IEICE TRANS. INF. & SYST., VOL.E91–D, NO.4 APRIL 2008

PAPER

Small Number of Hidden Units for ELM with Two-Stage Linear Model

Hieu Trung HUYNH[†a)], *Student Member and* **Yonggwan WON**[†b)], *Member*

SUMMARY The single-hidden-layer feedforward neural networks (SLFNs) are frequently used in machine learning due to their ability which can form boundaries with arbitrary shapes if the activation function of hidden units is chosen properly. Most learning algorithms for the neural networks based on gradient descent are still slow because of the many learning steps. Recently, a learning algorithm called extreme learning machine (ELM) has been proposed for training SLFNs to overcome this problem. It randomly chooses the input weights and hidden-layer biases, and analytically determines the output weights by the matrix inverse operation. This algorithm can achieve good generalization performance with high learning speed in many applications. However, this algorithm often requires a large number of hidden units and takes long time for classification of new observations. In this paper, a new approach for training SLFNs called least-squares extreme learning machine (LS-ELM) is proposed. Unlike the gradient descent-based algorithms and the ELM, our approach analytically determines the input weights, hidden-layer biases and output weights based on linear models. For training with a large number of input patterns, an online training scheme with sub-blocks of the training set is also introduced. Experimental results for real applications show that our proposed algorithm offers high classification accuracy with a smaller number of hidden units and extremely high speed in both learning and testing.

key words: *neural networks, single hidden-layer feedforward neural networks, extreme learning machine, least-squares scheme, linear model*

1. Introduction

An approach massively used in machine learning is the neural network. It can provide proper models for the various types of problems which are difficult to be solved by classical parametric techniques and approximate complex nonlinear mappings directly from the input patterns. Traditionally, training networks has based on gradient descent methods. However, they are generally very slow due to improper learning rates or local minima. There are many improvements proposed by researchers to obtain better learning performance [1]–[5]. A choice of the initial values for weights to improve the learning speed of networks was proposed by D. Nguyen and B. Widrow [1]. Jim Y. F. Yam and Tommy W. S. Chow [2] proposed a method based on multidimensional geometry to determine optimal biases and the magnitude of initial weight vectors. A recursive Levenberg-Marquardt algorithm uses the second-order information to overcome slow training convergence instead of using the first-order information of the cost function [6], [7]. In addition, there are many methods proposed by many researchers to overcome overfitting in training [8], [9]. However, most training algorithms based on the gradient descent are still slow due to the many learning steps which may be required to achieve the goal. In addition, several studies have reported that support vector machines (SVMs) are able to obtain higher classification accuracy than the other existing data classification algorithms [10]–[13]. However, they may take long time to select proper models for some applications. Therefore, the model selection has become a critical issue for the SVM which has been addressed by a number of recent works [14]–[16].

Nevertheless, Huang *et al.* [17] showed that a single hidden-layer feedforward neural network (SLFN) with any continuous bounded non-constant activation function or any bounded activation function which has unequal limits at infinities can form decision regions with arbitrary shapes. A learning algorithm called extreme learning machine (ELM) was proposed for training SLFNs [18], [19]. It randomly chooses the input weights and hidden-layer biases, and then the output weights of the SLFN can be calculated through the inverse operation of the output matrix of hidden layer. This algorithm provides better generalization performance with high learning speed in many applications, even when compared to the SVM approaches [18]. However, the ELM often requires a large number of hidden units and takes long time for classification of input patterns.

In this paper, we investigate a new approach for training SLFNs called least squares extreme learning machine (LS-ELM) which can reduce the number of hidden units while producing better performance and faster classification than other ELMs. Unlike the ELM algorithms, our approach analytically determines the input weights, hidden-layer biases and output weights based on a two-stage linear model. First, the input weights and the hidden-layer biases are determined through the pseudo-inverse operation of the matrix of training data, and then the output weights are determined through the pseudo-inverse operation of the output matrix of hidden layer. For training with a large number of input patterns, an online training scheme with sub-blocks of the training data set is also introduced. Experimental results show that this approach yields good classification performance with significant reduction of the number of hidden units, and therefore both learning and testing speeds are extremely high.

The rest of this paper is organized as follows. Section 2 reviews the single hidden-layer feedforward neural net-

Manuscript received July 25, 2007.
Manuscript revised December 14, 2007.

[†]The authors are with the Department of Computer Engineering, Chonnam National University, Gwangju 500–757, Korea.
a) E-mail: hthieu@hcmut.edu.vn
b) E-mail: ykwon@chonnam.ac.kr (Corresponding author)

DOI: 10.1093/ietisy/e91–d.4.1042

Copyright © 2008 The Institute of Electronics, Information and Communication Engineers

works (SLFNs) and the extreme learning machine (ELM). In Sect. 3, we propose the new approach for training SLFNs called least-squares extreme learning machine (LS-ELM), which is based on the two-stage linear model for determining the network weights. Section 4 presents the online training scheme with the two-stage linear model. Experimental results and analysis are shown in Sect. 5. Finally, we make a conclusion in Sect. 6.

2. Review of Related Works

2.1 Single Hidden-Layer Feedforward Neural Networks

If hidden units adopt an activation function $f(\cdot)$, then the i^{th} output of the SLFN with N hidden units and C output units can be expressed as

$$o_{ji} = \mathbf{h}_j \cdot \mathbf{a}_i, \tag{1}$$

where $\mathbf{h}_j = [f(\mathbf{w}_1 \cdot \mathbf{x}_j + b_1), f(\mathbf{w}_2 \cdot \mathbf{x}_j + b_2), \ldots, f(\mathbf{w}_N \cdot \mathbf{x}_j + b_N)]^T$ is the output vector of the hidden layer corresponding to the j^{th} input pattern $\mathbf{x}_j \in \mathbf{R}^d$, $\mathbf{a}_i = [a_{i1}, a_{i2}, \ldots, a_{iN}]^T$ is the output weight vector connecting from hidden units to the i^{th} output unit, b_m is the bias of the m^{th} hidden unit, and $\mathbf{w}_m = [w_{m1}, w_{m2}, \ldots, w_{md}]^T$ is the input weight vector connecting from input units to the m^{th} hidden unit. Note that $\mathbf{p} \cdot \mathbf{q} =\prec \mathbf{p}, \mathbf{q} \succ$ is the inner product of two vectors $\mathbf{p}$ and $\mathbf{q}$. This notion of SLFNs is depicted in Fig. 1.

For n training patterns $(\mathbf{x}_j, \mathbf{t}_j), j = 1, 2, \ldots, n$, where $\mathbf{x}_j = [x_{j1}, x_{j2}, \ldots, x_{jd}]^T$ and $\mathbf{t}_j = [t_{j1}, t_{j2}, \ldots, t_{jC}]^T$ are the j^{th} input pattern vector and its corresponding target vector, respectively, the main goal of training process is to adjust the network weights $\mathbf{w}_m$, b_m and $\mathbf{a}_i$ in order that they minimize the error function defined by

$$E = \sum_{j=1}^{n} (\mathbf{o}_j - \mathbf{t}_j)^2 = \sum_{j=1}^{n} \sum_{i=1}^{C} (\mathbf{h}_j \cdot \mathbf{a}_i - t_{ji})^2. \tag{2}$$

In gradient descent algorithms, the minimization procedure is performed by iteratively adjusting the set of vectors $\mathbf{z}$ consisting of weights $(\mathbf{w}_m, \mathbf{a}_i)$ and biases b_m as

$$\mathbf{z}_k = \mathbf{z}_{k-1} - \eta \frac{\partial E}{\partial \mathbf{z}}, \tag{3}$$

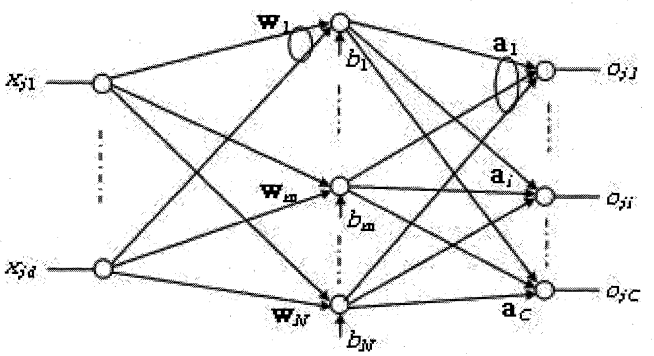

Fig. 1 The architecture of the SLFN.

where η is a learning rate. A popular training algorithm for the feedforward neural networks is the back-propagation (BP) algorithm in which gradients can be calculated and the vectors $\mathbf{z}$ can be updated by error propagation from the output to the input.

This iterative gradient-descent-based algorithm has many drawbacks such as slow convergence, oscillation or divergence due to the improper learning rate, over-training and local minima. Many approaches have been proposed to improve the BP learning algorithm [1]–[5], [8]. However, most learning algorithms for neural networks based on the gradient descent are still slow. Recently, an efficient learning algorithm for SLFNs called extreme learning machine (ELM) was proposed by Huang *et al.* [18], [19].

2.2 Extreme Learning Machine (ELM)

In the ELM algorithm, the input weights and hidden layer biases are randomly assigned, and the output weights of the SLFN can be analytically determined by the simple inverse operation of the output matrix of hidden layer. Clearly, an SLFN with N hidden units can approximate N input patterns $(n = N)$ with zero error. This means there exist network weights $\mathbf{w}, \mathbf{a}$ and biases b such that

$$t_{ji} = \mathbf{h}_j \cdot \mathbf{a}_i, \quad j = 1, 2, \ldots, n; \; i = 1, 2, \ldots, C. \tag{4}$$

This equation can be written as

$$\mathbf{HA} = \mathbf{T}, \tag{5}$$

where $\mathbf{H}$ is called the hidden layer output matrix of the SLFN and defined as [18]

$$\mathbf{H} = \begin{bmatrix} \mathbf{h}_1^T \\ \vdots \\ \mathbf{h}_n^T \end{bmatrix} = \begin{bmatrix} f(\mathbf{w}_1 \cdot \mathbf{x}_1 + b_1) & \cdots & f(\mathbf{w}_N \cdot \mathbf{x}_1 + b_N) \\ \vdots & \ddots & \vdots \\ f(\mathbf{w}_1 \cdot \mathbf{x}_n + b_1) & \cdots & f(\mathbf{w}_N \cdot \mathbf{x}_n + b_N) \end{bmatrix}, \tag{6}$$

$$\mathbf{T} = [\mathbf{t}_1, \mathbf{t}_2, \ldots, \mathbf{t}_n]^T, \tag{7}$$

and

$$\mathbf{A} = [\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_C]. \tag{8}$$

In [18], authors proved that the hidden layer output matrix $\mathbf{H}$ is invertible when the number of training patterns equals the number of hidden units $(n = N)$ and the activation function is infinitely differentiable in any interval of $\mathbf{R}$. So, we can determine the output weight matrix $\mathbf{A}$ with zero training error by simply inverting $\mathbf{H}$ in which the input weights $\mathbf{w}_m$ and the biases b_m can be randomly chosen. When the number of hidden units is less than the number of training patterns $(N < n)$, the output weight matrix $\mathbf{A}$ can also be determined by the pseudo-inverse of $\mathbf{H}$, which is formulated as

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

where $\mathbf{H}^{\dagger}$ is the Moore-Penrose generalized inverse of the hidden layer output matrix $\mathbf{H}$ [20]. We can summarize the extreme learning machine (ELM) algorithm as follows:

Step 1. Randomly assign the input weights $\mathbf{w}_m$ and the hidden layer biases $b_m, m = 1, 2, \ldots, N$.
Step 2. Determine the output matrix $\mathbf{H}$ of the hidden layer by Eq. (6).
Step 3. Determine the output weight matrix $\hat{\mathbf{A}}$ by Eq. (9).

When the whole training set is not available, a development of the ELM called online sequential extreme learning machine (OS-ELM) was proposed by N. Y. Liang *et al.* [21]. It is an online sequential learning algorithm for SLFNs based on the ELM and can learn one-by-one or block-by-block of data. In the OS-ELM, the input weights and the hidden layer biases are also randomly chosen and the output weights can be updated by arriving data. Because of random selection, the input weights and hidden layer biases in both ELM and OS-ELM algorithms might be non-optimal which tends to require more hidden units than conventional tuning-based algorithms in many applications.

3. Two-Stage Linear Model

The ELM algorithm often requires a large number of hidden units for achieving a proper level of training accuracy. Therefore, a larger network size is required which causes a larger computation for training and limitation for classification applications. This drawback is mainly caused by the random selection of the input weights and hidden layer biases. Thus, we claim that SLFNs can be improved if the input weights and the hidden layer biases are properly chosen.

The aim of our study was to develop an efficient learning algorithm for SLFNs with a smaller number of hidden units while producing better generalization capability and faster computation. Instead of randomly choosing or iteratively adjusting the input weights and biases as in the ELM, OS-ELM or BP, our approach estimates them analytically by using a least-squares scheme. Determining the weights and biases of SLFNs consists of two stages. In the first stage, the input weights and the hidden layer biases are estimated based on the linear model. Then, the output weights are determined by the second linear model.

From Eq. (5), if we assume that the output weight matrix $\mathbf{A}$ is determined then the hidden layer output matrix $\mathbf{H}$ can be estimated as

$$\mathbf{H} = \mathbf{T}\mathbf{A}^{\dagger}, \tag{10}$$

where $\mathbf{A}^{\dagger}$ is the Moore-Penrose generalized inverse of $\mathbf{A}$. For an invertible function $f(\cdot)$, we have

$$\mathbf{\Pi} = f^{-1}[\mathbf{T}\mathbf{A}^{\dagger}], \tag{11}$$

where $f^{-1}[\mathbf{T}\mathbf{A}^{\dagger}]_{jm} = f^{-1}([\mathbf{T}\mathbf{A}^{\dagger})]_{jm})$ and

$$\mathbf{\Pi} = \begin{bmatrix} \mathbf{w}_1 \cdot \mathbf{x}_1 + b_1 & \ldots & \mathbf{w}_N \cdot \mathbf{x}_1 + b_N \\ \vdots & \ddots & \vdots \\ \mathbf{w}_1 \cdot \mathbf{x}_n + b_1 & \ldots & \mathbf{w}_N \cdot \mathbf{x}_n + b_N \end{bmatrix}. \tag{12}$$

If we define the matrix $\mathbf{B} \in \mathbf{R}^{C\mathrm{x}N}$ by

$$\mathbf{B} = \mathbf{T}^{\dagger} f^{-1}[\mathbf{T}\mathbf{A}^{\dagger}], \tag{13}$$

where $\mathbf{T}^{\dagger}$ is the pseudo-inverse of $\mathbf{T}$, then Eq. (11) becomes

$$\mathbf{\Pi} = \mathbf{T}\mathbf{B}. \tag{14}$$

Define the input matrix as

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_1 & \mathbf{x}_2 & \ldots & \mathbf{x}_n \\ 1 & 1 & \ldots & 1 \end{bmatrix}^T \tag{15}$$

and let $\mathbf{W}$ be the matrix of input weights and biases defined by

$$\mathbf{W} = \begin{bmatrix} \mathbf{w}_1 & \mathbf{w}_2 & \ldots & \mathbf{w}_N \\ b_1 & b_2 & \ldots & b_N \end{bmatrix}. \tag{16}$$

Then, it follows that

$$\mathbf{X}\mathbf{W} = \mathbf{T}\mathbf{B}. \tag{17}$$

The minimum norm solution for $\mathbf{W}$ among all possible solutions is

$$\hat{\mathbf{W}} = \mathbf{X}^{\dagger}\mathbf{T}\mathbf{B}, \tag{18}$$

where $\mathbf{X}^{\dagger}$ is the pseudo-inverse of $\mathbf{X}$.

At the beginning of the learning process, the matrix $\mathbf{A}$ is unknown. Therefore, instead of estimating matrix $\mathbf{B}$ by Eq. (13), we can randomly assign values for $\mathbf{B}$, and then estimate the matrix of input weights and biases $\mathbf{W}$ by Eq. (18). After estimating the input weights $\mathbf{w}_m$ and the hidden layer biases $b_m (m = 1, 2, \ldots, N)$, we can calculate the hidden layer output matrix $\mathbf{H}$ by Eq. (6) and the output weight matrix $\mathbf{A}$ by Eq. (9). In summary, the two-stage linear model for training SLFNs can be described as follows:

Given a training set $\mathbf{S} = \{(\mathbf{x}_j, \mathbf{t}_j) | j = 1, \ldots, n\}$, *an activation function* $f(\cdot)$ *and the number of hidden units* N,

1. Randomly assign values for the matrix $\mathbf{B}$.
2. Estimate the input weights $\mathbf{w}_m$ and biases b_m by Eq. (18).
3. Calculate the hidden layer output matrix $\mathbf{H}$ by Eq. (6).
4. Determine the output weights by Eq. (9).

Thus, the parameters of networks can be determined by the non-iterative procedure, which results in very fast training process compared to conventional iterative learning algorithms for SLFNs.

Although we select random values for the matrix $\mathbf{B} \in \mathbf{R}^{C\mathrm{x}N}$, in comparison with the original ELM or the OS-ELM, the number of random values required is significantly reduced from $(d + 1)\mathrm{x}N$ to $C\mathrm{x}N$ when the number of classes denoted by C is much smaller than the number of features denoted by d, which is usual in most of the pattern classification applications. Furthermore, SLFNs trained by

our proposed method can have a small number of hidden units, which can further reduce the number of random values. In addition, the solution for the input weights and hidden-layer biases by Eq. (18) is the minimum norm solution. As analyzed by Peter L. Bartlett [22], the networks tend to have better generalization performance with small weights. Therefore, our proposed approach with small norm weights can be expected to give better performance than the original ELM.

4. Online Training with Two-Stage Linear Model

When the amount of training data is very large or when memory costs are very expensive, an online training method should be addressed. Updating the output weight matrix based on the recursive least-squares solution is given by

$$\mathbf{P}_{k+1} = \mathbf{P}_k - \mathbf{P}_k\mathbf{H}_{k+1}^T(\mathbf{I}+\mathbf{H}_{k+1}\mathbf{P}_k\mathbf{H}_{k+1}^T)^{-1}\mathbf{H}_{k+1}\mathbf{P}_k \tag{19}$$

$$\mathbf{A}^{(k+1)} = \mathbf{A}^{(k)} + \mathbf{P}_{k+1}\mathbf{H}_{k+1}^T(\mathbf{T}_{k+1} - \mathbf{H}_{k+1}\mathbf{A}^{(k)}) \tag{20}$$

where the initialization of $\mathbf{P}$ and $\mathbf{A}$ is given by $\mathbf{P}_0 = (\mathbf{H}_0^T\mathbf{H}_0)^{-1}$ and $\mathbf{A}^{(0)} = \mathbf{P}_0\mathbf{H}_0^T\mathbf{T}_0$ [21].

Now, we must estimate the input weights and biases for SLFNs. From Eq. (18), the matrix $\mathbf{B} \in \mathbf{R}^{C\times N}$ is randomly chosen which does not depend on arriving data. Therefore, it is just randomly chosen once, and then the parameters of SLFNs are recursively adjusted. We consider the case when rank($\mathbf{X}$)=$d+1$, where d is the number of input features. The pseudo-inverse of $\mathbf{X}$ is given by $\mathbf{X}^\dagger = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T$. Hence, the estimation of $\mathbf{W}$ in Eq. (18) is given by

$$\hat{\mathbf{W}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{T}\mathbf{B}. \tag{21}$$

Thus, the minimum norm solution for the initial training subset $\mathbf{S}_0 = \{(\mathbf{x}_j, \mathbf{t}_j)|j = 1,\ldots,n_0\}$ can be $\mathbf{W}^{(0)} = \mathbf{L}_0^{-1}\mathbf{X}_0^T\mathbf{T}_0\mathbf{B}$, where $\mathbf{X}_0 = \begin{bmatrix} \mathbf{x}_1 & \mathbf{x}_2 & \cdots & \mathbf{x}_{n_0} \\ 1 & 1 & \cdots & 1 \end{bmatrix}^T$, $\mathbf{T}_0 = \begin{bmatrix} \mathbf{t}_1 & \mathbf{t}_2 & \cdots & \mathbf{t}_{n_0} \end{bmatrix}^T$ and $\mathbf{L}_0 = \mathbf{X}_0^T\mathbf{X}_0$.

Suppose now that there are n_1 observations of the second training subset $\mathbf{S}_1 = \{(\mathbf{x}_j, \mathbf{t}_j)|j = n_0+1,\ldots,n_0+n_1\}$. The input weights and biases can be estimated by

$$\mathbf{W}^{(1)} = \mathbf{L}_1^{-1}\begin{bmatrix} \mathbf{X}_0 \\ \mathbf{X}_1 \end{bmatrix}^T\begin{bmatrix} \mathbf{T}_0 \\ \mathbf{T}_1 \end{bmatrix}\mathbf{B}, \tag{22}$$

where $\mathbf{X}_1 = \begin{bmatrix} \mathbf{x}_{n_0+1} & \mathbf{x}_{n_0+2} & \cdots & \mathbf{x}_{n_0+n_1} \\ 1 & 1 & \cdots & 1 \end{bmatrix}^T$,

$\mathbf{T}_1 = \begin{bmatrix} \mathbf{t}_{n_0+1} & \mathbf{t}_{n_0+2} & \cdots & \mathbf{t}_{n_0+n_1} \end{bmatrix}^T$ and

$$\begin{aligned}\mathbf{L}_1 &= \begin{bmatrix} \mathbf{X}_0 \\ \mathbf{X}_1 \end{bmatrix}^T\begin{bmatrix} \mathbf{X}_0 \\ \mathbf{X}_1 \end{bmatrix} \\ &= \mathbf{X}_0^T\mathbf{X}_0 + \mathbf{X}_1^T\mathbf{X}_1 \\ &= \mathbf{L}_0 + \mathbf{X}_1^T\mathbf{X}_1.\end{aligned}$$

By expanding the last three terms on the right side of Eq. (22), we have

$$\begin{aligned}\begin{bmatrix} \mathbf{X}_0 \\ \mathbf{X}_1 \end{bmatrix}^T\begin{bmatrix} \mathbf{T}_0 \\ \mathbf{T}_1 \end{bmatrix}\mathbf{B} &= \mathbf{X}_0^T\mathbf{T}_0\mathbf{B} + \mathbf{X}_1^T\mathbf{T}_1\mathbf{B} \\ &= \mathbf{L}_0\mathbf{W}^{(0)} + \mathbf{X}_1^T\mathbf{T}_1\mathbf{B}.\end{aligned}$$

By substitution into Eq. (22), we obtain

$$\begin{aligned}\mathbf{W}^{(1)} &= \mathbf{L}_1^{-1}(\mathbf{L}_0\mathbf{W}^{(0)} + \mathbf{X}_1^T\mathbf{T}_1\mathbf{B}) \\ &= \mathbf{L}_1^{-1}((\mathbf{L}_1 - \mathbf{X}_1^T\mathbf{X}_1)\mathbf{W}^{(0)} + \mathbf{X}_1^T\mathbf{T}_1\mathbf{B}) \\ &= \mathbf{W}^{(0)} + \mathbf{L}_1^{-1}\mathbf{X}_1^T(\mathbf{T}_1\mathbf{B} - \mathbf{X}_1\mathbf{W}^{(0)})\end{aligned}$$

In generalization, for n_{k+1} observations of the $(k+1)^{th}$ training subset $\mathbf{S}_{k+1} = \{(\mathbf{x}_j, \mathbf{t}_j)|j = \sum_{j=0}^{k} n_j + 1,\ldots,\sum_{j=0}^{k+1} n_j\}$, the input weights and biases can be determined by

$$\mathbf{W}^{(k+1)} = \mathbf{W}^{(k)} + \mathbf{L}_{k+1}^{-1}\mathbf{X}_{k+1}^T(\mathbf{T}_{k+1}\mathbf{B} - \mathbf{X}_{k+1}\mathbf{W}^{(k)}) \tag{23}$$

$$\mathbf{L}_{k+1} = \mathbf{L}_k + \mathbf{X}_{k+1}^T\mathbf{X}_{k+1} \tag{24}$$

where

$$\mathbf{X}_{k+1} = \begin{bmatrix} \mathbf{x}_{\sum_{j=0}^{k} n_j+1} & \mathbf{x}_{\sum_{j=0}^{k} n_j+2} & \cdots & \mathbf{x}_{\sum_{j=0}^{k+1} n_j} \\ 1 & 1 & \cdots & 1 \end{bmatrix}^T$$

and

$$\mathbf{T}_{k+1} = \begin{bmatrix} \mathbf{t}_{\sum_{j=0}^{k} n_j+1} & \mathbf{t}_{\sum_{j=0}^{k} n_j+2} & \cdots & \mathbf{t}_{\sum_{j=0}^{k+1} n_j} \end{bmatrix}^T.$$

Let $\mathbf{Q}_{k+1} = \mathbf{L}_{k+1}^{-1}$, then $\mathbf{Q}_{k+1}$ is expressed by Woodbury identity as [23]

$$\begin{aligned}\mathbf{Q}_{k+1} &= (\mathbf{L}_k + \mathbf{X}_{k+1}^T\mathbf{X}_{k+1})^{-1} \\ &= \mathbf{L}_k^{-1} - \mathbf{L}_k^{-1}\mathbf{X}_{k+1}^T(\mathbf{I}+\mathbf{X}_{k+1}\mathbf{L}_k^{-1}\mathbf{X}_{k+1}^T)^{-1}\mathbf{X}_{k+1}\mathbf{L}_k^{-1} \\ &= \mathbf{Q}_k - \mathbf{Q}_k\mathbf{X}_{k+1}^T(\mathbf{I}+\mathbf{X}_{k+1}\mathbf{Q}_k\mathbf{X}_{k+1}^T)^{-1}\mathbf{X}_{k+1}\mathbf{Q}_k.\end{aligned}$$

Finally, the updating formula for $\mathbf{W}^{(k+1)}$ is given by

$$\mathbf{W}^{(k+1)} = \mathbf{W}^{(k)} + \mathbf{Q}_{k+1}\mathbf{X}_{k+1}^T(\mathbf{T}_{k+1}\mathbf{B} - \mathbf{X}_{k+1}\mathbf{W}^{(k)}), \tag{25}$$

where

$$\mathbf{Q}_{k+1} = \mathbf{Q}_k - \mathbf{Q}_k\mathbf{X}_{k+1}^T(\mathbf{I}+\mathbf{X}_{k+1}\mathbf{Q}_k\mathbf{X}_{k+1}^T)^{-1}\mathbf{X}_{k+1}\mathbf{Q}_k. \tag{26}$$

In summary, the online training scheme for SLFNs with the two-stage linear model can be described as follows: *Given a training set* $\mathbf{S} = \{(\mathbf{x}_j, \mathbf{t}_j)|j = 1,\ldots,n\}$, *an activation function* $f(\cdot)$ *and the number of hidden units* N,

1) **Initialization:** For the initial training subset $\mathbf{S}_0 = \{(\mathbf{x}_j, \mathbf{t}_j)|j = 1,\ldots,n_0\}$,
   i. Assign random values for the matrix $\mathbf{B} \in \mathbf{R}^{C\times N}$.
   ii. Calculate the initial input weights and biases by $\mathbf{W}^{(0)} = \mathbf{Q}_0\mathbf{X}_0^T\mathbf{T}_0\mathbf{B}$, where

$$\mathbf{Q}_0 = (\mathbf{X}_0^T\mathbf{X}_0)^{-1},$$

$$\mathbf{X}_0 = \begin{bmatrix} \mathbf{x}_1 & \mathbf{x}_2 & \cdots & \mathbf{x}_{n_0} \\ 1 & 1 & \cdots & 1 \end{bmatrix}^T$$

and

$$\mathbf{T}_0 = \begin{bmatrix} \mathbf{t}_1 & \mathbf{t}_2 & \cdots & \mathbf{t}_{n_0} \end{bmatrix}^T.$$

iii. Calculate the initial hidden layer output matrix

$$\mathbf{H}_0=\begin{bmatrix}\mathbf{h}_1^T\\ \vdots\\ \mathbf{h}_{n_0}^T\end{bmatrix}$$
$$=\begin{bmatrix}f(\mathbf{w}_1\cdot\mathbf{x}_1+b_1)&\dots&f(\mathbf{w}_N\cdot\mathbf{x}_1+b_N)\\ \vdots&\ddots&\vdots\\ f(\mathbf{w}_1\cdot\mathbf{x}_{n_0}+b_1)&\dots&f(\mathbf{w}_N\cdot\mathbf{x}_{n_0}+b_N)\end{bmatrix}.$$

iv. Determine the initial output weights by

$$\mathbf{A}^{(0)}=\mathbf{P}_0\mathbf{H}_0^T\mathbf{T}_0,\text{ where }\mathbf{P}_0=(\mathbf{H}_0^T\mathbf{H}_0)^{-1}.$$

2) **Training process:** For the $(k+1)^{th}$ training subset $\mathbf{S}_{k+1}=\{(\mathbf{x}_j,\mathbf{t}_j)|j=\sum_{j=0}^{k}n_j+1,\dots,\sum_{j=0}^{k+1}n_j\}$,

i. Estimate the input weights and biases by

$$\mathbf{Q}_{k+1}=\mathbf{Q}_k-\mathbf{Q}_k\mathbf{X}_{k+1}^T(\mathbf{I}+\mathbf{X}_{k+1}\mathbf{Q}_k\mathbf{X}_{k+1}^T)^{-1}\mathbf{X}_{k+1}\mathbf{Q}_k$$
$$\mathbf{W}^{(k+1)}=\mathbf{W}^{(k)}+\mathbf{Q}_{k+1}\mathbf{X}_{k+1}^T(\mathbf{T}_{k+1}\mathbf{B}-\mathbf{X}_{k+1}\mathbf{W}^{(k)}).$$

ii. Calculate the partial hidden layer output matrix $\mathbf{H}_{k+1}$ by

$$\mathbf{H}_{k+1}=\begin{bmatrix}\mathbf{h}_{\sum_{j=0}^{k}n_j+1}^T\\ \vdots\\ \mathbf{h}_{\sum_{j=0}^{k+1}n_j}^T\end{bmatrix}.$$

iii. Determine the output weights by

$$\mathbf{P}_{k+1}=\mathbf{P}_k-\mathbf{P}_k\mathbf{H}_{k+1}^T(\mathbf{I}+\mathbf{H}_{k+1}\mathbf{P}_k\mathbf{H}_{k+1}^T)^{-1}\mathbf{H}_{k+1}\mathbf{P}_k$$
$$\mathbf{A}^{(k+1)}=\mathbf{A}^{(k)}+\mathbf{P}_{k+1}\mathbf{H}_{k+1}^T(\mathbf{T}_{k+1}-\mathbf{H}_{k+1}\mathbf{A}^{(k)}).$$

iv. Set $k=k+1$ and repeat.

The training algorithm consists of two processes which are the initialization process and the training process. In the initialization process, the matrix $\mathbf{B}$ is assigned randomly, and then the initial weights and the hidden-layer biases are determined based on the initial training subset $\mathbf{S}_0$. The number of patterns for $\mathbf{S}_0$ should be at least $max\{N,d+1\}$. In the training process, the weights and biases of SLFNs are updated for each arriving training subset $\mathbf{S}_{k+1}(k=0,\dots,K-1)$, where $\mathbf{x}_n\in\mathbf{S}_K$, which implies that each input pattern is involved in training only once and the total number of training subsets is $K+1$ including $\mathbf{S}_0$.

5. Experimental Results and Analysis

In this section, we report the performance comparison of our least squares extreme learning machine (LS-ELM) with the ELM algorithm and other popular training algorithms for SLFNs such as the back-propagation (BP) and support vector machines (SVMs). We investigated two real application problems of medical diagnosis and image segment classification. Experiments were run in a low power personal computer that has the CPU of Pentium M with 1.3 GHz and RAM of 256 MB. The algorithms BP, ELM and our LS-ELM were implemented in MATLAB 7.0 environment. The activation function used in our proposed algorithms is a simple sigmoidal function $f(x)=1/(1+exp(-x))$. The SVM was implemented by using the compiled C-coded SVM packages: LIBSVM[†] running in the same PC and using the radial basis function kernel.

5.1 Medical Diagnosis Application: Diabetes

We first evaluate our LS-ELM with the real medical diagnosis problem of diabetes using the "Pima Indians Diabetes Databases". The binary-valued diagnostic was investigated whether a patient showed signs of diabetes according to World Health Organization criteria. The data set consists of 768 patterns of female patients. Each pattern consisting of 8 input features with values in the range from 0.0 to 1.0 belongs to either positive or negative class. In our experiments, the outputs have been normalized into the range $[-1,1]$. As usually done in [18], [24]–[27], the training set is 75 percent of the pattern data set and the rest 25 percent is used as the test set.

The average results of fifty trials are shown in Tables 1 and 2. The number of hidden units of our approach is equal to that of the BP and five times smaller than that of the ELM. Thus, with analytically determined weights and biases, the training and testing speeds are extremely high, even though they are compared to the ELM algorithm. Our LS-ELM algorithm spent 0.0032s CPU time for training; it runs about 839 times faster than the BP, 46 times faster than the SVM, and three times faster than the ELM. The testing time of our approach is about 116 times shorter than that of the SVM, 12 times shorter than that of the BP and five times shorter than that of the ELM. In addition, it can be seen from Table 3 that our approach can obtain the testing accuracy of 77.60 % , which is compatible with the ELM, while better than the BP and all results for the same data set shown in the literature using various popular algorithms such as SAOCIF [25], SVM [18], Cascade-Correlation [25], AdaBoost [26], C4.5 [26] and RBF [27].

Table 1 Performance comparison for diabetes: Time(s).

Algorithms	Training	Testing	# SVs/nodes
Our approach	0.0032	0.0004	4
ELM	0.0097	0.0022	20
BP	2.6849	0.0050	4
SVM	0.1484	0.0466	317.28

Table 2 Performance comparison for diabetes: Classification Accuracy(%).

Algorithms	Training		Testing		# SVs/
	Mean	Dev.	Mean	Dev.	nodes
Our approach	77.95	0.98	77.60	2.67	4
ELM	78.65	1.17	77.57	2.80	20
BP	81.81	1.85	75.20	3.19	4
SVM	78.68	0.90	77.29	2.35	317.28

[†]SVM Source Codes: www.csie.ntu.edu.tw/~ cjlin/libsvm/

Table 3 Performance comparison with other methods: diabetes.

Algorithms	Test accuracy (%)
LS-ELM	77.60
ELM [18]	77.57
BP	75.20
SVM [18]	77.31
SAOCIF [25]	77.32
Cascade-Correlation [25]	76.58
AdaBoost [26]	75.60
C4.5[26]	74.30
RBF [27]	76.30
Heterogeneous RBF [27]	76.30
Incremental PRBFsplit [28]	75.90

Table 4 Performance comparison for Image segmentation: time(s).

Algorithms	Training	Testing	# nodes
LS-ELM	0.7629	0.0543	180
	0.2452	0.0256	95
ELM	0.8648	0.0555	190
BP	2920.5	0.0341	100

Table 5 Performance comparison for Image segmentation: classification accuracy(%).

Algorithms	Training		Testing		# nodes
	Mean	Dev.	Mean	Dev.	
LS-ELM	97.07	0.31	95.60	0.70	180
	95.91	0.34	95.12	0.81	95
ELM	97.22	0.30	95.11	0.62	190
RBF [27]	80.48	N/A	N/A	N/A	N/A
BP	97.00	0.45	86.43	1.72	100

5.2 Image Segmentation Application

In this section, we present the evaluation of our LS-ELM for the image segmentation data set consisting of seven classes [30]. Each pattern in the data set is a 3x3 region randomly drawn from a database of 7 outdoor images. The aim is to classify each region into one of seven classes: brickface, sky, foliage, cement, window, path and grass. 19 attributes extracted from the square region were used as the input features. In our experiments, 1500 training patterns and 810 testing patterns were randomly drawn from the entire data set for each trial and all attributes were normalized into the range $[-1, 1]$.

The average results of fifty trials are shown in Tables 4 and 5. As we expected based on the results of medical diagnosis application, our approach can obtain the testing accuracy of 95.60 % using 180 hidden units with learning of about 3828 times faster than the BP and slightly faster than the ELM. The testing time is shorter than that of the ELM and longer than that of the BP because our LS-ELM uses more hidden units than the BP for the testing accuracy of 95.60 % . However, for testing accuracy of 95.12 % which is compatible with the ELM and higher than the BP, our approach uses 95 hidden units which is slightly fewer than the BP and twice fewer than the ELM. In this case, our approach is much faster than the BP and the ELM for both training and testing. In literature, results for the same data set using other popular algorithms such as OS-ELM [21], GAP-RBF [29] and MRAN [31] are shown in Table 6. For this image segment classification problem, our approach is also better than others in various performance criteria.

Table 6 Performance comparison with other methods: Image segmentation.

Algorithms	Test accuracy (%)
LS-ELM	95.60
ELM	95.11
BP	86.43
OS-ELM [21]	94.88
GAP-RBF [29]	89.93
MRAN	93.30

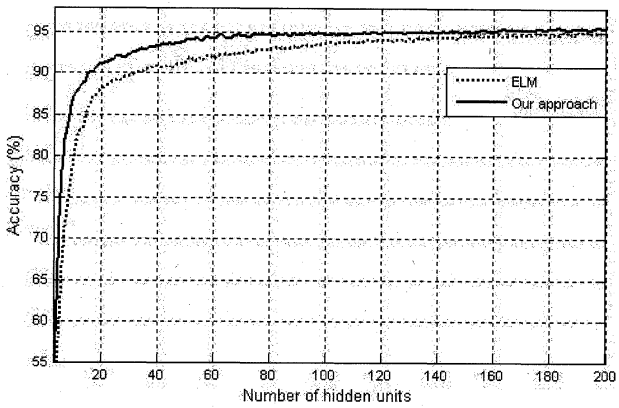

Fig. 2 Comparison of our approach with ELM for different numbers of hidden units.

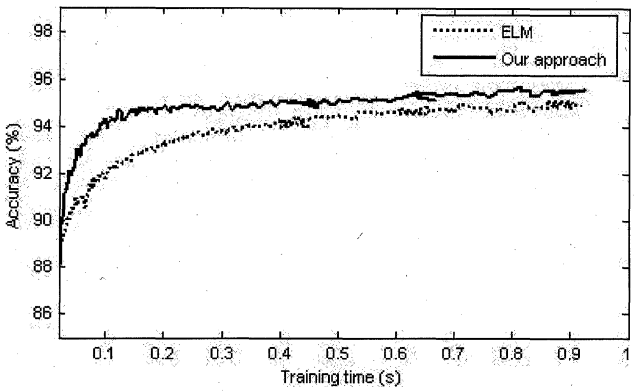

Fig. 3 Comparison of our approach with ELM on the training time.

The performance comparison of our approach with the ELM on the testing set for different numbers of hidden units ranging from 2 to 200 at the interval of 1 is shown in Fig. 2. It can be seen that our approach can obtain better performance than the ELM with the same number of hidden units,

which implies that our approach can obtain good performance with a smaller number of hidden units. Hence, its training time is shorter than that of the ELM for the same performance as shown in Fig. 3.

6. Conclusion

In this paper, a new approach for training single hidden-layer feedforward neural networks (SLFNs), least-squares extreme learning machine (LS-ELM), was proposed. The main distinction of the proposed algorithm is that the weights and biases can be determined by the non-iterative two-stage linear model. They do not need to be neither iteratively adjusted as the back-propagation (BP) nor randomly chosen as the original extreme learning machine (ELM). The recursive online training method with sub-blocks of the training data set can be applied when the volume of training data set is very large. The important advantage of the proposed algorithm, in comparison with ELMs, is that the network weights including hidden layer biases are determined by the smallest norm least-squares solution and the number of hidden units can be reduced, which results in time reduction for both training and testing. It also produces a good generalization capability by showing compatible or better results than those of the ELM and other popular classification algorithms for the single hidden-layer feedforward neural networks.

Acknowledgments

This work was supported by grant No. RTI-04-03-03 from the Regional Technology Innovation Program of the Ministry of Commerce, Industry and Energy(MOCIE) of Korea.

References

[1] D. Nguyen and B. Widrow, "Improving the learning speed of 2-layer neural networks by choosing initial values of the adaptive weights," Int'l Joint Conf. Neural Neural Networks, vol.3, pp.21–26, 1990.
[2] J.Y.F. Yam and T.W.S. Chow, "Feedforward networks training speed enhancement by optimal initialization of the synaptic coefficients," IEEE Trans. Neural Netw., vol.12, no.2, pp.430–434, 2001.
[3] N.B. Karayiannis and A.N. Venetsanopoulos, Artificial neural networks: Learning algorithms, performance evaluation, and applications, Kluwer Academic, Boston, MA, 1993.
[4] Y. LeCun, L. Bottou, G.B. Orr, and K.R. Müler, "Efficient backprop," Lecture Notes in Computer Science, vol.1524, pp.9–50, 1998.
[5] S.M.A. Burney, T.A. Jilani, and C. Ardil, "A comparison of first and second order training algorithms for artificial neural networks," International Journal of Computational Intelligence, vol.1, no.2, pp.218–224, 2004.
[6] L.S. Nghia, J. Sjöerg, and M. Viberg, "Adaptive neural nets filter using a recursive levenberg-marquardt search direction," Proc. Asilomar Conf. Signals, Syst., Comput., vol.1-4, pp.697–701, Nov. 1998.
[7] V.S. Asirvadam, S.F. McLoone, and G. Irwin, "Parallel and separable recursive levenberg-marquardt training algorithm," Proc. 12th IEEE Workshop Neural Net. Signal Process., pp.129–138, Sept. 2002.
[8] S. Lawrence and C.L. Giles, "Overfitting and neural networks: Conjugate gradient and backpropagation," Proc. IEEE-INNS-ENNS Int'l Joint Conf. on Neural networks, vol.1, pp.114–119, July 2000.
[9] Z.P. Liu and J.P. Castagna, "Avoiding overfitting caused by noise using a uniform training mode," Int'l Joint Conf. on Neural Networks, vol.3, pp.1788–1793, July 1999.
[10] S. Dumais, J. Platt, and D. Heckerman, "Inductive learning algorithms and representations for text categorization," Proc. Int'l Conf. Information and Knowledge Management, vol.3, pp.148–154, 1998.
[11] C.W. Hsu and C.J. Lin, "A comparison of methods for multiclass support vector machines," IEEE Trans. Neural Netw., vol.13, no.2, pp.415–425, March 2002.
[12] S.X. Lu and X.Z. Wang, "A comparison among four svm classification methods: Lsvm, nlsvm, ssvm and nsvm," Proc. Int'l Conf. on Machine Learning and Cybernetics, pp.4277–4282, Aug. 2004.
[13] T. Joachims, "Text categorization with support vector machines: Learning with many relevant features," Proc. Eur. Conf. Machine Learning, pp.137–142, 1998.
[14] O. Chapelle, V. Vapnik, O. Bousquet, and S. Mukherjee, "Choosing multiple parameters for support vector machines," Mach. Learn., vol.46, no.1, pp.131–159, 2002.
[15] D. Decoste and K. Wagstaff, "Alpha seeding for support vector machines," Proc. Intl Conf. Knowledge Discovery and Data Mining, pp.345–349, 2000.
[16] S.S. Keerthi, "Efficient tuning of svm hyperparameters using radius/margin bound and iterative algorithms," IEEE Trans. Neural Netw., vol.13, no.5, pp.1225–1229, Sept. 2002.
[17] G.B. Huang, Y.Q. Chen, and H.A. Babri, "Classification ability of single hidden layer feedforward neural networks," IEEE Trans. Neural Netw., vol.11, no.3, pp.799–801, May 2000.
[18] G.B. Huang, Q.Y. Zhu, and C.K. Siew, "Extreme learning machine: Theory and applications," Neurocomputing, vol.70, pp.489–501, 2006.
[19] G.B. Huang, Q.Y. Zhu, and C.K. Siew, "Extreme learning machine: A new learning scheme for feedforward neural networks," Proc. Int'l Joint Conf. on Neural Networks, July 2004.
[20] D. Serre, Matrices: Theory and Applications, Springer, New York, 2002.
[21] N.Y. Liang, G.B. Huang, P. Saratchandran, and N. Sundararajan, "A fast and accurate online sequential learning algorithm for feedforward networks," IEEE Trans. Neural Netw., vol.17, no.6, pp.1411–1423, 2006.
[22] 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 Trans. Inf. Theory, vol.44, no.2, pp.525–536, March 1998.
[23] G.H. Golub and C.F.V. Loan, Matrix computations, 3rd ed., Johns Hopkins University Press, Baltimore, MD, USA, 1996.
[24] T.O.G. Räsch and K.R. Müler, "An improvement of adaboost to avoid overfitting," Proc. 5th Int'l Conf. on Neural Information Processing, 1998.
[25] E. Romero and R. Alquézar, "A new incremental method for function approximation using feedforward neural networks," Proc. INNS-IEEE Int'l Joint Conf. on Neural Networks, pp.1968–1973, 2002.
[26] Y. Freund and R.E. Schapire, "Experiments with a new boosting algorithm," Proc. 13th Int'l Conf. on Machine Learning, pp.148–156, 1996.
[27] D.R. Wilson and T.R. Martinez, "Heterogeneous radial basis functionnetworks," Proc. Int'l Conf. on Neural Networks, pp.1263–1267, June 1996.
[28] C. Constantinopoulos and A. Likas, "An incremental training method for probabilistic rbf network," IEEE Trans. Neural Netw., vol.17, no.4, pp.966–974, July 2006.
[29] G.B. Huang, P. Saratchandran, and N. Sundararajan, "An efficient sequential learning algorithm for growing and pruning rbf (gap-rbf) networks," IEEE Trans. Syst., Man Cybern. B, Cybern., vol.34, no.6, pp.2284–2292, 2004.
[30] D. Newman, S. Hettich, C. Blake, and C. Merz, "UCI repository of machine learning databases," University of California, Irvine, 1998,

http://www.ics.edu/~mlearn/MLRepository.html

[31] L. Yingwei, N. Sundararajan, and P. Saratchandran, "Performance evaluation of a sequential minimal radial basis function (rbf) neural network learning algorithm," IEEE Trans. Neural Netw., vol.9, no.2, pp.308–318, March 1998.

Hieu Trung Huynh received the B.S. and M.S. in Computer Engineering from HoChiMinh City University of Technology in 1998 and 2003, respectively. He is currently pursuing the Ph.D. degree in computer engineering at Chonnam National University. From 1998 to 2005, he worked as a lecturer and researcher at the Faculty of Electrical and Electronics Engineering, HoChiMinh City University of Technology. His research interests focus on the computational intelligence for image analysis, pattern recognition, network and communication security, biological and medical data analysis.

Yonggwan Won received the B.S. in Electronics Engineering from Hanyang University in 1987, and M.S. and Ph. D. degrees in Electrical and Computer Engineering from University of Missouri-Columbia in 1991 and 1995, respectively. He worked with Electronics, and Telecommunication Research Institute (ETRI) from 1995 to 1996, and Korea Telecomm(KT) from 1996 to 1999. He is currently a professor in Chonnam National University in Korea, and the director of Korea Bio-IT Foundry Center at Gwangju. His major research interest is the computational intelligence for image analysis, pattern recognition, network and communication security, bio and medical data analysis.