Neural Comput & Applic (2006)
DOI 10.1007/s00521-006-0029-y

ORIGINAL ARTICLE

Aristoklis D. Anastasiadis · George D. Magoulas

Analysing the localisation sites of proteins through neural networks ensembles

Received: 21 July 2004 / Accepted: 10 January 2006

Abstract Scientists involved in the area of proteomics are currently seeking integrated, customised and validated research solutions to better expedite their work in proteomics analyses and drug discoveries. Some drugs and most of their cell targets are proteins, because proteins dictate biological phenotype. In this context, the automated analysis of protein localisation is more complex than the automated analysis of DNA sequences; nevertheless the benefits to be derived are of same or greater importance. In order to accomplish this target, the right choice of the kind of the methods for these applications, especially when the data set is drastically imbalanced, is very important and crucial. In this paper we investigate the performance of some commonly used classifiers, such as the K nearest neighbours and feed-forward neural networks with and without cross-validation, in a class of imbalanced problems from the bioinformatics domain. Furthermore, we construct ensemble-based schemes using the notion of diversity, and we empirically test their performance on the same problems. The experimental results favour the generation of neural network ensembles as these are able to produce good generalisation ability and significant improvement compared to other single classifier methods.

Keywords Feedforward neural networks · Neural ensembles · Protein localisation · Imbalanced datasets · K-nearest neighbour

1 Introduction

The ability to link known proteins through sequence similarity and cellular localisation is becoming increasingly important to accompany information on protein primary sequence and tertiary structure. In particular, the study of protein localisation (in order to function properly, proteins must be transported to various locations within a particular cell) is considered very useful in the post-genomics and proteomics era, as it provides information about each protein that is complementary to the protein sequence and structure data [1].

Two of the most thoroughly studied single-cell organisms are the bacterium, *Escherichia coli* (*E. coli*) and the brewer's yeast, *Saccharomyces cerevisiae* (*S. cerevisiae*). Both organisms use mainstream metabolic pathways, the majority of which are recognisable, similar to the corresponding metabolic functions in all life forms including higher eukaryotes. The entire genome sequence has been determined for both organisms. The relations between genetics and biochemistry, which constitute the fundamental processes of life in these single-cell organisms, serve as a foundation for on-going investigations on the processes that operate in the more complex, higher forms of life [2]. *E. coli* and *S. cerevisiae* are both well-characterised organisms and can be efficiently manipulated genetically. They have rapid growth rate and very simple nutritional requirements. Many studies have detailed both gene and protein expression of these organisms [3, 4]. Recently a neuro-fuzzy approach for functional genomics has been proposed. More precisely, the objective of this approach was to learn and predict the functional classes of the *E. coli* genes [5]. In this study, we are interested in a different problem, which is the prediction of the localisation sites of proteins, such as the *E. coli* and the *S. cerevisiae*.

The first approach for predicting the localisation sites of proteins from their amino acid sequences was an expert system developed by Nakai and Kanehisa [6, 7]. Later, expert identified features were combined with a probabilistic model, which could learn its parameters from a set of training data [8]. Better prediction accuracy has been achieved by using standard classification algorithms, such as K nearest neighbours (KNN), binary

A. D. Anastasiadis (✉) · G. D. Magoulas
School of Computer Science and Information Systems,
Birkbeck College, University of London, Malet Street,
London, WC1E 7HX, UK
E-mail: gmagoulas@dcs.bbk.ac.uk
E-mail: aris@dcs.bbk.ac.uk
Tel.: +44-1895-203397
Fax: +44-1895-211686

decision trees and naïve Bayesian classifiers. The KNN achieved the best classification accuracy compared to these methods [9]: *E. coli* proteins were classified into eight classes with an average accuracy of 86%, while *S. cerevisiae* proteins were classified into ten classes with an average accuracy of 60% by applying cross-validation. More recently, attempts to improve the classification success have been made using back-propagation neural networks, genetic algorithms, growing cell structures and expanding range rules, but no significant improvements over the KNN algorithm were reported [10]. A data selection method for probabilistic neural networks was also applied to the *E. coli* dataset achieving better performance than the KNN (90% accuracy) [11]. Lastly, an empirical study showed that combined methods can achieve better performance than individual ones [12].

This paper advocates a neural network-based approach for classifying the localisation sites of proteins. First, we describe the classification methods used in our work. Next, we introduce the datasets and the evaluation methodology adopted. We finally present experimental results and comparisons.

2 Classification methods

2.1 The Horton–Nakai model

The first method we discuss is a probabilistic model, referred to as the Horton–Nakai (HN) model, which has been specifically designed for the protein localisation problem [8]. The HN model consists of a rooted binary tree of classification variables. Each non-leaf node of the binary tree is a feature variable. The leaves of the tree are the possible classes that a new pattern is going to be classified. A non-leaf node n represents all the classes that belong to leaves that are descendants of n. Each node has a probability associated with it. The probability of n being true represents the probability that an object belongs to n class. In our experiments, we have used a version of the HN model that employs sigmoid conditional probability functions, as those exhibited the best results in previously published studies [8].

2.2 The K nearest neighbours algorithm

The KNN is a simple and effective classification algorithm. It is widely used in machine learning and has numerous variants. Given a test sample of unknown labels, it finds the KNN in the training set and assigns the label of the test sample according to the labels of these neighbours. The vote from each neighbour is weighted by its rank in terms of the distance to the test sample.

In a more formal way, we can express the operation of the KNN algorithm as follows: let $X = \{x^i = (x^i_1, \ldots, x^i_d), i = 1, \ldots, N\}$ be a collection of d-dimensional training samples and $C = \{C_1, ..., C_M\}$ is a set of M classes. Each sample x^i will first be assumed to possess a class label $L_i \in \{1, \ldots, M\}$ indicating with certainty its membership to a class in C. Assume also that x^s is the incoming sample to be classified. Classifying x^s corresponds to assigning it to one of the classes in C, i.e. deciding among a set of M hypotheses: $x^s \in C_q$, $q = 1, \ldots, M$. Let Φ^s be the set of the KNN of x^s in X. For any $x^i \in \Phi^s$, the knowledge that $L_i = q$ can be regarded as evidence that increases our belief that x^s also belongs to one of the classes of C. However, this piece of evidence does not provide certainty by itself.

The KNN method requires selecting a distance metric and choosing a value for parameter K. The KNN, as suggested by Duda and Hart [13], stores training data, denoted as a pair (X, L), and classifies new sample to the majority class among the K closest examples in the training data. This is usually done by calculating the Euclidean distance D, between the new point x and the prototype p^i of the d-dimensional training examples:

$$D^i = \|x - p^i\|, \quad \text{for all } i,$$

where $\|\ \|$ denotes the Euclidean norm.

One of the drawbacks of KNN algorithm is that it needs to compare a test sample with all samples in the training set. In addition, the performance of the algorithm greatly depends on the appropriate choice for the parameter K. KNN-based classification techniques are very popular in the biological domain because of their simplicity and their ability to capture sequential constraints present in the sequences. In order to classify a test sequence, the KNN first locates K training sequences which are most similar to the test sequence. It then assigns the class label that most frequently occurs among those K sequences (majority function) to the test sequence. The key component of the KNN classifier is the method used for computing the similarity between the two sequences.

2.3 Feed-forward neural networks

In a multilayer feed-forward neural network (FNN) nodes are organised in layers and connections are from input nodes to hidden nodes and from hidden nodes to output nodes. The notation I–H–O is used to denote a network with I inputs, H hidden layer nodes and O outputs nodes. In our experiments, we have used FNNs with sigmoid activations for the hidden and output nodes. One of the most popular training algorithms for FNNs is the batch back-propagation (BP), which is a first order method that minimises the error function using the steepest descent with constant learning rate [14]. A small learning rate is usually chosen, i.e. $0 < \eta < 1$, in order to secure the convergence of the BP training algorithm and avoid oscillations in a steep direction of the error surface. However, it is well known that this approach tends to be inefficient [15], and adaptive learning rate algorithms have been proposed to alleviate this situation.

Adaptive algorithms with different learning rates for each weight, try to overcome the inherent difficulty of choosing the appropriate learning rate. This is done by

controlling the weight update for every single connection during the learning process in order to minimise oscillations and maximise the length of the minimisation step. One of the better techniques, in terms of convergence speed, accuracy and robustness with respect to its parameters, is the Rprop algorithm [16–18]. The basic principle of Rprop is that takes into account only the sign of the partial derivatives to indicate the direction of the weight updates. The Rprop algorithm requires setting the following parameters: (1) the learning rate increase factor $\eta^+ = 1.2$; (2) the learning rate decrease factor $\eta^- = 0.5$; (3) the initial update-value is set to $\Delta_0 = 0.1$; and (4) the maximum update-value $\Delta_{\max} = 50$, which is used to prevent weight updates from becoming too large.

3 Ensemble-based methods

Ensemble-based methods enable an increase in generalisation performance by combining several individual neural networks trained on the same task. The ensemble approach has been justified both theoretically [19, 20] and empirically [21]. The creation of an ensemble is often divided into two steps [22]: (1) generate individual ensemble members; and (2) appropriately combine individual members outputs to produce the output of the ensemble. The simplest method for creating ensemble members is to train each member network using randomly initialised weights. A more advanced approach is to train the different networks on different subsets of the training set as bagging [23] does, where each training set is created by resampling and replacement of the original one with uniform probability. Boosting, [24], also uses resampling of the training set, but the data points previously poorly classified, receive a higher probability.

Finally, there is a class of methods for creating ensembles that focuses on creating classifiers that disagree partially on their decisions. In general terms, these methods alter the training process in an attempt to produce classifiers that will generate different classifications. In the neural networks context, these methods include techniques for training with different network topologies, different initial weights, different learning parameters and/or learning different portions of the training set (see [25] for a review and comparisons).

3.1 The notion of diversity and its levels

Networks belonging to an ensemble are thought to be diverse with respect to a test set if they make different generalisation errors on that test set. Different patterns of generalisations can be produced when networks are trained either on different training sets, or from different initial conditions, or with different numbers or hidden nodes, or using different algorithms [25].

In 1997, Sharkey and Sharkey, [25], introduced the term 'Levels of diversity'. They proposed four levels of diversity ranging from the best cases of levels 1 and 2 diversity to the minimum diversity of level 4. Level 1 diversity requires more than two members in an ensemble and considers that for every test input there is always a member that produces the correct output. Level 2 diversity corresponds to at least five members in an ensemble. It is possible diversity of level 2 to lead to level 1 diversity by removing some of the members. An ensemble of level 2 diversity is also known as *upwardly mobile* and eliminates coincident failures. However, in ensembles that belong to this level, the majority is always correct. The level 3 diversity may contain subsets of classifiers with either level 1 or 2 diversity and can be *upwardly mobile*. In this case, the correct output for each input pattern from a test set is always produced by at least one member. Finally, the level 4 diversity is equivalent to the minimal diversity that can be used to improve generalisation. This level can never be reliable since the ensemble members exhibit similar failures (see [25] for details).

3.2 Measuring ensemble diversity

As we have already mentioned the measure of diversity is an important factor in order to create ensembles that can generalise well. There are many ways to quantify the ensemble diversity; these are usually associated with the use of a particular error measure.

In the context of regression problems, Krogh and Vedelsby suggest to calculate the diversity as [20]:

$$d_i(p_k) = [A_i(p_k) - A^*(p_k)]^2,$$

where d_i is the diversity of the ith classifier on pattern p_k, $A_i(p)$ is the ith classifier prediction and $A^*(p)$ is the ensemble prediction. Then in this case, we obtain for the mean squared ensemble error the following equation:

$$E_{\text{ens}} = \bar{E} - \bar{D},$$

where $\bar{D}$ is the mean ensemble's diversity and $\bar{E}$ is the mean single classifier's errors.

Another measure of the diversity is related with a conditional entropy error measure [26]. In our experiments, we have focused on determining the contribution of the individual ensemble member to diversity. In this case, an entropy-based measure would not have been useful because it does not allow determining individual contributions [26].

Thus, for classification problems, the most widely used diversity measure is a simple 0/1 loss function. More precisely, if $A_i(p)$ is the prediction accuracy of the ith classifier for the label of p and assuming that the ensemble's accuracy is $A^*(p)$ then the diversity of the ith classifier on our tested example p is:

$$d_i(p) = \begin{cases} 0 & \text{if } A_i(p) = A^*(p), \\ 1 & \text{otherwise.} \end{cases}$$

The equation for the ensemble error is the same as in regression problems, provided that the loss function used is the squared error function, and that the ensemble prediction is still given as the weighted average of the single classifier predictions.

To compute the diversity of an ensemble of size n, on a training set of size m, the average of the above term is:

$$D_{i,j} = \frac{1}{nm}\sum_{i=1}^{n}\sum_{j=1}^{m} d_i(p_j).$$

In this paper, our approach was to build ensembles that are consistent with the training data and attempt to maximise this diversity term. The average diversity is an interesting factor for the operation of a learning scheme. We took into account small values for the mean squared ensemble error $E_{\text{ens}} = \bar{E} - \bar{D}$, $E_{\text{ens}}<0.1$ and used the disagreement of an ensemble member from the ensemble's prediction as a measure of diversity. Thus, the mean squared ensemble error is equal to the average squared error of the individual networks minus the average diversity. Generally, in order to obtain small ensemble error, we want the diversity to be large and the individual errors to be small [26].

Below an example is given to determine the contribution of an individual ensemble member to diversity. An ensemble of three different individual members (FNNs in our case) try to improve the accuracy for the *S. cerevisiae* data, particularly for the class *me2*, using tenfold cross-validation. Each member of the ensemble produces an output vector with ten components (ten output nodes FNNs are used), which are combined giving the ensemble's output. In Table 1, we take into account the output of the winner node, using five patterns and three members in the ensemble.

By applying the values in the equation for the diversity term we take $\bar{D} = 0.33$. The mean squared ensemble error ($E_{\text{ens}} = \bar{E} - \bar{D}$) is shown in Table 2.

Figure 1 illustrates how FNNs are combined to produce the ensemble. The major aim is to create an ensemble of networks with good predictive performance. Therefore, we consider a population of neural networks (100 FNNs). First we consider the properties of each individual FNN and then we combine FNNs that achieve better performance for the desired application.

Table 1 Example with three classifiers and five data patterns of *S. cerevisiae* dataset for the class *me2*

Target output values	p_1	p_2	p_3	p_4	p_5	Classification errors
	1	1	1	1	1	
FNN1: $A_1(x)$	0	1	0	1	0	$E1 = 0.6$
FNN2: $A_2(x)$	1	0	1	1	0	$E2 = 0.4$
FNN3: $A_3(x)$	1	1	1	0	1	$E3 = 0.2$
Ensemble: $A_n(x)$	1	1	1	1	0	$E = 0.2$

Table 2 The mean squared ensemble error for the example

Average error of the three classifiers$\sim$ ($\bar{E}$)	Mean diversity$\sim$ ($\bar{D}$)	$E_{\text{ens}} = \bar{E} - \bar{D}$
0.4	0.33	0.07

The implementation of the ensemble-based method consists of the following steps:

Step 1. Create n FNNs where each one uses the same training set and differs only in its random initial weights.
Step 2. Select the k neural networks ($k < n$), which when they do fail to classify the data, they fail on different inputs patterns so that failures on one FNN can be compensated by successes of others.
Step 3. Combine the ensemble members' outputs using majority voting to get ensemble's output.

In our experiments, reported in the next section, we investigated two different approaches for creating ensembles. The first approach consisted of creating a simple neural network ensemble by combining FNNs that use the full training set but differs in their random weight initialisations. The second approach was based on the notion of diversity that was mentioned above and used three different FNNs. Training and testing sets were generated by applying cross-validation. Similar approaches often produce results as good as bagging [21]. In these experiments, we also investigated the behaviour of neural ensembles which belong to levels 3 or 4 of diversity, [25], when cross-validation method is applied, and level 3 diversity, [25], when the test set contains all proteins.

4 Experimental study

4.1 The PSORT system

The PSORT system (http://www.psort.ims.u-tokyo.ac.jp/) is a tool for the prediction of protein sub-cellular localisation in the sense that it can deal with proteins localised at almost all the sub-cellular compartments. The latest version of PSORT is a widely used computational method to predict the sub-cellular localisation sites of proteins from their amino acid sequences. The reasoning algorithm is the KNN classifier. It is used to assess the probability of localising at each candidate sites [9]. For each query protein, such as the Gram-positive or Gram-negative or eukaryotic proteins, the output values of the subprograms for these proteins are normalised and simple Euclidean distances to all of the data points contained in the training data are calculated. Then, the prediction is performed using the KNN. For example, if let's say KNN contain 50% nuclear proteins, then the query is predicted to be localised to the nucleus class with a probability of 50%.

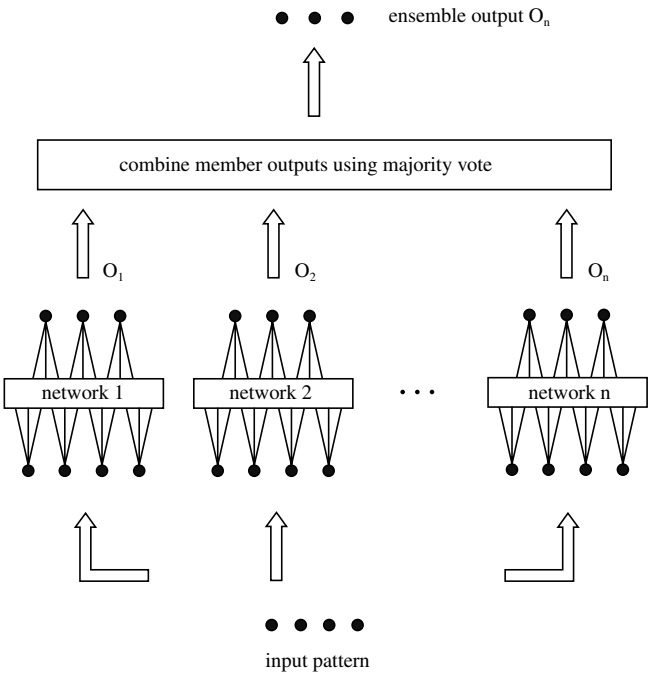

Fig. 1 Neural ensemble formulation

4.2 Description of datasets

Two of the datasets used have been submitted to the UCI machine learning data repository by Murphy and Aha [27] and are described in [7–9]. They include the *E.coli* dataset, which consists of 336 different proteins labelled according to eight localisation sites, and the *S. cerevisiae* data set that contains 1,484 proteins labelled according to ten sites.

Escherichia coli, being a prokaryotic Gram-negative bacterium, is an important component of the biosphere. It colonises the lower gut of animals and survives, as it is a facultive anaerobe, when realise to the natural environment, allowing widespread to new hosts [28, 29]. Three major and distinctive types of proteins are characterised in *E. coli*: enzymes, transporters and regulators. The largest number of genes encodes enzymes (34%) (this should include all the cytoplasm proteins), followed by the genes for transport functions and the genes for regulatory process (11.5%) [2].

Saccharomyces cerevisiae is the simplest eukaryotic organism, but still it is more complicated form of life than *E. coli*. It possesses different types of proteins related to the cytoskeletal structure of the cell, the nucleus organisation, membrane transporters and metabolic related proteins, such as mitochondrial proteins. Of major importance are the yeast membrane transporter proteins as they are responsible for nutrient uptake, drug resistance, salt tolerance, control of cell volume, efflux of undesirable metabolites and sensing of extracellular nutrients [28, 29, 30].

Protein patterns in the *E. coli* data set are organised as follows: 143 patterns of cytoplasm (cp), 77 of inner membrane without signal sequence (im), 52 of periplasm (pp), 35 of inner membrane with uncleavable signal sequence (imU), 20 of outer membrane without lipoprotein (om), 5 of outer membrane with lipoprotein (omL), 2 of inner membrane with lipoprotein (imL) and 2 patterns of inner membrane with cleavable signal sequence (imS).

Saccharomyces cerevisiae proteins are organised as follows: there are 463 patterns of cytoplasm (cyt), 429 of nucleus (nuc), 244 of mitochondria (mit), 163 of membrane protein without N-terminal signal (me3), 51 of membrane protein with uncleavable signal (me2), 44 of membrane protein with cleavable signal (me1), 35 of extracellular (exc), 30 of vacuole (vac), 20 of peroxisome (pox) and 5 patterns of endoplasmic reticulum (erl).

The last dataset used in our experiments is the recent sequence of *Arabidopsis thaliana chromosomes 4* [31]. The *A. thaliana* chromosome 4 data consist of four different sequences:

- C—chloroplast, i.e. this sequence contains a chloroplast transit peptide, cTP;
- M—mitochondrion, i.e. it contains a mitochondrial targeting peptide, mTP;
- S—secretory pathway, i.e. this sequence contains a signal peptide, SP;
- OL—sequence of any other location.

The dataset is organised as follows: 2,247 cTP patterns, 386 mTP patters, 657 SP patterns and 533 patterns of unknown location.

4.3 Evaluation methods

4.3.1 Cross-validation

In k-fold cross-validation a dataset D is randomly split into k mutually exclusive subsets $D_1,\ldots, D_k$ of approximately equal size. The classifier is trained and tested k times; each time $t \in \{1, 2, \ldots, k\}$ it is trained on all D_i, $i=1,\ldots,k$, with $i \neq t$, and tested on D_t. The cross-validation estimate of accuracy is the overall number of correct classifications divided by the number of instances in the dataset.

We have used cross-validation to estimate the accuracy of the classification methods by considering the proportion of patterns for all classes to be equal in each partition; this approach provides more accurate results than plain cross-validation [31].

4.3.2 The Wilcoxon test of statistical significance

The Wilcoxon signed rank test (WSRT) is a non-parametric method [32]. It is an alternative to the paired t test, but is less tolerant of overlapping variance. It has been introduced by Wilcoxon in 1945, and it is designed to test whether a particular sample comes from a population with a specific median. It can also be used in paired difference experiments. This test assumes that

there is information in the magnitudes of the differences between paired observations, as well as the signs. It is a very popular statistical test used by researchers to prove the significance of the experimental results [33, 34].

Next, we briefly describe the method. Firstly, the paired observations are taken, the differences are calculated and then ranked from smallest to largest depending on their absolute value. Adding all the ranks associated with positive and negative differences gives the so called T_+ and T_- statistics, respectively. Finally, the probability value associated with this statistic is found from the appropriate Tables.

More precisely the data consists of n^* observations on the respective bivariate random variables. Assume that the sample of differences,DF_i, is randomly selected from the population of differences. The DF_is are mutually independent and the probability distribution for the sampled paired differences is continuous. Let $|DF_i| = |X_i - Y_i|$ be the absolute differences for $i = 1, 2, ..., n^*$, where $X_i = (x_1, \ldots, x_n^*)$ is the population A, $Y_i = (y_1, \ldots, y_n^*)$ is the population B and n is the number of non-zero differences. T_+ denotes the sum of signed rank of positive DF_i, T_- denotes the sum of signed rank of negative DF_i and $T = \min(T_+, T_-)$. Ranks are assigned to these n absolute differences according to their relative size.

We implemented the right-tailed test. In this case, the population A is shifted to the right of B. The statistical hypothesis is: H_0: $DF_i = 0$, and H_a: $DF_i > 0$, and the rejection region is $T \leq T_0$ for small sample sizes, where T_0 is given by a standard table and it is the critical value of T [35]. When the sample sizes are greater than 25, the large sample approximation procedure can be used; this is $Z_c > Z_a$, where Z_c is equal to:

$$Z_c = \frac{T_+ - [n(n+1)/4]}{\sqrt{n(n+1)(2n+1)/24}}$$

and Z_a takes a standard value $Z_a = 1.96$, and n is the number of the paired differences, which are not zero. This approach was used in our experiments in order to analyse the statistical significance of the results.

The implementation consists of the following steps [35]:

Step 1. Create two lists of experimental data in a pairwise fashion.
Step 2. Take the absolute difference $|DF_i| = |X_i - Y_i|$ for each pair.
Step 3. Omit from consideration those cases where $|X_i - Y_i| = 0$.
Step 4. Rank, from smallest to largest, the remaining absolute differences using tied ranks where this is appropriate.
Step 5. Assign to each such rank a "+" sign when the difference of $(X_i - Y_i) > 0$, and a "−" sign when the difference of $(X_i - Y_i) < 0$. T_+ is the sum of signed rank of positive and T_- is the sum of signed rank of negative.
Step 6. Calculate the value of Z_c for the Wilcoxon test if the sample sizes are greater than 25, or calculate the value of T for small samples, which is equal to the minimum of the sum of the signed ranks $T = \min(T_+, T_-)$.

All statements, above, refer to a significance level of 5%, [32], which corresponds, for example, to $T < 8$ for the *E. coli* dataset and $T < 14$ for the *S. cerevisiae* dataset [35].

4.4 Experiments and results

In previous works, [8, 9], Horton and Nakai showed that the KNN algorithm achieves better performance than the HN probabilistic model. Thus, in our experiments, reported below, only KNN results are presented and compared against FNNs and neural ensembles.

4.4.1 Classifying E. coli proteins using a feed-forward neural network

We conducted a set of preliminary experiments to find the most suitable FNN architecture in terms of training speed. The Rprop algorithm was used to train several networks with one hidden layer with various combinations of hidden nodes, i.e. 8, 12, 14, 16, 24, 32, 64, 120 hidden nodes. Each FNN architecture was trained ten times with different initial weights. The best available architecture was a 7-16-8 FNN, and this architecture was used throughout the experiments. In order to explore the effect of the network error on the accuracy of the classifications, one hundred (100) independent trials were performed with two different termination criteria, which are based on the Mean Squared Error, $E_{\mathrm{MSE}} < 0.05$ and $E_{\mathrm{MSE}} < 0.015$. Following previous work in this area [6–10], we conducted experiments using all the data for testing, as well as data sets produced by fourfold cross-validation. Also, we did additional experiments by applying leave one out cross-validation to further investigate the performance of the methods.

4.4.1.1 Classification of E. coli patterns using the full dataset The particular nature of the problem makes important for biologists to know exactly the performance of a method on each individual protein that is included in the dataset. Thus, this experiment trained and tested FNNs using the entire dataset. Table 3 presents the classification success for each class.

The results of the FNN for each class exhibit average performance over 100 trials. It is evident that the FNNs outperforms KNN (the value $K = 7$ was used as suggested in [9]) in almost every class. It is very important to highlight the FNN classification success in imS and imL classes. In the first case, FNNs trained with $E_{\mathrm{MSE}} < 0.05$ exhibit a 6.5% success and FNNs with $E_{\mathrm{MSE}} < 0.015$ have 49% success. In the second case, FNNs with

Table 3 The accuracy of classification of *E. coli* proteins for each class

No. of patterns	Class label	KNN (%)	FNN (%) ($E_{MSE} < 0.05$)	FNN (%) ($E_{MSE} < 0.015$)
77	im	75.3	82.5	89.0
143	cp	98.6	98.1	99.0
2	imL	0.0	85.0	91.8
5	omL	80.0	100.0	100.0
35	imU	65.7	68.0	88.3
2	imS	0.0	6.5	49.0
20	om	90.0	86.6	93.2
52	pp	90.3	88.7	93.6
Mean (%)		62.5	76.9	88.0
Stdv		39.8	30.1	16.3

$E_{MSE} < 0.05$ and FNNs with $E_{MSE} < 0.015$ exhibit 85 and 91.8% success, respectively. The KNN method fails in both cases to produce correct classifications.

It is worth noticing that FNNs results become better when an $E_{MSE} < 0.015$ was used in training. This obviously caused an increase to the average number of epochs required to converge (approximately 2,000 epochs were needed on average), but at the same time led to considerable improvements: the average classification accuracy over 100 runs was 93% with a standard deviation of 0.33. This behaviour provides evidence that a small variation in the value of the error goal might affect the classification success. This might provide explanation for the unsatisfactory FNNs performance reported in [10], where no details on the error goals used in the training phase were provided.

In order to identify common misclassifications, we calculated the confusion matrix for the KNN and the FNN that exhibited the best training speed for the error goal $E_{MSE} < 0.015$. These results are shown in Tables 4 and 5. The aforementioned neural network achieved high percentage of classification compared to other methods [9]. These results also show that fast convergence achieved by the FNN by no means affect its classification success.

4.4.1.2 Classification of E. coli patterns using fourfold cross-validation In this experiment, we performed a fourfold cross-validation test by randomly partitioning the dataset into four equally sized subsets, as suggested in [8, 9]. Three subsets were used for training, while the remaining one was used for testing. Table 6 presents the best KNN and FNN classification accuracy for each class in the second partition. The overall classification success is given in Table 7, where results are in terms of percentage of success for each partition. Lastly, it is important to mention that the performance of KNN algorithm is considerably improved when fourfold cross-validation is used but still lacks in performance compared to the best FNN.

Table 4 Confusion matrix for *E. coli* proteins with KNN

No. of patterns	Class label	cp	imL	imS	imU	im	omL	om	pp
143	cp	141	0	0	0	0	0	0	2
2	imL	0	0	0	0	1	1	0	0
2	imS	0	0	0	1	0	0	0	1
35	imU	0	0	0	23	11	0	0	0
77	im	3	0	0	14	58	0	0	2
5	omL	0	0	0	0	0	4	1	0
20	om	0	0	0	0	0	0	18	2
52	pp	4	0	0	0	1	0	0	47

Table 5 Confusion matrix for *E. coli* proteins with FNN

No. of patterns	Class label	cp	imL	imS	imU	im	omL	om	pp
143	cp	142	0	0	0	0	0	0	1
2	imL	0	2	0	0	0	0	0	0
2	imS	0	0	1	1	0	0	0	0
35	imU	0	0	0	31	4	0	0	0
77	im	2	0	0	6	69	0	0	0
5	omL	0	0	0	0	0	5	0	0
20	om	0	0	0	0	0	0	19	1
52	pp	3	0	0	0	0	0	0	49

4.4.1.3 Classification of E. coli patterns using leave one out cross-validation In this experiment, we classified *E. coli* proteins using a FNN by applying a leave one out cross-validation. Table 8 gives the best results for each method using leave one out cross-validation. As shown in the table, the KNN exhibited better performance than the FNN in this case. Nevertheless, it still lacks compared to KNN with fourfold cross-validation.

The overall pattern classification success for all classes using FNNs with $E_{MSE} < 0.015$ was 85.42%, which can be considered slightly worst compared to the 86% of the KNN. Table 8 exhibits the average classification success for each class. The differences in mean performance of the two methods in Table 8 reflect the fact that the KNN classified correctly the five patterns of the class omL while the FNN misclassified one of these patterns. In order to identify the misclassifications in the *E. coli* dataset we created the confusion matrix for the KNN and FNN, which are shown in Tables 9 and 10, respectively.

Table 6 Best classification success for each method with fourfold cross-validation for *E. coli* proteins (results are for the second partition)

No. of patterns	Class label	KNN (%)	FNN (%) ($E_{MSE} < 0.015$)
77	im	84.0	79.5
143	cp	100.0	97.2
2	imL	0.0	0.0
5	omL	100.0	100.0
35	imU	62.2	87.5
2	imS	0.0	0.0
20	om	80.0	80.0
52	pp	92.2	100.0
Mean (%)		64.8	68.1
Stdv		41.8	42.7

Table 7 Best overall performance on each partition of the fourfold cross-validation

Cross-validation	Partition	KNN (%)	FNN (%) ($E_{MSE} < 0.015$)
	0	89.3	91.7
	1	95.2	88.1
	2	84.5	84.5
	3	76.2	88.1
Mean (%)		86.3	88.1
Stdv		8.0	2.9

Table 8 Classification success of *E. coli* proteins for each class using leave one out cross-validation

No. of patterns	Class label	KNN (%)	FNN (%) ($E_{MSE} < 0.015$)
77	im	76.6	79.3
143	cp	97.9	97.2
2	imL	0.0	0.0
5	omL	100.0	80.0
35	imU	57.1	65.7
2	imS	0.0	0.0
20	om	80.0	80.0
52	pp	88.5	84.6
Mean (%)		62.5	60.8
Stdv		40.8	38.5

4.4.2 Classifying S. cerevisiae proteins using a feed-forward neural network

We conducted a set of preliminary experiments as with the *E. coli* dataset in order to find the most suitable architecture. An 8-16-10 FNN architecture exhibited the best performance. One hundred FNNs were trained with the Rprop algorithm using different initial weights. As previously we conducted the experiments following the guidelines of [8, 9], and used $K = 21$ as suggested in [9].

4.4.2.1 Classification of S. cerevisiae patterns using the full dataset This experiment concerns training and testing the FNN using the whole dataset. The FNN outperformed significantly the other methods. The average classification accuracy of the FNNs was 67%; the worst-case performance was 64% (which is still an improvement over other methods) and the best one 69%. The result of the KNN was 59.5%. FNNs were trained for 10,000 epochs or until $E_{MSE} < 0.05$. On average, 3,500 epochs were necessary in order to reach convergence.

Table 9 Confusion matrix for the KNN with leave one out cross-validation in the *E. coli* data set

No. of patterns	Class label	cp	imL	imS	imU	im	omL	om	pp
143	cp	140	0	0	0	0	0	0	3
2	imL	0	0	0	0	1	1	0	0
2	imS	0	0	0	1	0	0	0	1
35	imU	1	0	0	20	14	0	0	0
77	im	4	0	0	10	59	0	0	4
5	omL	0	0	0	0	0	5	0	0
20	om	0	0	0	0	0	1	16	3
52	pp	5	0	0	0	1	0	0	46

Table 10 Confusion matrix for the FNN with leave one out cross-validation in the *E. coli* data set

No. of patterns	Class label	cp	imL	imS	imU	im	omL	om	pp
143	cp	139	0	0	0	0	0	0	4
2	imL	0	0	0	0	1	1	0	0
2	imS	0	0	0	1	0	0	0	1
35	imU	0	0	0	23	11	0	0	1
77	im	3	0	0	11	61	0	1	1
5	omL	0	0	0	0	0	4	1	0
20	om	0	0	0	0	0	2	16	2
52	pp	4	0	0	0	3	0	1	44

Table 11 shows the classification success achieved for each class. The results of the FNN represent the average of 100 trials. The neural network outperforms the other methods in almost every class. It is very important to highlight its classification success in the POX and ERL classes.

To identify the misclassifications in the *S. cerevisiae* dataset we created the confusion matrix for the FNN that exhibited the fastest convergence to an $E_{MSE} < 0.05$. The results are shown in Table 12. It is important to highlight the significant improvements in classifying the sites for each class compared with previous attempts as shown in Table 13 [9].

4.4.2.2 Classification of S. cerevisiae patterns using tenfold cross-validation This experiment involved the use of tenfold cross-validation method, as proposed in previous published works [8, 9]. The results obtained are shown in Table 14. The KNN algorithm exhibited an improved generalisation achieving an overall classification success of 59.5%. The performance of the FNNs was also improved, achieving average classification success of 64.9%. In these experiments, we used an $E_{MSE} < 0.05$ to train our networks. The results of the Wilcoxon test for the runs shown in Table 14, gives $T = 7$, which satisfies the condition $T < 14$ (satisfy WSRT).

Table 11 The accuracy of classification of *S. cerevisiae* proteins for each class

No. of patterns	Class label	KNN (%)	FNN (%) ($E_{MSE} < 0.05$)
463	cyt	70.7	66.7
5	erl	0.0	99.6
35	exc	62.9	62.7
44	me1	75.0	82.9
51	me2	21.6	47.8
163	me3	74.9	85.6
244	mit	57.8	61.3
429	nuc	50.7	57.7
20	pox	55.0	54.6
30	vac	0.0	4.1
Mean (%)		46.8	62.3
Stdv		29.1	25.9

Table 12 Confusion matrix of *S. cerevisiae* proteins for each class using an FNN

No. of patterns	Class label	cyt	erl	exc	me1	me2	me3	mit	nuc	pox	vac
463	cyt	309	0	0	3	0	12	30	109	0	0
5	erl	0	5	0	0	1	1	0	0	0	0
35	exc	4	0	23	2	2	0	2	2	0	0
44	me1	3	0	1	37	0	2	1	0	0	0
51	me2	6	0	2	3	25	5	8	2	0	0
163	me3	9	0	0	0	0	140	2	10	0	2
244	mit	50	0	0	2	4	10	150	28	0	0
429	nuc	110	0	0	0	42	10	20	247	0	0
20	pox	6	0	0	0	0	0	2	1	11	0
30	vac	11	0	2	0	1	6	2	7	0	1

Table 13 Confusion matrix of *S. cerevisiae* proteins for each class using the KNN algorithm

No. of patterns	Class label	cyt	erl	exc	me1	me2	me3	mit	nuc	pox	vac
463	cyt	314	0	1	0	2	3	32	91	1	0
5	erl	0	0	3	1	1	0	0	0	0	0
35	exc	4	0	22	4	2	0	2	1	0	0
44	me1	0	0	8	33	0	1	2	0	0	0
51	me2	9	0	7	10	11	3	7	4	0	0
163	me3	18	0	0	0	1	122	6	16	0	0
244	mit	62	0	4	2	5	8	141	19	3	0
429	nuc	171	0	0	0	2	10	27	216	0	0
20	pox	4	0	1	1	0	0	1	2	11	0
30	vac	13	0	3	1	1	6	1	5	0	0

4.4.2.3 Classification of S. cerevisiae patterns using leave one out cross-validation In this experiment, we implemented a leave one out cross-validation to explore further the performance of the tested methods. The best overall classification success using a FNN with leave one out cross-validation was 59.9%, while the KNN algorithm exhibited approximately 58%. Both methods performance still lacks compared to tenfold cross-validation. Table 15 shows the comparative results for the tested methods using leave one out cross-validation in terms of average classification success for each class. The statistics reveal that the FNN outperforms the KNN, exhibiting a more balanced performance.

4.4.3 Classifying proteins using ensemble-based techniques

We have created ensembles focusing on classifiers that disagree on their decisions so that when they do fail to classify the data, they fail on different inputs. In this way, failures of one ensemble member can be compensated by successes of others. Combining diverse neural networks (DNNs) into an ensemble formulation allows weighting the outputs of the networks in such a way that either the correct answer is obtained or at least it is obtained often enough so that the generalisation is improved. In order to get a small ensemble error, the diversity should get large values but the number of individual errors should be small.

Table 14 Best performance for each method using tenfold cross-validation for *S. cerevisiae* proteins

Cross-validation	Partition	KNN (%)	FNN (%) ($E_{MSE} < 0.05$)
	0	55.8	65.1
	1	59.2	66.4
	2	61.0	63.2
	3	65.8	65.8
	4	48.6	66.4
	5	62.3	68.5
	6	68.5	61.8
	7	58.9	59.8
	8	56.9	66.4
	9	58.2	65.6
Mean (%)		59.5	64.9
Stdv		5.5	2.6

Table 15 Best performance for each method using onefold cross-validation for *S. cerevisiae* proteins

No. of patterns	Class label	KNN (%)	FNN (%) ($E_{MSE} < 0.05$)
463	cyt	55.4	66.7
5	erl	0.0	80.0
35	exc	61.7	62.7
44	me1	65.1	82.9
51	me2	26.0	47.8
163	me3	75.8	85.6
244	mit	58.4	61.3
429	nuc	50.3	57.7
20	pox	57.8	54.6
30	vac	0.0	4.1
Mean (%)		46.05	60.3
Stdv		26.9	23.4

4.4.3.1 Experiments using the full datasets In this section we report results using a simple network ensemble (SNE) and a DNNs ensemble (DNE). The DNE implementation consisted of five networks and belonged to the so called level 3 diversity [25]. Table 16 exhibits the results of the two ensemble techniques on the *E. coli* dataset. The overall classification success of the SNE was 93.5%, while the overall classification success of the DNE was 96.8%. We decided to concentrate on the DNE approach and created an ensemble for classifying the patterns of the *S. cerevisiae* dataset. The results are shown in Table 17. The DNE significantly outperforms all other methods used so far in the literature (cf. with the results reported in [8–10]).

4.4.3.2 Experiments using leave one out cross-validation In this case, it is difficult to build an ensemble with FNNs that make errors at different inputs patterns, because when leave one out cross-validation is used most

Table 16 Accuracy of classification for *E. coli* proteins using ensemble-based techniques

No. of patterns	Class label	SNE (%) ($E_{MSE} < 0.015$)	DNE (%) ($E_{MSE} < 0.015$)
77	im	87.0	92.2
143	cp	98.0	100.0
2	imL	100.0	100.0
5	omL	100.0	100.0
35	imU	77.2	94.3
2	imS	50.0	50.0
20	om	80.0	100.0
52	pp	91.4	96.1
Mean (%)		85.5	91.7
Stdv		16.8	17.1

Table 18 Best performance for each method using leave one out cross-validation for *E. coli* proteins

No. of patterns	Class label	FNN (%) ($E_{MSE} < 0.015$)	DNE (%) ($E_{MSE} < 0.015$)
77	im	79.3	80.6
143	cp	97.2	97.2
2	imL	0.0	0.0
5	omL	80.0	80.0
35	imU	65.7	65.7
2	imS	0.0	0.0
20	om	80.0	80.0
52	pp	84.6	86.6
Mean (%)		60.8	61.3
Stdv		38.5	38.8

of the FNNs fail on the same patterns. Thus, the classification ability of the ensemble cannot be improved significantly compared to a single FNN. The corresponding is shown in Tables 18 and 19 below.

4.4.3.3 Experiments using fourfold cross-validation for the E. coli dataset In these experiments, an ensemble of three FNNs was investigated. This ensemble corresponded to level 4 diversity, as all FNNs fail to predict patterns of imS class. This ensemble can never be reliable but it can still be used to improve the generalisation overall. In order to train properly the FNNs, a smaller error target was set $E_{MSE} < 0.01$ and the process focused on FNNs that can classify correctly classes with very few patterns, such the imL class. The diversity obtained in these experiments was $D = 0.33$ and the ensemble was able to classify correctly the two patterns of the imL class.

Table 20 presents the number of mistakes for each member of the ensemble and for different combinations. Ensemble members FNN1, FNN2 and FNN3 fail to classify 15, 16 and 11 mistakes, respectively. Combining different FNNs produces various numbers of common mistakes (failures) depending on the diversity of the members which are combined. For example, as shown in Table 20, although FNN1 and FNN2 make 15 and 16 mistakes, a combination of the two produces significantly smaller number of common failures, i.e. only two common mistakes. In our experiments we used a combination of the three FNNs using majority voting. This ensemble also produces two common failures but the overall success is improved. Detailed results are shown in Table 21.

It is important to mention that using a DNE allows to classify correctly the imL class, which has only two patterns. This can increase the average class accuracy for *E. coli* proteins as shown in Table 21.

Various combinations of diverse FNNs can be produced focusing on improving the classification for particular class labels. For example, if predicting correctly classes with a few patterns (e.g. imS) is not a priority then it is possible to improve the overall classification success by focusing on classifying correctly classes with many patterns (e.g. cp). In this case the ensemble would combine diverse neural networks that achieve better performance in classes with many patterns, such as the cp class.

4.4.3.4 Experiments using tenfold cross-validation for the S. cerevisiae dataset In this case, we created an ensemble of three neural networks, which corresponds to level 3 diversity [25]. In this case a simple majority vote will not

Table 17 Accuracy of classification for *S. cerevisiae* proteins combining diverse neural networks

No. of patterns	Class label	DNE (%) ($E_{MSE} < 0.05$)
463	cyt	69.2
5	erl	100.0
35	exc	64.3
44	me1	84.9
51	me2	54.9
163	me3	88.4
244	mit	61.7
429	nuc	57.7
20	pox	55.0
30	vac	10.0
Mean (%)		64.6
Stdv		24.6

Table 19 Best performance for each method using leave one out cross-validation for *S. cerevisiae* proteins

No. of patterns	Class label	FNN (%) ($E_{MSE} < 0.05$)	DNE (%) ($E_{MSE} < 0.05$)
463	cyt	66.7	66.7
5	erl	80.0	100.0
35	exc	62.7	62.7
44	me1	82.9	80.0
51	me2	47.8	47.8
163	me3	85.6	91.7
244	mit	61.3	61.3
429	nuc	57.7	57.7
20	pox	54.6	54.6
30	vac	4.1	4.1
Mean (%)		60.3	62.6
Stdv		23.4	26.4

Table 20 Number of *E. coli* patterns that are not classified correctly for each ensemble member (FNN1, FNN2, FNN3), and for combinations of diverse FNNs (FNN1&FNN2, FNN1&FNN3, FNN2&FNN3, FNN1&FNN2&FNN3). Training used fourfold cross-validation

Number of FNNs	Failures
FNN1	15
FNN2	16
FNN3	11
FNN1&FNN2	2
FNN1&FNN3	5
FNN2&FNN3	3
FNN1&FNN2&FNN3	2

always produce the correct answer, but at least one member in the ensemble will produce the correct output for each input pattern in the test set. Table 22 shows classification success for an FNN and two ensembles (DNE1 and DNE2) using tenfold cross-validation. The FNN performance is the best available from a set of 100 trials, while the diversity value for DNN1, which focuses on predicting cyt patterns, is $D = 0.36$ giving $E_{ens} = 0.023$, and for DNN2, which is built based on vac patterns, is $D = 0.33$ and $E_{ens} = 0.022$.

It is important to mention that DNEs achieved better performance compared to other classification methods. The KNN overall success was 59.3% and the FNN success was 63.4% when $E_{MSE} < 0.05$ was used in training. DNN1 shows significant improvement in overall pattern classification success, reaching 67.6% but it cannot classify correctly any vac pattern; it focuses on cyt patterns instead. The overall classification success for DNN2 was 62.75% with predictions for all classes of the *S. cerevisiae* proteins, and an average success per class of 66.2%, as shown in Table 22.

4.4.3.5 Classifying A. thaliana chromosome 4 data In order to test further the effectiveness of the DNE approach we applied to a new problem, the classification of the *A. thaliana* chromosome 4 [31]. An ensemble of three neural networks exhibiting level 3 diversity, [25], was created. Tenfold cross-validation was used to estimate the accuracy of the classification methods by considering the proportion of patterns for all classes to be equal in each partition. One subset was used for training, while the remaining ones were used for testing. Using this procedure we can achieve an overall mean classification performance of approximately 99.9%. The FNNs in the ensemble were combined using a simple majority vote.

Table 23 shows the classification success for each different sequence of the *A. thaliana* chromosome 4 data. The performance of the FNN is the average of 100 runs. As before, two different ensembles were created; each one focused on better prediction for particular sequences. DNE2 is better able to classify correctly more sequences of the cTP and SP peptides class. As shown in Table 23, DNE2 performs significantly better in SP and cTP classes compared to a single FNN, and the DNE1, which focused on mTP and OL patterns, improves its testing classification success on these classes. DNE2 achieves the best mean performance for each class.

Table 21 Classification results for *E. coli* proteins using fourfold cross-validation and a diverse neural networks ensemble

No. of patterns	Class label	FNN (%) ($E_{MSE} < 0.01$)	DNE (%) ($E_{MSE} < 0.01$)
77	im	89.0	88.8
143	cp	97.2	97.2
2	imL	0.0	100.0
5	omL	100.0	100.0
35	imU	75.0	87.5
2	imS	0.0	0.0
20	om	80.0	80.0
52	pp	100.0	92.3
Mean (%)		67.6	80.7
Stdv		42.7	33.3

Table 22 Classification results for *S. cerevisiae* proteins using tenfold cross-validation and diverse neural networks ensembles

No. of patterns	Class label	KNN (%)	FNN (%) ($E_{MSE} < 0.05$)	DNE1 (%) ($E_{MSE} < 0.05$)	DNE2 (%) ($E_{MSE} < 0.05$)
463	cyt	70.7	71.7	76.1	59.0
5	erl	0.0	100.0	100.0	100.0
35	exc	62.9	25.0	75.0	50.0
44	me1	75.0	80.0	60.0	80.0
51	me2	21.6	60.0	40.0	60.0
163	me3	74.9	87.5	100.0	100.0
244	mit	57.8	75.0	91.7	58.3
429	nuc	50.7	42.9	52.5	54.8
20	pox	55.0	66.7	33.3	66.7
30	vac	0.0	0.0	0.0	33.3
Mean (%)		46.8	60.9	62.9	66.2
Stdv		29.1	30.3	32.2	21.3

5 Conclusions

In this paper we explored the use of a neural network approach in predicting the localisation sites of proteins in the organisms. Neural Networks performance both in single mode and in ensemble formulation was investigated. We particularly investigated the use of network ensembles exhibiting levels 3 and 4 diversity with and without cross-validation, creating ensembles from

Table 23 Classification results for *A. thaliana* chromosome 4 patterns using tenfold cross-validation

No. of patterns	Class label	FNN (%) ($E_{MSE} < 0.001$)	DNE1 (%) ($E_{MSE} < 0.001$)	DNE2 (%) ($E_{MSE} < 0.001$)
2,247	OL	98.1	99.5	98.5
386	mTP	83.5	96.8	87.4
657	SP	77.9	78.9	89.9
533	cTP	87.0	88.3	95.5
Mean (%)		86.6	90.9	92.8
Stdv		8.5	9.3	5.0

classifiers that disagree on their decisions in order to improve the generalisation success for each class. In many case, these ensembles were able to achieve significant improvements compared to all other approaches in the literature for the *E. coli* and *S. cerevisiae* data sets, and satisfactory performance for the *A. thaliana* chromosome 4 data set.

Although our test produced encouraging results, it is important to improve the performance of the ensembles by generating and combining networks that exhibit higher diversity, when possible. To this end, our future work will explore the training and use of network ensembles with higher degrees of diversity, taking into account the drastically imbalanced nature of certain datasets.

Acknowledgements We would like to thank Dr Maria Roubelakis of Oxford University for assistance in biological aspects of this work.

References

1. Boland MV, Murphy RF (1999) After sequencing: quantitative analysis of protein localization. IEEE Eng Med Biol Sept/Oct:115–119
2. Liang P, Labedan B, Riley M (2002) Physiological genomics of *Escherichia coli* protein families. Physiol Genomics 9(1):15–26
3. Lu Z, Szafron D, Greiner R, Lu P, Wishart DS, Poulin B, Anvik J, Macdonell C, Eisner R (2004) Predicting subcellular localization of proteins using machine learned classifiers. Bioinformatics 20:547–556
4. Clare A, King RD (2003) Predicting gene function in *Saccharomyces cerevisiae*. Bioinformatics 19:42–49
5. Neagu D, Palade V (2003) A neuro-fuzzy approach for fuctional genomics data interpretation and analysis. Neural Comput Appl 12:153–159
6. Nakai K, Kanehisa M (1991) Expert system for predicting protein localization sites in gram-negative bacteria. Proteins: Struct Funct Genet 11:95–110
7. Nakai K, Kanehisa M (1992) A knowledge base for predicting protein localization sites in eukaryotic cells. Genomics 14:897–911
8. Horton P, Nakai K (1996) A probabilistic classification system for predicting the cellular localization sites of proteins. In: Proceedings of the 4th international conference on intelligent systems for molecular biology, AAAI Press, St. Louis, pp 109–115
9. Horton P, Nakai K (1997) Better prediction of protein cellular localization sites with the *k* nearest neighbors classifier. In: Proceedings of intelligent systems in molecular biology, Halkidiki, Greece, pp 368–383
10. Cairns P, Huyck C, Mitchell I, Wu W (2001) A comparison of categorisation algorithms for predicting the cellular localization sites of proteins. In: Proceedings of IEEE international workshop on database and expert systems applications, pp 296–300
11. Bolat B, Yıldırım T (2003) A data selection method for probabilistic neural networks. In: International XII. Turkish symposium on artificial intelligence and neural networks—TAINN, pp 1137–1140
12. Tan AC, Gilbert D (2003) An empirical comparison of supervised machine learning techniques in bioinformatics. In: Proceedings of the first Asia Pacific bioinformatics conference (APBC 2003), Adelaide, Australia. Australian Computer Society, Sydney. Chen P (ed) Conferences in research and practice in information technology, vol 19, pp 219–222
13. Duda RO, Hart PE (1973) Pattern classification and scene analysis. Wiley, New York
14. Rumelhart DE, Hinton GE, Williams RJ (1986) Learning internal representations by error propagation. In: Rumelhart DE, McClellend JL (eds) Parallel distributed processing: explorations in the microstructure of cognition. MIT Press, Cambridge, pp 318–362
15. Sima J (1996) Back propagation is not efficient. Neural Netw 6:1017–1023
16. Riedmiller M, Braun H (1993) A direct adaptive method for faster backpropagation learning: the RPROP algorithm. In: Proceedings of international conference on neural networks, San Francisco, CA, pp 586–591
17. Riedmiller M (1994) RPROP-description and implementation details. Technical Report, University of Karlsruhe, Germany
18. Udelhoven T, Schutt B (2000) Capability of feed-forward neural networks for a chemical evaluation of sediments with diffuse reflectance spectroscopy. Chemometr Intell Lab Syst 51:9–22
19. Hansen LK, Salamon P (1990) Neural network ensembles. IEEE Trans Pattern Anal Mach Intell 12:993–1001
20. Krogh A, Vedelsby J (1995) Neural network ensembles, cross validation, and active learning. In: Tesauro G, Touretzky D, Leen T (eds) Advances in neural information processing systems, vol 2, pp 650–659
21. Opitz D, Maclin R (1999) Popular ensemble methods: an empirical study. J Artif Intell Res 11:169–198
22. Sharkey AJC (1996) On combining artificial neural nets. Connect Sci 8:299–314
23. Breiman L (1996) Bagging predictors. Mach Learn 24:123–140
24. Freund Y, Schapire RE (1996) Experiments with a new boosting algorithm. In: Proceedings of the 13th international machine learning conference, pp 148–156
25. Sharkey AJC, Sharkey NE (1997) Combining diverse neural nets. Knowl Eng Rev 12:231–247
26. Zenobi G, Cunningham P (2001) Using diversity in preparing ensembles of classifiers based on different feature subsets to minimize generalization error. In: Proceedings of the European conference on machine learning, pp 576–587
27. Murphy PM, Aha DW (1996) UCI repository of machine learning databases. http://www.ics.uci.edu/mlearn
28. Blattner FR, Plunkett G, Bloch CA, Perna NT, Burland V, Riley M, Collado-Vides J, Glasner JD, Rode CK, Mayhew GF, Gregor J, Davis NW, Kirkpatrick HA, Goeden MA, Rose DJ, Mau B, Shao Y (1997) The complete genome sequence of *Escherichia coli* K-12. Science 277(5331):1453–1474
29. Lodish H, Berk A, Zipursky SL, Matsudaira P, Baltimore D, James Darnell J (2003) Molecular cell biology, 5th edn. Freeman, San Francisco, CA
30. Van Belle D, Andre B (2001) A genomic view of yeast membrane transporters. Curr Opin Cell Biol 13(4):389–398
31. Emanuelsson O, Nielsen H, Brunak S, von Heijne G (2000) Predicting Subcellular localization of proteins based on their N-terminal amino acid sequence. J Mol Biol 300:1005–1016
32. Igel C, Husken M (2003) Empirical evaluation of the improved Rprop learning algorithms. Neurocomputing 50:105–123
33. Kohavi R (1995) A study of cross-validation and bootstrap for accuracy estimation and model selection. In: International joint conference on artificial intelligence, AAAI Press and MIT Press, pp 223–228
34. Nugent CD, Lopez JA, Smith AE 1, Black ND (2002) Prediction models in the design of neural network based ECG classifiers: a neural network and genetic programming approach. BMC Med Inform Decis Making 2(1)
35. Snedecor G, Cochran W (1989) Statistical methods, 8th edn. Iowa State University Press, Ames, IA