In compliance with the
Canadian Privacy Legislation
some supporting forms
may have been removed from
this dissertation.

While these forms may be included
in the document page count,
their removal does not represent
any loss of content from the dissertation.

An artificial neural network based prediction system for the classification of Golgi resident proteins

Yannick Daoudi

McGill University, Montreal
February 2003

A thesis submitted to the Faculty of Graduate Studies and Research in partial fulfillment of the requirements for a Master of Science degree.

Your file Votre référence
ISBN: 0-612-88180-6
Our file Notre référence
ISBN: 0-612-88180-6

To my loving wife and parents

Table of contents

Abstract

The Golgi apparatus of the cell is responsible for crucial mechanisms such as the intra-cellular transport of molecules, and the glycosylation process. It is linked to several serious and deadly diseases, including cancer. Understanding which proteins make up this organelle, and their associated functions, is essential to provide scientists with more insight into these related processes. The localization of Golgi resident proteins is a complex and not yet fully understood process. It has kept biologists using slow, manual, "wet lab" methods. Artificial neural networks have the capability of learning complex relationships from training data, and generalizing beyond these examples. Here, we present a neural network prediction system based on evolutionary information, which we extracted from the amino acid sequences in our data set. Our predictor proved to correctly classify, with a 90% success rate, whether a candidate sequence was coding for a protein located in the Golgi apparatus.

L'appareil de Golgi est une organelle cellulaire impliquée dans la glycolysation et le transport intracellulaire des protéines. Il est lié à plusieurs maladies mortelles, y compris le cancer. Il est essentiel de déterminer quelles protéines composent cette organelle, ainsi que leurs fonctions associées, pour fournir aux scientifiques une compréhension accrue du rôle cellulaire du Golgi. La localisation des protéines résidentes du Golgi est un processus complexe qui n'est pas encore entièrement compris. Ce dernier contraint Les biologistes à utiliser des techniques de laboratoire lentes et fastidieuses. Les réseaux de neurones artificiels ont la capacité d'apprendre des rapports complexes à partir d'un ensemble de données d'apprentissage, et de généraliser au-delà de ces exemples. Ici, nous présentons un système de prédiction qui utilise un réseau de neurones basé sur de l'information évolutive que nous avons extraite à partir de séquences d'acides aminés de notre ensemble de données. Notre système s'est avéré très efficace pour prédire si une séquence d'acides aminés codait pour une protéine résidente de l'appareil de Golgi, avec un taux de classification de 90%.

The author would like to thank his supervisors, Michael T. Hallett and Doina Precup, whose knowledge, support, and guidance have made this work possible.

1. Introduction

1.1 Proteomics

With the human genome recently completed, we now have access to the full DNA sequence of the human species, among many others. These DNA sequences encode genes, which in turn, translate into peptides that are assembled into proteins. Within each cell in our body, proteins are the molecules responsible for cellular function. Since there exist over 30,000 genes coding for approximately half a million proteins, the next logical task is to understand the complexity and mechanisms of biological systems. Proteomics is part of the post-genomic era in computational biology; it studies protein expression, protein functions, interactions, pathways, and post-translational modifications.

1.2 The problem

Bioinformatics was in part driven by the vast amount of DNA data in the late 90's, and the need for good, automated analysis tools that would allow processing it. There are three types of analysis: functional, comparative, and structural. Here, we deal with functional analysis; we predict the final location of a protein within a cell, given its amino acid sequence. To be more exact, the problem consists of constructing an algorithm that correctly classifies proteins located in the Golgi apparatus of the cell, given their amino acid sequence. The only existing methods used for this task are slow, manual, and require specialized personnel.

Many diseases are related to the glycosylation procedure that takes place in the Golgi apparatus. A relatively small number of the Golgi proteins have been classified to present. The ability to classify a wider number of proteins among the huge set of unknown proteins in the databases would provide researchers with more insight into diseases such as cancer, autoimmune and inflammatory bowel diseases. Additionally, classifying a new protein with a well-characterized function, that was never before assigned to the Golgi apparatus, might lead to discovering a new function of this intra-cellular compartment. Conversely, classifying a new protein with an unknown function as belonging to the Golgi, would give a hint as to the function of that protein.

1.3 Machine learning

Machine learning addresses the problem of building computer programs that improve their performance on a given task through experience. For a given analytical system there are some patterns that have known desired responses; these data types form pairs, which we refer to as inputs and targets. The goal of supervised learning is to find a mapping that correctly associates the inputs with the targets. A wide range of adaptive algorithms have been developed in this field including decision trees, artificial neural networks, and k-nearest neighbour. Many of these result from the recent interest in "data mining" which is the application of machine learning algorithms to analyze large amounts of corporate and scientific data.

A very popular approach is to use Artificial Neural Networks (ANNs). ANNs are inspired by the organization of the brain. They are robust to noise and very useful for learning real-world sensor data such as interpreting visual scenes, speech recognition, and learning robot control strategies [1].

1.4 Our approach

There are no effective automated systems in place for predicting whether a given protein is Golgi resident, based on its amino acid sequence.

The localization system of Golgi resident proteins is highly complex, and far from completely understood. There are no specific known patterns, and although we have some example sequences obtained from "wet lab" methods, a huge number of sequences are yet to be analyzed.

The sequence of a protein is not only responsible for its localization but also for its structure and function. Therefore we need to design an adaptable prediction system that is robust to noise, and can learn to extract location information from example sequences, if we are to generalize beyond those examples. Artificial neural networks have been successfully trained to perform fairly accurately on several related problems such as secondary protein structure prediction [15,16], protein-coding region detection [19], and sorting signal prediction [17,18]. In all these cases, the prediction was based on amino acid or DNA sequences.

In this thesis, we present a similar approach for the problem of classifying Golgi resident proteins in a cell. We use the amino acid sequences of the proteins in our data set as the initial input. We preprocess these sequences using the DARWIN [8] bioinformatics tool in order to extract useful features. Finally, we use these computed features to train our ANN system, which, coupled with other machine learning techniques, learns to output a prediction for whether a given candidate sequence is coding for a Golgi resident protein or not.

Golgi Prediction Problem

input: a set $\mathbf{S} = \{E_1,...,E_n\}$ where E_i is the amino acid sequence of protein i
a mapping f: $\mathbf{S} \rightarrow \{+,-\}$ where + represents a Golgi resident protein, and – represents a non-Golgi resident protein.

output: a predictor X: P $\rightarrow$ {+,–} where P is an amino acid sequence for a candidate protein and + (respectively, –) represents the residence (respectively, non-residence) of P in the Golgi.

In the first part of this thesis, we present the biological information and background necessary to understand the localization mechanisms present in the cell, and properties of Golgi resident proteins. Next, we introduce the Artificial Neural Network, the backpropagation learning algorithm, and various design and data issues related to our problem. We then review and analyze related approaches and problems in bioinformatics. In Chapters 5 and 6, we present our data set and the preliminary data analysis, which enables us to determine what types of features to use with our predictor. Finally, in Chapters 7 and 8, we describe our experimental setup and results, analyzing different ANN architectures and settings. We conclude by summarizing the problems encountered, our most successful approach, and future work considerations.

2. Protein sorting and the Golgi apparatus

2.1 Protein Sorting

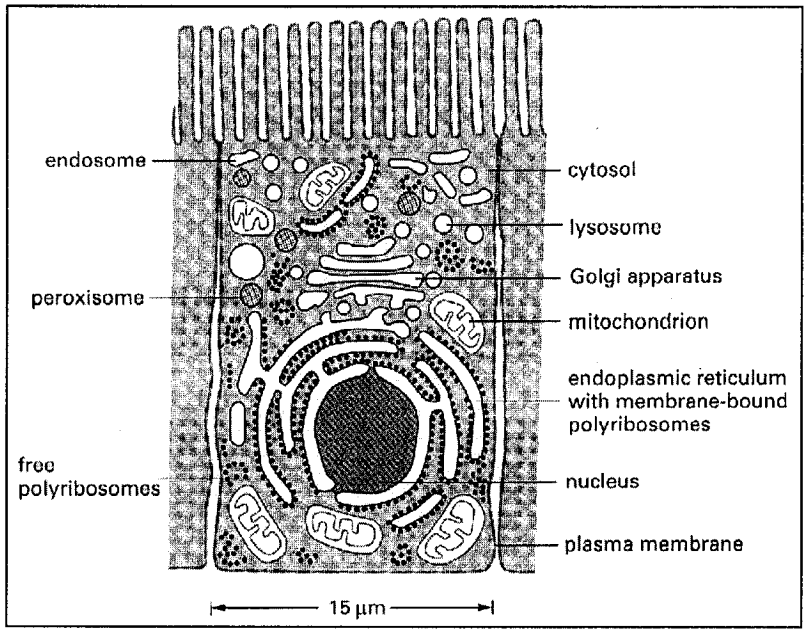

Figure 2.1: The major intracellular compartments of an animal cell [4,p.660]

A eucaryotic cell is subdivided into different *organelles*, which are functionally distinct, intracellular, membrane bounded compartments. Each of these compartments holds a specific set of molecules and enzymes. It is not only necessary to understand the complex processes behind the transport of molecules from one organelle to the other, but also how the organelles are created and maintained, as well as their metabolic function [4]. Proteins play an important role in keeping the eucaryotic cell subdivided into distinct compartments. They catalyze the reactions that occur in each organelle and transport specific molecules to and from their *lumen* (their interior). Most proteins are first synthesized in the cytosol, and then delivered to their corresponding compartment. This delivery mechanism depends on the amino acid sequence of the transported protein, which is the sequence of basic chemical structural units that encodes a protein. This sequence can contain sorting signals [17]. A *sorting signal* is responsible for directing proteins to different organelles. There are three main ways by which proteins move between organelles:

- *Gated transport* relies on the nuclear pores to selectively transport macromolecules from the cytosol to the nucleus.
- *Transmembrane transport* relies on membrane-bound translocators to transport a usually unfolded protein through the membrane.
- *Vesicular transport* relies on transport vesicles to carry folded proteins from one organelle to another. These proteins do not have to cross a membrane.

If a protein is going to be transported through a membrane or via a vesicle, then it must have a sorting signal that is recognized by a complementary receptor protein. There are at least two types of sorting signals:

- *Signal sequences* consist of 15 to 60 contiguous amino acid residues (a substring of the original amino acid sequence), usually located at the N-terminus (referred to as the beginning of the amino acid sequence).
- *Signal patches* correspond to specific spatial arrangements when the protein folds, such that amino acid residues that may be far apart in the linear sequence come in contact when the protein folds to its native three-dimensional state. In such a case, the amino acid residues that form the signal patch represent a subsequence of the original amino acid sequence.

The types of sorting signals vary from exact sequences of amino acids, to patterns of hydrophobicity and charge [4]. Therefore, in some cases, physical properties of amino acids (such as hydrophobicity or charge) appear to be more important in the signal recognition process than the exact amino acid sequence.

2.2 Golgi resident proteins

The Golgi apparatus is the organelle responsible for receiving lipids and proteins from the Endoplasmic Reticulum (ER), and dispatching them to various destinations. It represents the hub of the secretory pathway. It is organized as stacked cisternae (compartments), in which vesicular transport carries molecules from one compartment to the next. On the way through these cisternae, Golgi resident proteins carry out functions on the transported molecules such as glycosylation and proteolytic processing, membrane transport, recycling to the ER, as well as the organizational maintenance of the organelle itself. Despite the intense flux of proteins in the Golgi and the complex regulatory

functions associated with it, its structural integrity (the set of molecules and enzymes that comprise the Golgi) is conserved. This suggests that either specific sorting signals are present for the retention of Golgi resident proteins [20], or that Golgi resident proteins are missing sorting signals found in other exported proteins. Additionally, experimental studies have shown that, although the cisternae are physically distinct entities, there is continuity in the mixture of enzymes present, rather than each compartment having a different set. The Golgi resident proteins can be divided into five classes: type I and type II membrane proteins, multimembrane-spanning proteins, peripheral membrane proteins, and soluble lumenal proteins. Furthermore, based on biochemical and functional properties, these proteins can be divided into at least seven groups: glycolysation machinery, Viral glycoproteins, recycling TGN membrane proteins, retrieval receptors, matrix and cytoskeletal-binding proteins, membrane transport, and others. It has been become very clear over the past decade that in order to have such diverse types of proteins within the Golgi, multiple localization mechanisms must operate. Experiments have shown the following examples of these mechanisms [20]:

- Sorting signals located in different parts of the sequence, such as the transmembrane domains of mammalian Golgi glycosyltransferases; the stem region in a medial-Golgi enzyme (GlcNAc-TI) independent of the transmembrane domain; or the cytoplasmic tail of the membrane proteins in the trans-Golgi network.
- The aggregation of molecules inside the correct Golgi compartment that would prevent those molecules from exiting through vesicles.
- Retrograde transport (or intra-Golgi retrieval) which recycles proteins back to previous cisternae. This would also indicate that Golgi retention does not depend on protein immobilization as Golgi resident proteins may move around within the cisternae of the Golgi apparatus.
- In a multiple membrane-spanning protein (IBV M protein), only the first of the three membrane spanning domains is required to retain it in the Golgi, whereas in the MHV M protein, two signals are necessary, one in a transmembrane domain, and the other in the cytoplasmic tail.

- Peripheral membrane proteins interacting with motifs on the cytoplasmic domains of membrane proteins.

These results indicate that two sets of membrane proteins within the same Golgi compartment can have very different mechanisms of localization. Thus, the localization of Golgi proteins is very complex, and far from being fully understood.

Furthermore, the integrity of the Golgi is currently being debated. It is not clear whether the Golgi is an independent organelle rebuilt from a template that divides during cell division, or whether it is a dynamic self-organizing aggregation of proteins and lipid membrane that assembles and disassembles constantly [22]. Thus, the Golgi apparatus is still a very controversial organelle. It plays a crucial role in the intra-cellular environment, and is linked to various serious or deadly diseases. Therefore, determining which proteins comprise this organelle is fundamental not only to understanding more fully the processes associated with it, but also to give an insight into the functions of those proteins, if they are unknown. It has been estimated that there could be up to 1000 proteins resident in the Golgi. At the time of writing, less than one tenth of these have been identified.

3. The artificial neural network

Artificial neural networks (ANNs) are a well-known means for uncovering complex, non-linear relationships in multivariate data, although they are still able to map linearities. Examples of domains where neural nets are used include natural language processing, character recognition, image compression [3], biotechnology [3,7], robotics and interactive "intelligent" systems [4]. Feed-forward neural networks consist of very simple interconnected computational units (neurons) that can take numerical inputs and, via weighted summation and a transfer function, transform these values into an output. ANNs have the ability to learn (or approximate) target functions from a given data set (consisting of pairs of inputs and desired targets), by modifying the connection weights, until the output nodes match the desired target for the given input, to a certain degree of accuracy. Any function can be approximated to arbitrary accuracy by a network of four layers of units [1]. The first layer, or *input layer*, consists of input nodes directly related to the type of data being input into the network. The output from the ANN is located on the *output layer*, which encodes the type of results expected. All layers in between the input and output layers are called *hidden layers* as their inputs and outputs are only available within the network and not seen by the user.

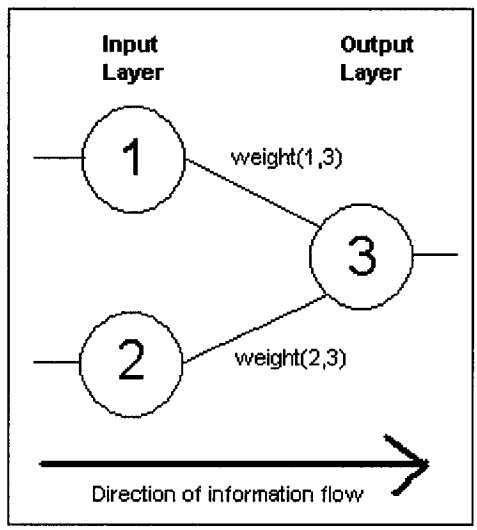

Figure 3.1: A simple feed-forward artificial neural network

Figure 3.1 shows a very simple neural network consisting of only an input layer with two input nodes (1 and 2), connected to a single output node (3). The edges represent the connections between the two layers, and a weight is associated with each edge (weight(1,3) and weight(2,3)). This is called a feed-forward neural network as the information starts at the input nodes and flows towards the output nodes without loops or connections back to a previous layer.

3.1 The data set

One of the most important aspects of ANN is that they can learn from examples. These examples (pairs of inputs and targets) are grouped into a data set that is used to train and test the accuracy of the network.

The size and distribution of the data is important since neural networks have good interpolation but bad extrapolation [9]. Therefore we need sufficient well-distributed examples to fill the space of features (used to represent these examples), and allow generalization. If the training set is small compared to number of weights in the neural network, then there is the risk of memorizing input-output pairs [16] without any generalization. This process is known as *overfitting*, and causes the network to have good accuracy on training data but very poor accuracy on new data.

3.2 The Transfer function

Typically, the output produced by a node in a neural network is computed as a function of its inputs. This function is called the *transfer function*. Most transfer functions include the weighted summation operation:

$$net = \sum_{i=1}^{n} x_i w_{ij} = x_1 w_{1j} + x_2 w_{2j} + \ldots + x_n w_{nj}$$

where w_{ij} represents the weight of the connection between node i and node j, and x_i represents the output of node i.

Using a simple transfer function, we can compare the result of this operation to a threshold θ and output a 1 if the summation is greater than θ, or a 0 otherwise. Most ANN applications, however, use a differentiable approximation of the step function. This is also known as the *squashing* or *sigmoid function*:

$$Output = \frac{1}{1+e^{-net}}$$

where *net* represents the summation defined above. This outputs a floating-point value in the range (0.0,1.0):

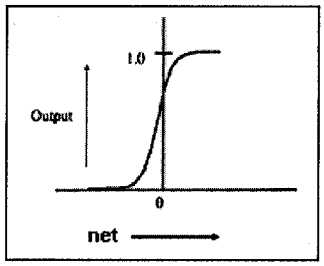

Figure 3.2: The sigmoid transfer function

3.3 Backpropagation learning algorithm

Most ANN applications that use a feed-forward neural network architecture adjust their weights using a gradient descent search through the space of possible connection weights. This is done in order to minimize the error between the network outputs and the target values in the training set. The typical error function on a training example *d* would be:

$$E_d = \frac{1}{2}\sum_{i=1}^{n}(o_i - t_i)^2$$

where n is the number of output units, o_i represents the output value associated with output unit i, and t_i represents the target value associated with output unit i.

This error is propagated backwards through the network, and the weights modified accordingly using the following rule:

$$w_{ij} := w_{ij} + \eta\delta_j x_{ij}$$

where $\eta \in (0.0,1.0)$ is the *learning rate*, x_{ij} is the i^{th} input to unit j, and δ_j is proportional to the error and defined as follows:

- For each output unit k: $\delta_k = o_k(1 - o_k)(t_k - o_k)$
- For each hidden unit j: $\delta_j = o_j(1 - o_j)\,\Sigma\,\delta_k w_{jk}$

The weights are often initialized to small random values, allowing them to be more easily adjusted towards 1 or towards 0 as a result of the training. A training pass through all the vectors of the input data is called an *epoch*. This procedure is repeated for as many epochs as specified, or until a desired level of accuracy is reached (if possible).

3.4 The local minima problem

The gradient descent search used in the backpropagation algorithm can become trapped in one of the many possible local minima on the error surface.

There are several techniques to reduce the risk of falling in a local minimum, including adding *momentum* to the learning rate. In this case, the weight update rule becomes:

$$w_{ij} := w_{ij} + \eta \delta_j x_{ij} + \alpha \Delta_{ij}$$

where Δ_{ij} represents the last weight change applied to w_{ij} and α is called the momentum rate and is in the range (0.0,1.0).

Like the learning rate, the momentum is a user-controlled learning parameter. Lower values may cause longer training times, but generalization accuracy is typically good. Higher values may not converge to an acceptable error, but might prevent getting caught into a local minimum. There is no fixed rule to determine what values to choose initially, and such parameters are typically determined experimentally (see Chapter 8).

Another common approach used to avoid the local minima problem consists of trying true gradient descent instead of stochastic gradient descent [1]. In true gradient descent, the error is summed over all training examples before updating weights, whereas in stochastic gradient descent, weights are updated after each training example.

An alternative solution to the local minima problem is to use ensemble methods, which consist of forming a committee of neural networks (or other classifiers) trained on the same task. The prediction for an input is based on the output of every neural network in the ensemble [1,11]. For instance, we can use a majority vote of a few different classifiers. This should give a better approximation of the true hypothesis than using just one such classifier. Using an initial weight randomization can serve as a diversification factor among the neural networks in an ensemble.

3.5 Internal structure

The number of input and output nodes is determined directly based on the attributes in the data set. This is not true for the number of hidden nodes. The main problem is to learn the general features of the relationship without learning the idiosyncrasies of the training data set. When too many hidden nodes are used, this may lead to overfitting the training data.

When too few hidden nodes are used, this may lead to over-generalization. Often the number of hidden nodes is chosen to be between log(n) and $n^{1/2}$ where n is the dimensionality of the input. This is fixed throughout the training. Alternatively, the network structure can be dynamically modified to find the optimal number of units in the hidden layer. A number of such methods have been developed, two of which we outline below:

- *Pruning connections*: This method increases generalization, reduces required computation, and may help to pinpoint important inputs. If a weight remains close to 0 with little variance, then it is most likely unnecessary and may be removed. One such pruning algorithm is called Optimal Brain Damage [9,10]. It uses connection weight saliencies, and has been shown to be very efficient.
- *Node splitting*: This method increases efficiency in learning by locating undecided (useless) nodes. If the composite input and output variance is greater than the respective composite input and output mean, then the node is split in two, and a new node is created on either side of the original node (half the variance apart) [5,6].

4. Review of related problems and approaches

One of the most important issues in using a neural network to deal with amino acid sequences is to be able to determine which features to use (i.e., extract a fixed number of inputs), and to determine how to encode these. In this chapter, we review some of the work on related problems and the existing approaches to this issue.

4.1 Secondary structure prediction

The first system we discuss, called NnPredict, consists of predicting secondary structure of proteins using a neural network [16]. An initial experiment uses a neural net that takes 13 amino acids as input. This choice is centered on the amino acid to be predicted. It outputs one of three predictions corresponding to the three possible structure configurations of amino acids: ∀ helix, ∃ sheet, or turn. Each amino acid is encoded using a 1-of-n (unary) encoding, representing each amino acid by a different permutation of twenty 0's and a single 1. This results in a total of 13 * 21 = 273 input nodes. The training set consists of 91 proteins, and the test set contains 14 proteins, for a total of 105 proteins, or 20760 amino acids. After 100 epochs, the accuracy is of 65% on the training set and 64% on the test set. The NnPredict system enhances this approach by adding two input units to encode for periodicity of ∀ helices and ∃ sheets (calculated using the hydrophobicity of amino acids). This brings the accuracy up to 68% on the training set, and 65% on the test set. In a second experiment, the authors divide the data set into tertiary structural classes: all-∀, all-∃, ∀/∃, and others, corresponding to the predominant secondary structure of the proteins. They perform a leave-one-out cross-validation (jackknife) procedure on each set separately. This consists of removing one protein at a time from the training set and testing only that protein. This process is repeated for every protein in the training set. The resulting accuracies are of 79% on all-∀ class, 70% on all-∃ class, and 64% on ∀/∃ class. The main problem with this algorithm is that it is short sighted. It only looks at a small local window at a time and does not take into account the global view of the protein.

The next approach, called PHDsec [15], is also aimed at predicting the secondary structure of proteins. Similarly to NNPredict, this algorithm uses a neural network, but

achieves higher accuracy due to the use of evolutionary information, and global information about the protein as additional inputs. This is achieved by not looking only at the sequence of 13 adjacent amino acids, but also at a multiple sequence alignment (MSA) of that subsequence with all sequences in the database. The MSA is an indication of how well a particular subsequence was conserved over time. Other inputs describe the protein length, the distance of the subsequence from each end, and the percentage occurrence, in the whole protein, of the amino acid to predict. The output is identical to NnPredict. The resulting accuracy is of 72% over all 3 configurations, and an average of 74% when the proteins are divided into structural classes.

Both of these approaches make a prediction based on a sliding window that advances along the amino acid sequence, predicting one amino acid at a time. Once the end of the sequence is reached, the results of these predictions are put together in order to predict the whole structure. This type of technique is well suited when trying to predict a certain property for one amino acid based on its neighbouring amino acids. This idea cannot be applied to our problem, since we want to base the prediction on the relationship between potentially distant amino acids, and a sliding window would miss existing relationships between amino acids that do not belong to the same window.

4.2 Exon prediction

The GRAIL II system predicts which regions in a DNA sequence actually code for a protein [19]. These protein-coding regions are called *exons*. GRAIL II (version 1.3) uses different algorithms to extract features by separately recognizing exon-edge signals, using a frame-dependent 6-tuple preference model and a 5^{th} order non-homogeneous Markov chain model. Coding signals are recognized using a simple neural network trained on frequency measures of certain short sequences of DNA bases that are found around these sites. Then, an integrating neural network outputs a prediction based on these measures combined with other general features such as GC composition (this pair of bases is more common in coding regions), predicted length, and properties of the adjacent regions. The network has 13 input nodes (corresponding to the measures previously listed), 2 hidden layers of 7 and 3 hidden nodes respectively, and one output node. It was trained to score the "partial correctness" of each potential exon candidate. The training set of 2000

training examples has three types of exon candidates: true, partially true (sequences that overlap an exon), and false. On the test set, GRAIL II located 93% of all exons, with a false positive rate of 12%. Among the true positives, 62% match the actual exons exactly, and 93% match at least one edge correctly (either the start or the end of the exon is correctly predicted).

Unlike secondary structure prediction, which uses a sliding window of amino acid sequence in the protein, this exon prediction system uses more general sequence pattern information (frequencies) as input. This is a useful encoding for global information, and is independent of the size of the protein. The problem with using such features is that the neural network does not learn to pinpoint the exon within a sequence, but relies on preliminary algorithms to detect a potential protein-coding region, and then outputs a score for that potential candidate. This introduces a second opportunity to make errors.

4.3 Sorting signals and localization prediction

The system called TargetP [18] predicts the sorting signal of a protein based only on its first 100 amino acid residues (N-terminal). The sorting signal usually corresponds to a short sequence of amino acids which code for the localization of the protein within the cell. The system then enters this information into a second level neural network to predict its localization among four different classes: the mitochondrion, the chloroplast, the secretory pathway, and "other". The TargetP predictor has two layers of neural networks. The first layer consists of one network for each type of sorting signal. These networks have a sliding window that scans the first 100 amino acid residues (N-terminal). Each network scores each candidate residue. The first layer networks are trained to recognize whether or not the residue in the middle of the window is part of a sorting signal. The sliding windows contain 55 positions for the chloroplast class, 35 positions for the mitochondrion class, and 31 positions for the secretory pathway class. Each position in the input sequence is encoded using a 1-of-20 encoding. The outputs from these networks are fed into the second layer (or integrating network), which outputs one score per sequence and the probable localization class. The input layer of this network consists of 300 positions corresponding to the scores output by first layer networks: one for each of the three classes and each of the 100 amino acid residues in the query sequence. Its

accuracy averages at approximately 85% prediction rate, and builds on top of a previous prediction system called SignalP [17].

This approach solves, in part, the problem of the sliding window that occurred in systems described in Section 4.1 by using an integrating network. However it is still limited to a fixed number of input amino acids residues (the first 100 N-terminal in this case). Dividing the data set into structural classes, and training the neural network on individual classes has been shown empirically to greatly improve accuracy [16] over one network that tries to approximate each different class of output. This division into subclasses is not suitable for our problem, as the classification (which recognizes proteins within a particular location of the secretory pathway) is much more specific than the four classes that they predict. We want to predict the localization of a protein in the Golgi without differentiating between different types of Golgi resident proteins, or the different compartments of the Golgi.

5. The Data Set

The initial data set consists of accession numbers (into the SwissProt and GenBank protein databases) of 201 proteins: 81 positive examples corresponding to proteins located in the Golgi apparatus, and 120 negative examples corresponding to 3 proteins located in the cytosol, 72 proteins in the endoplasmic reticulum, 1 protein in the endosome, 18 proteins in the microsome, 16 proteins in the mitochondria, and 2 proteins in the peroxisome. We retrieve the sequences from the corresponding online databases. In some cases, we notice that two or more different accession numbers sometimes refer to the same amino acid sequence. We compare them for redundancy by aligning every sequence with every other sequence in the data set (see Chapter 6 on sequence alignments). The final data set consists of 155 proteins (73444 amino acid residues): 58 positive examples; and 97 negative examples where 3 are located in the cytosol, 64 in the endoplasmic reticulum, 1 in the endosome, 11 in the microsome, 16 in the mitochondria, and 2 in the peroxisome (see Table 5.1, and Appendix A for the actual name and description of the proteins).

These accession numbers give us access to the amino acid sequence of each protein, but not all of them have additional information such as domain structure, localization, and secondary structure information. Therefore we cannot readily use any information other than the sequence. In order to extract additional features, we use the bioinformatics tool DARWIN [8], and prediction tools from the web, such as DAS [23] in order to predict transmembrane domains. This is consistent with the ultimate goal of our prediction system, where the only information about the candidate sequence to be predicted is its amino acid sequence (any other features used to predict would have to be extracted using existing tools).

As stated in Section 2.2, experimental results show that Golgi resident proteins are not necessarily localized to one Golgi compartment, but may move around between the different compartments [20]. This allows us to analyze the Golgi as an entity rather than viewing it as a set of different units. This is important to our experiments, since we have only 58 positive examples; dividing these into smaller sets would increase the difficulty of the learning. Therefore, we focus on classifying them as either being Golgi resident or

non-Golgi resident, without taking into account the different compartments of the Golgi apparatus.

Positive examples (58) :	**42** *326* BAA94492	**75** *504* P10633	**121** *456* P39656
	43 *427* Q9WUZ9	**76** *526* P51870	**122** *287* P16232
GOLGI APPARATUS :	**44** *371* Q9Z0F0	**77** *465* Q9R233	**123** *300* P20070
1 *571* Q10471	**45** *371* O88178	**78** *490* P05179	**124** *133* P00173
2 *454* AAD55350	**46** *1325* P55937	**79** *502* P51590	**125** *95* P38391
3 *403* P13721	**47** *1171* U08136	**80** *513* P04799	
4 *578* O08832	(AAB03365)	**81** *534* P97501	ENDOSOME (1):
5 *641* gi/5729913	**48** *986* Q62839	**82** *532* P36365	**126** *663* Q99805
6 *659* O02773	**49** *1150* P27046	**83** *504* P12939	
7 *511* P53619	**50** *959* U14192	**84** *490* P05178	MICROSOME
8 *415* O43505	(AAA62632)	**85** *494* P24470	(ER)(11) :
9 *517* Q62902	**51** *882* Q02353	**86** *507* P11715	**127** *530* P97886
10 *510* gi/1352382	**52** *357* P19814	**87** *504* P05183	**128** *530* P08542
11 *451* O35390	**53** *641* P39098	**88** *493* P05182	**129** *535* Q64550
12 *447* Q09325	**54** *655* P45700	**89** *491* P00176	**130** *530* P08541
13 *380* Q64689	**55** *533* Q10468	**90** *490* P20814	**131** *531* Q64638
14 *653* P33908	**56** *436* P46892	**91** *491* P04167	**132** *285* Q64671
15 *446* O19071	**57** *356* P49256	**92** *500* P10634	**133** *530* P19488
16 *442* Q09326	**58** *154* P08011	**93** *504* P04800	**134** *529* P09875
17 *331* Q9Y274		**94** *492* Q9N2C7	**135** *530* P36511
18 *403* Q64685	**Negative examples (97):**	**95** *500* Q64680	**136** *699* P18163
19 *399* P15535	**-no** cargo	**96** *492* P33272	**137** *317* P50169
20 *586* AAF67014	**-no** trafficking	**97** *500* P12938	
21 *380* P97877	**-no** novel/unknown	**98** *455* P07687	MITOCHONDRIA
22 *398* Q14523		**99** *439* P10867	(16) :
23 *370* O70281	CYTOSOL (3):	**100** *497* Q64581	**138** *519* P19643
24 *333* P97354	**59** *433* P17182	**101** *490* P56656	**139** *573* P19226
25 *308* Q12910	**60** *231* Q9NJ03	**102** *490* P11510	**140** *545* P08428
26 *337* P54751	**61** *375* P02571	**103** *398* Q9QZH8	**141** *543* P56480
27 *385* Q14509		**104** *441* O54734	**142** *529* O75306
28 *380* JC6321	ENDOPLASMIC	**105** *492* P11711	**143** *529* P15105
29 *767* CAA02402	RETICULUM (64):	**106** *489* P19225	**144** *463* P09606
30 *388* BAA94791	**62** *532* P97872	**107** *492* P15149	**145** *373* Q04467
31 *356* O88464	**63** *505* P11598	**108** *471* Q9Z2Z8	**146** *373* P09034
32 *278* P24668	**64** *500* P08683	**109** *444* Q9Z0R9	**147** *452* P30038
33 *356* O08889	**65** *579* P97612	**110** *416* Q02769	**148** *412* P15999
34 *327* BAA83414	**66** *532* Q04799	**111** *372* P27364	**149** *563* P10719
35 *365* Q95577	**67** *491* P13107	**112** *480* P00180	**150** *296* Q60932
36 *327* AF142675	**68** *504* P20816	**113** *475* P38377	**151** *227* P00406
(AAF22225)	**69** *531* gi/2137357	**114** *61* P02802	**152** *169* P10888
37 *374* Q02734	**70** *507* P20817	**115** *677* P00388	**153** *70* P29419
38 *356* O93336	**71** *522* P51869	**116** *509* P09103	
39 *326* O54904	**72** *484* P30839	**117** *509* P04785	PEROXISOME (2):
40 *336* Q61420	**73** *524* P33274	**118** *414* Q63662	**154** *405* Q9QYL1
41 *299* gi/7305627	**74** *503* Q64654	**119** *591* P35565	**155** *302* P09118
		120 *605* P07153	

Table 5.1: Our data set (**Protein number** *Sequence Length* Accession number)

From here on, graphs and tables will refer to the data set examples by protein number as listed in table 5.1.

6. Preliminary data analysis

The first step of our research consists of looking for the presence of obvious motifs, domains, or patterns within the sequences, that allow for the differentiation of positive example sequences and negative example sequences.

6.1 Evolutionary tree

One fact of biological sequence analysis is that evolutionary and functionally related amino acid sequences can differ significantly throughout much of the sequence but preserve the same 3D structure, or the same domains, motifs, active sites, or related dispersed (non-continuous) residues. One such tool we employ is the *multiple sequence alignment* (MSA). This allows us to identify the conserved features (areas of similarity) that correlate with structure and function. A MSA consists of finding the best scoring alignment of two or more sequences according to a particular scoring matrix, by inserting gaps within these sequences so that they all have the same length. We use DARWIN v2.1: This package uses a probabilistic model to create multiple sequence alignments from the sequences in the data set. We then use the alignments to build the evolutionary trees shown in Figures 6.1-6.4. The leaves represent proteins and are labeled with protein numbers from table 5.1. The distances labeling the edges of the trees represent evolutionary distances (or *PAM distances*). The PAM (Point Accepted Mutations) distance does not correspond to time in an immediate way. One PAM unit is the amount of evolution that changes, on average, 1% of the amino acids in the sequence. Note that a PAM distance of 10 does not necessarily mean that 10% of the residues disagree, because of backflips (an amino acid being mutated to another and back to its original one). A *pairwise alignment* (MSA of only two sequences) A has a *similarity score sim(A)* if it is $10^{sim(A)/10}$ more likely that the alignment is between two sequences of common ancestry than between two unrelated sequences. Thus, the similarity score of an alignment represents an estimate of the likelihood that the two sequences evolved from a common ancestor (versus it being a random alignment).

The existing algorithms to compute alignments and trees are not exact and each has its own bias; however they do give a sense of how related the sequences are. Our purpose is

not to compute an exact evolutionary tree for the sequences but to determine if there is some fixed pattern or feature that would indicate a clear evolutionary split between the sequences in the positive set (Golgi proteins), and those in the negative set. We also show evolutionary trees based only on Golgi proteins, in order to analyze their distribution within their own set.

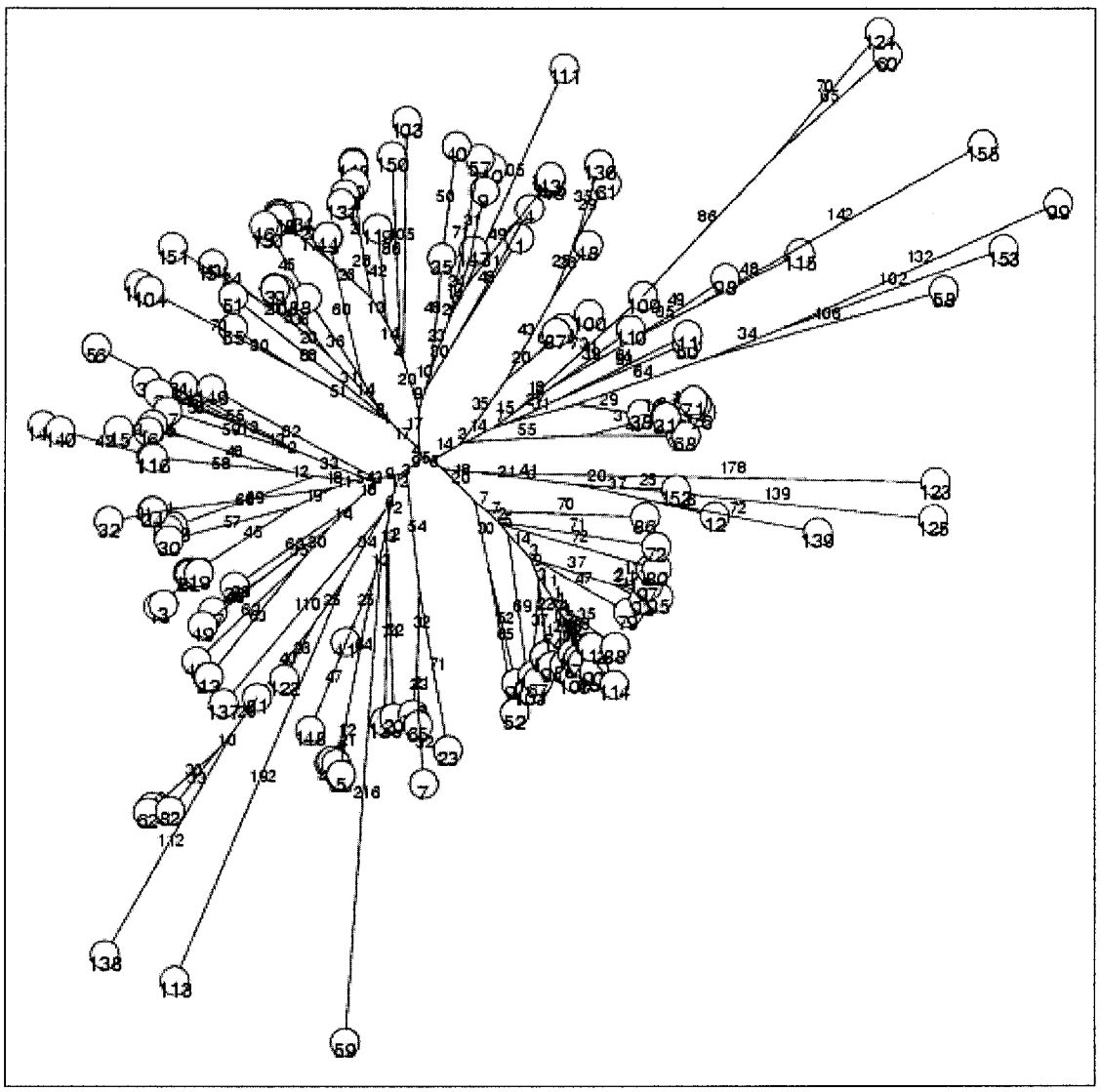

Figure 6.1: Unrooted tree for all 155 proteins in the data set

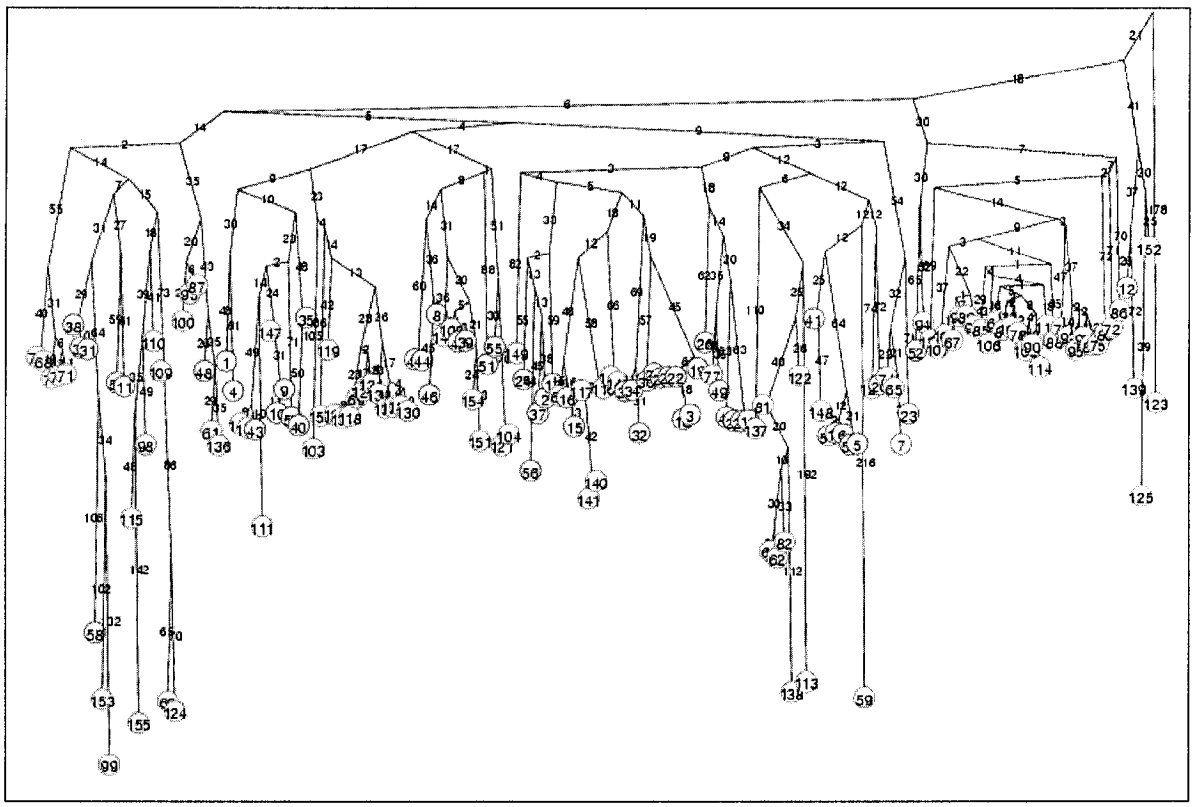

Figure 6.2: Rooted tree for all 155 proteins in the data set

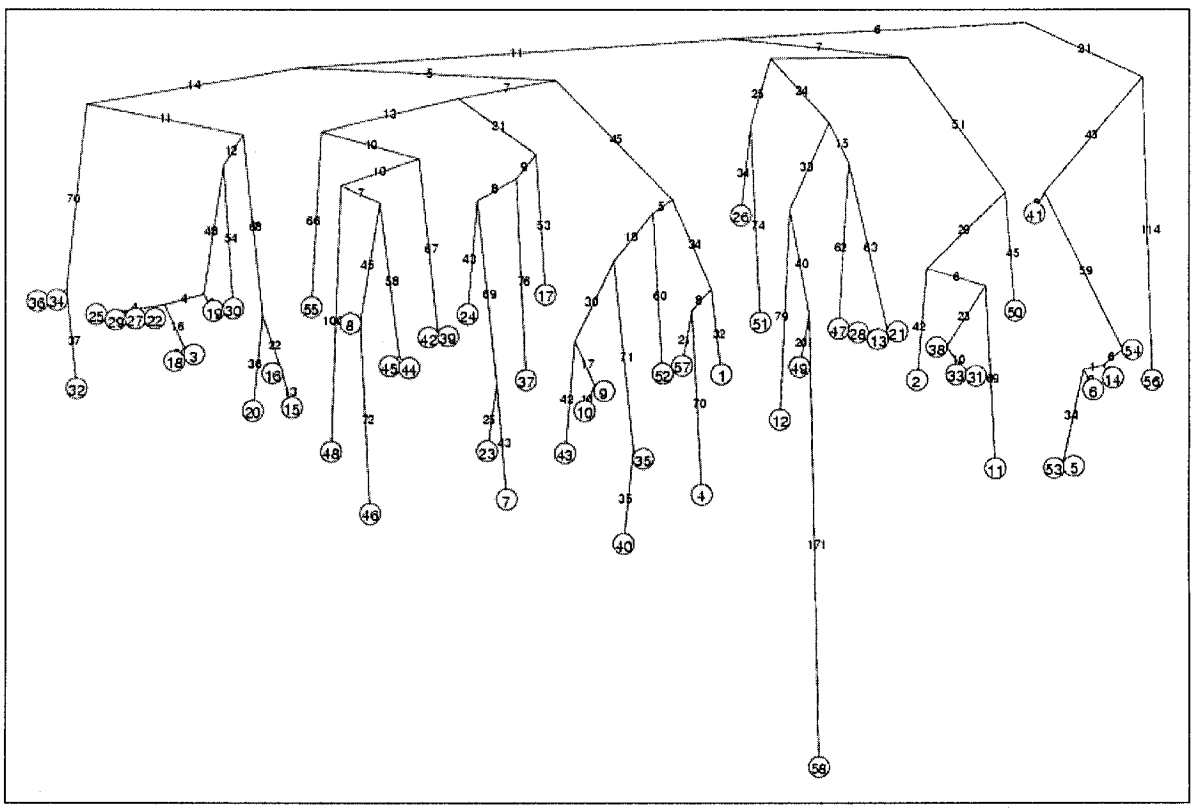

Figure 6.3: Rooted tree for Golgi resident proteins

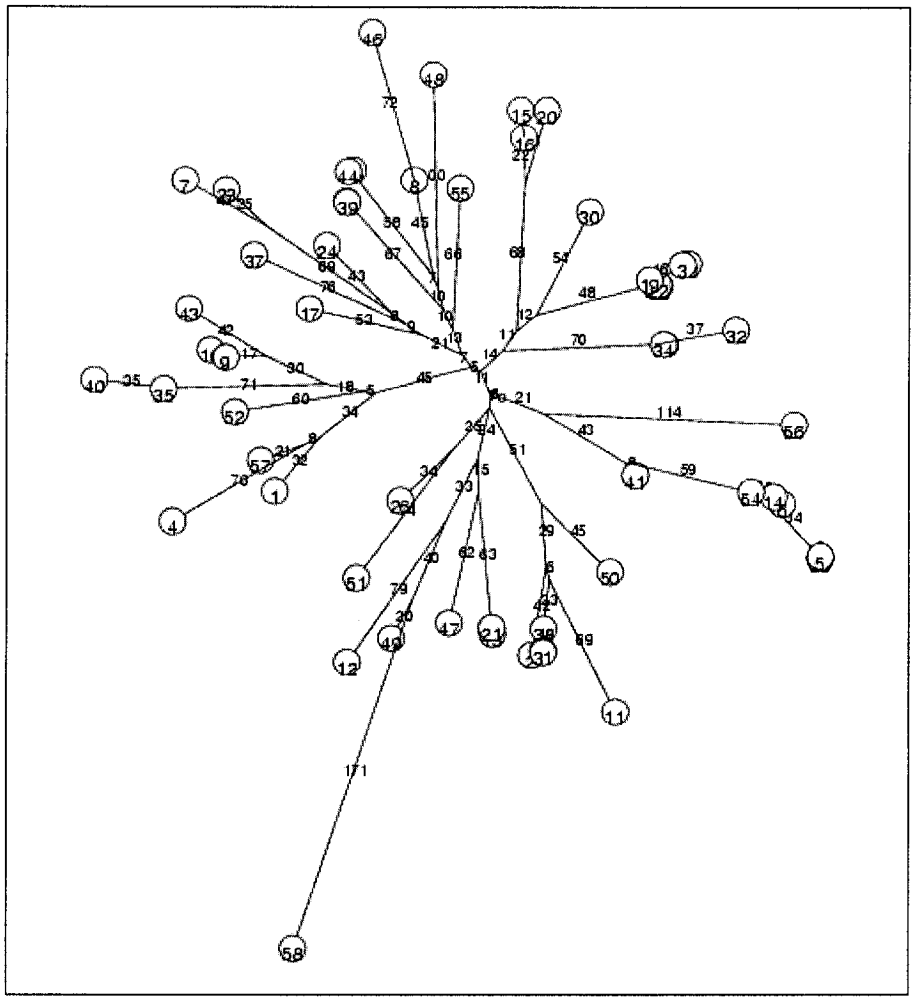

Figure 6.4: Unrooted tree for Golgi resident proteins

From Figures 6.1 and 6.2, we clearly see that there is no obvious split between the positive and negative sequences, and that they seem to be distributed uniformly throughout the tree. However, we do notice clusters of positive examples, and clusters of negative examples, which indicate high similarity between the sequences within the same cluster. Therefore, the PAM distance and similarity score used to construct these trees seem to be measures that may be useful for our final predictor. Within Golgi resident proteins, except for small clusters, we do not see any significant pattern dividing the proteins into subgroups; therefore we initially treat all positive examples together. In all trees, we notice that some sequences seem to be clearly set apart from the rest; these outliers may help improve the generalization of our prediction system, but may also have difficulty being learned (correctly classified). We now analyze the individual pairwise alignments in more detail.

6.2 Pairwise sequence alignments

Another fact of biological sequence analysis is that high sequence similarity between a pair of sequences usually implies significant structural or functional similarity. We used DARWIN v2.1 [8] to compute an all-versus-all alignment (a pairwise alignment of each of the sequences against every other sequence). DARWIN uses a dynamic programming algorithm together with a set of scoring matrices called the *Dayhoff matrices*, to find the most probable alignment of the sequences. Note that although the dynamic programming algorithm is guaranteed to find the alignment with the highest score (i.e. the most probable), this computation is based on the scoring matrices, which are not necessarily correct for our problem. For instance, the Dayhoff matrices are more biased towards finding a better alignment over a longer subsequence than finding a short motif in a transmembrane region or finding a hydrophobicity pattern. These matrices help score the alignments by assigning a score to the individual alignments of any two amino acid residues. One problem is that these matrices are built from sequences that are chosen according to a set of rules, determined by biologists based on their knowledge. This implies a bias. The Dayhoff matrices we use are built using the SwissProt v.38 protein database.

In the previous section, we used *global alignments* where gaps were inserted to make all sequences the same length (which is necessary in order to compare sequences evolutionarily). Here, we use *local alignments*, which consist of finding subregions with high similarity, instead of forcing an alignment over the whole length of the sequences. This may indicate particular motifs, or well-conserved regions on which we can base our prediction.

Although for space and relevance reasons, we do not list the $\binom{155}{2}$ different alignments, we have extracted the similarity score, the estimated PAM distance, and the PAM variance of each pairwise sequence alignment. These results can be seen as three-dimensional graphs (Figures 6.5-6.7), and the tables used can be downloaded online [26]. A PAM distance of 250 or above is roughly equivalent to randomly aligned sequences. An alignment can be classified as good when the similarity score is 300 or above.

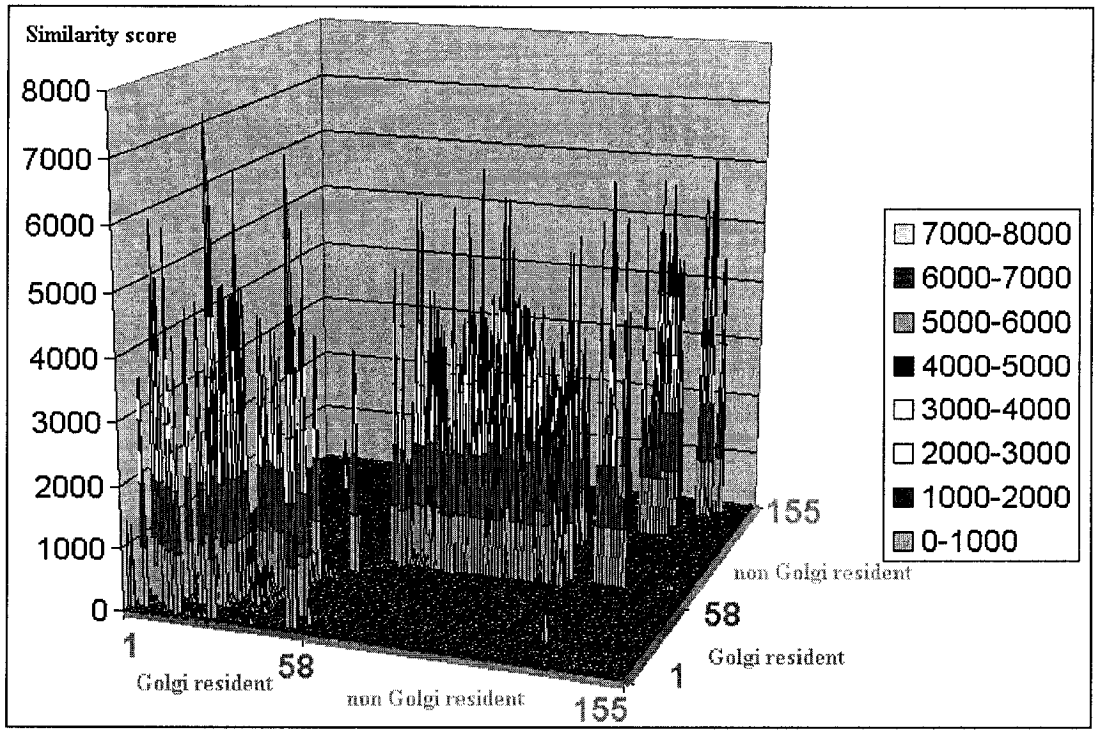

Figure 6.5: Similarity scores of all vs. all alignments

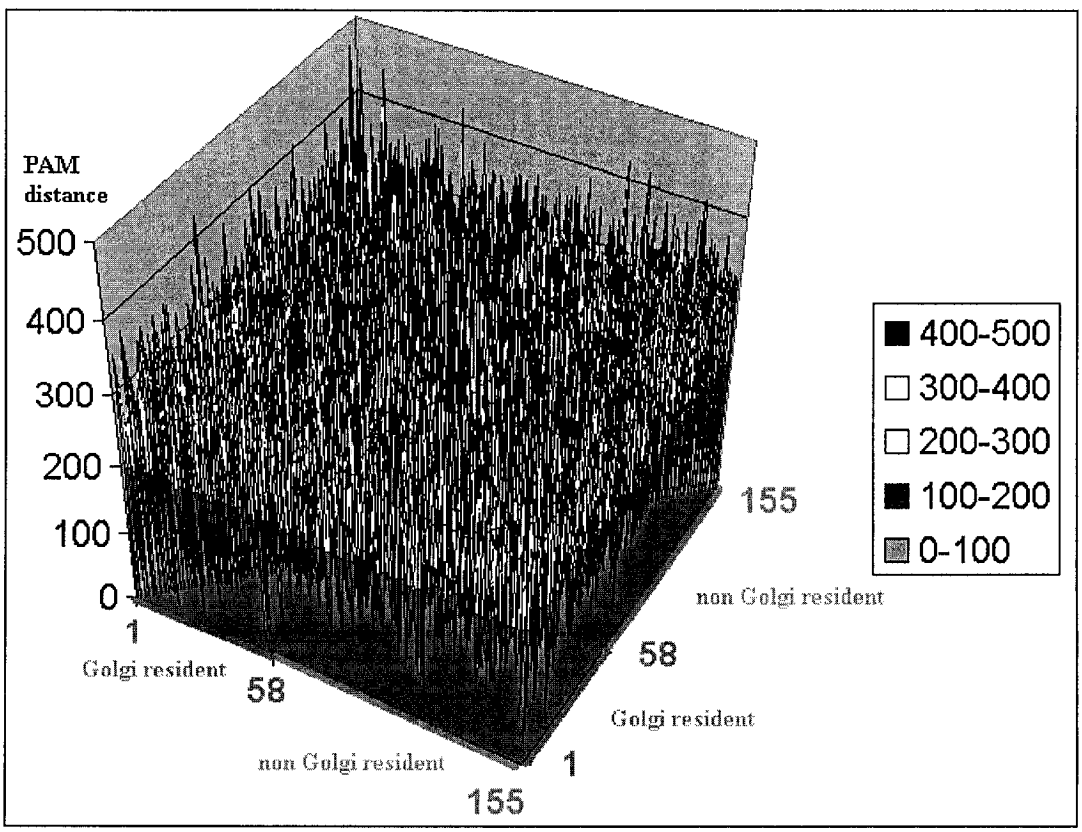

Figure 6.6: PAM distances of all vs. all alignments

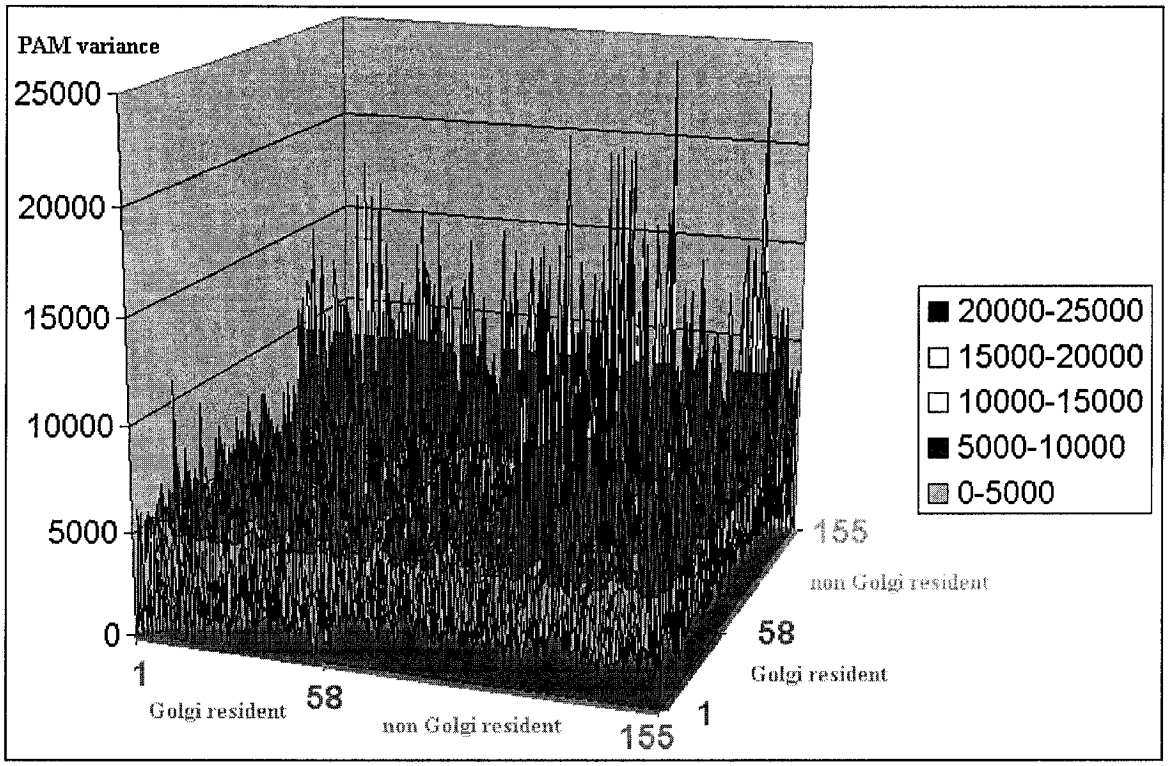

Figure 6.7: PAM variances of all vs. all alignments

We clearly see a pattern in the similarity graph (Figure 6.5), since quite a few of the positive (Golgi resident proteins) vs. positive alignments (bottom left), and of the negative (non-Golgi resident proteins) vs. negative alignments (top right) have similarity scores much higher than most of the positive vs. negative alignments. This verifies the observation from the previous section, where we noticed clusters of positive sequences and clusters of negative sequences, and supports the fact that the similarity score is an important feature that we should use in our prediction system. However, we cannot rely entirely on this score as an important number of positive vs. positive, and negative vs. negative alignments have low similarity scores.

Unlike the similarity graph, the PAM distance (Figure 6.6) and PAM variance (Figure 6.7) graphs do not show any obvious pattern. In the PAM distance graph, we still have very large distances between sequences in the same set, as well as very low distances in the positive vs. negative alignments. It is not clear from these alignments how useful the PAM distance will be in our final predictor. However, the PAM distance did show some relevance for the tree clusters in the previous section, where sequences in the same cluster were a short PAM distance apart. The PAM variance graph shows a high variance for the

negative set in general. This is consistent with the fact that this set contains sequences from many different organelles. All of these proteins have very different functions, and it is not surprising that their sequences are far apart. The PAM variance may help in the classification of positive examples, but may lead to some false positives. We keep in mind that where one measure may give a false classification, another may correct that classification. Therefore all these measures will be tested experimentally in our neural network in order to determine how relevant they are in classifying the examples in the data set (see Chapter 8).

As seen in Section 2.1, a sorting signal may consist of a short signal sequence (15-60 residues). Therefore, the next step consists of looking at particular alignments to see if we can recognize such a particular domain or motif. We accomplish this by extracting the best alignment (with respect to the similarity score) for each sequence in the all vs. all alignment. We then manually choose the shorter alignments. These become the domain candidates (listed in appendix B). If the best alignment that a particular sequence has in common with the whole data set is a short sequence of amino acids, this may well be a domain responsible for some function of this protein. Additionally, if such a domain is found in all proteins within a particular organelle, but not in the other organelles, it is reasonable to infer that this short domain is responsible for targeting proteins to that organelle. We then perform a local alignment of each of these candidate domains versus every sequence in the data set, however we do not find any relevant domain that is present within most of the positive set, but not in the negative set (or vice-versa).
The other type of sorting signal presented in Section 2.1 is a pattern of hydrophobicity within the amino acid sequence. Hydrophobicity is taken into account when an alignment is performed. However, it is hard to detect a hydrophobic pattern from just looking at the alignments. Therefore, we reanalyze the candidate domains, and recompute the alignments, focusing on the hydrophobicity of the amino acids: each amino acid was classified and marked as either hydrophobic or hydrophilic (see appendix C). This allows us to detect any clear pattern that would characterize the positive set versus the negative set. No such pattern is apparent.

6.3 Short substrings

Signal sequences with a length as short as 4 amino acid residues have been shown to be responsible for localization into different organelles (ex: KDEL for the Endoplasmic Reticulum) [1]. It is still possible that a short sequence of amino acids is acting as a sorting signal, but because of its length, was not detected using regular alignment tools in the previous section. Here, we compute every possible *substring* (fixed sequence of contiguous amino acids) of length 4 (20^4), and look for their occurrence in the data set [26]. Figure 6.8 shows only the substrings that occur more than 10 times in either the positive or negative data sets.

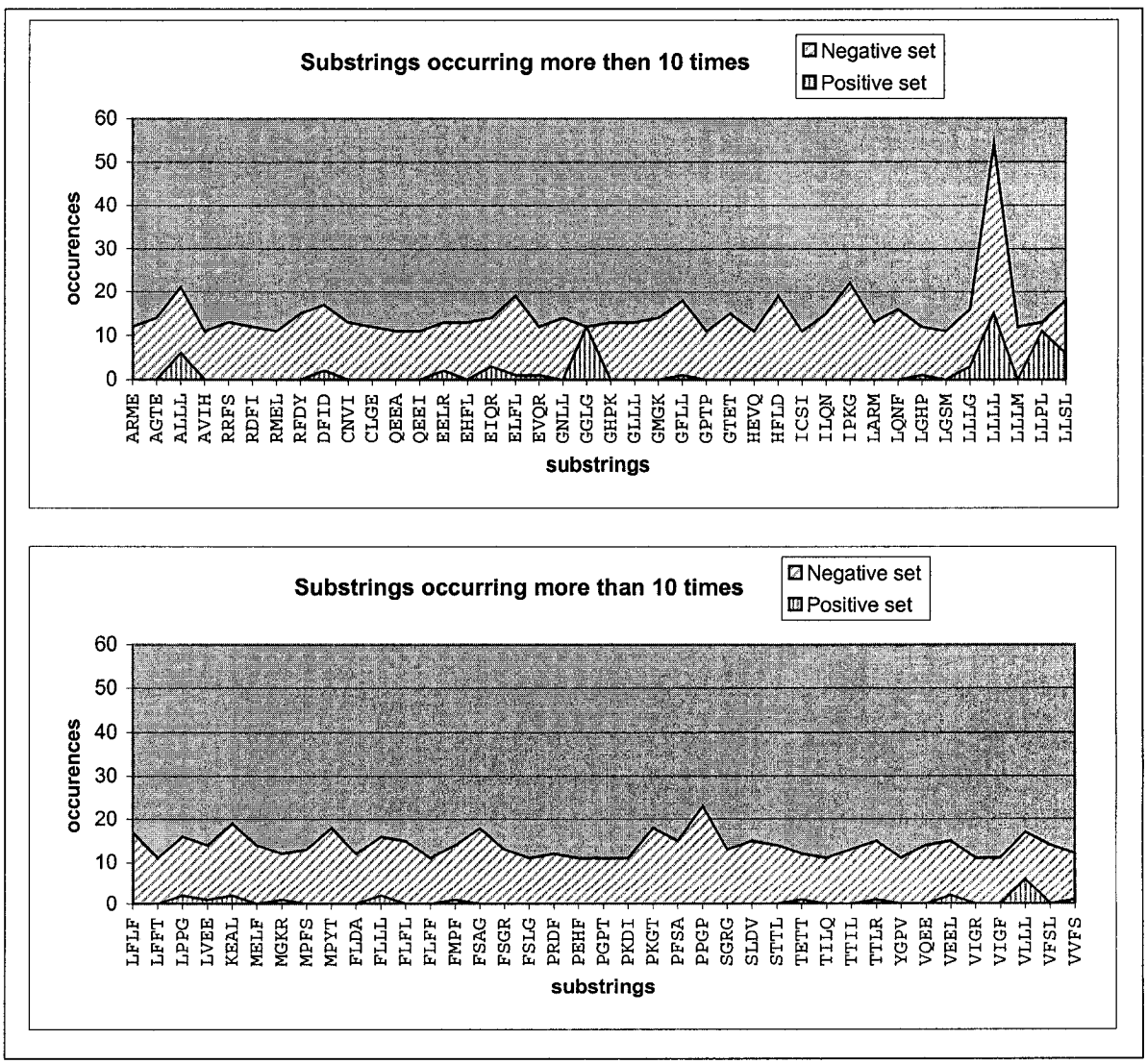

Figure 6.8: Substrings of length 4 that occur more than 10 times

We see that only one substring (GGLG) is present more than 10 times in the positive data set, and never in the negative data set. Every other substring occurs either in both sets, none of the sets, or only in the negative set. However GGLG occurs only 12 times, so it cannot be a universal Golgi signal sequence, as it is not found in each of the 58 Golgi resident proteins. We then analyze the individual Golgi resident proteins in which GGLG is located, and find it in 7 different proteins (protein # [index]: 5 [395], 6 [16,20,24,415], 14 [381], 46 [96], 53 [395], 54 [16,24,411], 56 [399]). We compare their properties and sequences, and find that most of them are very similar throughout their sequences. We conclude that the fact that this substring is only found in Golgi resident proteins is likely to be just a coincidence.

Additionally, this shows that any sorting signal for the Golgi apparatus does not consist of a simple substring of length greater than 3. Therefore, signal patches (as defined in Section 2.1) must be responsible for the localization of proteins to the Golgi. This implies looking for a *subsequence* (sequence of amino acid residues, where each two are separated possibly by a *gap* or arbitrary substring). Finding such a subsequence is a hard problem that cannot be yet efficiently solved. Therefore, we try to find a simpler signal patch, which consists of two or more substrings of length 4 that would always occur together in Golgi resident proteins. Results of this computation is available online [26]. Similarly to the previous experiment on single substrings, the sets of short substrings that occur together only in Golgi resident proteins are always found among similar sequences, and at most on 7 Golgi resident proteins.

6.4 Frequencies

Frequencies of amino acids have been shown to be relevant for some structural properties of proteins [21]. Therefore, we analyze these frequencies in the Golgi resident sequences (positive set), as well as in those of other organelles (negative set), and compare them to the average frequencies over the whole SwissProt v.38 database (see table 6.1).

Positive Examples: 28643 AA residues	Negative Examples: 44801 AA residues	SwissProt v.38: 29085965 AA residues
A: 0.065915 (-0.009930)	A: 0.062253 (-0.013592)	A: 0.075845
C: 0.016479 (-0.000545)	C: 0.016183 (-0.000841)	C: 0.017024
D: 0.049750 (-0.003302)	D: 0.050713 (-0.002339)	D: 0.053052
E: 0.065601 (+0.002427)	E: 0.060133 (-0.003041)	E: 0.063174
F: 0.045037 (+0.004636)	F: 0.055401 (+0.015000)	F: 0.040401
G: 0.065147 (-0.003749)	G: 0.066003 (-0.002893)	G: 0.068896
H: 0.028000 (+0.005659)	H: 0.025848 (+0.003507)	H: 0.022341
I: 0.047062 (-0.009973)	I: 0.057677 (+0.000642)	I: 0.057035
K: 0.059910 (+0.000754)	K: 0.062610 (+0.003454)	K: 0.059156
L: 0.107531 (+0.014609)	L: 0.107565 (+0.014643)	L: 0.092922
M: 0.023147 (-0.000521)	M: 0.027566 (+0.003898)	M: 0.023668
N: 0.040359 (-0.004881)	N: 0.036584 (-0.008656)	N: 0.045240
P: 0.052962 (+0.003662)	P: 0.057365 (+0.008065)	P: 0.049300
Q: 0.047062 (+0.006682)	Q: 0.034731 (-0.005649)	Q: 0.040380
R: 0.057222 (+0.005434)	R: 0.049084 (-0.002704)	R: 0.051788
S: 0.068708 (-0.003060)	S: 0.062521 (-0.009247)	S: 0.071768
T: 0.047656 (-0.010042)	T: 0.052700 (-0.004998)	T: 0.057698
V: 0.060783 (-0.004494)	V: 0.067610 (+0.002333)	V: 0.065277
W: 0.015815 (+0.003165)	W: 0.014174 (+0.001524)	W: 0.012650
Y: 0.035855 (+0.003737)	Y: 0.033191 (+0.001073)	Y: 0.032118

Table 6.1: amino acid frequencies ($\Delta_{frequency}$ with SwissProt v.38)

We see that the maximum difference for the frequency of any amino acid between the positive set, negative set, and the SwissProt database, is at most around 1%, which is not significant enough to characterize one set versus the other, or to even use this feature in our predictor.

7. Experimental Setup

We need to divide the data set into a training set and a test set. The training set is used to adjust the parameters of the neural network, while the test set is needed to quantify the accuracy of our predictor on data it has not yet seen. It is important that the training set does not have a disproportionate number of a certain type of examples (positive or negative), as the network may become biased towards predicting a candidate example of that type. Also, as we have seen in Section 3.1, it is important that the training set represents the whole range of possible examples, in order to have good interpolation. A validation set is sometimes used to detect overfitting, preventing the ANN from becoming too specific to the training data. A validation set is similar to a test set, as its data examples are not seen during training. We use four types of partitioning techniques throughout our experiments:

- *10-fold cross-validation*: This consists of splitting the data into 10 folds, then training the system on 9 of them, and testing on the remaining one. This process is repeated ten times by rotating the sets (using a different test set, and a new neural network every time). Since the examples used in the test set are not used during training, it allows us to check the stability and error of our prediction system on unseen data. Each fold contains the same proportion of positive and negative examples as in the whole data set. The folds are kept constant throughout all the experiments using the 155 examples in the data set. This allows us to compare the different implementations in a more consistent way.
- *Leave-one-out cross-validation*: This training technique consists of training the network on all but one example, and using that example to test the predictor. The process is repeated for every example in the data set. This technique is very useful for a more detailed analysis, and enables the detection of examples that are predicted incorrectly. Another advantage is that we can use more examples to train, while still being able to test the accuracy of the neural network. This is very useful when the data set is relatively small (as it is in our case).

- *Random testing*: This technique consists in choosing a proportion of examples to be taken randomly from the data set for testing. The rest of the examples are used to train the system.
- *Sampling*: This technique is used when the ratio of negative to positive examples is too low or too high. The data set is divided into multiple smaller sets. Each of these sets contains all of the examples of the minority class, and a proportional number of examples of the majority class. We then run each experiment on all of these smaller data sets, and either compute an average or report all of the results when quantifying the accuracy of our predictor in general.

The actual experimental platform used to train neural networks, run tests, and view results is our own graphical user interface application called ProteiNNet v.1.0 (see appendix F).

8. Results

The most important aspect of our prediction system is the set of features that will be used to represent the data. If we extract a good set of features, which somehow encode for the protein location, then, with enough proper training and an appropriate structure, the neural network system should learn the relationship. Our main problem is the fact that the amino acid sequences in the data set have variable length from 150 to 1500 amino acids. Since a neural network cannot have a variable number of inputs, we cannot use the whole sequence by encoding each amino acid. Another option includes using general features of proteins, such as length, molecular weight, type of domains present, the percentage of each kind of amino acid, secondary structure information if available, chemical properties, or evolutionary distance and similarity with other proteins. We explore several of these features in the following experiments. Some features have shown to give good accuracy on prediction (see Chapter 4, and results 8.3). Initially, we want to gain some insight into the localization process that takes place in the cellular environment by trying to find local features that could be responsible for this. The results for most of the experiments referred to in this chapter are available online [26]. For the sake of space and clarity, we describe in detail only the most relevant results.

8.1 First n amino acids

As seen in Chapter 2, proteins are directed to their final location by means of a sorting signal, and there are different types of such signals. Our biggest challenge is to decide what type to look for, and then to find an appropriate encoding for the input into the network. Initially we focus on the simplest sorting signal (as described in Section 2.1), which consists of a short sequence of amino acids usually located within the first 60 positions on the N-terminus of the amino acid chain.

Since the unary encoding for amino acids has been shown to be quite successful, we use a 1-of-20 encoding. Each amino acid is encoded using 20 input nodes (the node corresponding to the amino acid is switched on, while the others remain off). In preliminary experiments, we vary the number of amino acids used up to 60 [26]. We find 40 to give the best accuracy on training and testing. This results in an input layer of 800

nodes (20*40). Since one hidden layer is generally enough for most problems [1,3], and more than one layer increases the computational complexity drastically, we decide to use just one hidden layer. Next, we determine how many hidden units to include in this layer. Preliminary experiments show that a fully connected network (with various numbers of hidden nodes) cannot learn successfully, as the number of connections is too large relative to the amount of training data available [26]. This is as expected. We require additional structural groupings of the inputs that have a direct relationship: with all 20 inputs corresponding to an amino acid going to one hidden node, we get a total of 40 hidden nodes, one for each residue. Since the prediction is a Boolean function, we only need one output node, which can take values in the range (0.0,1.0). We define this value to be the confidence measure for our prediction. For the percent misclassified reported in our results, we consider a training example as correctly classified if the predicted value is within 0.4 of the target value (0 or 1), and misclassified otherwise.

After varying different parameters, such as learning rate, momentum, and the number of epochs, we find that a learning rate of 0.3 and a momentum of 0.1 give the best results. The accuracy usually stabilizes after approximately 500 epochs. Figure 8.1 shows the averaged 10-fold cross-validation results on 1000 epochs. The mean squared error graph on the training and test set, as well as the results for each individual fold, are available online for every graph shown here [26]. We use an initial weight randomization between 0.5 and –0.5 for the neural network connections.

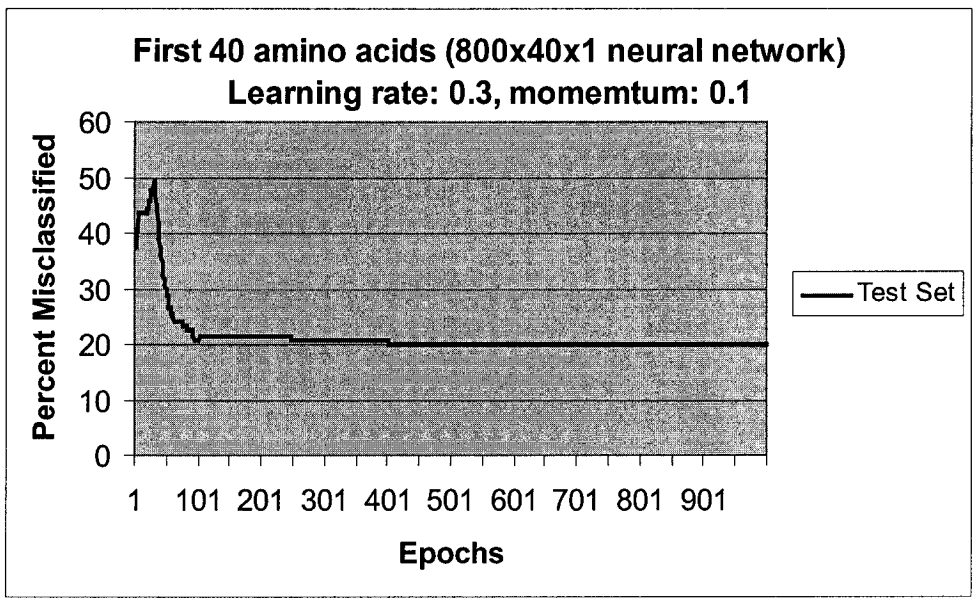

Figure 8.1: Averaged results of 10-fold cross-validation on first 40 amino acids

These results seem unexpectedly good for a first set of experiments, with a prediction rate of 80% (misprediction rate of 20%, standard deviation of 8.4%). After analyzing the proteins in the data set, we notice (in the cases where that information is available) that an important number of Golgi proteins (at least half) have a very similar domain structure, consisting of a short cytoplasmic domain (<10 amino acids), followed by a single hydrophobic membrane spanning domain (16-25 AA), and a large carboxylterminal catalytic domain. These represent type II membrane proteins, and we believe their similarity caused the neural network to predict well. Furthermore, if the data set is a good representative of the actual Golgi resident protein distribution, this result is consistent with the fact that Glycosyltransferases (which are all Type II membrane proteins) represent a very significant portion of the Golgi resident proteins [20]. To understand if the network is detecting the similarity of these proteins, we compute an all-versus-all alignment of the proteins based only on the first 40 amino acids, and keep the best alignment for each one [26]. We then go through the alignments by hand, classifying a protein as its best match: Golgi resident if the sequence that gives the best alignment codes for a Golgi resident protein, and non-Golgi resident otherwise. The quality of the alignments is based on similarity and PAM distance. Out of the 155 best-matched alignments, we misclassify 32, which is exactly a 20.6% misclassification rate. Therefore, in this experiment, the network is predicting slightly better than we can. We conclude that

there is some important location information in the first 40 amino acids, but that not all proteins have that information. It also seems likely that type II membrane proteins are being classified correctly. To check this hypothesis, we run a leave-one-out cross-validation on the data set to determine which proteins are not classified correctly. The results on the positive set are displayed in table 8.1, where:

"n:": is the protein number as assigned in table 5.1

underlined&bold: represents type II membrane proteins

"√ ": means the protein was correctly classified

"X": means the protein was misclassified.

1: √	**13: √**	**24: √**	35: X	47: X
2: X	**14: √**	25: X	36: √	48: X
3: √	**15: √**	**26: √**	**37: X**	**49: √**
4: X	**16: √**	**27: √**	38: √	50: X
5: √	17: X	28: √	**39: √**	**51: √**
6: X	**18: √**	29: √	40: X	52: X
7: X	**19: √**	**30: X**	41: √	**53: √**
8: X	20: √	31: √	**42: √**	**54: √**
9: √	**21: √**	32: √	43: X	**55: √**
10: X	**22: √**	33: √	**44: √**	56: X
11: X	**23: X**	**34: √**	**45: √**	57: X
12: X			46: X	58: X

Table 8.1: Leave-one-out cross-validation on the positive set

These results show that 23 out of the known 29 type II membrane proteins are being correctly classified (80%), versus 37 out of 58 total Golgi resident proteins (64%). In this case, it is possible that the 6 proteins are misclassified due to the noise in the training set: If the network is really learning the classification of Golgi resident type II membrane proteins, then the training set contains many false positive examples. We check this hypothesis in the next experiment. These results support the fact that most type II membrane proteins are being classified correctly compared to all other types of Golgi resident proteins put together.

Next, we run tests extracting only the known Golgi resident type II membrane proteins as positive examples, and we sample the negative set to get a proportional negative set size (we also run experiments without sampling, as expected these are unsuccessful for correctly classifying positive examples). This results in two sets of negative examples; we run experiments using both sets. We do not use any other proteins from the Golgi to

avoid using any false negatives. We achieve around 80% accuracy overall, and in the positive set (using the prediction from both experiments) [26].

We believe the prediction overall did not improve partly because of the sampling: we had to remove some examples that might have helped correctly predict a certain other example in the set that we kept, and similarly in the positive set where we might have excluded some relevant type II proteins that had not yet been identified as such. The results of this experiment suggest that we reached an upper bound on the accuracy that can be obtained using only the first n amino acids. Therefore, in order to improve our prediction engine, we require more biologically motivated features to use as inputs to the neural network. Evolution mutates sequences by, for example, domain shuffling, inserting, and deleting amino acids from an amino acid sequence [4]. Therefore, if a certain type of domain is responsible for localization into the Golgi apparatus, it may possibly be located anywhere within the sequence. Anytime the first 40 amino acids are used, a misclassification would result when this domain is located past the first 40 amino acids. In the next section we focus on finding such a domain, and using it to predict.

8.2 Transmembrane domains

As seen in Chapter 2, experimental chimeric studies [20] show that the membrane spanning domains of mammalian Golgi glycosyltransferases are responsible for directing the localization of molecules to the Golgi apparatus. This is consistent with the results from the previous section, as most type II membrane proteins have their transmembrane domain located within the first 40 amino acids. Therefore, we base our next set of experiments on transmembrane domains, and use as input, 40 amino acid residues centered on such domains. This length is large enough to encompass each transmembrane domain in the data set, but still yields a reasonable number of connections in the neural network compared to the size of the data set. Initially, we use the same input encoding, output encoding and structure as in the previous experiment.

Not all proteins have information about the location of any existing transmembrane domains. Also, this information is likely not to be available for new candidate sequences either. Therefore, we use various online tools to extract the transmembrane domains, and we try to determine their exact location manually by inspecting the results of the prediction algorithms [26]. We use the following three tools: DAS (Dense Alignment Surface) which bases its prediction on low-stringency dot-plots of the query sequence against a collection of non-homologous membrane proteins using a special scoring matrix [23]; HMMtop which uses a Hidden Markov Model, and is based on the difference in the amino acid distributions in various structural parts of proteins [24]; and TMAP which uses a Kyte-Doolittle hydropathy profile to detect transmembrane spanning domains [25]. The resulting data set from this analysis is summarized in appendix D. An example of how we put these different results together can be seen in table 8.2.

Prediction algorithm used	Transmembrane domains predicted for Golgi resident protein # 5 start index - stop index
TMAP	30-58 267-290
HMMtop	37-56 218-237 264-283 298-315
DAS (Cutoff: 2.2)	37-55 271-278
Sets of 40 amino acids used in the data set	***27-66*** ***260-299***

Table 8.2: Extracting transmembrane domains from protein #5

This new data set consists of 390 data examples, with 121 transmembrane domains located in Golgi resident proteins, and 269 in proteins from other organelles. For the positive set, we cannot use all of the transmembrane domains as positive examples, since for this experiment we assume that there is only one transmembrane domain per protein, responsible for the targeting of that protein to the Golgi. We run several tests using the whole data set (121 positive, and 269 negative examples), but as expected, these results are only moderately good (70% prediction rate, standard deviation of 4.8%) [26]. This is probably due to the large number of false positive examples given our assumption: if there is actually only one transmembrane domain per protein responsible for localization into the Golgi then using all the transmembrane domains will provide many false positive examples. Therefore we choose one transmembrane domain for each Golgi resident protein. We proceed by comparing the previous results on the first 40 amino acids with the prediction of the transmembrane domains. We keep the first 40 amino acids when the prediction is correct and a transmembrane domain is present. Otherwise, we cross-reference the different prediction algorithms, as well as the information from SwissProt (when available) to find the domain most likely to be correct. Even with all of these

sources of information, we have some proteins with either too many transmembrane domains (in the case of multiple membrane-spanning proteins), or no such domains at all. This is probably true for the soluble and peripheral proteins. In this case we decide to initially remove them from the set, and consider them as "special" cases. This is done in order to compute the accuracy of our system without this noise. The resulting positive data set consists of 49 positive examples (listed in table 8.3), and 269 negative examples (see appendix D), since every transmembrane domain in the negative set should be considered negative (according to our assumption).

1 (1->40)	14 (30->69)	28 (1->40)	41 (1->40)
2 (1->40)	15 (1->40)	29 (11->50)	42 (1->40)
3 (1->40)	16 (1->40)	30 (11->50)	44 (1->40)
4 (1->40)	17 (247->286)	31 (1->40)	45 (1->40)
5 (27->66)	18 (1->40)	32 (1->40)	47 (1130->1169)
6 (40->79)	19 (11->50)	33 (1->40)	49 (1->40)
7 (1->40)	21 (1->40)	34 (21->60)	51 (1->40)
8 (1->40)	22 (11->50)	35 (302->341)	52 (300->339)
9 (1->40)	23 (1->40)	36 (21->60)	53 (21->60)
10 (466->505)	24 (1->40)	37 (1->40)	54 (31->70)
11 (160->199)	26 (1->40)	38 (1->40)	55 (1->40)
12 (1->40)	27 (1->40)	39 (1->40)	57 (310->349)
13 (1->40)			

Table 8.3: Positive transmembrane data set

Table 8.3 shows the new positive transmembrane data set. Each data example is represented by the original protein number followed by the starting and ending index of the 40 amino acids surrounding the transmembrane domain.

The next obstacle we have to overcome is the disproportional size of the negative set compared to the positive set: training the neural network with a 5:1 negative to positive ratio is not likely to give very meaningful results. We run leave-one-out cross-validation experiments using the best-predicted positive examples, and all of the negative examples [26], but as expected these are unsuccessful (only 45% of the Golgi transmembrane domains are correctly classified). We proceed by sampling, and divide the data set into 3 smaller data sets. Each of these consists of all of the positive examples listed in table 8.3, and one third of the negative examples [26]. The results are shown in Figure 8.2.

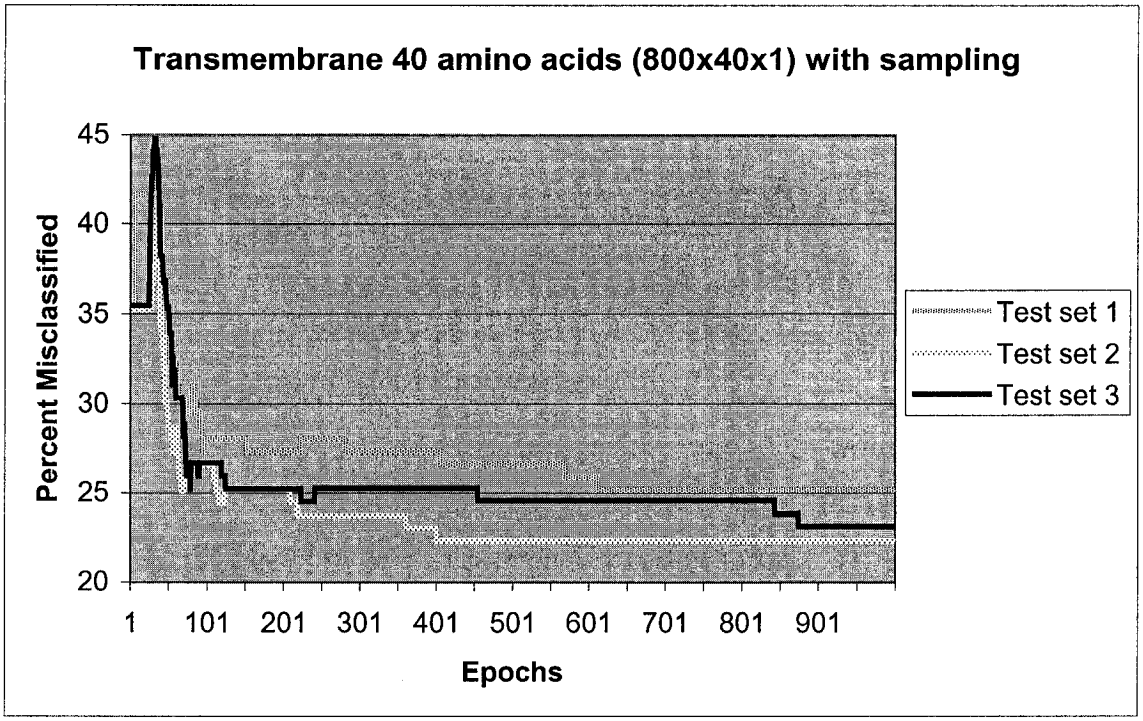

Figure 8.2: Averaged 10-fold cross-validation on transmembrane data sets 1, 2 and 3

We see from these results that the success rate is lower using the transmembrane domains than the success rate using the first 40 amino acids. The average over the 3 data sets is approximately 77% prediction accuracy (23% misclassification rate, standard deviation of 8.8%). We can expect such results based on the fact that we use prediction tools to detect the transmembrane domains. Therefore we compute a prediction on top of a previous prediction. This is likely to compound error. Also, the same sampling problem as in the previous section may have arisen: by dividing the negative set on the training, we may misclassify some examples that would have otherwise been correctly classified (if one or more other relevant examples had not been in a different data set). We also run experiments using one transmembrane domain per protein, but this gives very similar results [26].

In order to verify how well the neural network is learning given these data sets, we compute an all-versus-all alignment of all the sequences in the transmembrane data set (318 examples). We align on the 40 amino acids around the predicted transmembrane domains. We then extract the best alignment (based on similarity) of each sequence against all other sequences [26], and manually classify each one as its best match in a

nearest neighbour fashion. Using this technique we misclassify 93 examples, which is exactly 23.8 % of misclassification rate (76.2% prediction rate). This is similar to the average prediction rate of the neural network over the three data sets, therefore it seems that sampling does not affect its performance significantly, since overall it is predicting with a slightly better accuracy than we can using only the alignments. Again here, we see that there is some relevant location information within the transmembrane domains, but only encoding the amino acid residues does not allow the neural network to learn anything more than strict sequence similarity. Thus we need to add more biologically relevant information into our system.

As seen in Chapter 2, some sorting signals may consist of a pattern of hydrophobicity of the amino acids. During the preliminary data analysis, we could not detect any obvious such pattern, but since membrane-spanning domains consist of hydrophobic amino acids, we decide to incorporate the hydrophobicity of the amino acids into the network. This can hopefully allow the network to recognize the important segment of the 40 amino acids used as input, because the amino acids located within this transmembrane segment would naturally have more importance in the prediction.

In order to include the hydrophobicity level, we experiment with two different encodings and structures for the neural network. The first consists of keeping the same internal structure, but adding one extra input node for each amino acid. This represents the hydrophobicity level of the corresponding amino acid, scaled from 0 to 1 (as listed in appendix C). Each amino acid is now represented using 21 input nodes; each set of these 21 input nodes is connected to one internal hidden node. This results in a 3-layered 840x40x1 neural network. We train our system on the same three transmembrane data sets; the results are shown in Figure 8.3. But this experiment is unsuccessful, as it does not improve our prediction rate, achieving a misclassification rate on predicted transmembrane domains of approximately 24% on average over the three data sets (standard deviation of 8.3%). This seems to indicate that the hydrophobicity level of the amino acids around the transmembrane domain is not a useful measure for localization prediction.

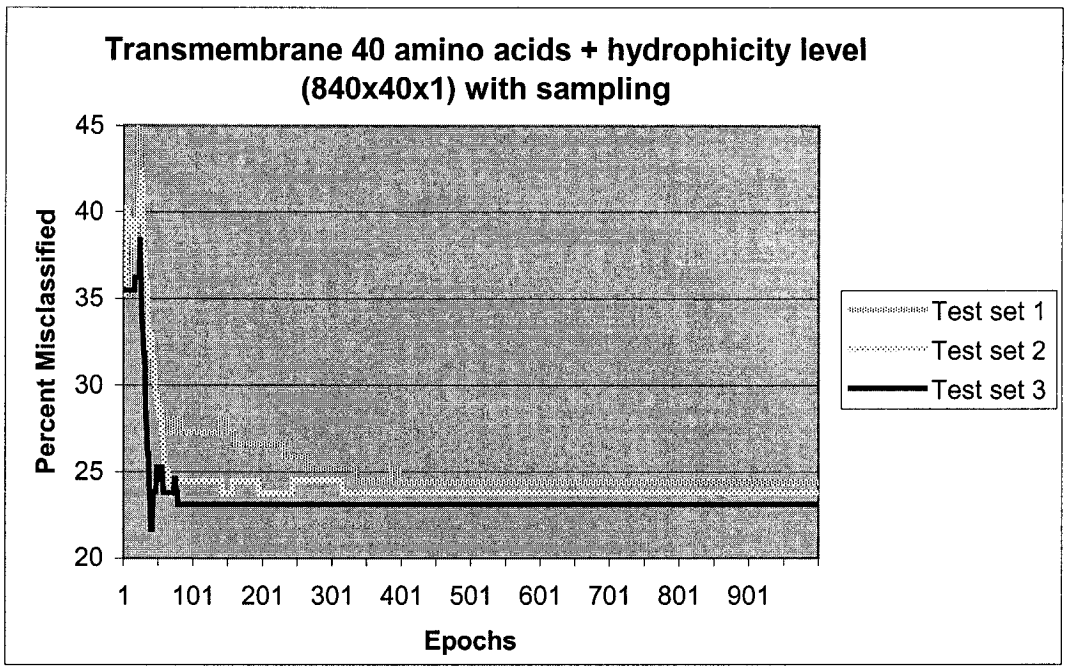

Figure 8.3: Averaged 10-fold cross-validation on transmembrane data sets 1, 2 and 3 with 1-of-20 encoding and hydrophobicity level

In order to verify that the hydrophobicity level of the amino acids around the transmembrane domain is not a useful measure for localization prediction, we try a completely different architecture. It consists of using a single node per residue representing only the hydrophobicity level also scaled from 0 to 1. This results in a small 40x40x1 3-layered neural network. We train this new network on the three transmembrane data sets, and find a very poor prediction rate, of approximately 67% on average (standard deviation of 10.1%) [26]. This is probably due to the fact that in both positive and negative examples, the amino acids located in the membrane-spanning domain are hydrophobic. Therefore this representation may have introduced noise in the training by adding the same signal in both positive and negative training examples. These results agree with the previous experiment and lead us to conclude that the hydrophobicity within transmembrane domains does not seem like a strong measure for predicting protein localization.

Using transmembrane domains seemed to be a biologically meaningful approach, but from the results in this section, we see that even pushing the "special" cases aside, and taking into account the many different levels where noise could have been introduced into

the data, we still see that any sorting signal might be more complex than a simple pattern within the transmembrane domain, or the first 40 amino acids. We need to be able to encode a more global view of the proteins in the data set, using features that can encompass more properties.

8.3 Similarity score and PAM distance

In the previous experiments, we concluded that using a fixed number of amino acids was successful for some types of sorting signals which we were able to detect, but could not detect more complex signals, where we must use more global features.

As seen in the preliminary data analysis (Chapter 6), evolutionary trees and sequence alignments, the similarity score, PAM distance, and PAM variance of pairwise alignments seem to contain useful information regarding the localization of proteins in the data set. Furthermore, we determined that the neural networks are learning similarity between short 40 residues sequences, which allows a prediction rate of 80% at best. Thus, using a more general measure of similarity, and averaging this measure over the positive and negative data set, may give us a more accurate prediction, for all the different types of proteins. This amounts to approximating a nearest neighbour method based on the tree representation in Section 6.1, but weighing and adapting to the different measures according to their relevance for the prediction. Assuming that the data set is representative of all types of Golgi resident proteins, this method should even classify soluble, and peripheral Golgi proteins that were hard to classify in the previous experiments.

Using DARWIN, we compute, for each sequence in the data set, the average similarity score, PAM distance, and PAM variance against the positive set, and against the negative set (see appendix E). In order to use these measures, we design a 2-layered neural network, with 6 input nodes, and one output node (see Figure 8.4).

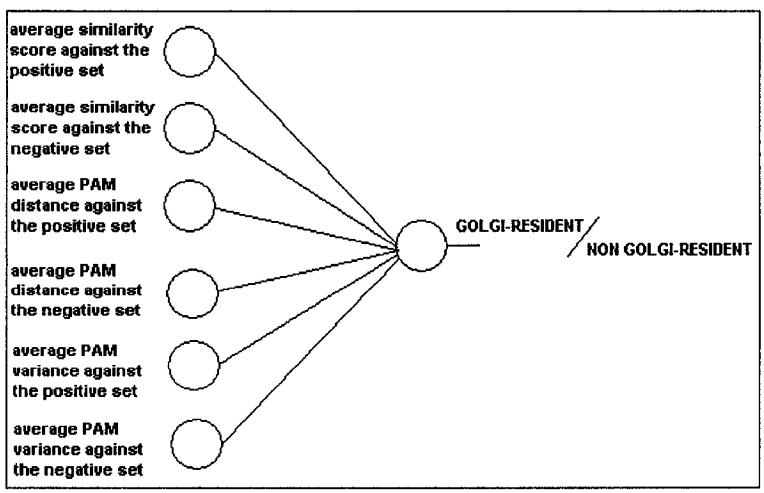

Figure 8.4: Neural network structure

The output encoding is the same as in the previous experiments. As for the prediction of a candidate sequence, we use the natural output of the neural network, which is a floating number in the range (0.0,1.0), as the confidence measure for the given prediction.

Each input is scaled individually between 0 and 1, because this has been shown to improve the learning ability of artificial neural networks [1]. After analyzing the data values, we decide to use the following scaling:

- Similarity score: scaled between 0 and 300, where any value greater than 300 is set to 1.0.
- PAM distance: scaled between 180 and 250, where any value greater than 250 is set to 1.0, and any value smaller than 180 is set to 0.0.
- PAM variance: scaled between 2000 and 6000, where any value greater than 6000 is set to 1.0, and any value smaller than 2000 is set to 0.0.

We run various tests [26], including random test sets, 10-fold cross-validation, adding a hidden layer, and varying the number epochs ($\leq$10,000), as well as the learning rate and momentum. We find that the network has the best accuracy using the two-layer structure, training with a classification threshold of 0.5 (the lowest possible: a data example is correctly classified if the output is within 0.5 of the target value), a learning rate of 0.3, a momentum of 0.1 and approximately 350 epochs (overfitting seem to occur in several experiments after that many epochs). Results are shown in Figure 8.5.

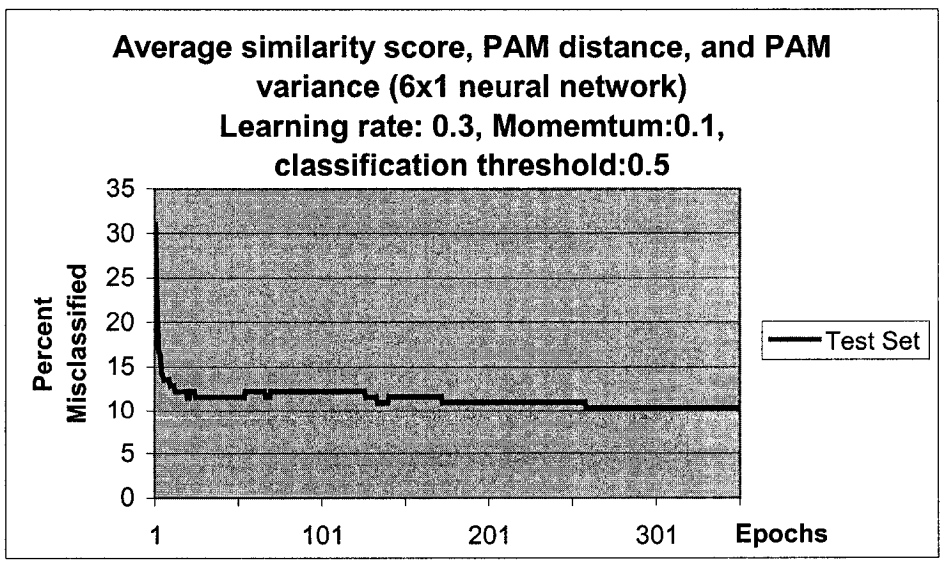

Figure 8.5: Averaged results of 10-fold cross-validation

As we can see from Figure 8.5, we obtain the most successful results by correctly classifying approximately 90% of the data examples (10 % misclassification rate, standard deviation of 6.8%). Setting the classification threshold to 0.4 gives a slightly worst prediction rate of 87% (standard deviation of 7.8%) [26].

Therefore, using these general features allows the network to recognize proteins using more complex sorting signals that we could not detect using other methods. This corresponds to our observation that the neural network is approximating a nearest neighbour method based on the tree clusters in Section 6.1. In order to verify this, we run a leave-one-out cross-validation test on the data set, to detect the misclassified examples (**bold** in table 8.4).

1: √	**32: X**	63: √	94: √	125: √
2: √	33: √	64: √	95: √	126: √
3: √	34: √	65: √	96: √	127: √
4: √	**35: X**	66: √	97: √	128: √
5: √	36: √	67: √	98: √	129: √
6: √	37: √	68: √	99: √	130: √
7: √	38: √	69: √	100: √	131: √
8: √	39: √	70: √	101: √	132: √
9: √	**40: X**	71: √	102: √	133: √
10: √	41: √	72: √	103: √	134: √
11: X	42: √	73: √	104: √	135: √
12: X	**43: X**	74: √	105: √	136: √
13: √	44: √	75: √	106: √	137: √
14: √	45: √	76: √	107: √	138: √
15: √	46: √	77: √	108: √	139: √
16: √	**47: X**	78: √	**109: X**	140: √
17: √	48: √	79: √	**110: X**	141: √
18: √	49: √	80: √	111: √	142: √
19: √	50: √	81: √	112: √	143: √
20: X	51: √	82: √	113: √	144: √
21: √	**52: X**	83: √	114: √	145: √
22: √	53: √	84: √	115: √	146: √
23: √	54: √	85: √	116: √	147: √
24: √	55: √	86: √	117: √	148: √
25: √	**56: X**	87: √	118: √	149: √
26: √	57: √	88: √	**119: X**	150: √
27: √	**58: X**	89: √	120: √	151: √
28: √	59: √	90: √	121: √	152: √
29: √	60: √	91: √	122: √	**153: X**
30: √	61: √	92: √	123: √	**154: X**
31: √	62: √	93: √	124: √	155: √

Table 8.4: Leave-one-out cross-validation on the whole data set

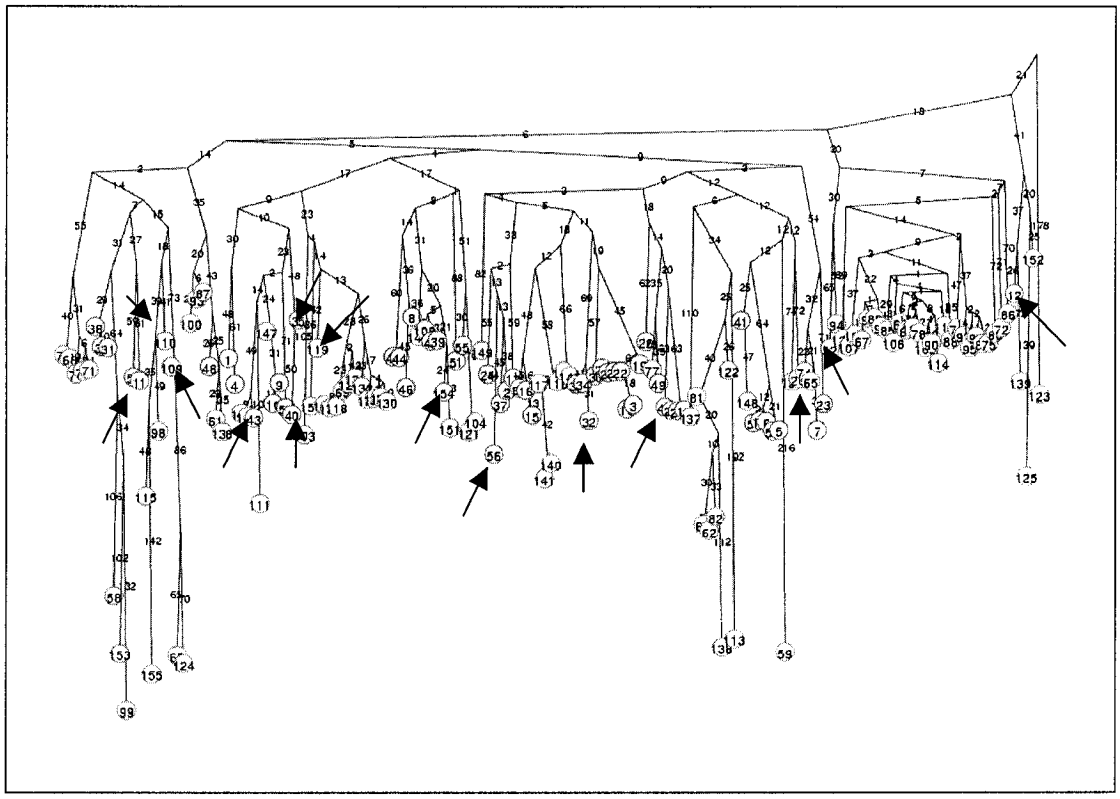

Figure 8.6: Misclassified examples in the whole data set

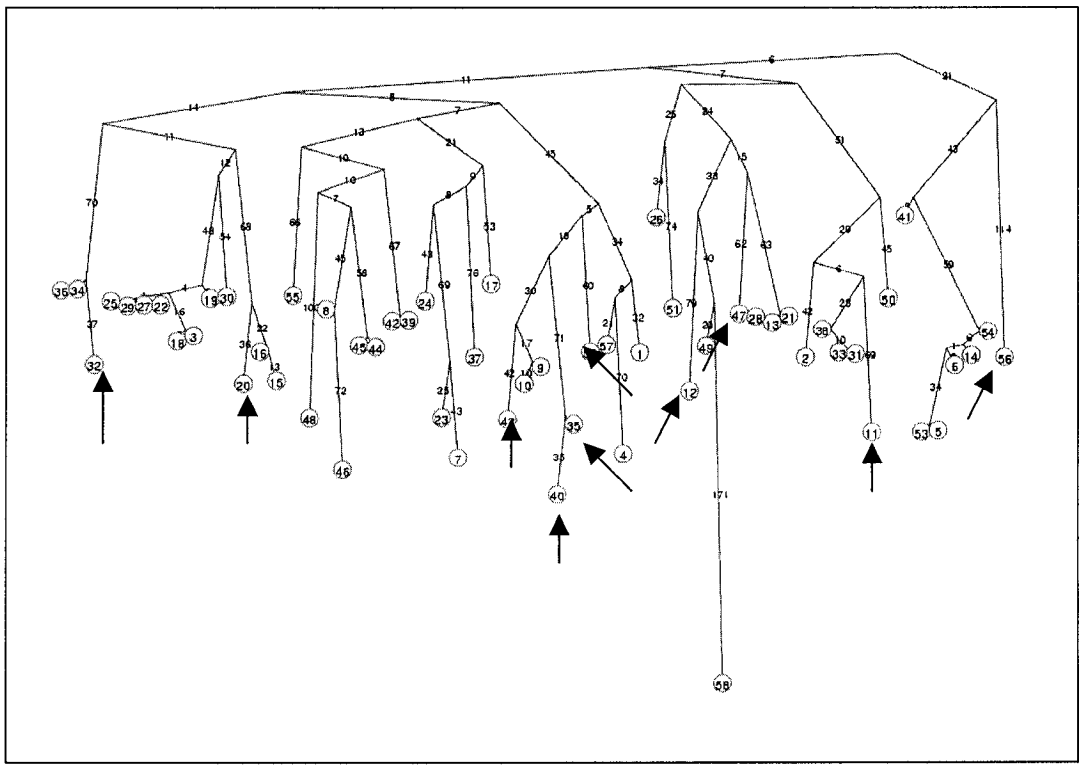

Figure 8.7: Misclassified examples in the positive data set

Examining the results in table 8.4, and Figures 8.6-7, we see that most of the misclassified examples are proteins that are relatively far evolutionarily from any other, or are somehow closer to proteins in the complementary set. Therefore, a technique based on evolutionary relationship such as ours, cannot possibly classify those correctly.

We try other implementations including using the similarity score, PAM distance and PAM variance of the best alignment (based on similarity) against the positive and negative sets, instead of using the average. We modify the scaling on the inputs accordingly. This gives worse results with a best prediction accuracy of 84% (standard deviation of 8%)[26]. We also try using the first 40 amino acids and the same encoding and structure as in Section 8.1, but we add the average similarity score, PAM distance, and PAM variance connected to a 41st hidden node. This results in an 806x41x1 structure. This experiment gives good accuracy with an 86% prediction rate (standard deviation of 8.7%)[26], but not as successful as using only these six measures alone.

8.4 Comparison of conjectures

In this section, we run leave-one-out cross-validation experiments for each conjecture (or input encoding) used in this chapter and compare the sensitivity, specificity, and overall prediction rate of the most relevant experiments. Results are shown in table 8.5.

The specificity indicates the number of sequences that will be predicted as Golgi resident despite being non-Golgi resident. It is defined as the number of true negatives (TN) divided by a sum of the number of true negatives and false positives (FP). Sensitivity indicates the amount of sequences that will be predicted as non-Golgi resident despite being Golgi resident. It is defined as the number of true positives divided by a sum of the number of true positives and false negatives. The overall prediction rate, expressed as a percentage, is computed as the number of misclassified examples (FP+FN) divided by total number of data examples. For all three measures, the range is (0.0,1.0) where a higher value represents a better conjecture.

Conjecture (input into the neural network)	**Overall prediction rate**	**Sensitivity** TP/(TP+FN)	**Specificity** TN/(TN+FP)
First 40 amino acids	76.1%	0.60	0.86
Type II membrane proteins set 1	75.3%	0.59	0.85
Type II membrane proteins set 2	75.3%	0.66	0.81
40 Transmembrane amino acids ("good" positives, all negatives)	73.9%	0.44	0.84
Average Similarity, PAM distance and PAM variance	90%	0.81	0.95
First 40 amino acids with average Similarity, PAM distance and variance	86%	0.79	0.91

Table 8.5: Comparison of some of the different conjectures used (data taken from leave-one-out cross-validation experiments)

"Data suggests that Golgi retention in many instances is not determined by a discrete and continuous sequence motif, but rather by disparate regions of the molecules" [20]. Our final results support this last statement, as our prediction system is able to predict (up to a

certain accuracy) whether a candidate test sequence it has never seen before is coding for a Golgi resident protein or not: the best prediction rate is achieved by looking at different measures of similarity over the whole protein (Average similarity, PAM distance, and variance). This does represent to a certain degree disparate regions of the molecule, whereas earlier experiments, trying to identify a discrete and continuous sequence motif, either failed or were unsuccessful in giving acceptable prediction accuracy.

9. Conclusions & Future work

We have seen that the localization of Golgi resident proteins is a complex and not yet completely understood process. There are a multitude of different localization mechanisms, and sometimes, combinations of these are required for properly directing a protein to its final location. Not only is this process complex, but the very identity of the Golgi as an independent organelle is still being debated. Yet, without differentiating between the different types of proteins in the data set, or the different compartments of the Golgi apparatus, we designed a neural network prediction system capable of successfully classifying (approximately 90% accuracy) whether a given candidate sequence codes for a Golgi resident protein or not.

The data set consists of amino acid sequences of 155 proteins, among which, 58 are actual Golgi resident proteins. The analysis of the data for any obvious patterns or different types of motifs within the sequences did not yield any results. We required a framework that would be robust to noise due to the nature of the data: amino acid sequences that are subject to mutations. Ideally, this system would be able to represent complex relationships in multivariate data by learning them from the data set. If the data set was sufficiently representative of the actual protein distributions within the Golgi, and the system set up in a proper way, we would get an accurate predictor. Thus we used an artificial neural network with a backpropagation learning algorithm. After multiple experiments trying to recognize signals within the actual amino acid sequences, and encoding those in different ways, our system was predicting at a relatively good accuracy (80%). But this approach was limited by the failure to recognize certain types of signals such as signal patches, and other complicated non-linear localization mechanisms. We found the best features to be the general measures available to represent evolutionary similarity of protein sequences. We showed that the evolutionary comparison of sequences was relevant for our problem, and allowed us to extract what may be the only features available for a general classification of proteins into the Golgi.

We found our results to be surprisingly good given the different biases we were faced with, both within the bioinformatics tools we used, and when choosing how to encode amino acid sequence.

One possible direction for future work would be to design a specific predictor for each type of Golgi resident protein: type I, type II, multiple membrane-spanning, soluble, and peripheral proteins. Each predictor would use its own set of features extracted from the amino acid sequence to make a prediction. The problem would be to generate training sets of sufficient size for each type. Our data set was too small to be divided in such a way.

Another approach would be to design a specific predictor for each type of known sorting signal, and group these together using a second level neural network on top of the first level predictors. The advantage of such a system is the possibility of recognizing proteins that require multiple sorting signals. The data set would need to be divided into different classes based on what is believed to be the localization mechanism for each protein. A problem that can be foreseen is again, gathering a data set that would yield large enough training subsets when divided. More importantly this approach would assume we know every localization mechanism at work in the Golgi. Since there exists controversy surrounding this organelle (Section 2.2), this is not likely to be the case. Therefore, this may prevent using an important part of the data set for which we know the localization but not the localization mechanism. This also implies that a candidate sequence that uses a different type of mechanism could not be predicted.

References

[1] Tom M. Mitchell (1997), *Machine Learning*, chap. 4&7.

[2] Michael C. Mozer, Robert Dodier, Cesar Buerra, Richard Wolniewicz, Lian Yan, Michael Colagrosso, David Grimes (2000), *Prediction and Classification: Pitfalls for the Unwary*, http://www.cs.colorado.edu/~mozer/papers/white-paper3.html.

[3] Pierre Baldi, Soren Brunak (1998), *Bioinformatics The Machine Learning Approach*, chap. 5&6.

[4] Bruce Alberts, Alexander Johnson, Julian Lewis, Martin Raff, Keith Roberts, Peter Walter (2002), *Molecular Biology of the Cell 4th edition*, chap. 12&13.

[5] Stephen Jose Hanson (1989), *Meiosis Networks*, Proceedings of NIPS 2, p.533-541.

[6] Mike Wynne-Jones (1991), *Node Splitting: a Constructive Algorithm for Feed-forward Neural Networks*, Proceedings of NIPS 4, p.1072-1079.

[8] G. H. Gonnet, M. T. Hallett, C. Korostensky, and L. Bernardin (2000), *Darwin v. 2.0: an interpreted computer language for the biosciences*, Bioinformatics, 16:101-103.

[9] Royston Goodacre, Mark J. Neal, Douglas B. Kell (1996), *Quantitative Analysis of Multivariate Data Using Artificial Neural Networks,* Zentralblatt für Bakteriologie, in the press.

[10] *A Quantitative Comparison of Three Design Algorithms for Feed-forward Neural Nets*, Adriana Dumitras, Adrian Traian Murgan.

[11] Dietterich, T. G. (2000), *An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization*, Machine Learning, 40(2):139-158.

[12] Michael P. Washburn, Dirk Wolters, John R. Yates (2001), *Large-Scale Analysis of the Yeast Proteome by Multidimensional Protein Identification Technology*, Nature Biotechnology, 19(3):242-247.

[13] Pavel A. Pevzner (2000), *Computational Molecular Biology: An Algorithmic Approach*, Chapter 11.

[15] Burkhard Rost (1996), *PHD: Predicting One-Dimensional Protein Structure by Profile Based Neural Networks*, Methods in Enzymology, 266:525-539.

[16] D.G. Kneller, F.E. Cohen, R. Langridge (1990), *Improvements in Protein Secondary Structure Prediction by an Enhanced Neural Network*, Journal of Molecular Biology, 214:171-182.

[17] Manuel G Claros, Soren Brunak, Gunnar von Heijne (1997), *Prediction of N-terminal protein sorting signals*, Current Opinion in Structural Biology, 7:394-398.

[18] Olof Emanuelsson, Henrik Nielsen, Soren Brunak, Gunnar von Heijne, *Prediction Subcellular Localization of Proteins Based on their N-terminal Amino Acid Sequence*, Journal of Molecular Biology, 300:1005-1016.

[19] Ying Xu, Edward C. Uberbacher (1998), *Computational Gene Prediction Using Neural Networks and Similarity Search*, Computational Methods in Molecular Biology, p.109-128.

[20] Paul A. Gleeson (1998), *Targeting of proteins to the Golgi apparatus*, Histochem Cell Biol, 109:517-532.

[21] Jacquelyn S. Fetrow, Michael J. Palumbo, George Berg (1997), *Patterns, Structures, and Amino Acid Frequencies in Structural Building Blocks, a Protein Secondary*

Structure Classification Scheme, PROTEINS : Structure, Function, and Genetics, 27:249-271.
[22] Erika Check (2002), *Will the real Golgi please stand up,* NATURE, 416:780-781.
[23] DAS transmembrane prediction system
http://www.sbc.su.se/~miklos/DAS/maindas.html
[24] HMMtop transmembrane prediction system
http://www.enzim.hu/hmmtop/submit.html
[25] Tmap transmembrane prediction system
http://130.237.130.31/tmap/single.html
[26] http://www.mcb.mcgill.ca/~yannick/thesis/
User name: golgi
Password: resident

APPENDIX A: THE DATA SET

#*	Protein description	Species
1	Polypeptide n-acetylgalactosaminyltransferase	homo sapiens
2	Golgi stacking protein homolog GRASP55	rattus norvegicus
3	CMP-N-acetylneuraminate-beta-galactosamide-alpha-2,6-sialyltransferase	rattus norvegicus
4	Polypeptide n-acetylgalactosaminyltransferase	mus musculus
5	Alpha-1,2-mannosidase	homo sapiens
6	Man9-mannosidase	sus scrofa
7	COPD_BOVIN Coatomer delta subunit	bos taurus
8	N-acetyllactosaminide beta-1,3-N-acetylglucosaminyltransferase	homo sapiens
9	P58	rattus norvegicus
10	ERGIC-53 protein precursor	homo sapiens
11	Endo-alpha-D-mannosidase	rattus norvegicus
12	Alpha-1,3-mannosyl-glycoprotein beta-1,2-N-acetylglucosaminyltrans.	rattus norvegicus
13	Alpha-2,8-sialyltransferase	mus musculus
14	Mannosyl-oligosaccharide 1,2-alpha-mannosidase IA	homo sapiens
15	Alpha-1,6-mannosyl-glycoprotein beta-1,2-N-acetylglucosaminyltrans.	sus scrofa
16	Alpha-1,6-mannosyl-glycoprotein beta-1,2-N-acetylglucosaminyltrans.	rattus norvegicus
17	ALPHA2,3-SIALYLTRANSFERASE ST3GAL VI	homo sapiens
18	CMP-N-acetylneuraminate-beta-galactosamide-alpha-2,6-sialyltransferase	mus musculus
19	Beta-1,4-galactosyltransferase 1	mus musculus
20	endomembrane protein emp70 precursor isolog	Homo sapiens
21	alpha-N-acetyl-neuraminnide alpha-2,8-sialyltransferase	rattus norvegicus
22	Beta-1,4-galactosyltransferase 1	Homo sapiens
23	Protein-tyrosine sulfotransferase 1	mus musculus
24	CMP-N-acetylneuraminate-beta-1,4-galactosamide-alpha-2,3-sialyltrans.	mus musculus
25	beta-1,4-galactosyltransferase	Homo sapiens
26	CMP-N-acetylneuraminate-beta-galactosamide-alpha-2,3-sialyltrans.	mus musculus
27	Beta-1,4-galactosyltransferase 1	Homo sapiens
28	CMP-NeuAc--GM3 alpha2-8 sialyltransferase	rattus norvegicus
29	galactosyltransferase-sialyltransferase hybrid protein	?
30	Beta-1,4-galactosyltransferase 5	mus musculus
31	Heparan sulfate 2-sulfotransferase	mus musculus
32	Cation-dependent mannose-6-phosphate receptor	mus musculus
33	Heparan sulfate 2-sulfotransferase	Cricetulus longicaud.
34	Beta-1,4-galactosyltransferase 7	homo sapiens
35	Major histocompatibility complex class I	rattus norvegicus
36	Beta-1,4-galactosyltransferase VII	homo sapiens
37	CMP-N-acetylneuraminate-beta-1,4-galactoside alpha-2,3-sialyltransferase	rattus norvegicus
38	Heparan sulfate 2-sulfotransferase	xenopus laevis
39	UDP-GALACTOSE:beta-n-acetyl-glucosamine-beta-1,3-galactosyltrans II	mus musculus
40	CMP-sialic acid transporter	mus musculus
41	CMP-NeuAC:(beta)-N-acetylgalactosaminide (alpha)2,6-sialyltransferase	homo sapiens
42	UDP-GALACTOSE:beta-n-acetyl-glucosamine-beta-1,3-galactosyltrans I	homo sapiens
43	Ectonucleoside triphosphate diphosphohydrolase 5 [precursor]	mus musculus
44	Beta-1,3-galactosyltransferase 4	mus musculus
45	Beta-1,3-galactosyltransferase 4	rattus norvegicus
46	Golgin-160	mus musculus
47	Golgi apparatus protein 1 [Precursor]	rattus norvegicus
48	Cis-golgi matrix protein GM130	rattus norvegicus
49	Alpha-mannosidase II	mus musculus
50	General vesicular transport factor p115	rattus norvegicus

51	Heparan sulfate N-deacetylase/N-sulfotransferase	rattus norvegicus
52	Trans-golgi network integral membrane protein TGN38 [Precursor]	rattus norvegicus
53	Mannosyl-oligosaccharide 1,2-alpha-mannosidase IB	mus musculus
54	Mannosyl-oligosaccharide 1,2-alpha-mannosidase IA	mus musculus
55	Beta-1,4 N-acetylgalactosaminyltransferase	rattus norvegicus
56	Galactosyltransferase associated protein kinase P58/GTA	rattus norvegicus
57	Vesicular integral-membrane protein VIP36 [Precursor]	Canis familiaris
58	Polypeptide n-acetylgalactosaminyltransferase	rattus norvegicus
59	Alpha enolase	Mus musculus
60	Elongation factor 1 alpha [Fragment]	Dryoxylon onoharae.
61	Actin, cytoplasmic 2	Hs,Mm,Rn,Bt,Tv,Aaa
62	Dimethylaniline monooxygenase [N-oxide forming] 5	Mus musculus
63	Protein disulfide isomerase A3 [Precursor]	Rattus norvegicus
64	Cytochrome P450 2C11	Rattus norvegicus
65	Fatty-acid amide hydrolase	Rattus norvegicus
66	Dimethylaniline monooxygenase [N-oxide forming] 5	Oryctolagus cunicul.
67	Cytochrome P450 2B3	Rattus norvegicus
68	Cytochrome P450 4A2 [Precursor]	Rattus norvegicus
69	glucuronosyltransferase (EC 2.4.1.17) Ugt1.6	Mus musculus
70	Cytochrome P450 4A3	Rattus norvegicus
71	Cytochrome P450 4F4	Rattus norvegicus
72	Fatty aldehyde dehydrogenase	Rattus norvegicus
73	Cytochrome P450 4F1	Rattus norvegicus
74	Cytochrome P450 51	Rattus norvegicus
75	Cytochrome P450 2D1	Rattus norvegicus
76	Cytochrome P450 4F5	Rattus norvegicus
77	Tapasin [Precursor]	Mus musculus
78	Cytochrome P450 2C7	Rattus norvegicus
79	Cytochrome P450 2J3	Rattus norvegicus
80	Cytochrome P450 1A2	Rattus norvegicus
81	Dimethylaniline monooxygenase [N-oxide forming] 3	Mus musculus
82	Dimethylaniline monooxygenase [N-oxide forming] 1	Rattus norvegicus
83	Cytochrome P450 2D5	Rattus norvegicus
84	Cytochrome P450 2C6	Rattus norvegicus
85	Cytochrome P450 2C23	Rattus norvegicus
86	Cytochrome P450 17	Rattus norvegicus
87	Cytochrome P450 3A2	Rattus norvegicus
88	Cytochrome P450 2E1	Rattus norvegicus
89	Cytochrome P450 2B1	Rattus norvegicus
90	Cytochrome P450 2C13, male-specific	Rattus norvegicus
91	Cytochrome P450 2B2	Rattus norvegicus
92	Cytochrome P450 2D2	Rattus norvegicus
93	Cytochrome P450 3A1	Rattus norvegicus
94	Cytochrome P450c21	Canis familiaris
95	Cytochrome P450 2D18	Rattus norvegicus
96	Cytochrome P450 2B12	Rattus norvegicus
97	Cytochrome P450 2D3	Rattus norvegicus
98	Epoxide hydrolase 1	Rattus norvegicus
99	L-gulonolactone oxidase	Rattus norvegicus
100	Cytochrome P450 3A18	Rattus norvegicus
101	Cytochrome P450 2C39	Mus musculus
102	Cytochrome P450 2C12, female-specific	Rattus norvegicus
103	Arylacetamide deacetylase	Rattus norvegicus

104	Oligosaccharyltransferase	Mus musculus
105	Cytochrome P450 2A1	Rattus norvegicus
106	Cytochrome P450 2C22	Rattus norvegicus
107	Cytochrome P450 2A2	Rattus norvegicus
108	7-dehydrocholesterol reductase	Rattus norvegicus
109	Delta-6 fatty acid desaturase	Mus musculus
110	Farnesyl-diphosphate farnesyltransferase	Rattus norvegicus
111	3 beta-hydroxysteroid dehydrogenase type III	Rattus norvegicus
112	Cytochrome P450 2C1 [Fragment]	Oryctolagus cunicul.
113	Protein transport protein Sec61 alpha subunit isoform 1	Canis familiaris
114	Metallothionein-I	Mus musculus
115	NADPH-cytochrome P450 reductase	Rattus norvegicus
116	Protein disulfide isomerase [Precursor]	Mus musculus
117	Protein disulfide isomerase [Precursor]	Rattus norvegicus
118	Rat 3-methylcholanthrene-inducible truncated UDP-glucuronosyltrans.	Rattus norvegicus
119	Calnexin [Precursor]	Rattus norvegicus
120	Dolichyl-diphosphooligosaccharide--glycosyltr. 67 kDa subunit[Precursor]	Rattus norvegicus
121	Dolichyl-diphosphooligosaccharide--glycosyltr. 48 kDa subunit[Precursor]	Homo sapiens
122	Corticosteroid 11-beta-dehydrogenase, isozyme 1	Rattus norvegicus
123	NADH-cytochrome b5 reductase	Rattus norvegicus
124	Cytochrome b5	Rattus norvegicus
125	Protein transport protein SEC61 beta subunit	Homo s. , Canis f.
126	Transmembrane 9 superfamily protein member 2 [Precursor]	Homo sapiens
127	UDP-glucuronosyltransferase	Rattus norvegicus
128	UDP-glucuronosyltransferase 2B3 precursor, microsomal	Rattus norvegicus
129	UDP-glucuronosyltransferase 1-1 precursor, microsomal	Rattus norvegicus
130	UDP-glucuronosyltransferase 2B2 precursor, microsomal	Rattus norvegicus
131	UDP-glucuronosyltransferase 1-5 precursor, microsomal	Rattus norvegicus
132	UDP GLUCURONOSYLTRANSFERASE [Fragment]	Rattus norvegicus
133	UDP-glucuronosyltransferase 2B6 precursor, microsomal	Rattus norvegicus
134	UDP-glucuronosyltransferase 2B1 precursor, microsomal	Rattus norvegicus
135	UDP-glucuronosyltransferase 2B12 precursor, microsomal	Rattus norvegicus
136	Long-chain-fatty-acid--CoA ligase, liver isozyme	Rattus norvegicus
137	Retinol dehydrogenase type I	Rattus norvegicus
138	Amine oxidase [flavin-containing] B	Rattus norvegicus
139	60 kDa heat shock protein, mitochondrial [Precursor]	Mus m. , Rattus n.
140	ATP synthase alpha chain, mitochondrial [Precursor]	Xenopus laevis
141	ATP synthase beta chain, mitochondrial [Precursor]	Mus musculus
142	NADH-ubiquinone oxidoreductase 49 kDa subunit, mitochon. [Precursor]	Homo sapiens
143	Glutamine synthetase	Mus musculus
144	Glutamine synthetase	Rattus norvegicus
145	Isocitrate dehydrogenase [NADP], mitochondrial [Precursor]	Bos taurus
146	Argininosuccinate synthase	Rattus norvegicus
147	Delta-1-pyrroline-5-carboxylate dehydrogenase, mitochondrial [Precursor]	Homo sapiens
148	ATP synthase alpha chain, mitochondrial [Precursor] [Fragment]	Rattus norvegicus
149	ATP synthase beta chain, mitochondrial [Precursor]	Rattus norvegicus
150	Voltage-dependent anion-selective channel protein 1	Mus musculus
151	Cytochrome c oxidase polypeptide II	Rattus norvegicus
152	Cytochrome c oxidase subunit IV isoform 1, mitochondrial [Precursor]	Rattus norvegicus
153	ATP synthase e chain, mitochondrial	Rattus norvegicus
154	PEX13	Cricetulus griseus
155	Uricase	Rattus norvegicus

*Protein number as defined in table 5.1

APPENDIX B: DOMAIN CANDIDATES

("short" best local alignments using Darwin)

```
Alignment of 2,111:
lengths=19,19 simil=73.7, PAM_dist=69.295, offsets=76969949,80071766, identity=57.9%
LLKANVEKPVKMLIYSSKT
||.|.||..|..:||||.|
LLDACVEASVPAFIYSSST

Alignment of 11,100:
lengths=14,14 simil=66.5, PAM_dist=92.8845, offsets=77015482,80060750, identity=50.0%
MGHSLPMFYIYDSY
!..|| !||||.!|
LATSLMLFYIYGTY

Alignment of 12,102:
lengths=14,13 simil=86.8, PAM_dist=55.2992, offsets=79987594,80062813, identity=71.4%
LLLLFFWTRPAPGR
||||!.| ||:|||
LLLLYLW_RPSPGR

Alignment of 51,84:
lengths=31,31 simil=81.6, PAM_dist=145.42, offsets=80015286,80044609, identity=32.3%
SSNYPSSETFEEIQFFNGHNYHKGIDWYMEF
|.:!|..|.|!. :|::|:...|..|!!|.|
SKEFPDPEIFDPGHFLDGNGKFKKSDYFMPF

Alignment of 52,51:
lengths=27,27 simil=64.1, PAM_dist=151.071, offsets=80014698,80011771, identity=29.6%
RGLEPSADASESDCGDPPPVAPSRLLP
|:|..::..::::...:.||!.||..||
RALPSASKPNNTSSENNPPIQPSTPLP

Alignment of 56,96:
lengths=18,18 simil=84.6, PAM_dist=86.3411, offsets=80018920,80056820, identity=55.6%
LSDQGFDLMNKFLTYYPG
||.| |!|.:.||.|!||
LSSQVFELYSGFLKYFPG

Alignment of 58,19:
lengths=15,15 simil=79.4, PAM_dist=64.6616, offsets=77686757,78801537, identity=66.7%
LGVTLVYYLSGRDLS
||!.|.|.|||.|||
LGIGLLYSLSGPDLS

Alignment of 60,69:
lengths=25,22 simil=69.5, PAM_dist=87.1791, offsets=80014287,80029016, identity=44.0%
TGEFEAGISKNGQTREHALLAFTLG
|.|||| !:.:|   ||.::.|!||
TQEFEAYVNASG___EHGIVVFSLG

Alignment of 65,149:
lengths=11,11 simil=77.4, PAM_dist=57.5646, offsets=80024940,80106402, identity=63.6%
MPSPAMRRALI
!|:|||!|||:
LPAPALRRALL

Alignment of 99,82:
lengths=16,16 simil=74.8, PAM_dist=61.2582, offsets=80042332,80059747, identity=62.5%
KRVLVVGMGNSGTDIA
|!|.||| |:|.!|||
KKVKVVGGGHSPSDIA

Alignment of 103,40:
lengths=49,49 simil=74.5, PAM_dist=171.981, offsets=79997754,80063594, identity=22.4%
GRFKASLSENVLGSPKELAKLSVPSLVYAVQNNMAFLALSNLDAAVYQV
|.:.|:::::!|.:|....||.|.:|!|.:.:.!.: :.|:.!.:.|.:
GNLTAAVTQQILQDPDVKIKLKVQALIYPALQALDMNVPSQQENSQYPL
```

```
Alignment of 111,89:
lengths=19,19 simil=89.6, PAM_dist=73.0096, offsets=80049622,80072009, identity=52.6%
QEILDYIGHIVEKHRATLD
|:. !:||.:||:||.|||
QKTSEWIGTLVEQHRETLD

Alignment of 113,123:
lengths=34,34 simil=68.4, PAM_dist=178.624, offsets=80073720,80083146, identity=23.5%
GLGSGISLFIATNICETIVWKAFSPTTVNTGRGM
||..|..!!!:|.| .:!|.!.!:|::.:..!|:
GLPIGQHIYLSTRIDGNLVIRPYTPVSSDDDKGF

Alignment of 114,69:
lengths=12,12 simil=69.3, PAM_dist=119.334, offsets=80029238,80066963, identity=58.3%
CCAYGCRKCFGG
||..||.|| .|
CCPVGCSKCAQG

Alignment of 115,98:
lengths=25,26 simil=87.7, PAM_dist=102.155, offsets=80058675,80075209, identity=46.2%
ELVLASLL_GFVIYWFVSRDKEETLP
!!|| ||: |.:.|||! |.|:|.:|
DMVLFSLIVGVLTYWFIFRKKKEEIP

Alignment of 119,50:
lengths=18,18 simil=77.9, PAM_dist=93.9508, offsets=80013542,80080267, identity=38.9%
EEEDESGDQEDDDDELDD
|||...||:|!!!!:|!!
EEEKNKGDEEEEEEKLEE

Alignment of 120,46:
lengths=48,48 simil=88.9, PAM_dist=125.495, offsets=80004752,80081699, identity=31.2%
SDLTSAQKEMKTKHKAYENAVSILSRRLQEALASKEATDAELNQLRAQ
|.|:|.:|.!:|:|||..:.!:!|..||:.. :.. .. :|!::|.||
STLNSGKKSLETEHKAVTSEIAVLQSRLKTEGSDLCDRVSEMQKLDAQ

Alignment of 125,64:
lengths=18,21 simil=61.9, PAM_dist=88.5788, offsets=80023574,79063632, identity=42.9%
PVLVLVLTL___SSLLLLSLW
||.|||!!|   :|::!|.:|
PVPVLVMSLLFIASVFMLHIW

Alignment of 139,6:
lengths=11,11 simil=88.2, PAM_dist=65.6683, offsets=79985275,80097549, identity=72.7%
GGLGGGLGGGL
|||!|||!|||!
GGMGGGMGGGM

Alignment of 147,6:
lengths=34,34 simil=66.8, PAM_dist=158.238, offsets=79985508,80104599, identity=32.4%
DALDTLFIMKMKNEFEEAKAWVEEHLNFNVNAEV
|.:::!| .:.|:||!:.|.|.|.:|  ::.|:|
DIFQAIFEKHYKTEFDKHKIWYEHRLIDDMVAQV

Alignment of 148,61:
lengths=18,18 simil=78.7, PAM_dist=78.6609, offsets=80020917,80105640, identity=50.0%
FLGMESCGIHETTFNSIM
|:||:|.||:||. .!|!
FIGMKSRGIYETPAGTIL

Alignment of 150,19:
lengths=9,9 simil=68.9, PAM_dist=37.1479, offsets=77686880,80107125, identity=77.8%
KKNPEIKTG
|||..||||
KKNAKIKTG

Alignment of 151,126:
lengths=35,35 simil=68.6, PAM_dist=136.33, offsets=80084841,80107457, identity=28.6%
NSLVIVLFLSGMVAMIMLRTLHKDIARYNQMDSTE
::|:||::!|.!|..|!...|...!::.:.||:.|
HTLMIVFLISSLVLYIISLMLTTKLTHTSTMDAQE
```

```
Alignment of 152,103:
lengths=36,36 simil=65.9, PAM_dist=221.518, offsets=80063409,80105200, identity=19.4%
FLLISVVLVAYYIYIPLPDDIEEPWKIILGNTLLKL
|..!.:!....|!| |:|:.:!:.|..:..:.!|:!
FTALVLIWEKSYVYGPIPHTFDRDWVAMQTKRMLDM

Alignment of 153,71:
lengths=6,6 simil=58.1, PAM_dist=41.3497, offsets=80030920,80103636, identity=83.3%
YSFLKP
||:|||
YSYLKP

Alignment of 154,19:
lengths=49,49 simil=91.7, PAM_dist=182.098, offsets=77686807,80108589, identity=32.7%
PPPPLGVSPKPRPGLDSSPGAASGPGLKSNLSSLPVPTTTGLLSLPACP
||||.. ..!. ||..|:||:::||:::|. .:...:.|..|..:|...|
PPPPKPWETRRIPGAGSGPGTGPGPAFQSADLGPTLLTRPGQPTLTRVP

Alignment of 155,3:
lengths=8,8 simil=70.0, PAM_dist=28.9482, offsets=79983922,80109430, identity=87.5%
FEKNMVKH
|||| |||
FEKNGVKH
```

APPENDIX C: PROPERTIES OF AMINO ACIDS

Amino Acid	3-Letter Code	1-Letter Code	Molecular Weight	Hydrophobicity*
Alanine	Ala	A	89.09	0.616
Cysteine	Cys	C	121.16	0.680
Aspartate	Asp	D	133.10	0.028
Glutamate	Glu	E	147.13	0.043
Phenylalanine	Phe	F	165.19	1.00
Glycine	Gly	G	75.07	0.501
Histidine	His	H	155.16	0.165
Isoleucine	Ile	I	131.18	0.943
Lysine	Lys	K	146.19	0.283
Leucine	Leu	L	131.18	0.943
Methionine	Met	M	149.21	0.738
Asparagine	Asn	N	132.12	0.236
Proline	Pro	P	115.13	0.711
Glutamine	Gln	Q	146.15	0.251
Arginine	Arg	R	174.20	0.000
Serine	Ser	S	105.09	0.359
Threonine	The	T	119.12	0.450
Valine	Val	V	117.15	0.825
Tryptophan	Trp	W	204.23	0.878
Tyrosine	Tyr	Y	181.19	0.880

* Scaled between 0 and 1.

APPENDIX D: PREDICTED TRANSMEMBRANE DOMAINS

"x: y (a->b)"
where "x" represent the data example number in this new data set, "y" represent the protein number in the original data set, "a" represents the starting index of the 40 amino acids around the transmembrane domain, and "b" the end index.

1: 1 (1->40)
2: 2 (1->40)
3: 3 (1->40)
4: 3 (306->345)
5: 4 (1->40)
6: 5 (27->66)
7: 5 (260->299)
8: 6 (40->79)
9: 6 (170->209)
10: 6 (280->319)
11: 7 (1->40)
12: 8 (1->40)
13: 9 (1->40)
14: 9 (471->510)
15: 10 (1->40)
16: 10 (466->505)
17: 11 (1->40)
18: 11 (160->199)
19: 12 (1->40)
20: 12 (190->229)
21: 12 (253->292)
22: 13 (1->40)
23: 13 (218->257)
24: 13 (286->325)
25: 14 (30->69)
26: 14 (271->310)
27: 15 (1->40)
28: 15 (146->185)
29: 16 (1->40)
30: 16 (142->181)
31: 17 (1->40)
32: 17 (247->286)
33: 18 (1->40)
34: 18 (308->347)
35: 19 (11->50)
36: 19 (130->169)
37: 20 (1->40)
38: 20 (209->258)
39: 20 (281->320)
40: 20 (316->355)
41: 20 (341->380)
42: 20 (436->475)
43: 20 (470->509)
44: 20 (501->540)
45: 20 (541->580)
46: 21 (1->40)
47: 21 (220->259)
48: 22 (11->50)
49: 23 (1->40)
50: 24 (1->40)
51: 24 (176->215)
52: 24 (250->289)
53: 25 (1->40)
54: 25 (25->64)
55: 26 (1->40)
56: 26 (251->290)
57: 27 (1->40)
58: 28 (1->40)
59: 28 (220->259)
60: 29 (11->50)
61: 29 (670->709)
62: 30 (11->50)
63: 31 (1->40)
64: 32 (1->40)
65: 32 (180->219)
66: 33 (1->40)
67: 34 (21->60)
68: 35 (1->40)
69: 35 (302->341)
70: 36 (21->60)
71: 37 (1->40)
72: 38 (1->40)
73: 39 (1->40)
74: 39 (61->100)
75: 40 (1->40)
76: 40 (36->75)
77: 40 (126->165)
78: 40 (161->200)
79: 40 (201->240)
80: 40 (235->274)
81: 40 (274->313)
82: 41 (1->40)
83: 41 (200->239)
84: 42 (1->40)
85: 42 (60->99)
86: 43 (1->40)
87: 44 (1->40)
88: 44 (60->99)
89: 44 (250->289)
90: 45 (1->40)
91: 45 (56->95)
92: 45 (250->289)
93: 46 (1->40)
94: 47 (1->40)
95: 47 (1130->1169)
96: 48 (1->40)
97: 48 (70->109)
98: 49 (1->40)
99: 50 (270->309)
100: 50 (300->339)
101: 50 (360->399)
102: 50 (480->519)
103: 50 (521->560)
104: 51 (1->40)
105: 51 (70->109)
106: 51 (140->179)
107: 51 (281->320)
108: 51 (501->540)
109: 52 (1->40)
110: 52 (300->339)
111: 53 (21->60)
112: 53 (253->292)
113: 54 (31->70)
114: 54 (266->305)
115: 55 (1->40)
116: 56 (1->40)
117: 57 (1->40)
118: 57 (310->349)
119: 58 (1->40)
120: 58 (71->110)
121: 58 (90->129)
122: 59 (101->140)
123: 59 (366->405)
124: 60 (46->85)
125: 60 (106->145)
126: 60 (192->231)
127: 61 (121->160)
128: 61 (336->375)
129: 62 (315->354)
130: 62 (355->394)
131: 62 (493->532)
132: 63 (1->40)
133: 63 (30->69)
134: 64 (1->40)
135: 64 (300->339)
136: 64 (425->464)
137: 65 (1->40)
138: 65 (396->435)
139: 66 (355->404)
140: 66 (493->532)
141: 67 (1->40)
142: 67 (160->199)
143: 67 (280->319)
144: 67 (431->460)

145: 68 (1->40)
146: 68 (101->140)
147: 68 (385->424)
148: 68 (450->489)
149: 69 (1->40)
150: 69 (285->214)
151: 69 (481->520)
152: 70 (1->40)
153: 70 (105->144)
154: 70 (396->435)
155: 70 (451->490)
156: 71 (1->40)
157: 71 (96->135)
158: 71 (391->430)
159: 71 (461->500)
160: 72 (95->134)
161: 72 (445->484)
162: 73 (1->40)
163: 73 (90->129)
164: 73 (401->440)
165: 73 (461->500)
166: 74 (14->53)
167: 74 (190->229)
168: 74 (295->334)
169: 75 (1->40)
170: 75 (170->209)
171: 75 (295->334)
172: 75 (435->474)
173: 76 (1->40)
174: 76 (80->119)
175: 76 (401->440)
176: 76 (470->509)
177: 77 (1->40)
178: 77 (127->166)
179: 77 (410->449)
180: 78 (1->40)
181: 78 (291->330)
182: 78 (430->469)
183: 79 (1->40)
184: 79 (61->100)
185: 79 (305->344)
186: 79 (450->489)
187: 80 (1->40)
188: 80 (66->105)
189: 80 (306->345)
190: 80 (446->485)
191: 81 (350->389)
192: 81 (495->534)
193: 82 (130->169)
194: 82 (350->389)
195: 82 (494->533)
196: 83 (1->40)
197: 83 (170->209)
198: 83 (295->334)
199: 83 (450->489)
200: 84 (1->40)
201: 84 (300->339)
202: 84 (426->465)
203: 85 (1->40)
204: 85 (290->329)
205: 85 (430->469)
206: 86 (1->40)
207: 86 (155->194)
208: 86 (285->324)
209: 86 (426->465)
210: 87 (1->40)
211: 87 (205->244)
212: 87 (285->324)
213: 88 (1->40)
214: 88 (431->470)
215: 89 (1->40)
216: 89 (156->195)
217: 89 (275->314)
218: 89 (431->470)
219: 90 (1->40)
220: 90 (201->239)
221: 90 (280->319)
222: 90 (426->465)
223: 91 (1->40)
224: 91 (155->194)
225: 91 (276->315)
226: 91 (430->469)
227: 92 (1->40)
228: 92 (210->249)
229: 92 (295->334)
230: 92 (445->484)
231: 93 (1->40)
232: 93 (205->244)
233: 93 (291->330)
234: 94 (1->40)
235: 94 (151->190)
236: 94 (426->465)
237: 95 (1->40)
238: 95 (60->99)
239: 95 (171->210)
240: 95 (216->255)
241: 95 (295->334)
242: 95 (436->475)
243: 96 (1->40)
244: 96 (275->314)
245: 96 (431->470)
246: 97 (1->40)
247: 97 (180->219)
248: 97 (206->245)
249: 97 (296->235)
250: 97 (435->474)
251: 98 (1->40)
252: 98 (340->379)
253: 99 (110->149)
254: 99 (156->195)
255: 99 (245->284)
256: 100 (1->40)
257: 100 (210->249)
258: 100 (285->224)
259: 100 (436->475)
260: 101 (1->40)
261: 101 (300->339)
262: 101 (230->269)
263: 102 (1->40)
264: 102 (280->319)
265: 102 (426->465)
266: 103 (1->40)
267: 103 (91->130)
268: 104 (1->40)
269: 104 (402->441)
270: 105 (1->40)
271: 105 (285->324)
272: 105 (426->465)
273: 106 (1->40)
274: 106 (156->195)
275: 106 (276->315)
276: 106 (425->464)
277: 107 (1->40)
278: 107 (280->319)
279: 107 (426->465)
280: 108 (21->60)
281: 108 (85->124)
282: 108 (147->186)
283: 108 (242->281)
284: 108 (320->359)
285: 108 (405->444)
286: 109 (130->169)
287: 109 (255->294)
288: 109 (295->334)
289: 110 (51->90)
290: 110 (161->200)
291: 110 (275->314)
292: 110 (377->416)
293: 111 (60->99)
294: 111 (285->324)
295: 112 (421->460)
296: 113 (21->60)
297: 113 (66->105)
298: 113 (106->145)
299: 113 (135->174)
300: 113 (161->200)
301: 113 (195->234)
302: 113 (231->270)
303: 113 (346->385)
304: 113 (416->455)
305: 114 (1->40)
306: 115 (11->50)
307: 116 (1->40)
308: 117 (1->40)
309: 118 (1->40)
310: 118 (291->330)
311: 119 (1->40)
312: 119 (411->450)
313: 119 (475->514)
314: 120 (1->40)
315: 120 (425->454)

316: 121 (11->50)
317: 121 (417->456)
318: 122 (1->40)
319: 122 (136->175)
320: 122 (196->235)
321: 123 (1->40)
322: 124 (94->133)
323: 125 (56->95)
324: 126 (1->40)
325: 126 (295->334)
326: 126 (360->399)
327: 126 (391->430)
328: 126 (425->464)
329: 126 (460->499)
330: 126 (515->554)
331: 126 (547->586)
332: 126 (581->620)
333: 126 (620->659)
334: 127 (1->40)
335: 127 (290->329)
336: 127 (476->515)
337: 128 (1->40)
338: 128 (190->229)
339: 128 (300->339)
340: 128 (485->524)
341: 129 (1->40)
342: 129 (273->312)
343: 129 (481->520)
344: 130 (1->40)
345: 130 (162->201)
346: 130 (300->339)
347: 130 (485->524)
348: 131 (1->40)
349: 131 (146->185)
350: 131 (270->309)
351: 131 (477->516)
352: 132 (1->40)
353: 133 (1->40)
354: 133 (191->230)
355: 133 (300->339)
356: 133 (485->524)
357: 134 (1->40)
358: 134 (300->339)
359: 134 (390->429)
360: 134 (484->523)
361: 135 (1->40)
362: 135 (485->524)
363: 136 (11->50)
364: 136 (170->209)
365: 136 (225->464)
366: 136 (580->619)
367: 137 (1->40)
368: 137 (124->163)
369: 138 (480->519)
370: 139 (419->458)
371: 140 (1->40)
372: 141 (1->40)
373: 142 (50->89)
374: 142 (185->224)
375: 143 (50->89)
376: 143 (185->224)
377: 144 (424->463)
378: 145 (1->40)
379: 146 (1->40)
380: 147 (310->349)
381: 148 (1->40)
382: 149 (189->234)
383: 150 (1->40)
384: 151 (16->55)
385: 151 (55->94)
386: 152 (95->134)
387: 153 (1->40)
388: 154 (165->204)
389: 154 (226->265)
390: 155 (1->40)

APPENDIX E: AVERAGE SIMILARITY SCORE, PAM DISTANCE, AND PAM VARIANCE AGAINST POSITIVE AND NEGATIVE DATA SETS

Protein number	average Similarity score against positive examples	average Similarity score against negative examples	average PAM distance against positive examples	average PAM distance against negative examples	average PAM variance against positive examples	average PAM variance against negative examples
1	74.073475	47.599768	239.998283	233.361526	3398.492752	3652.811054
2	48.734857	46.434791	244.471172	224.452117	3473.227087	3685.460425
3	200.821406	45.660555	219.710330	248.060425	3327.592907	4093.205026
4	74.804655	46.912512	246.045009	245.371455	3315.662192	3419.673368
5	365.929896	49.621727	227.963204	244.322917	2878.653955	3192.925250
6	389.605258	50.120134	218.145673	232.983985	2690.191408	3236.101254
7	48.766320	44.821857	239.669698	242.133855	2996.676544	3744.956660
8	51.393063	49.957234	216.926531	231.752036	3006.748314	3344.694827
9	150.505049	47.303188	221.709405	241.898779	2978.732564	3237.629020
10	150.625291	47.468643	221.091175	231.124326	2935.260389	3345.186437
11	48.710264	46.368035	243.405311	256.209112	3352.925409	3885.221548
12	48.971226	48.853335	240.746247	237.176103	3493.246561	3783.966867
13	224.568454	48.517083	235.072545	251.704194	3030.581909	3499.446651
14	388.127923	49.039955	206.589113	244.731345	2957.227120	3391.830568
15	127.029065	47.859931	242.335038	233.426379	3494.143045	3659.522182
16	126.619904	47.789094	233.650918	241.967373	3516.333173	3959.813253
17	89.544045	48.591853	230.801013	225.489460	3718.849146	3623.202220
18	197.697034	46.895781	227.816420	242.663378	3449.969901	3795.888464
19	325.926569	47.526414	213.299688	222.958226	3106.082516	3583.715951
20	50.223638	57.697203	231.803608	236.793013	3480.745929	3642.944648
21	230.590157	48.061558	235.685657	253.057713	3116.895077	3527.745741
22	371.422610	45.654052	209.341689	241.375514	2934.183715	3891.385716
23	47.833011	46.598561	223.762625	233.314841	3286.791750	3647.493684
24	89.417442	45.755103	222.241425	248.900860	3024.066991	3890.879272
25	314.848389	45.198001	203.766139	243.166612	3468.504892	3953.933194
26	84.124089	46.984935	210.538848	225.422568	3158.217190	3915.969218
27	373.224094	45.266566	200.608543	246.117942	2939.383090	3932.151643
28	230.649896	48.044894	229.863983	253.724030	2889.355446	3598.688182
29	507.999446	48.917770	181.485147	247.374941	1970.253010	3514.079986
30	137.641617	47.161547	222.534246	230.711229	3265.872447	3791.765569
31	188.082799	47.738040	229.765826	245.664768	3553.891811	3480.264314
32	45.470291	43.499967	241.354906	249.620799	4163.384419	4369.876933
33	188.764316	47.558886	230.594855	240.971169	3488.761389	3481.929360
34	162.792642	46.767647	207.739511	242.570086	2952.837854	3961.384271
35	46.756342	44.918298	226.758424	236.830831	3965.517749	4109.545265
36	162.130715	46.844842	212.361803	239.055444	2952.058497	3921.329755
37	91.481048	47.480575	205.885811	244.959980	2910.236520	3678.714550
38	177.430167	47.335468	231.236649	245.086284	3377.623767	3644.822145
39	134.914187	45.342553	227.999700	252.860424	3630.391256	4006.966023
40	45.659094	44.762077	235.479612	235.021107	3841.591338	4250.555479
41	75.382006	44.804611	231.992126	248.924296	3159.055790	4107.047402
42	134.852635	45.311371	229.139832	253.334426	3637.663760	4037.056149
43	46.627042	49.203661	240.078704	236.296385	3810.882123	3315.826882
44	131.810117	48.399177	226.606642	239.891492	3586.185098	3901.445755
45	133.080893	47.881978	206.276318	238.726473	3478.901765	3847.925030
46	62.565264	52.613810	221.085255	235.634803	2313.766182	2614.864108
47	53.002949	52.319350	233.181325	235.584408	2991.914705	2708.338950
48	58.447418	48.619086	231.228891	238.522855	2545.483181	2952.104627
49	53.221630	52.630513	256.252482	241.414105	2888.783145	2833.901107
50	56.625608	50.875887	233.745899	261.651412	2748.839182	3061.283674
51	54.295158	53.353759	254.722153	232.726246	2923.576471	2968.578070
52	46.003405	43.511021	237.729687	225.958579	4009.146414	3814.698027

53	363.449657	49.381345	230.857898	229.903580	3183.719516	3232.654541
54	395.782034	49.452192	226.876579	241.892952	3017.251960	3429.256107
55	48.954298	49.629987	242.279350	224.643619	3198.666050	3123.852087
56	46.718841	49.229904	245.140041	238.825862	3447.787137	3410.972202
57	66.625628	47.512048	222.638776	227.777775	3743.470615	3778.616425
58	44.518622	42.104325	221.756679	235.991679	3938.624068	4433.543084
59	48.348535	47.606710	243.238768	242.889709	3379.085464	3616.066568
60	43.590907	43.929461	234.686946	230.048693	4630.021116	4334.571883
61	45.251389	47.924770	240.611975	241.830434	3797.845470	3501.756220
62	47.175131	148.092781	262.021009	227.185743	3726.333570	3142.426200
63	47.685311	65.838043	251.829389	245.318043	3377.664126	3254.924985
64	50.558040	579.795848	212.223917	195.295390	2819.290141	2110.582133
65	47.989041	49.947691	235.967642	225.629302	3604.641107	3220.628127
66	48.705664	149.595887	236.909234	223.847921	3227.324949	2965.329088
67	51.339857	493.548109	231.346572	204.350345	2845.326393	2199.785717
68	49.787923	232.994772	219.486614	211.428607	3096.553824	2567.063229
69	49.349846	329.498111	261.491511	219.802848	3295.842289	2976.567977
70	48.962903	232.230394	223.697691	218.018387	3315.538281	2455.641124
71	48.171214	236.361018	262.465853	210.368507	3411.233025	2341.153613
72	45.788586	51.175026	258.027865	242.431961	3746.740957	2890.648836
73	47.530337	233.830273	258.813794	230.874559	3882.613285	2436.394132
74	49.406645	110.386616	231.667791	231.192380	3173.289408	2268.509144
75	47.627995	484.002238	233.318627	194.247165	3411.101779	2416.901940
76	47.781748	225.241457	250.111640	222.740189	3807.018836	2575.199472
77	49.208347	47.473224	226.297593	235.822693	3551.368483	3892.845772
78	48.534213	582.881510	236.670738	195.634805	3298.400627	2146.617871
79	49.224877	395.124142	238.645806	210.463086	3071.138718	2292.625658
80	46.518114	233.636490	254.811576	204.573831	3751.917466	2322.736206
81	47.149866	127.440456	265.213755	228.687657	4082.997639	3383.257455
82	48.999356	124.229582	250.116359	236.745469	3586.781923	3149.965986
83	47.227011	484.586934	238.207214	197.455023	3551.483431	2493.263804
84	49.152166	585.537497	237.622433	180.182763	3206.013466	2102.451834
85	48.461179	529.463080	237.145891	206.043658	3350.601786	2187.823258
86	45.756318	228.884914	253.720180	213.332397	3830.121908	2807.633087
87	48.140947	252.663378	255.017634	226.590736	3601.833696	2322.700396
88	50.724197	484.652926	220.478565	208.701323	2932.254900	2313.373907
89	48.481156	565.509342	237.473612	187.855389	3173.094495	2422.652866
90	49.888050	550.826957	231.598439	205.635808	3393.820757	2363.521192
91	48.859934	564.730826	224.284090	187.914781	3167.243114	2297.469121
92	48.660540	468.904866	249.061245	197.372320	3248.320847	2328.899896
93	48.729698	247.435922	252.336131	233.736826	3622.386108	2424.033075
94	48.340816	194.802362	220.480013	220.401897	3397.414978	2488.382363
95	50.176104	453.160462	245.195281	199.281414	3172.844651	2305.518815
96	48.804608	524.452350	232.767320	194.851582	2948.436439	2284.536689
97	46.224707	481.783075	254.475171	196.377440	3578.378338	2432.558488
98	47.253632	48.135832	244.358814	240.078588	4040.587335	3710.097638
99	50.552069	50.174008	236.925446	237.610168	3376.013253	3284.000004
100	49.764691	223.472584	248.523213	226.655251	3295.041980	2547.966421
101	50.021466	596.861252	224.693953	192.668594	3132.016207	2152.600160
102	49.767053	536.875774	219.098794	195.949424	3180.092889	2303.297815
103	47.489821	47.070903	236.098645	250.486821	3662.233485	3498.990380
104	45.328141	99.429669	247.450252	227.697471	3892.304365	3267.626802
105	49.908084	472.855236	238.572562	203.086346	2957.263849	2482.306737
106	50.702772	508.538057	226.652682	205.905353	3123.792922	2398.676742
107	49.527577	450.139497	232.330034	204.920406	3366.879893	2235.941113
108	50.322266	48.684705	230.206055	238.645442	3740.962045	4022.212863
109	50.015017	49.665612	245.747957	242.596339	3317.622741	3706.699733
110	49.937208	47.532559	219.764686	238.013273	3180.925211	3529.384883
111	50.208974	49.384274	246.404859	228.845757	3615.750666	3675.075232
112	52.482402	565.912285	223.466487	183.958743	2691.439594	2311.121078
113	45.901879	48.305229	245.164365	254.139030	3783.593028	3786.805679
114	33.227528	32.633955	206.999677	211.274846	9606.571005	9897.600327
115	48.414030	51.182766	252.341366	229.065488	3291.762126	3157.714501

116	47.725085	117.173326	242.033608	241.250536	3670.737957	3448.537609
117	48.571260	117.466679	237.779229	242.519816	3587.938102	3516.964242
118	47.355465	278.500429	259.270815	213.967872	3389.169229	2992.491656
119	50.558782	50.191596	234.375589	219.857240	3228.117079	3311.782875
120	50.600646	48.639516	241.453312	256.080694	3082.057592	3321.083820
121	45.717568	98.907128	245.596577	234.670617	3658.929694	3405.656068
122	41.966896	48.322618	256.626992	223.724817	4105.828874	3622.536359
123	44.438279	48.862317	244.060455	220.587090	3883.132322	3531.727561
124	40.428569	42.951350	235.936415	234.815688	5173.602148	4899.954325
125	39.343600	40.025550	220.259139	219.425461	5902.128818	5937.494722
126	64.880220	51.574920	239.438335	243.025901	3512.436258	3036.084749
127	48.092727	345.603289	256.369829	217.240722	3298.320870	2875.782712
128	48.171289	314.567837	246.949397	223.824819	3519.223092	3143.772446
129	47.429519	266.019926	235.803311	225.396204	3472.372803	2877.595134
130	48.429805	302.907446	245.414214	226.545746	3196.021755	2904.254681
131	48.060714	269.958362	241.650101	219.718778	3547.953436	2839.464114
132	47.023102	185.242446	248.590001	218.340013	3610.263213	3211.606812
133	48.153569	313.413450	249.669846	226.355424	3690.320026	3122.283995
134	49.015496	253.814221	257.610592	220.452310	3321.441175	2865.033835
135	48.770736	296.545994	250.122340	216.662644	3415.281970	3127.129295
136	48.989356	49.874091	242.343013	244.543714	3563.105206	3305.149286
137	45.938242	47.392949	238.219245	247.429490	3782.953222	3818.280662
138	48.806224	50.694119	265.168536	240.749470	3625.549321	3391.614010
139	48.837895	46.738823	231.855378	240.929159	3356.867035	3314.722792
140	47.906513	108.648094	220.853498	226.060978	2754.368873	3296.978388
141	46.350531	109.511798	241.801092	224.555694	3324.377177	3253.564153
142	47.495731	116.738310	244.590311	225.519945	3316.578369	2990.766278
143	47.311779	117.287838	240.968963	223.529396	3282.155913	2935.872549
144	47.586318	49.546635	229.824531	254.827295	3576.292531	3569.121830
145	47.233634	91.015289	240.327476	248.348506	3772.662614	4087.577766
146	46.987136	92.436365	245.629253	234.829700	4004.729027	3583.784528
147	46.376370	46.294379	242.277314	243.280971	3727.183317	3801.653058
148	44.995728	45.894231	246.408347	222.591256	4000.136643	3839.397346
149	47.893964	49.871405	229.599489	240.075721	3274.915064	3599.917061
150	44.783344	43.907188	226.934765	236.130576	4186.743418	4534.393477
151	45.034897	45.901417	219.545475	221.703801	4126.147280	4050.391275
152	39.948817	42.179853	252.972182	248.897299	5791.411310	5121.918748
153	38.928428	35.341034	199.564502	214.091458	4856.602268	6037.207077
154	50.647880	46.139871	243.705076	239.703644	3389.454527	3736.841386
155	45.613110	44.982016	240.833739	245.261844	3775.821798	3978.721069

APPENDIX F: PROTEINNET USER MANUAL

The user starts the program by double clicking on the file ProteiNNet.exe. Figure F.1 shows the opening dialog of the program. This dialog offers three options to the user, accessible through the following buttons: *Train Mode*, *Test Mode*, *Results Mode*, and *Predict Mode.*

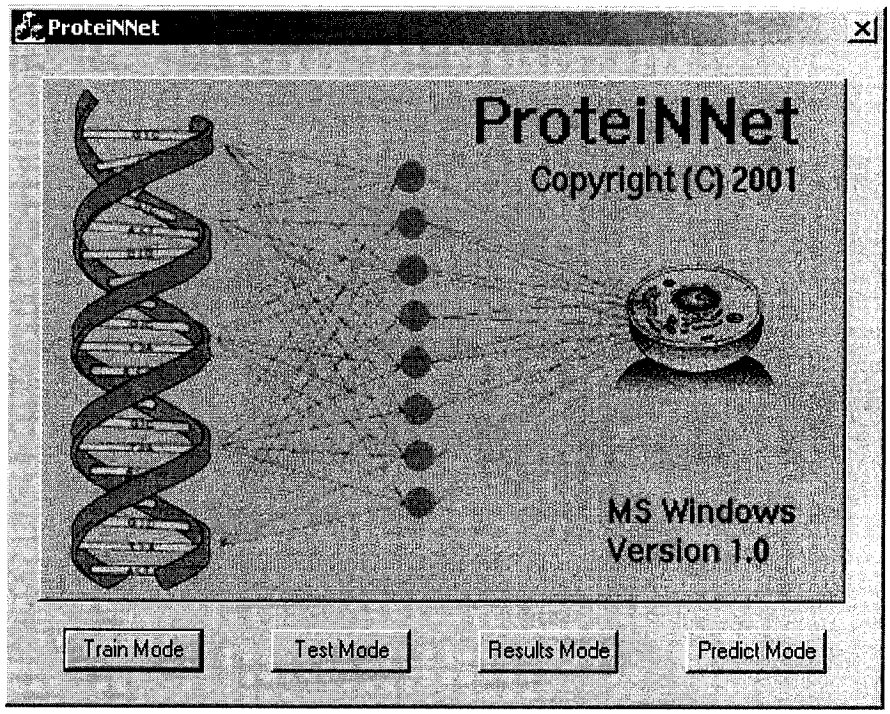

Figure F.1: start dialog

F.1 Train mode

This Mode enables the user to train a new neural network or continue training an existing one. The following attributes must be specified (see Figure F.2):

- *Use existing Neural Net*: If checked, the user needs to enter the name of a neural network file (.net), manually or by using the *browse...* button. This option is checked when the user wants to continue training an existing neural network

- *Load training data*: The user is required to enter the name of a data file (.dat, see section F.2) that will be used to train the network; this can be done manually or by using the *browse...* button.
- *Learning rate*: The learning algorithm uses this parameter as the step size to update the connection weights in the network (floating point value between 0 and 1).
- *Momentum*: This parameter allows the user to add momentum to the weight update to achieve higher learning speed (floating point value between 0 and 1).
- *Number of epochs*: The number of times that the whole training set will be presented to the network (unsigned integer value greater than 0).
- *Training mode*: The learning algorithm uses this parameter to decide when to update the connection weights in the neural network.
    - *Online*: Update after every example (default).
    - *Batch*: Update after every epoch.
- *Shuffle data*: This is optional. If checked, the user will have the choice between having the training data shuffled
    - *Once*: before training starts (default).
    - *Before every epoch.*
- *Test Set*: This option is dependent on the percentage of training examples chosen to be in the test set. By default, it is disabled since the percentage is set to a default value of 0%. The user must specify the relative size of the test set using a drop-down list where values are between 0 and 100, in increments of 5%.
    - When the *Amount of training data to use* is larger than 0%, the user can choose the *cross-validation option*. The basic idea is that we take a certain portion of the data (determined by the size of the test set) and test the performance of the network on that set. We do this repeatedly using many different subsets, so that in the end, every data example has been part of the validation set at some point, and part of the training set at some other point. The number of runs depends on the size of the test set.

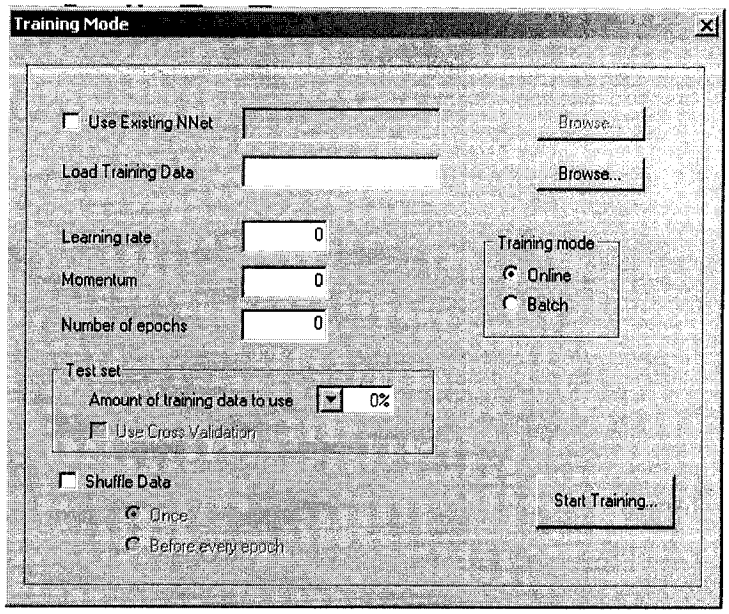

Figure F.2 : the Train Mode dialog

After pressing the *Start training...* button, the window becomes idle (the *Start training...* button becomes disabled). Once the training is finished the user is presented with the results in a new dialog window (Figure F.3). The result of the training is displayed as curves of error measures for the training and, if available, the test data and each cross validation set. The choices at this point are:

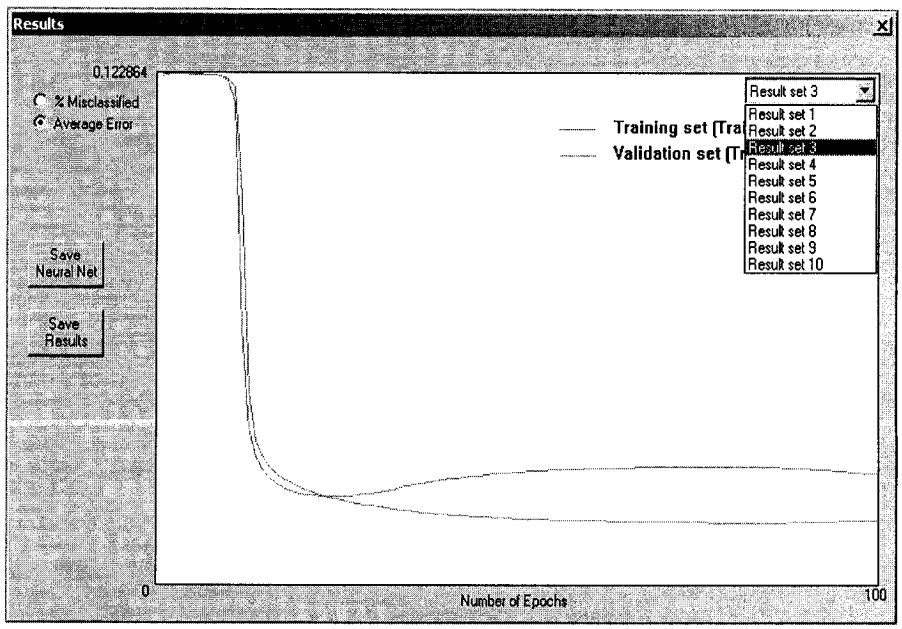

Figure F.3 : Results dialog

- Error measure:
    - *% Misclassified*: The percentage of misclassified examples (y-axis) versus the number of epochs (x-axis).
    - *Average Error*: the mean squared error on the output versus the target output. Here, the application plots two curves (if test set is greater than 0%) along with the legend:
        - Average error on the training set (y-axis) versus the number of epochs (x-axis).
        - Average error on the test set (y-axis) versus the number of epochs (x-axis).
- *Result set*: Only enabled if cross validation is enabled, this drop-down list in the top right hand corner allows to switch from one cross-validation result file to another.
- *Save results*: This button allows the user to save the results from his experiment in a .res file.
- *Save Neural Net*: This button allows the user to save the trained neural network to a .net file.

F.2 Test mode

This mode enables the user to test a previously trained neural network. The user must specify a neural network file to use (.net) to test its prediction accuracy; this can be done manually or by using the *browse...* button. The user must also specify the data file (.dat) that will be input into the network; this can be done manually or by using the *browse...* button. This data file must have the same format as the data file used to train the network: the list of features (Average Similarity against the positive set, Average Similarity against the negative set, Average PAM distance against the positive set, Average PAM distance against the negative set, Average PAM variance against the positive set, Average PAM variance against the negative set) separated by spaces, and followed by a space and a class label (1 for Golgi resident, and 0 for non-Golgi resident), where each line represents

a data example (note: The last data example in the file must have a carriage return character after its class label).

The user must also specify the following:

- *Classification threshold*: This drop down list enables the user to choose between different threshold values when classifying a data example, where a lower value represents a stricter classification (more restrictive). By default the classification threshold is set to the least restrictive value (0.5).

After pressing the *Test!* button, the performance of neural network on that test set is displayed in the *results* text box (not editable by the user), in terms of the Mean Squared Error, and the prediction accuracy (percentage of correctly classified examples). The list of misclassified example is also displayed in the result box.

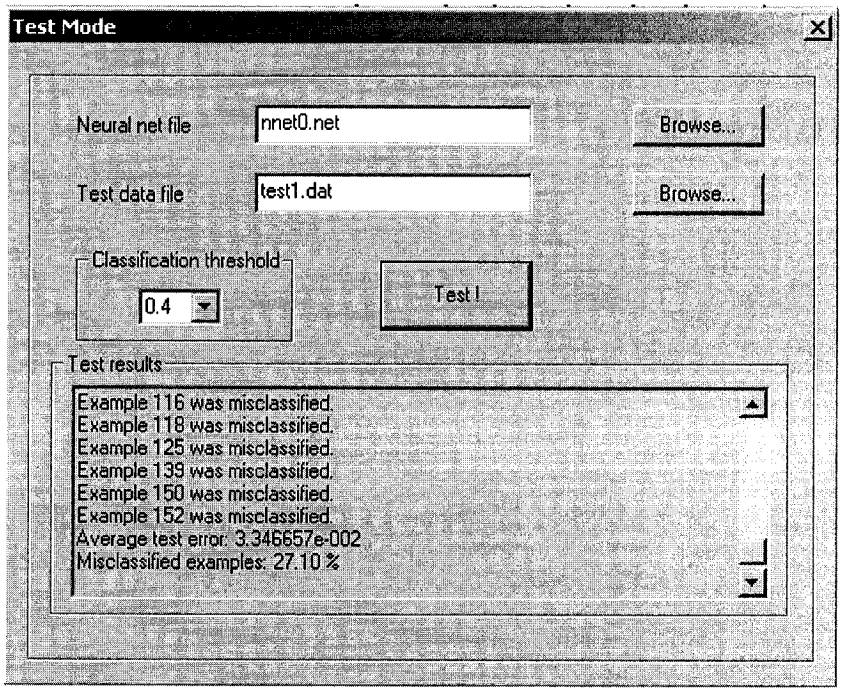

Figure F.4: Test mode dialog

F.3 Results mode

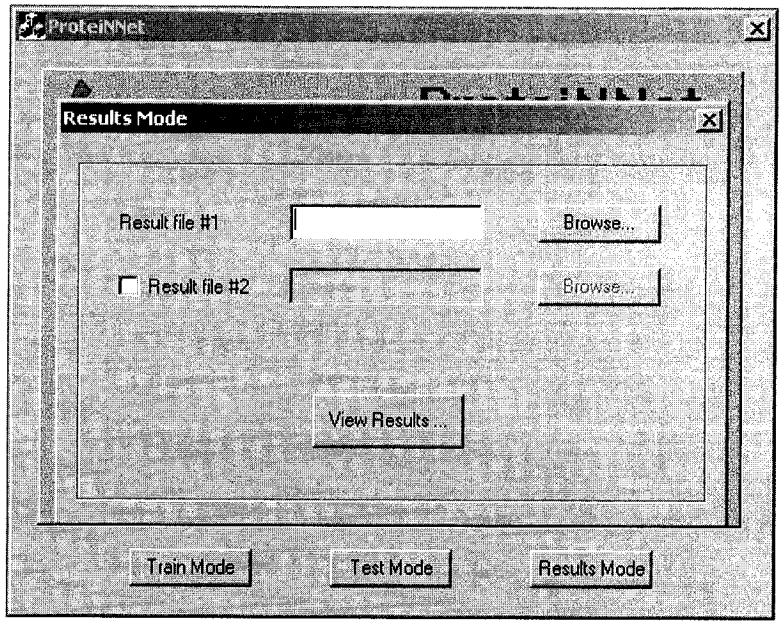

Figure F.5 : Results mode dialog

Pressing the *Result Mode* button from the main dialog opens up a new dialog window. This mode allows the user to view and compare up to two different result sets previously obtained. The user is asked to enter one or two names for result files. This can be done manually or by pressing the *browse...* button. If comparing two files, the user must enable the check box *Result file #2*, and then enter a second file name. This allows comparing the performance of two neural nets on the same input, or of the same network on two different inputs, or just comparing two different nets and two different input sets. (See Figure F.5).

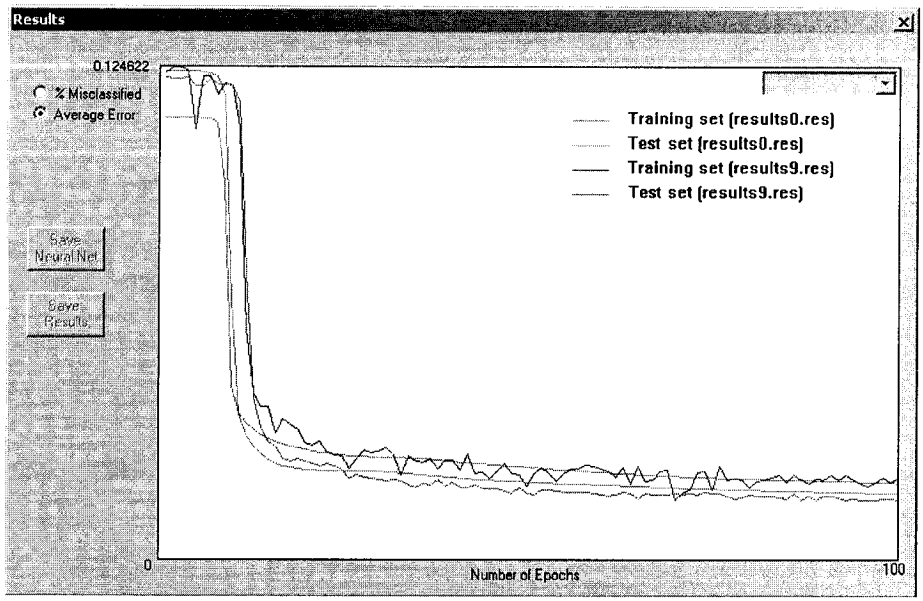

Figure F.6: Results dialog called from results mode

When the button "View Results..." is pressed, another dialog window is opened (Figure F.6) with the results of the comparison, if any, or just the results of the one file. At this point the choices will be:

- Error measure:
  - *% Misclassified*: The percentage of misclassified examples (y-axis) versus the number of epochs (x-axis).
  - *Average Error*: the mean squared error on the output versus the target output. Here, the application plots two curves or four curves (depending on the mode) along with the legend:
    - Average error on the training set (y-axis) versus the number of epochs (x-axis), one per result file.
    - Average error on the test set (y-axis) versus the number of epochs (x-axis), one per result file (if testing was enabled).
- *Result set*: This drop-down list in the top right hand corner is always disabled in this mode.
- *Save results*: This button is always disabled in this mode.
- *Save Neural Net*: This button is always disabled in this mode.

F.4 Predict mode

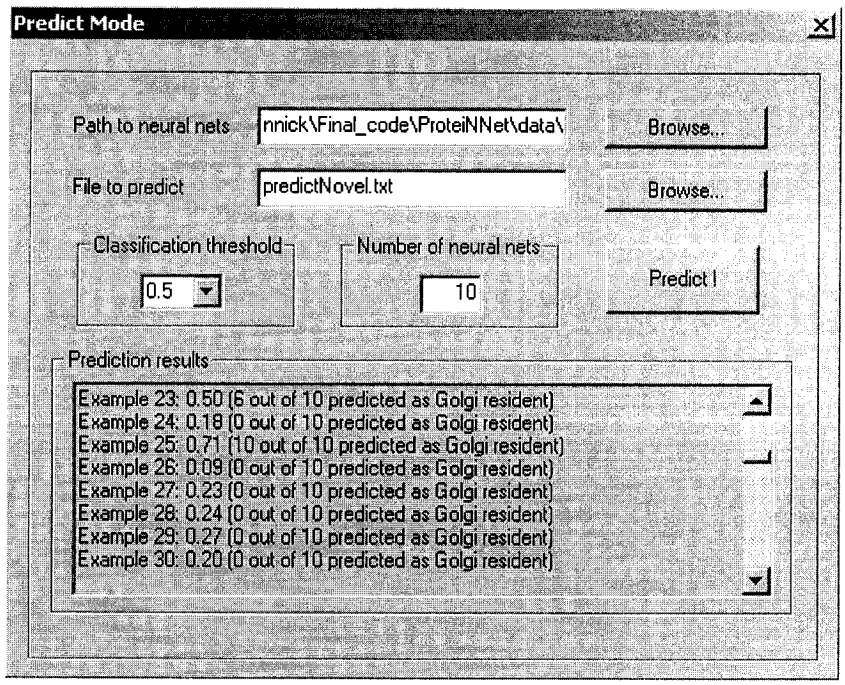

Figure F.7 : Predict mode dialog

This mode enables the user to predict one or more candidate examples using one or more existing neural networks. The user must specify the path to the neural network files: there should be at least at least one file named nnet0.net, and any additional files must be named nnet1.net, nnet2.net, etc. This can be done manually by typing in an existing path containing these files or by using the *browse...* button. The user must also specify the text file that contains the candidate examples to be predicted. This can be done manually or by using the *browse...* button. This file must have the same format as the .dat files, except for the " 0" or the " 1" at the end of each line (note: unlike .dat files, there does not need to be a carriage return after the last candidate example).

The user must also specify the following:

- *Classification threshold*: This drop down list enables the user to choose between different threshold values when classifying a data example, where a lower value represents a stricter classification (more restrictive). By default the classification threshold is set to the least restrictive value (0.5).

- *Number of neural nets*: The number of neural networks, from the given path, to use for the prediction (must be consecutive in their names: nnet0.net, nnet1.net, etc.).

After pressing the *Predict!* button, the candidate examples in the prediction file will be input into the neural networks. For each example, the average prediction value, in the range (0,1), will be displayed in the *results* text box (not editable by the user), followed by the number of neural networks that predicted it as Golgi resident.

F.5 Control Flow Diagram

The following figure represents the control flow diagram of ProteiNNet: The user goes down to the next dialog window by pressing the corresponding button, and comes back to the previous dialog window by closing the current dialog window.