Protein Sequence Classification Using Extreme Learning Machine

Dianhui Wang
Department of Computer Science and
Computer Engineering
La Trobe University
Melbourne, VIC 3083
Australia
E-mail: dh.wang@latrobe.edu.au

Guang-Bin Huang
School of Electrical and
Electronic Engineering
Nanyang Technological University
Nanyang Avenue
Singapore 639798
E-mail: egbhuang@ntu.edu.sg

Abstract— In this paper, a recently developed machine learning algorithm for neural networks referred to as Extreme Learning Machine (ELM) is used to classify protein sequences with ten classes of super-families. Performance of ELM is compared in terms of training time as well as generalization performance with the main conventional neural network classifier - Backpropagation Neural Networks. Results show that ELM needs four orders of magnitude less training time compared to Backpropagation Neural Networks classifier. The classification accuracy of ELM is also higher than that of BP network. Compared with conventional learning methods, the ELM can be easily implemented.

I. Introduction

It is known that traditionally all the parameters of the feedforward networks need to be tuned and thus there exists the dependency between different layers of parameters (weights and biases). For past decades gradient descent-based methods have mainly been used in various learning algorithms of feedforward neural networks. However, it is clear that gradient descent based learning methods are generally very slow due to improper learning steps or may easily converge to local minimums. And many iterative learning steps are required by such learning algorithms in order to obtain better learning performance. It is also known that many tuning or adjustment based learning methods may face difficulty in training the networks with non-differential activation functions.

Unlike popular implementations such as Back-Propagation (BP), Huang, et al[1] have recently proposed a new learning algorithm called Extreme Learning Machine (ELM) for Single-hidden Layer Feedforward neural Networks (SLFNs) which are either additive neurons or kernel based schemes. For additive neurons based SLFNs one may randomly choose the input weights and the hidden neurons' biases and analytically determine the output weights of SLFNs[1]. Input weights are the weights of the connections between input neurons and hidden neurons and output weights are the weights of the connections between hidden neurons and output neurons. For Radial Basis Function (RBF) kernel based SLFNs, instead of tuning the centers and impact widths of RBF kernels we may just simply randomly choose these kernel parameters and analytically calculate the output weights of RBF networks[2], [3]. After the input weights and the hidden layer biases are chosen arbitrarily, SLFNs can be simply considered as a linear system and the output weights (linking the hidden layer to the output layer) of SLFNs can be analytically determined through simple generalized inverse operation of the hidden layer output matrices. As analyzed by Huang, et al[1], [2], [3], ELM tends to have good generalization performance and can be implemented easily. Unlike other tuning/adjustment methods (e.g.,[4]) which may neither be suitable for non-differential activation functions nor prevent the troubling issues such as stopping criteria, learning rate, learning epoches, and local minima, the ELM algorithm can avoid these difficulties very well.

In this paper, the performance of ELM with sigmoidal activation function for protein sequence analysis is investigated and compared with the main neural network classifier - BP based neural networks classifiers. Raw protein sequences data with ten classes of super-families were obtained from a well-known database[1]. Feature extraction from this raw data was first performed and then these were used to train the classifiers. The trained classifiers were then tested with data not seen during the training to evaluate its accuracy. The results indicate that the ELM classifier produces similar classification accuracy but requires training time of orders of magnitude less than the BP and SVM.

The paper is organized as follows. Section 2 gives a brief description of different neural network classifiers including ELM and BP. Section 3 presents a brief description of the protein sequence database and the applied feature selection method. Performance comparison of ELM classifier with BPNN classifiers are reported in Section 4 along with a discussion of the results. Conclusions from this study are summarized in Section 5.

[1]http://pir.Georgetown.edu

II. Brief Description of Main Neural Network Classifiers

A. Extreme Learning Machine for SLFNs

Unlike popular implementations such as Back-Propagation (BP), Huang, et al[1] have recently proposed a new learning algorithm called Extreme Learning Machine (ELM) for Single-hidden Layer Feedforward neural Networks (SLFNs) which are either additive neurons or kernel based schemes. For additive neurons based SLFNs one may randomly choose the input weights and the hidden neurons' biases and analytically determine the output weights of SLFNs[1]. Input weights are the weights of the connections between input neurons and hidden neurons and output weights are the weights of the connections between hidden neurons and output neurons. After the input weights and the hidden layer biases are chosen arbitrarily, SLFNs can be simply considered as a linear system and the output weights (linking the hidden layer to the output layer) of SLFNs can be analytically determined through simple generalized inverse operation of the hidden layer output matrices.

For N arbitrary distinct samples $(\mathbf{x}_i, \mathbf{t}_i)$, where $\mathbf{x}_i = [x_{i1}, x_{i2}, \cdots, x_{in}]^T \in \mathbf{R}^n$ and $\mathbf{t}_i = [t_{i1}, t_{i2}, \cdots, t_{im}]^T \in \mathbf{R}^m$, standard SLFNs with $\tilde{N}$ hidden neurons and activation function $g(x)$ are mathematically modeled as

$$\sum_{i=1}^{\tilde{N}} \beta_i g(\mathbf{w}_i \cdot \mathbf{x}_j + b_i) = \mathbf{o}_j, \quad j = 1, \cdots, N, \tag{1}$$

where $\mathbf{w}_i = [w_{i1}, w_{i2}, \cdots, w_{in}]^T$ is the weight vector connecting the ith hidden neuron and the input neurons, $\beta_i = [\beta_{i1}, \beta_{i2}, \cdots, \beta_{im}]^T$ is the weight vector connecting the ith hidden neuron and the output neurons, and b_i is the threshold of the ith hidden neuron. $\mathbf{w}_i \cdot \mathbf{x}_j$ denotes the inner product of $\mathbf{w}_i$ and $\mathbf{x}_j$.

The fact that standard SLFNs with $\tilde{N}$ hidden neurons each with activation function $g(x)$ can approximate these N samples with zero error, means that $\sum_{j=1}^{\tilde{N}} \|\mathbf{o}_j - \mathbf{t}_j\| = 0$. i.e., there exist β_i, $\mathbf{w}_i$ and b_i such that

$$\sum_{i=1}^{\tilde{N}} \beta_i g(\mathbf{w}_i \cdot \mathbf{x}_j + b_i) = \mathbf{t}_j, \quad j = 1, \cdots, N. \tag{2}$$

The above N equations can be written compactly as:

$$\mathbf{H}\beta = \mathbf{T} \tag{3}$$

where

$$\mathbf{H}(\mathbf{w}_1, \cdots, \mathbf{w}_{\tilde{N}}, b_1, \cdots, b_{\tilde{N}}, \mathbf{x}_1, \cdots, \mathbf{x}_N) = \begin{bmatrix} g(\mathbf{w}_1 \cdot \mathbf{x}_1 + b_1) & \cdots & g(\mathbf{w}_{\tilde{N}} \cdot \mathbf{x}_1 + b_{\tilde{N}}) \\ \vdots & \cdots & \vdots \\ g(\mathbf{w}_1 \cdot \mathbf{x}_N + b_1) & \cdots & g(\mathbf{w}_{\tilde{N}} \cdot \mathbf{x}_N + b_{\tilde{N}}) \end{bmatrix}_{N \times \tilde{N}} \tag{4}$$

$$\beta = \begin{bmatrix} \beta_1^T \\ \vdots \\ \beta_{\tilde{N}}^T \end{bmatrix}_{\tilde{N} \times m} \quad \text{and} \quad \mathbf{T} = \begin{bmatrix} \mathbf{t}_1^T \\ \vdots \\ \mathbf{t}_N^T \end{bmatrix}_{N \times m} \tag{5}$$

$\mathbf{H}$ is called the hidden layer output matrix of the neural network[5], [6]; the ith column of $\mathbf{H}$ is the ith hidden neuron output with respect to inputs $\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_N$. In theory, it has been shown [7], [6] that when $\tilde{N} = N$ for any randomly given input weights and hidden neurons' biases the hidden layer output matrix $\mathbf{H}$ is invertible and there exists output weights β such that $\mathbf{H}\beta = \mathbf{T}$.

In most cases the number of hidden neurons is much less than the number of distinct training samples, $\tilde{N} \ll N$, $\mathbf{H}$ is a nonsquare matrix and there may not exist $\mathbf{w}_i, b_i, \beta_i$ $(i = 1, \cdots, \tilde{N})$ such that $\mathbf{H}\beta = \mathbf{T}$. Thus, $\hat{\mathbf{w}}_i, \hat{b}_i, \hat{\beta}$ $(i = 1, \cdots, \tilde{N})$ are to be found such that they satisfy :

$$\begin{aligned} &\|\mathbf{H}(\hat{\mathbf{w}}_1, \cdots, \hat{\mathbf{w}}_{\tilde{N}}, \hat{b}_1, \cdots, \hat{b}_{\tilde{N}})\hat{\beta} - \mathbf{T}\| \\ &= \min_{\mathbf{w}_i, b_i, \beta} \|\mathbf{H}(\mathbf{w}_1, \cdots, \mathbf{w}_{\tilde{N}}, b_1, \cdots, b_{\tilde{N}})\beta - \mathbf{T}\| \end{aligned} \tag{6}$$

which is equivalent to minimizing the cost function

$$E = \sum_{j=1}^{N} \left(\sum_{i=1}^{\tilde{N}} \beta_i g(\mathbf{w}_i \cdot \mathbf{x}_j + b_i) - \mathbf{t}_j \right)^2 \tag{7}$$

When $\mathbf{H}$ is unknown, gradient-based learning algorithms are generally used to search the minimum of $\|\mathbf{H}\beta - \mathbf{T}\|$. The popular learning algorithm used in feed-forward neural networks is the back-propagation learning algorithm where gradients can be computed efficiently by propagation from the output to the input. There are several issues for these gradient-descent based algorithms such as local minimal, overfitting, slow convergence rate, etc.[1]

It is very interesting[6] to note that unlike the most common understanding one has that all the parameters of SLFNs need to be adjusted, the input weights $\mathbf{w}_i$ and the hidden layer biases b_i are in fact not necessarily tuned and the hidden layer output matrix $\mathbf{H}$ can actually remain unchanged once arbitrary values have been assigned to these parameters in the beginning of learning.

For fixed input weights $\mathbf{w}_i$ and the hidden layer biases b_i, from equation (6), to train an SLFN is simply equivalent to finding a least-squares solution $\hat{\beta}$ of the linear system $\mathbf{H}\beta = \mathbf{T}$:

$$\begin{aligned} &\|\mathbf{H}(\mathbf{w}_1, \cdots, \mathbf{w}_{\tilde{N}}, b_1, \cdots, b_{\tilde{N}})\hat{\beta} - \mathbf{T}\| = \\ &\min_{\beta} \|\mathbf{H}(\mathbf{w}_1, \cdots, \mathbf{w}_{\tilde{N}}, b_1, \cdots, b_{\tilde{N}})\beta - \mathbf{T}\| \end{aligned} \tag{8}$$

The unique smallest norm least-squares solution of the above linear system is:

$$\hat{\beta} = \mathbf{H}^{\dagger}\mathbf{T} \tag{9}$$

where $\mathbf{H}^{\dagger}$ is the Moore-Penrose generalized inverse of hidden layer output matrix $\mathbf{H}$[8].

The special solution $\hat{\beta} = \mathbf{H}^{\dagger}\mathbf{T}$ is the smallest norm least-squares solutions of a general linear system $\mathbf{H}\beta = \mathbf{T}$, meaning that the smallest training error can be reached by this special solution. As analyzed by Huang, et al[1], from the viewpoint of Vapnik-Chervonenkis (VC) dimension (and hence number of parameters)[9], this method

may tend to reach good generalization performance. (For detailed discussion, refer to [1], [9])

The three main steps involved in ELM algorithm can be summarized as:

ELM Algorithm[1]: Given a training set $\aleph = \{(\mathbf{x}_i, \mathbf{t}_i)|\mathbf{x}_i \in \mathbf{R}^n, \mathbf{t}_i \in \mathbf{R}^m, i = 1, \cdots, N\}$, activation function $g(x)$, and hidden neuron number $\tilde{N}$,

step 1 Assign arbitrary input weight $\mathbf{w}_i$ and bias b_i, $i = 1, \cdots, \tilde{N}$.

step 2 Calculate the hidden layer output matrix $\mathbf{H}$.

step 3 Calculate the output weight β: $\beta = \mathbf{H}^{\dagger}\mathbf{T}$.

Unlike other tuning/adjustment methods (e.g.,[4]) which may neither be suitable for non-differential activation functions nor prevent the troubling issues such as stopping criteria, learning rate, learning epoches, and local minima, the ELM algorithm can avoid these difficulties very well.

B. Backpropagation Neural Network (BPNN)

As mentioned above, gradient-descent based learning BP algorithm minimizes the cost function (cf. equation (7) also):

$$\min_{\mathbf{w}_i, \beta_i, b_i} E = \sum_{j=1}^{N} \left(\sum_{i=1}^{\tilde{N}} \beta_i g(\mathbf{w}_i \cdot \mathbf{x}_j + b_i) - \mathbf{t}_j \right)^2 \tag{10}$$

based on:

$$\mathbf{W}_k = \mathbf{W}_{k-1} - \eta \frac{\partial E(\mathbf{W})}{\partial \mathbf{W}} \tag{11}$$

where the vector $\mathbf{W}$ is the set of weights $(\mathbf{w}_i, \beta_i)$ and basis (b_i) parameters, and η is the learning rate. The method for calculating the gradients for the sigmoid approximation of neural networks is called the backpropagation method, where the gradients can be computed efficiently by propagation from the output layer to the input layer. It is known that learning rate η should be carefully selected. Another precaution is the local minima presented in the cost function surface. Moreover, early stopping or regularization method has to be used in the training process in order to improve the generalization performance.

III. Protein Sequence Classification Problem

In this paper, we use the same database as and adopt the preprocessing method similar to the work done in Wang, et al[10]. The protein sequences are transformed from DNA sequences using the predefined genome code. Protein sequences are more reliable than DNA sequence because of the redundancy of the genetic code. Two protein sequences are believed to be functional and structurally related if they show similar sequence identity or homology. These conserved patterns are of interest for the protein classification task.

A protein sequence is made from combinations of variable length of 20 amino acids $\sum = A, C, D, E, F, G, H, I, K, L, M, N, P, Q, R, S, T, V, W, Y$. The n-grams or k-tuples features will be extracted as an input vector of the neural network classifier. The n-gram features are a pair of values (v_i, c_i), where v_i is the feature i and c_i is the counts of this feature in a protein sequence for $i = 1, \cdots, 20^n$. In general, a feature is the number of occurrences of an animal in a protein sequence. These features are all the possible combinations of n letters from the set $\sum$. For example, the 2-gram (400 in total) features are $(AA, AC, \cdots, AY, CA, CC, \cdots, CY, \cdots, YA, \cdots, YY)$. Consider a protein sequence $VAAGTVAGT$, the extracted 2-gram features are $\{(VA, 2), (AA, 1), (AG, 2), (GT, 2), (TV, 1)\}$. The 6-letter exchange group is another commonly used piece of information. The 6-letter group actually contains 6 combinations of the letters from the set $\sum$. These combinations are $A = \{H, R, K\}$, $B = \{D, E, N, Q\}$, $C = \{C\}$, $D = \{S, T, P, A, G\}$, $E = \{M, I, L, V\}$ and $F = \{F, Y, W\}$. For example, the protein sequence $VAAGTVAGT$ mentioned above will be transformed using 6-letter exchange group as $EDDDDEDDD$ and their 2-gram features are $\{(DE, 1), (ED, 2), (DD, 5)\}$. We will use e_n and a_n to represent n-gram features from a 6-letter group and 20 letters set. Each sets of n-grams features, i.e., e_n and a_n, from a protein sequence will be scaled separately to avoid skew in the counts value using the following formula:

$$\overline{x} = \frac{x}{L - n + 1} \tag{12}$$

where x represents the count of generic gram feature, $\overline{x}$ is the normalized x, which will be the inputs of the classifiers; L is the length of the protein sequence and n is the size of n-gram features.

In this study, the protein sequences covering ten superfamilies (classes) were obtained from the PIR databases comprised by PIR1 and PIR2[2]. The ten superfamiles to be trained/classified in this study are: Cytochrome c (113/17), Cytochrome $c6$ (45/14), Cytochrome b (73/100), Cytochrome $b5$ (11/14), Triose-phosphate isomerase (14/44), Plastocyanin (42/56), Photosystem II D2 protein (30/45), Ferredoxin (65/33), Globin (548/204), and Cytochrome $b6 - f$ complex $4.2K$ (8/6). The 56 features were extracted and comprised by e_2 and a_1.

IV. Performance Evaluation

Levenberg-Marquardt (LM) learning algorithm, one of the fastest algorithm of BP's variants, is used in our simulations. Besides faster convergence achieved by the LM algorithm, we also avoid all the headaches of tuning the learning rate η. The LM provided in MATLAB has been well optimized at quite low level, such as the binary code level. The release MATLAB version of ELM codes can be downloaded from the ELM host site: http://www.ntu.edu.sg/home/egbhuang/. For BP,

[2]Protein Information Resources (PIR), http://pir.Georgetown.edu

the method of early stopping was applied to improve the generalization performance using validation data.

All the simulations are based on a 3.0GHz CPU with 768MB memory. All the simulation results are averaged over 50 trials. It was found during our simulations that BP required large memory and for this applications it would be out of memory when more than 40 hidden neurons are assigned to the BP networks.

A. Training data from PIR1 and testing from PIR2

The 949 protein sequences selected from PIR1 were used as the training data and the 533 protein sequences selected from PIR2 as the test data.

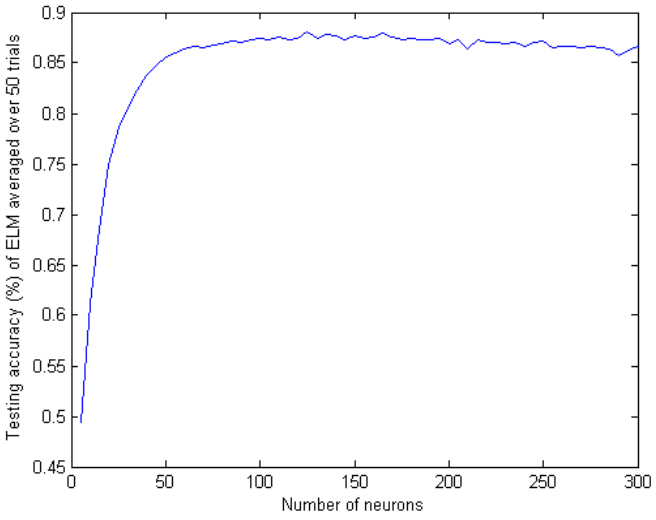

Fig. 1. The relationship between the generalization performance and number of hidden neurons: ELM where protein sequences training dataset (PIR1) and testing dataset (PIR2) are fixed.

The performance of BP classifier is shown in Table I and Figure 2. The performance of ELM classifier is shown in Figure 1. As observed from Table II, the best generalization performance obtained by ELM is 88.03% while the best generalization performance obtained by BP is 86.929%, thus ELM can obtain better generalization than BP. As observed from Table II, ELM can run 11038 times faster than BP in the case when best generalization performances are obtained for both BP and ELM. It also be seen from Table II, the standard deviation of the generalization performance of ELM is much smaller than BP's, meaning that ELM may run much more stable than BP. The relationship between the spent learning time and the number of neurons for ELM and BP are shown in Figure 3 and Figure 4, respectively.

B. Reshuffling training and testing data from PIR1 and PIR2

As shown above, the ELM can obtain better generalization performance than BP if PIR1 is chosen as the training dataset and PIR2 as the testing dataset. One might

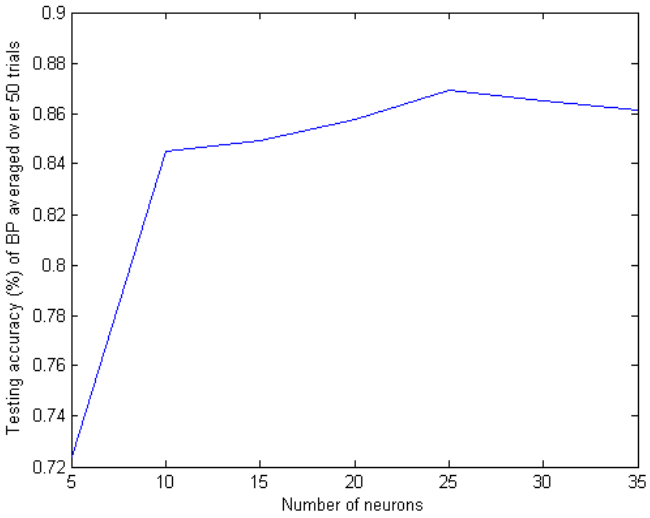

Fig. 2. The relationship between the generalization performance and number of hidden neurons: BP where protein sequences training dataset (PIR1) and testing dataset (PIR2) are fixed.

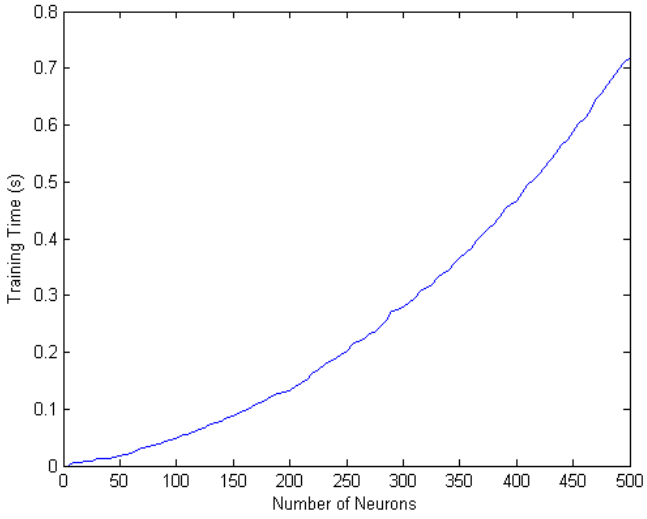

Fig. 3. The relationship between the spent training time and number of hidden neurons: ELM.

think that the this kind of selection of training dataset and testing dataset may inadvertently give preference to ELM classifier. In order to shown whether ELM can work generally better than BP for this protein sequence application, one method is to mix together the 949 protein sequences from PIR1 and the 533 protein sequences PIR2 and then randomly select the 949 protein sequences from the mixed dataset as the training dataset and the rest 533 protein sequences as the testing dataset at each trial. 50 trials have been conducted for both ELM and BP and the averaged results are obtained.

The performance of BP classifier is shown in Table

Training Time (seconds)	Training (%)		Testing (%)		No of Neurons
	Rate	Dev	Rate	Dev	
56.44	91.21	3.0284	72.397	4.9446	5
150.14	98.502	1.3539	84.502	3.2615	10
307.65	98.386	2.9586	84.906	5.6807	15
529.01	98.997	1.4379	85.775	3.1658	20
744.63	**99.364**	**0.8788**	**86.929**	**3.1565**	**25**
1045.7	99.355	1.3606	86.502	4.8249	30
1507.4	99.412	1.5512	86.15	4.2458	35
Out of memory (768MB memory, 3GHZ CPU) when number of neurons ≥ 40					

TABLE I

Performance of BP classifier: fixed protein sequences training dataset (PIR1) and testing dataset (PIR2).

Algorithms	Training Time		Training (%)		Testing (%)		No of Neurons
	(seconds)	Speedup	Rate	Dev	Rate	Dev	
ELM	0.06746	11038	99.63	0.15	88.03	1.19	125
BP	744.63	1	99.364	0.8788	86.929	3.1565	25

TABLE II

Performance comparison of different classifiers: fixed protein sequences training dataset (PIR1) and testing dataset (PIR2).

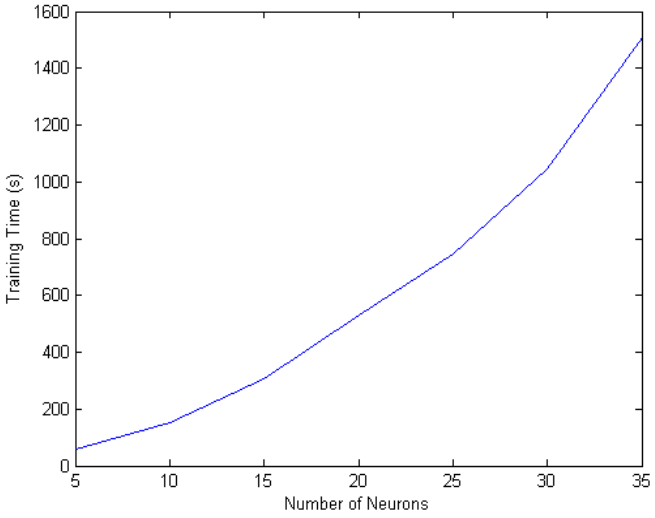

Fig. 4. The relationship between the spent training time and number of hidden neurons: BP.

III and Figure 6. The performance of ELM classifier is shown in Figure 5. As observed from Table IV, the best generalization performance obtained by ELM is 96.738% while the best generalization performance obtained by BP is 96.037%, thus ELM can still obtain better generalization than BP. As observed from Table IV, ELM can run 17417 times faster than BP in the case when best generalization performances are obtained for both BP and ELM. Similarly, it also be seen from Table IV, ELM may run much more stable than BP.

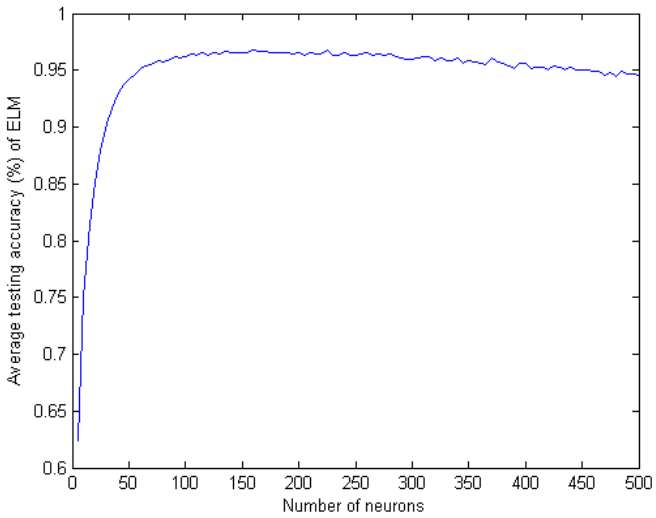

Fig. 5. The relationship between the generalization performance and number of hidden neurons: ELM where protein sequences training dataset and testing dataset are randomly generated from mixed protein sequences database.

V. Conclusions

In this paper, we have evaluated the performance of two main neural network classifiers, namely BP and ELM on classification of protein sequences with ten super-families (classes). This study demonstrates that ELM needs much

Training Time (seconds)	Training (%)		Testing (%)		No of Neurons
	Rate	Dev	Rate	Dev	
66.604	88.35	1.9389	85.685	1.9389	5
171.02	98.729	1.276	94.524	1.276	10
374.12	99.45	0.88195	94.757	0.88195	15
624.89	99.6	0.53561	95.558	0.53561	20
843.68	99.511	1.0176	95.551	1.0176	25
1228.4	99.576	1.2518	95.378	1.2518	30
1737.5	**99.739**	**0.53529**	**96.037**	**0.53529**	**35**
Out of memory (768MB memory, 3GHZ CPU) when number of neurons ≥ 40					

TABLE III

Performance of BP classifier: randomly generated protein sequences training dataset and testing dataset.

Algorithms	Training Time		Training (%)		Testing (%)		No of Neurons
	(seconds)	Speedup	Rate	Dev	Rate	Dev	
ELM	0.099758	17417	99.758	0.15967	96.738	0.86281	160
BP	1737.5	1	99.739	0.53529	96.037	1.2132	35

TABLE IV

Performance comparison of different classifiers: randomly generated protein sequences training dataset and testing dataset.

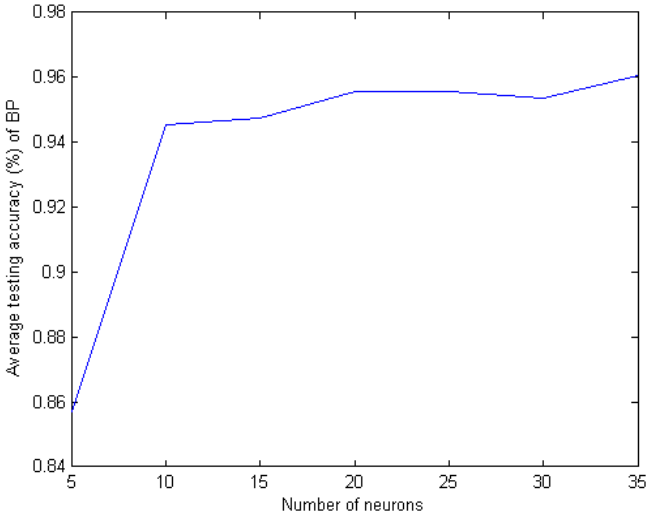

Fig. 6. The relationship between the generalization performance and number of hidden neurons: BP where protein sequences training dataset and testing dataset are randomly generated from mixed protein sequences database.

less training time compared to conventional BP classifiers. The classification accuracy of ELM is much better than BP. Compared with BP, ELM can be implemented easily since there is no parameter to be tuned except for network size which is common to BP.

References

[1] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme learning machine: A new learning scheme of feedforward neural networks," in Proceedings of International Joint Conference on Neural Networks (IJCNN2004), (Budapest, Hungary), 25-29 July, 2004.

[2] G.-B. Huang and C.-K. Siew, "Extreme learning machine: RBF network case," in Proceedings of the Eighth International Conference on Control, Automation, Robotics and Vision (ICARCV 2004), (Kunming, China), 6-9 Dec, 2004.

[3] G.-B. Huang and C.-K. Siew, "Extreme learning machine with randomly assigned RBF kernels," International Journal of Information Technology, vol. 11, no. 1, 2005.

[4] E. B. Baum., "On the capabilities of multilayer perceptrons," Journal of Complexity, vol. 4, pp. 193–215, 1988.

[5] G.-B. Huang and H. A. Babri, "Upper bounds on the number of hidden neurons in feedforward networks with arbitrary bounded nonlinear activation functions," IEEE Transactions on Neural Networks, vol. 9, pp. 224–229, Jan. 1998.

[6] G.-B. Huang, "Learning capability and storage capacity of two-hidden-layer feedforward networks," IEEE Transactions on Neural Networks, vol. 14, pp. 274–281, Mar. 2003.

[7] S. Tamura and M. Tateishi, "Capabilities of a four-layered feedforward neural network: Four layers versus three," IEEE Transactions on Neural Networks, vol. 8, no. 2, pp. 251–255, 1997.

[8] D. Serre, "Matrices: Theory and applications," Springer-Verlag New York, Inc, 2002.

[9] P. L. Bartlett, "The sample complexity of pattern classification with neural networks: The size of the weights is more important than the size of the network," IEEE Transactions on Information Theory, vol. 44, pp. 525–536, Mar. 1998.

[10] D. Wang, N. K. Lee, and T. S. Dillon, "Extraction and optimization of fuzzy protein sequence classification rules using grbf neural networks," Neural Information Processing - Letters and Review, vol. 1, no. 1, pp. 53–59, 2002.