cgi/astrid/nnpsl_mult.cgi
[30] Reinhardt A & Hubbard T. (1998) Using neural networks for prediction of the subcellular location of proteins. *Nucleic Acids Research,* Vol. 26, No. 9, 2230-2236.
[31] PredictNLS: Prediction and analysis of nuclear localization signals. (20 Mar 2002) http://maple.bioc.columbia.edu/predictNLS/
[32] Zhao L & Padmanabhan R. (1988) Nuclear transport of adenovirus dna polymerase is facilitated by interaction with preterminal protein. *Cell*, 55, 1005-1015.
[33] Minor DLJ & Kim PS. Context-dependent secondary structure formation of a designed protein sequence. *Nature,* 380, 730-734.

[34] Kenta Nakai, personal communication