The extension version of this paper can be found in G.-B. Huang, Q.-Y.

Extreme Learning Machine: A New Learning Scheme of Feedforward Neural Networks

Guang-Bin Huang, Qin-Yu Zhu, and Chee-Kheong Siew
School of Electrical and Electronic Engineering
Nanyang Technological University
Nanyang Avenue, Singapore 639798
E-mail: egbhuang@ntu.edu.sg

Abstract—It is clear that the learning speed of feedforward neural networks is in general far slower than required and it has been a major bottleneck in their applications for past decades. Two key reasons behind may be: 1) the slow gradient-based learning algorithms are extensively used to train neural networks, and 2) all the parameters of the networks are tuned iteratively by using such learning algorithms. Unlike these traditional implementations, this paper proposes a new learning algorithm called extreme learning machine (ELM) for single-hidden layer feedforward neural networks (SLFNs) which randomly chooses the input weights and analytically determines the output weights of SLFNs. In theory, this algorithm tends to provide the best generalization performance at extremely fast learning speed. The experimental results based on real-world benchmarking function approximation and classification problems including large complex applications show that the new algorithm can produce best generalization performance in some cases and can learn much faster than traditional popular learning algorithms for feedforward neural networks.

I. Introduction

From a mathematical point of view, research on the approximation capabilities of feedforward neural networks has focused on two aspects: universal approximation on compact input sets and approximation in a finite set. Many researchers have explored the universal approximation capabilities of standard multi-layer feedforward neural networks[1], [2], [3]. In real applications, the neural networks are trained in finite training set. For function approximation in a finite training set, Huang and Babri[4] shows that a sigle-hidden layer feedforward neural network (SLFN) with at most N hidden neurons and with almost any nonlinear activation function can learn N distinct observations with zero error. It should be noted that the input weights (linking the input layer to the first hidden layer) need to be adjusted in all these previous theoretical research works as well as in almost all practical learning algorithms of feedforward neural networks.

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 has been shown [5], [6] that SLFNs (with N hidden neurons) with arbitrarily chosen input weights can learn N distinct observations with arbitrarily small error. Unlike the popular thinking and most practical implementations that all the parameters of the feedforward networks need to be tuned, one may not necessarily adjust the input weights and first hidden layer biases in applications. In fact, some simulation results on artificial and real large applications in our work [7] have shown that this method not only makes learning extremely fast but also produces good generalization performance. Recently, it has been further rigorously proved in our work [8] that SLFNs with arbitrarily assigned input weights and hidden layer biases and with almost any nonzero activation function can universally approximate any continuous functions on any compact input sets. These research results imply that in the applications of feedforward neural networks input weights may not be necessarily adjusted at all.

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. Based on this concept, this paper proposes a simple learning algorithm for SLFNs called extreme learning machine (ELM) whose learning speed can be thousands of times faster than traditional feedforward network learning algorithms like back-propagation algorithm while obtaining better generalization performance. Different from traditional learning algorithms the proposed learning algorithm not only tends to reach the smallest training error but also the smallest norm of weights. Bartlett's theory on the generalization performance of feedforward neural networks[9] states that for feedforward neural networks reaching smaller training error, the smaller the norm of weights is, the better generalization performance the networks tend to have. Therefore, the proposed learning algorithm tends to have better generalization performance for feedforward neural networks.

Proceedings of International Joint Conference on Neural Networks (IJCNN2004), 25-29 July, 2004, B

As the new proposed learning algorithm tends to reach the smallest training error, obtain the smallest norm of weights and the best generalization performance, and runs extremely fast, in order to differentiate it from the other popular SLFN learning algorithms, it is called the Extreme Learning Machine (ELM) in the context of this paper.

This paper is organized as follows. Section II introduces the Moore-Penrose generalized inverse and the minimum norm least-squares solution of a general linear system which play an important role in developing our new ELM learning algorithm. Section III proposes the new ELM learning algorithm for single-hidden layer feedforward neural networks (SLFNs). Performance evaluation is presented in Section IV. Discussions and conclusions are given in Section V.

II. Preliminaries

In this section, the Moore-Penrose generalized inverse is introduced. We also consider in this section the minimum norm least-squares solution of a general linear system $\mathbf{Ax} = \mathbf{y}$ in Euclidean space, where $\mathbf{A} \in \mathbf{R}^{m\times n}$ and $\mathbf{y} \in \mathbf{R}^m$. As shown in [5], [6], the SLFNs are actually a linear system if the input weights and the hidden layer biases can be chosen arbitrarily.

A. Moore-Penrose Generalized Inverse

The resolution of a general linear system $\mathbf{Ax} = \mathbf{y}$, where $\mathbf{A}$ may be singular and may even not be square, can be made very simple by the use of the Moore-Penrose generalized inverse[10].

Definition 2.1: [10] A matrix $\mathbf{G}$ of order $n \times m$ is the Moore-Penrose generalized inverse of matrix $\mathbf{A}$ of order $m \times n$, if

$$\mathbf{AGA} = \mathbf{A}, \mathbf{GAG} = \mathbf{G}, (\mathbf{AG})^T = \mathbf{AG}, (\mathbf{GA})^T = \mathbf{GA} \tag{1}$$

For the sake of convenience, the Moore-Penrose generalized inverse of matrix $\mathbf{A}$ will be denoted by $\mathbf{A}^\dagger$.

B. Minimum Norm Least-Squares Solution of General Linear System

For a general linear system $\mathbf{Ax} = \mathbf{y}$, we say that $\hat{\mathbf{x}}$ is a least-squares solution (l.s.s) if

$$\|\mathbf{A}\hat{\mathbf{x}} - \mathbf{y}\| = \min_{\mathbf{x}} \|\mathbf{Ax} - \mathbf{y}\| \tag{2}$$

where $\|\cdot\|$ is a norm in Euclidean space.

Definition 2.2: $\mathbf{x}_0 \in \mathbf{R}^n$ is said to be a minimum norm least-squares solution of a linear system $\mathbf{Ax} = \mathbf{y}$ if for any $\mathbf{y} \in \mathbf{R}^m$

$$\|\mathbf{x}_0\| \le \|\mathbf{x}\|, \forall \mathbf{x} \in \{\mathbf{x} : \|\mathbf{Ax} - \mathbf{y}\| \le \|\mathbf{Az} - \mathbf{y}\|, \forall \mathbf{z} \in \mathbf{R}^n\} \tag{3}$$

That means, a solution $\mathbf{x}_0$ is said to be a minimum norm least-squares solution of a linear system $\mathbf{Ax} = \mathbf{y}$ if it has the smallest norm among all the least-squares solutions.

Theorem 2.1: {p. 147 of [10]} Let there exist a matrix $\mathbf{G}$ such that $\mathbf{Gy}$ is a minimum norm least-squares solution of a linear system $\mathbf{Ax} = \mathbf{y}$. Then it is necessary and sufficient that $\mathbf{G} = \mathbf{A}^\dagger$, the Moore-Penrose generalized inverse of matrix $\mathbf{A}$.

III. Extreme Learning Machine

In Section II we have briefed the Moore-Penrose inverse and the smallest norm least-squares solution of a general linear system $\mathbf{Ax} = \mathbf{y}$. We can now propose an extremely fast learning algorithm for the single hidden layer feedforward networks (SLFNs) with $\tilde{N}$ hidden neurons, where $\tilde{N} \le N$, the number of training samples.

A. Approximation problem of SLFNs

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

$$\sum_{i=1}^{\tilde{N}} \beta_i g(\mathbf{w}_i \cdot \mathbf{x}_j + b_i) = \mathbf{o}_j, \;\; j = 1, \cdots, N, \tag{4}$$

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 output neurons are chosen linear in this paper.

That standard SLFNs with $\tilde{N}$ hidden neurons 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

$$\sum_{i=1}^{\tilde{N}} \beta_i g(\mathbf{w}_i \cdot \mathbf{x}_j + b_i) = \mathbf{t}_j, \;\; j = 1, \cdots, N. \tag{5}$$

The above N equations can be written compactly as:

$$\mathbf{H}\beta = \mathbf{T} \tag{6}$$

where

$$\mathbf{H}(\mathbf{w}_1, \cdots, \mathbf{w}_{\tilde{N}}, b_1, \cdots, b_{\tilde{N}}, \mathbf{x}_1, \cdots, \mathbf{x}_N) = \begin{bmatrix} g(\mathbf{w}_1 \cdot \mathbf{x}_1 + b_1) & \cdots & g(\mathbf{w}_{\tilde{N}} \cdot \mathbf{x}_1 + b_{\tilde{N}}) \\ \vdots & \cdots & \vdots \\ g(\mathbf{w}_1 \cdot \mathbf{x}_N + b_1) & \cdots & g(\mathbf{w}_{\tilde{N}} \cdot \mathbf{x}_N + b_{\tilde{N}}) \end{bmatrix}_{N\times\tilde{N}} \tag{7}$$

$$\beta = \begin{bmatrix} \beta_1^T \\ \vdots \\ \beta_{\tilde{N}}{}^T \end{bmatrix}_{\tilde{N}\times m} \quad \text{and} \quad \mathbf{T} = \begin{bmatrix} \mathbf{t}_1^T \\ \vdots \\ \mathbf{t}_N^T \end{bmatrix}_{N\times m} \tag{8}$$

As named in Huang and Babri[4] and Huang[6], $\mathbf{H}$ is called the hidden layer output matrix of the neural network; the ith column of $\mathbf{H}$ is the ith hidden neuron's output vector with respect to inputs $\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_N$.

B. Gradient-Based Learning Algorithms

As analyzed in pervious works[4], [5], [6], if the number of hidden neurons is equal to the number of distinct training samples, $\tilde{N} = N$, matrix $\mathbf{H}$ is square and invertible, and SLFNs can approximate these training samples with zero error. However, 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, instead one may need to find specific $\hat{\mathbf{w}}_i, \hat{b}_i, \hat{\beta}$ $(i = 1, \cdots, \tilde{N})$ such that

$$\begin{aligned}&\|\mathbf{H}(\hat{\mathbf{w}}_1, \cdots, \hat{\mathbf{w}}_{\tilde{N}}, \hat{b}_1, \cdots, \hat{b}_{\tilde{N}})\hat{\beta} - \mathbf{T}\| \\ &= \min_{\mathbf{w}_i, b_i, \beta} \|\mathbf{H}(\mathbf{w}_1, \cdots, \mathbf{w}_{\tilde{N}}, b_1, \cdots, b_{\tilde{N}})\beta - \mathbf{T}\|\end{aligned} \tag{9}$$

which is equivalent to minimizing the cost function

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

When $\mathbf{H}$ is unknown gradient-based learning algorithms are generally used to search the minimum of $\|\mathbf{H}\beta - \mathbf{T}\|$. In the minimization procedure by using gradient-based algorithms, vector $\mathbf{W}$ which is the set of weights $(\mathbf{w}_i, \beta_i)$ and biases (b_i) parameters $\mathbf{W}$ is iteratively adjusted as follows:

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

Here η is a learning rate. The popular learning algorithm used in feedforward 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 on back-propagation learning algorithms:

1) When the learning rate η is too small, the learning algorithm converges very slowly. However, when η is too large, the algorithm becomes unstable and diverges.
2) Another peculiarity of the error surface that impacts the performance of the back-propagation learning algorithm is the presence of local minima[11]. It is undesirable that the learning algorithm stops at a local minimum if it is located far above a global minimum.
3) Neural network may be over-trained by using back-propagation algorithms and obtain worse generalization performance. Thus, validation and suitable stopping methods are required in the cost function minimization procedure.
4) Gradient-based learning is very time-consuming in most applications.

The aim of this paper is to solve the above issues related with gradient-based algorithms and propose an efficient learning algorithm for feedforward neural networks.

C. Minimum Norm Least-Squares Solution of SLFN

As mentioned in Section I, it is very interesting and surprising that unlike the most common understanding 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.

Our recent work [6] also shows that SLFNs (with infinite differential activation functions) with the input weights chosen arbitrarily can learn distinct observations with arbitrarily small error. In fact, simulation results on artificial and real world cases done in [7] as well as in this paper have further demonstrated that no gain is possible by adjusting the input weights and the hidden layer biases. Recently, it has been further rigorously proved in our work[8] that unlike the traditional function approximation theories[1], which require to adjust input weights and hidden layer biases, the feedforward networks with arbitrarily assigned input weights and hidden layer biases and with almost all nonzero activation functions can universally approximate any continuous functions on any compact input sets.

These research results (simulations and theories) show that the input weights and the hidden layer biases of SLFNs need not be adjusted at all and can be arbitrarily given. For fixed input weights $\mathbf{w}_i$ and the hidden layer biases b_i, seen from equation (9), to train an SLFN is simply equivalent to finding a least-squares solution $\hat{\beta}$ of the linear system $\mathbf{H}\beta = \mathbf{T}$:

$$\begin{aligned}&\|\mathbf{H}(\mathbf{w}_1, \cdots, \mathbf{w}_{\tilde{N}}, b_1, \cdots, b_{\tilde{N}})\hat{\beta} - \mathbf{T}\| \\ &= \min_{\beta} \|\mathbf{H}(\mathbf{w}_1, \cdots, \mathbf{w}_{\tilde{N}}, b_1, \cdots, b_{\tilde{N}})\beta - \mathbf{T}\|\end{aligned} \tag{12}$$

According to Theorem 2.1, the smallest norm least-squares solution of the above linear system is:

$$\hat{\beta} = \mathbf{H}^{\dagger}\mathbf{T} \tag{13}$$

Remarks: As discussed in Section II, we have the following important properties:

1) Minimum training error. The special solution $\hat{\beta} = \mathbf{H}^{\dagger}\mathbf{T}$ is one of the 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:

   $$\|\mathbf{H}\hat{\beta} - \mathbf{T}\| = \|\mathbf{H}\mathbf{H}^{\dagger}\mathbf{T} - \mathbf{T}\| = \min_{\beta} \|\mathbf{H}\beta - \mathbf{T}\| \tag{14}$$

   Although almost all learning algorithms wish to reach the minimum training error, however, most of them cannot reach it because of local minimum or infinite training iteration is usually not allowed in applications.
2) Smallest norm of weights and best generalization performance. In further, the special solution $\hat{\beta} = \mathbf{H}^{\dagger}\mathbf{T}$

has the smallest norm among all the least-squares solutions of $\mathbf{H}\beta = \mathbf{T}$:

$$\|\hat{\beta}\| = \|\mathbf{H}^{\dagger}\mathbf{T}\| \leq \|\beta\|,$$
$$\forall \beta \in \{\beta : \|\mathbf{H}\beta - \mathbf{T}\| \leq \|\mathbf{Hz} - \mathbf{T}\|, \forall \mathbf{z} \in \mathbf{R}^{\tilde{N}\times N}\} \qquad (15)$$

As pointed out by Bartlett[12], [9], for feedforward networks with many small weights but small squared error on the training examples, the Vapnik-Chervonenkis (VC) dimension (and hence number of parameters) is irrelevant to the generalization performance. Instead, the magnitude of the weights in the network is more important. The smaller the weights are, the better generalization performance the network tends to have. As analyzed above, our method not only reaches the smallest squared error on the training examples but also obtains the smallest weights. Thus, it is reasonable to think that our method tends to have better generalization performance. It should be worth pointing out that it may be difficult for gradient-based learning algorithms like back-propagation to reach the best generalization performance since they only try to obtain the smallest training errors without considering the magnitude of the weights.

3) The minimum norm least-squares solution of $\mathbf{H}\beta = \mathbf{T}$ is unique, which is $\hat{\beta} = \mathbf{H}^{\dagger}\mathbf{T}$.

D. Learning Algorithm for SLFNs

Thus, a simple learning method for SLFNs called the extreme learning machine (ELM)[1] can be summarized as follows:

Algorithm ELM : 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:

$$\beta = \mathbf{H}^{\dagger}\mathbf{T} \qquad (16)$$

where $\mathbf{H}$, β and $\mathbf{T}$ are defined as formula (7) and (8).

IV. Performance Evaluation

In this section, the performance of the proposed ELM learning algorithm is compared with the popular algorithms of feedforward neural networks like the conventional back-propagation (BP) algorithm on some benchmarking problems in the function approximation and classification areas. This paper mainly focuses on feedforward neural networks and aims to propose a new learning algorithm to train feedforward networks efficiently. Although Support Vector Machines (SVMs) are obviously different from standard feedforward networks and it is not the objective of the paper to systematically compare the differences between SVM and feedforward networks, the performance comparison between SVMs and our ELM are also simply conducted.

All the simulations for the BP and ELM algorithms are carried out in MATLAB 6.5 environment running in a Pentium 4, 1.9 GHZ CPU. Although there are many variants of BP algorithm, a faster BP algorithm called Levenberg-Marquardt algorithm is used in our simulations.

The simulations for SVM are carried out using compiled C-coded SVM packages: LIBSVM[13] running in the same PC. When comparing learning speed of ELM and SVMs, readers should bear in mind that a C implementation would be much faster than MATLAB. So a comparison with LIBSVM could give SVM an advantage. The kernel function used in SVM is radial basis function whereas the activation function used in our proposed algorithms is a simple sigmoidal function $g(x) = 1/(1+\exp(-x))$. In order to compare BP and our ELM, BP and ELM are always assigned the same of number of hidden neurons for same applications.

A. Benchmarking with Function Approximation Problem

California Housing is a dataset obtained from the StatLib repository[2]. There are 20,640 observations for predicting the price of houses in California. Information on the variables were collected using all the block groups in California from the 1990 Census. In this sample a block group on average includes 1425.5 individuals living in a geographically compact area. Naturally, the geographical area included varies inversely with the population density. Distances among the centroids of each block group were computed as measured in latitude and longitude. All the block groups reporting zero entries for the independent and dependent variables were excluded. The final data contained 20,640 observations on 9 variables, which consists of 8 continuous inputs (median income, housing median age, total rooms, total bedrooms, population, households, latitude, and longitude) and one continuous output (median house value). In our simulations, 8,000 training data and 12,640 testing data randomly generated from the California Housing database for each trial.

For simplicity, the 8 input attributes and one output have been normalized to the range $[0, 1]$ in our experiment. The parameter C is tuned and set as $C = 500$ in SVR algorithm. 50 trials have been conducted for all the algorithms and the average results are shown in Table I. Seen from Table I, the learning speed of our ELM algorithm is more than 1,000 and 2,000 times faster than BP and SVM for this case. The generalization performance obtained by the ELM algorithm is very close to the generalization performance of BP and SVMs.

[1] Source codes can be downloaded from http://www.ntu.edu.sg/eee/icis/cv/egbhuang.htm.

[2] http://www.niaad.liacc.up.pt/∼ltorgo/Regression/cal_housing.html

TABLE I
PERFORMANCE COMPARISON IN CALIFORNIA HOUSING PREDICTION APPLICATION.

Algo	Time (seconds)		Training	Testing	No of SVs/
	Training	Testing	RMS	RMS	Neurons
ELM	0.272	0.143	0.1358	0.1365	20
BP	295.23	0.286	0.1369	0.1426	20
SVM	558.4137	20.9763	0.1267	0.1275	2534.0

B. Benchmarking with Real Medical Diagnosis Application

The performance comparison of the new proposed ELM algorithm and many other popular algorithms has been conducted for a real medical diagnosis problem: Diabetes[3], using the "Pima Indians Diabetes Database" produced in the Applied Physics Laboratory, Johns Hopkins University, 1988. The diagnostic, binary-valued variable investigated is whether the patient shows signs of diabetes according to World Health Organization criteria (i.e., if the 2 hour post-load plasma glucose was at least 200 mg/dl at any survey examination or if found during routine medical care). The database consists of 768 women over the age of 21 resident in Phoenix, Arizona. All examples belong to either positive or negative class. All the input values are within $[0, 1]$. For this problem, 75% and 25% samples are randomly chosen for training and testing at each trial, respectively. The parameter C of SVM algorithm is tuned and set as: $C = 10$ and the rest parameters are set as default.

50 trials have been conducted for all the algorithms and the average results are shown in Table II. Seen from Table II, in our simulations SVM can reach the testing rate 77.31% with 317.16 support vectors at average. Rătsch et al[14] obtained a testing rate 76.50% for SVM which is slightly lower than the SVM result we obtained. However, the new ELM learning algorithm can achieve the average testing rate 76.54% with 20 neurons, which is higher than the results obtained by many other popular algorithms such as bagging and boosting methods[15], C4.5[15], and RBF[16] (cf. Table III). BP algorithm performs very poor in our simulations for this case. It can also be seen that the ELM learning algorithm run around 1,000 times faster than BP, and 12 times faster than SVM for this small problem without considering that C executable environment may run much faster than MATLAB environment.

TABLE II
PERFORMANCE COMPARISON IN REAL MEDICAL DIAGNOSIS APPLICATION: DIABETES.

Algo	Training Time	Success Rate		No of SVs/
	(Seconds)	Training	Testing	Neurons
ELM	0.015	78.71%	76.54%	20
BP	16.196	92.86%	63.45%	20
SVM	0.1860	78.76%	77.31%	317.16

TABLE III
PERFORMANCE COMPARISON IN REAL MEDICAL DIAGNOSIS APPLICATION: DIABETES.

Algorithms	Testing Rate
ELM	76.54%
SVM[14]	76.50%
AdaBoost[15]	75.60%
C4.5[15]	71.60%
RBF[16]	76.30%

C. Benchmarking with Real-World Large Complex Application

We have also tested the performance of our ELM algorithm for large complex applications such as Forest Cover Type Prediction.[4]

Forest Cover Type Prediction is an extremely large complex classification problem with seven classes. The forest cover type for 30 x 30 meter cells was obtained from US Forest Service (USFS) Region 2 Resource Information System (RIS) data. There are 581,012 instances (observations) and 54 attributes per observation. In order to compare with the previous work[17], similarly it was modified as a binary classification problem where the goal was to separate class 2 from the other six classes. There are 100,000 training data and 481,012 testing data. The parameters for the SVM are $C = 10$ and $\gamma = 2$.

50 trials have been conducted for the ELM algorithm, and 1 trial for SVM since it takes very long time to train SLFNs using SVM for this large complex case[5]. Seen from Table IV, the proposed ELM learning algorithm obtains better generalization performance than SVM learning algorithm. However, the proposed ELM learning algorithm only spent 1.5 minutes on learning while SVM need to spend nearly 12 hours on training. The learning speed has dramatically been increased 430 times. On the other hand, since the support vectors obtained by SVM is much larger than the required hidden neurons in ELM, the testing time spent SVMs for this large testing data set is more than 480 times than the ELM. It takes more than 5.5 hours for the SVM to react to the 481,012 testing samples. However, it takes only less than 1 minute for the obtained ELM to react to the testing samples. That means, after trained and deployed the ELM may react to new observations much faster than SVMs in such real application. It should be noted that in order to obtain as good performance as possible for SVM, long time effort has been made to find the appropriate parameters for SVM. In fact the generalization performance of SVM we obtained in our simulation for this case is much higher than the one reported in literature[17].

V. Discussions and Conclusions

This paper proposed a new learning algorithm for single-hidden layer feedforward neural networks (SLFNs)

[3] ftp://ftp.ira.uka.de/pub/neuron/proben1.tar.gz

[4] http://www.ics.uci.edu/~mlearn/MLRepository.html.

[5] Actually we have tested SVM for this case many times and always obtained similar results as presented here.

TABLE IV
PERFORMANCE COMPARISON OF THE ELM, BP and SVM LEARNING ALGORITHMS IN FOREST TYPE PREDICTION APPLICATION.

Algo	Time (minutes)		Success Rate		No of SVs/ Neurons
	Training	Testing	Training	Testing	
ELM	1.6148	0.7195	92.35	90.21	200
SLFN[17]	12	N/A	82.44	81.85	100
SVM	693.60	347.78	91.70	89.90	31,806

called extreme learning machine (ELM), which has several interesting and significant features different from traditional popular gradient-based learning algorithms for feedforward neural networks:

1) The learning speed of ELM is extremely fast. It can train SLFNs much faster than classical learning algorithms. Previously, it seems that there exists a virtual speed barrier which most (if not all) classic learning algorithms cannot break through and it is not unusual to take very long time to train a feedforward network using classic learning algorithms even for simple applications.
2) Unlike the traditional classic gradient-based learning algorithms which intend to reach minimum training error but do not consider the magnitude of weights, the ELM tends to reach not only the smallest training error but also the smallest norm of weights. Thus, the proposed ELM tends to have the better generalization performance for feedforward neural networks.
3) Unlike the traditional classic gradient-based learning algorithms which only work for differentiable activation functions, the ELM learning algorithm can be used to train SLFNs with non-differentiable activation functions.
4) Unlike the traditional classic gradient-based learning algorithms facing several issues like local minimum, improper learning rate and overfitting, etc, the ELM tends to reach the solutions straightforward without such trivial issues. The ELM learning algorithm looks much simpler than most learning algorithms for feedforward neural networks.

It should be noted that gradient-based learning algorithms like back-propagation can be used for feedforward neural networks which have more than one hidden layers while the proposed ELM algorithm at its present form is still only valid for single-hidden layer feedforward networks (SLFNs). Fortunately, it has been found that SLFNs can approximate any continuous function[1], [8] and implement any classification application[18]. Thus, reasonably speaking the proposed ELM algorithm can be used generally in many cases and it needs to be investigated further in the future. On the other hand, more comprehensive experimental results of ELM on different aritificial and real world benchmark problems can be found in our technical report[6] [19].

References

[1] K. Hornik, “Approximation capabilities of multilayer feedforward networks,” Neural Networks, vol. 4, pp. 251–257, 1991.

[2] M. Leshno, V. Y. Lin, A. Pinkus, and S. Schocken, “Multilayer feedforward networks with a nonpolynomial activation function can approximate any function,” Neural Networks, vol. 6, pp. 861–867, 1993.

[3] Y. Ito, “Approximation of continuous functions on $\mathbf{R}^d$ by linear combinations of shifted rotations of a sigmoid function with and without scaling,” Neural Networks, vol. 5, pp. 105–115, 1992.

[4] 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, no. 1, pp. 224–229, 1998.

[5] 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.

[6] G.-B. Huang, “Learning capability and storage capacity of two-hidden-layer feedforward networks,” IEEE Transactions on Neural Networks, vol. 14, no. 2, pp. 274–281, 2003.

[7] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, “Real-time learning capability of neural networks,” in Technical Report ICIS/45/2003, (School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore), Apr. 2003.

[8] G.-B. Huang, L. Chen, and C.-K. Siew, “Universal approximation using incremental feedforward networks with arbitrary input weights,” in Technical Report ICIS/46/2003, (School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore), Oct. 2003.

[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, no. 2, pp. 525–536, 1998.

[10] D. Serre, “Matrices: Theory and applications,” Springer-Verlag New York, Inc, 2002.

[11] S. Haykin, “Neural networks: A comprehensive foundation,” New Jersey: Prentice Hall, 1999.

[12] P. L. Bartlett, “For valid generalization, the size of the weights is more important than the size of the network,” in Advances in Neural Information Processing Systems’1996 (M. Mozer, M. Jordan, and T. Petsche, eds.), vol. 9, pp. 134–140, MIT Press, 1997.

[13] C.-C. Chang and C.-J. Lin, “LIBSVM – a library for support vector machines,” in http://www.csie.ntu.edu.tw/∼cjlin/libsvm/, Deptartment of Computer Science and Information Engineering, National Taiwan University, Taiwan, 2003.

[14] G. Rätsch, T. Onoda, and K. R. Müller, “An improvement of AdaBoost to avoid overfitting,” in Proceedings of the 5th International Conference on Neural Information Processing (ICONIP’1998), 1998.

[15] Y. Freund and R. E. Schapire, “Experiments with a new boosting algorithm,” in International Conference on Machine Learning, pp. 148–156, 1996.

[16] D. R. Wilson and T. R. Martinez, “Heterogeneous radial basis function networks,” in Proceedings of the International Conference on Neural Networks (ICNN 96), pp. 1263–1267, June 1996.

[17] R. Collobert, S. Bengio, and Y. Bengio, “A parallel mixtures of SVMs for very large scale problems,” Neural Computation, vol. 14, pp. 1105–1114, 2002.

[18] G.-B. Huang, Y.-Q. Chen, and H. A. Babri, “Classification ability of single hidden layer feedforward neural networks,” IEEE Transactions on Neural Networks, vol. 11, no. 3, pp. 799–801, 2000.

[19] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, “Extreme learning machine,” in Technical Report ICIS/03/2004, (School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore), Jan. 2004.

[6] http://www.ntu.edu.sg/eee/icis/cv/egbhuang.htm.