ONLINE SEQUENTIAL LEARNING BASED ON ENHANCED EXTREME LEARNING MACHINE USING LEFT OR RIGHT PSEUDO-INVERSE

Weiwei Zong, Yuan Lan and Guang-Bin Huang
School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore

Keywords: Feedforward network, Extreme learning machine, Online sequential learning.

Abstract: The latest development (Huang et al., 2011) has shown that better generalization performance can be obtained for extreme learning machine (ELM) by adding a positive value to the diagonal of $\mathbf{H}^T\mathbf{H}$ or $\mathbf{H}\mathbf{H}^T$, where $\mathbf{H}$ is the hidden layer output matrix. This paper further extends this enhanced ELM to online sequential learning mode. An online sequential learning algorithm is proposed for SLFNs and other regularization networks, consisting of two formulas for two kinds of scenarios: when initial training data is of small scale or large scale. Performance of proposed online sequential learning algorithm is demonstrated through six benchmarking data sets for both regression and multi-class classification problems.

1 INTRODUCTION

Training algorithms for feedforward networks, including least-square based extreme learning machine (ELM) (Huang et al., 2006b; Huang et al., 2006a; Huang and Chen, 2007; Huang and Chen, 2008) for single-hidden layer feedforward networks (SLFNs) and gradient-descent based backpropagation (BP) method (Rumelhart et al., 1986) for multi-layer feedforward neural networks, have attracted the attention of many researchers for the past years. Main focus is given to the batch learning mode of the aforementioned algorithms. However, in real world applications, the training data may not come at once or the size of training data may be too large. In such circumstances, online sequential learning instead of batch learning is preferred.

Sequential learning algorithms based on BP for SLFNs with additive nodes have been proposed in literature (Ngia et al., 1998; Asirvadam et al., 2002). Resource allocation network (RAN), one of the training algorithms for feedforward networks with RBF nodes, has been extended to sequential learning mode as well (Huang et al., 2004; Huang et al., 2005). These sequential learning algorithms may not be efficient enough due to the disadvantages in convergence rate, training speed and parameter tuning complexity. Moreover, the data can be learned only on a one by one basis.Online sequential extreme learning machine (OS-ELM) was proposed by Liang, et al (Liang et al., 2006) where the training data can be learned not only on a one-by-one basis but also on a chunk-by-chunk basis. OS-ELM is based on the original ELM where the SLFN can be viewed as a linear system with the solution being the left pseudo-inverse of the hidden layer output matrix $\mathbf{H}$ in the following form: $\mathbf{H}^\dagger = (\mathbf{H}^T\mathbf{H})^{-1}\mathbf{H}^T$. Inheriting the advantage of simplicity from original ELM, which randomly generates the hidden layer nodes, OS-ELM outperforms the state-of-art sequential learning algorithms both in generalization capability and in computational efficiency. Therefore, the performance comparison with the state-of-art method in this paper is conducted over OS-ELM.

As mentioned in ridge regression theory (Hoerl and Kennard, 1970), the solution tends to be more stable and better generalization performance can be achieved by adding a positive value to the diagonal of $\mathbf{H}^T\mathbf{H}$ or $\mathbf{H}\mathbf{H}^T$. The resultant enhanced ELM with *sigmoid* additive nodes has been studied in the work of Toh (Toh, 2008) and Deng, et al (Deng et al., 2009). In the recent work of Huang, et al (Huang et al., 2011), the idea of a unified solution based on ELM (referred to as *eELM*) for SLFNs and other regularization networks with wide type of feature mappings or kernels was first proposed and fulfilled.

OS-ELM is derived on the basis of batch mode original ELM using left pseudo-inverse. However, latest development in ELM (Huang et al., 2011) has shown much better advantage in generalization ca-

Zong W., Lan Y. and Huang G. (2012).
ONLINE SEQUENTIAL LEARNING BASED ON ENHANCED EXTREME LEARNING MACHINE USING LEFT OR RIGHT PSEUDO-INVERSE.
In *Proceedings of the 1st International Conference on Pattern Recognition Applications and Methods*, pages 300-305
DOI: 10.5220/0003777603000305
Copyright © SciTePress

pability than the original ELM. On the other hand, to the best of our knowledge, very few work has been done about the sequential learning using right pseudo-inverse. In this paper, we propose an online sequential learning algorithm based on eELM using the right pseudo-inverse (referred to as *OS-eELM-right*). Moreover, the online sequential learning for SLFNs and other regularization networks is built regarding both left pseudo-inverse (referred to as *OS-eELM-left*) and right pseudo-inverse (*OS-eELM-right*). The performance of the proposed framework is compared with original OS-ELM through six benchmarking data sets for regression and multi-class classification applications.

The rest of this paper is organized as follows. Section 2 gives a brief introduction of ELM and its enhanced version eELM. Section 3 derives the proposed framework including OS-eELM-left and OS-eELM-right. Performance evaluation over benchmarking data sets is provided in Section 4. A summary is presented in Section 5.

2 BRIEF REVIEW OF ELM

2.1 Review of ELM

ELM (Huang et al., 2006b) was originally proposed for the single-hidden layer feedforward *neural* networks and was then extended to the SLFNs where the hidden layer need not be neuron alike (Huang and Chen, 2007; Huang and Chen, 2008). The main feature of ELM lies in that the hidden layer need not be tuned. Instead of iterative tuning as in traditional learning algorithms, in ELM, the hidden nodes are randomly generated which is independent of the training data.

After randomly generating L hidden nodes, with the (row) vector $\mathbf{h}(\mathbf{x}) = [h_1(\mathbf{x}), \cdots, h_L(\mathbf{x})]$ presenting the outputs of the L hidden nodes with respect to the input $\mathbf{x}$, the SLFNs is essentially a linear system

$$\mathbf{H}\beta = \mathbf{T}, \tag{1}$$

where $\beta = [\beta_1, \cdots, \beta_L]$ is the vector of the output weights, and $\mathbf{H}$ is the hidden layer output matrix

$$\mathbf{H} = \begin{bmatrix} \mathbf{h}(\mathbf{x}_1) \\ \vdots \\ \mathbf{h}(\mathbf{x}_N) \end{bmatrix}. \tag{2}$$

In the original implementation of ELM, the minimal norm least square solution is

$$\beta = \mathbf{H}^{\dagger}\mathbf{T}, \tag{3}$$

where $\mathbf{T}$ is the label matrix, and $\mathbf{H}^{\dagger}$ is the *Moore-Penrose generalized inverse* of matrix $\mathbf{H}$ (Rao and Mitra, 1971; Serre, 2002). One of the methods to calculate Moore-Penrose generalized inverse of a matrix is the orthogonal projection method (Rao and Mitra, 1971): $\mathbf{H}^{\dagger} = \left(\mathbf{H}^T\mathbf{H}\right)^{-1}\mathbf{H}^T$ (called *left pseudo-inverse*) when $\mathbf{H}^T\mathbf{H}$ is nonsingular or $\mathbf{H}^{\dagger} = \mathbf{H}^T\left(\mathbf{H}\mathbf{H}^T\right)^{-1}$ (called *right pseudo-inverse*) when $\mathbf{H}\mathbf{H}^T$ is nonsingular. Usually the left pseudo-inverse is suitable when the size of training data is large; otherwise the right pseudo-inverse is better in terms of training speed.

2.2 Review of eELM

According to the ridge regression theory (Hoerl and Kennard, 1970), one can add a positive value to the diagonal of $\mathbf{H}\mathbf{H}^T$ or $\mathbf{H}^T\mathbf{H}$, the resultant solution is more stable and tends to have better generalization performance. In (Huang et al., 2011) a unified solution framework for SLFNs, SVM and other regularization networks were proposed where solution based on right pseudo-inverse for small scale of dataset and solution based on left pseudo-inverse for large scale dataset are given by

$$\begin{aligned} \text{Right: } \beta &= \mathbf{H}^{\dagger}\mathbf{T} = \mathbf{H}^T\left(\frac{\mathbf{I}}{C} + \mathbf{H}\mathbf{H}^T\right)^{-1}\mathbf{T} \\ \text{Left: } \beta &= \mathbf{H}^{\dagger}\mathbf{T} = \left(\frac{\mathbf{I}}{C} + \mathbf{H}^T\mathbf{H}\right)^{-1}\mathbf{H}^T\mathbf{T} \end{aligned} \tag{4}$$

where C is determined by users. Observed from simulations on wide types of datasets, the hidden node number L is normally set as a large value to obtain good generalization performance, while the regularization term C in (4) is the only parameter user needs to specify according to various datasets.

3 THE PROPOSED ONLINE SEQUENTIAL LEARNING ALGORITHM

In this section, the online sequential learning algorithm based on eELM is proposed, comprising of OS-eELM-left which calculates the left pseudo-inverse when small scale of initial training data is observed, and OS-eELM-right which calculates the right pseudo-inverse when large scale of initial training data is presented.

3.1 OS-eELM-left

The only difference between OS-eELM-left and OS-

ELM is that OS-eELM is derived on the basis of the latest development of ELM (Huang et al., 2011), which we call eELM in this paper. Hence, it is not difficult to find out that only a slight change need to be made to extend OS-ELM to OS-eELM-left. The solution is to obtain the initial output weight $\beta^{(0)}$ as $\beta^{(0)} = \left(\frac{\mathbf{I}}{C} + \mathbf{H}_0^T\mathbf{H}_0\right)^{-1}\mathbf{H}_0^T\mathbf{T}_0 = \mathbf{P}_0\mathbf{H}_0^T\mathbf{T}_0$. The rest of the equations in the training procedure remains exactly the same as OS-ELM. Therefore, more focus is given to the derivation of OS-eELM-right which makes a great difference to OS-ELM by using right pesudo-inverse.

3.2 The Proposed OS-eELM-right

Given the initial training set $\aleph_0 = \{(\mathbf{x}_i, \mathbf{t}_i)\}_{i=1}^{N_0}$, the minimum norm least square solution to minimize $\| \mathbf{H}_0\beta - \mathbf{T}_0 \|$ is given by $\beta^{(0)} = \hat{\mathbf{H}}_0^{\dagger}\mathbf{T}_0 = \mathbf{H}_0^T(\frac{\mathbf{I}}{C} + \mathbf{H}_0\mathbf{H}_0^T)^{-1}\mathbf{T}_0$. Given another chunk of data $\aleph_1 = \{(\mathbf{x}_i, \mathbf{t}_i)\}_{i=N_0+1}^{N_0+N_1}$ of size N_1, from the batch learning point of view, the problem is to minimize

$$\begin{bmatrix} \mathbf{H}_0 \\ \mathbf{H}_1 \end{bmatrix} \beta^{(1)} - \begin{bmatrix} \mathbf{T}_0 \\ \mathbf{T}_1 \end{bmatrix}. \tag{5}$$

According to (4), the output weight β becomes

$$\beta^{(1)} = \begin{bmatrix} \mathbf{H}_0 \\ \mathbf{H}_1 \end{bmatrix}^T \left(\frac{\mathbf{I}}{C} + \begin{bmatrix} \mathbf{H}_0 \\ \mathbf{H}_1 \end{bmatrix} \begin{bmatrix} \mathbf{H}_0 \\ \mathbf{H}_1 \end{bmatrix}^T \right)^{-1} \begin{bmatrix} \mathbf{T}_0 \\ \mathbf{T}_1 \end{bmatrix}. \tag{6}$$

Let $\mathbf{A} = \frac{\mathbf{I}}{C} + \begin{bmatrix} \mathbf{H}_0 \\ \mathbf{H}_1 \end{bmatrix} \begin{bmatrix} \mathbf{H}_0 \\ \mathbf{H}_1 \end{bmatrix}^T$, we have

$$\mathbf{A}^{-1} = \begin{bmatrix} \frac{\mathbf{I}}{C} + \mathbf{H}_0\mathbf{H}_0^T & \mathbf{H}_0\mathbf{H}_1^T \\ \mathbf{H}_1\mathbf{H}_0^T & \frac{\mathbf{I}}{C} + \mathbf{H}_1\mathbf{H}_1^T \end{bmatrix}^{-1} = \begin{bmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} \\ \mathbf{A}_{21} & \mathbf{A}_{22} \end{bmatrix}.$$

This 2×2 block matrix is invertible if and only if $(\frac{\mathbf{I}}{C} + \mathbf{H}_0\mathbf{H}_0^T)$ and its schur complement are invertible (Boyd et al., 1994). From the initial step, it is easy to show that $(\frac{\mathbf{I}}{C} + \mathbf{H}_0\mathbf{H}_0^T)$ is invertible. Similar to (Feng et al., 2009), it can be proved that schur complement of $(\frac{\mathbf{I}}{C} + \mathbf{H}_0\mathbf{H}_0^T)$, denoted by $\mathbf{S}$, is invertible with probability one. And according to (Boyd et al., 1994), we have

$$\begin{aligned}
\mathbf{A}_{11} &= \left(\frac{\mathbf{I}}{C} + \mathbf{H}_0\mathbf{H}_0^T\right)^{-1} + \left(\frac{\mathbf{I}}{C} + \mathbf{H}_0\mathbf{H}_0^T\right)^{-1} \\
&\quad \left(\mathbf{H}_0\mathbf{H}_1^T\right)\mathbf{S}^{-1}\left(\mathbf{H}_1\mathbf{H}_0^T\right)\left(\frac{\mathbf{I}}{C} + \mathbf{H}_0\mathbf{H}_0^T\right)^{-1} \\
\mathbf{A}_{12} &= -\left(\frac{\mathbf{I}}{C} + \mathbf{H}_0\mathbf{H}_0^T\right)^{-1}\left(\mathbf{H}_0\mathbf{H}_1^T\right)\mathbf{S}^{-1} \\
\mathbf{A}_{21} &= -\mathbf{S}^{-1}\left(\mathbf{H}_1\mathbf{H}_0^T\right)\left(\frac{\mathbf{I}}{C} + \mathbf{H}_0\mathbf{H}_0^T\right)^{-1} \\
\mathbf{A}_{22} &= \mathbf{S}^{-1}
\end{aligned} \tag{7}$$

where

$$\begin{aligned}
\mathbf{S} &= \left(\frac{\mathbf{I}}{C} + \mathbf{H}_1\mathbf{H}_1^T\right) - \left(\mathbf{H}_1\mathbf{H}_0^T\right)\left(\frac{\mathbf{I}}{C} + \mathbf{H}_0\mathbf{H}_0^T\right)^{-1}\left(\mathbf{H}_0\mathbf{H}_1^T\right) \\
&= \frac{\mathbf{I}}{C} + \mathbf{H}_1\left(\mathbf{I} - \mathbf{H}_0^{\dagger}\mathbf{H}_0\right)\mathbf{H}_1^T.
\end{aligned} \tag{8}$$

Denote $\mathbf{P}_0 = \mathbf{I} - \mathbf{H}_0^{\dagger}\mathbf{H}_0$, then $\mathbf{S}_1$, which represents the state of $\mathbf{S}$ after the first chunk data arrives, can be expressed as: $\mathbf{S}_1 = \frac{\mathbf{I}}{C} + \mathbf{H}_1\left(\mathbf{I} - \mathbf{H}_0^{\dagger}\mathbf{H}_0\right)\mathbf{H}_1^T = \frac{\mathbf{I}}{C} + \mathbf{H}_1\mathbf{P}_0\mathbf{H}_1^T$. Thus we have

$$\begin{aligned}
\beta^{(1)} &= \begin{bmatrix} \mathbf{H}_0^T & \mathbf{H}_1^T \end{bmatrix} \begin{bmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} \\ \mathbf{A}_{21} & \mathbf{A}_{22} \end{bmatrix} \begin{bmatrix} \mathbf{T}_0 \\ \mathbf{T}_1 \end{bmatrix} \\
&= \mathbf{H}_0^T\mathbf{A}_{11}\mathbf{T}_0 + \mathbf{H}_1^T\mathbf{A}_{21}\mathbf{T}_0 + \mathbf{H}_0^T\mathbf{A}_{12}\mathbf{T}_1 + \mathbf{H}_1^T\mathbf{A}_{22}\mathbf{T}_1 \\
&= \beta^{(0)} + \mathbf{H}_0^{\dagger}\mathbf{H}_0\mathbf{H}_1^T\mathbf{S}_1^{-1}\mathbf{H}_1\beta^{(0)} - \mathbf{H}_1^T\mathbf{S}_1^{-1}\mathbf{H}_1\beta^{(0)} \\
&\quad - \mathbf{H}_0^{\dagger}\mathbf{H}_0\mathbf{H}_1^T\mathbf{S}_1^{-1}\mathbf{T}_1 + \mathbf{H}_1^T\mathbf{S}_1^{-1}\mathbf{T}_1 \\
&= \beta^{(0)} - \left(\mathbf{I} - \mathbf{H}_0^{\dagger}\mathbf{H}_0\right)\mathbf{H}_1^T\mathbf{S}_1^{-1}\left(\mathbf{H}_1\beta^{(0)} - \mathbf{T}_1\right) \\
&= \beta^{(0)} - \mathbf{P}_0\mathbf{H}_1^T\mathbf{S}_1^{-1}\left(\mathbf{H}_1\beta^{(0)} - \mathbf{T}_1\right).
\end{aligned} \tag{9}$$

In completion of the learning of the first chunk of data, $\mathbf{P}$ is updated as follows

$$\begin{aligned}
\mathbf{P}_1 &= \mathbf{I} - \begin{bmatrix} \mathbf{H}_0 \\ \mathbf{H}_1 \end{bmatrix}^{\dagger} \begin{bmatrix} \mathbf{H}_0 \\ \mathbf{H}_1 \end{bmatrix} \\
&= \mathbf{I} - \mathbf{H}_0^{\dagger}\mathbf{H}_0 - \mathbf{H}_0^{\dagger}\mathbf{H}_0\mathbf{H}_1^T\mathbf{S}_1^{-1}\mathbf{H}_1\mathbf{H}_0^{\dagger}\mathbf{H}_0 + \mathbf{H}_1^T\mathbf{S}_1^{-1}\mathbf{H}_1 \\
&\quad \mathbf{H}_0^{\dagger}\mathbf{H}_0 + \mathbf{H}_0^{\dagger}\mathbf{H}_0\mathbf{H}_1^T\mathbf{S}_1^{-1}\mathbf{H}_1 - \mathbf{H}_1^T\mathbf{S}_1^{-1}\mathbf{H}_1 \\
&= (\mathbf{I} - \mathbf{H}_0^{\dagger}\mathbf{H}_0) + (\mathbf{I} - \mathbf{H}_0^{\dagger}\mathbf{H}_0)\mathbf{H}_1^T\mathbf{S}_1^{-1}\mathbf{H}_1\mathbf{H}_0^{\dagger}\mathbf{H}_0 \\
&\quad - (\mathbf{I} - \mathbf{H}_0^{\dagger}\mathbf{H}_0)\mathbf{H}_1^T\mathbf{S}_1^{-1}\mathbf{H}_1 \\
&= \mathbf{P}_0 - \mathbf{P}_0\mathbf{H}_1^T\mathbf{S}_1^{-1}\mathbf{H}_1\mathbf{P}_0.
\end{aligned} \tag{10}$$

As seen from equation (9), the output weight $\beta^{(1)}$ is expressed as a function of $\beta^{(0)}$, $\mathbf{S}_1$, $\mathbf{P}_0$ and newly arriving chunk of data $\mathbf{H}_1$; $\mathbf{S}_1$ can actually be expressed as the function of $\mathbf{P}_0$ and newly arriving chunk $\mathbf{H}_1$; $\mathbf{P}_0$ is updated using itself and newly arriving data only in each iteration. Therefore, whenever new data arrives, a recursive algorithm for updating the output weight β can be derived. Given $(k+1)th$ chunk of data, the hidden layer output matrix of which is $\mathbf{H}_{k+1}$, the recursive algorithm works as follows

$$\begin{aligned}
\mathbf{S}_{k+1} &= \frac{\mathbf{I}}{C} + \mathbf{H}_{k+1}\mathbf{P}_k\mathbf{H}_{k+1}^T \\
\beta^{(k+1)} &= \beta^{(k)} - \mathbf{P}_k\mathbf{H}_{k+1}^T\mathbf{S}_{k+1}^{-1}\left(\mathbf{H}_{k+1}\beta^{(k)} - \mathbf{T}_{k+1}\right)
\end{aligned} \tag{11}$$

Finally, the proposed OS-eELM-right can be summarized as follows.

The proposed algorithm can learn data chunk by chunk or one by one, thus is able to handle the situation where data is arriving sequentially. In general, there are two stages during training, namely the initialization stage and incremental learning stage.

(1) **Initialization.** Given the chunk of initial training data $\aleph_0 = \{(\mathbf{x}_i, \mathbf{t}_i)\}_{i=1}^{N_0}$ and $\mathbf{x}_i \in \mathbf{R}^n, \mathbf{t}_i \in \mathbf{R}^m$,

  a) Randomly generate L hidden nodes (additive hidden nodes or RBF hidden nodes). Good performance can be achieved normally when L is assigned a large value.
  b) Calculate the initial hidden layer output matrix $\mathbf{H}_0$.
  c) Compute the initial output weight $\beta^{(0)} = \mathbf{H}_0^T(\frac{\mathbf{I}}{C} + \mathbf{H}_0\mathbf{H}_0^T)^{-1}\mathbf{T}_0$.
  d) Calculate $\mathbf{P}_0 = \mathbf{I} - \hat{\mathbf{H}_0}^{\dagger}\mathbf{H}_0 = \mathbf{I} - \mathbf{H}_0^T(\frac{\mathbf{I}}{C} + \mathbf{H}_0\mathbf{H}_0^T)^{-1}\mathbf{H}_0$.
  e) Set $k = 0$.

(2) **Incremental Learning.** Given $(k+1)th$ chunk of data $\aleph_{k+1} = \{(\mathbf{x}_i, \mathbf{t}_i)\}_{i=(\sum_{j=0}^{k} N_j)+1}^{\sum_{j=0}^{k+1} N_j}$, where N_{k+1} presents the number of training samples in the $(k+1)th$ chunk,

  a) Calculate the hidden layer output matrix $\mathbf{H}_{k+1}$ for the $(k+1)th$ chunk of data.
  b) Update the output weight:

$$\mathbf{S}_{k+1} = \frac{\mathbf{I}}{C} + \mathbf{H}_{k+1}\mathbf{P}_k\mathbf{H}_{k+1}^T$$
$$\beta^{(k+1)} = \beta^{(k)} - \mathbf{P}_k\mathbf{H}_{k+1}^T\mathbf{S}_{k+1}^{-1}\left(\mathbf{H}_{k+1}\beta^{(k)} - \mathbf{T}_{k+1}\right) \tag{12}$$

  c) Update $\mathbf{P}_{k+1}$: $\mathbf{P}_{k+1} = \mathbf{P}_k - \mathbf{P}_k\mathbf{H}_{k+1}^T\mathbf{S}_{k+1}^{-1}\mathbf{H}_{k+1}\mathbf{P}_k$.
  d) Set $k = k+1$. Go to (2).

4 DISCUSSIONS

The choice of OS-eELM-left or OS-eELM-right is dependent on the size of initial training data N_0. It is wise to choose OS-eELM-right when a small scale of initial training data is provided, otherwise OS-eELM-left is a better choice. Since good performance is achieved normally when the number of hidden nodes is set a large value, which is 1,000 in (Huang et al., 2011) corresponding to up to 40,000 observations, the threshold for the choice of left or right is set the same as 1,000. OS-eELM-left is recommended when more than 1,000 observations are given during the initial training phase; otherwise OS-eELM-right is supposed to save the computation and time.

Both of OS-eELM-left and OS-eELM-right can learn data on a one-by-one basis or chunk-by-chunk basis where the chunk size can be varied. When only one observation $(\mathbf{x}_{k+1}, t_{k+1})$ is provided during the incremental learning phase, the updating equation for $\beta^{(k+1)}$ and $\mathbf{P}_{k+1}$ can be further simplified

$$\begin{aligned}\beta^{(k+1)} &= \beta^{(k)} - \frac{\mathbf{P}_k\mathbf{h}_{k+1}^T(\mathbf{h}_{k+1}\beta^{(k)} - t_{k+1})}{\frac{1}{C} + \mathbf{h}_{k+1}\mathbf{P}_k\mathbf{h}_{k+1}^T}\\ \mathbf{P}_{k+1} &= \mathbf{P}_k - \frac{\mathbf{P}_k\mathbf{h}_{k+1}^T\mathbf{h}_{k+1}\mathbf{P}_k}{\frac{1}{C} + \mathbf{h}_{k+1}\mathbf{P}_k\mathbf{h}_{k+1}^T}.\end{aligned} \tag{13}$$

Similar to OS-ELM, batch mode eELM-left can be considered as the special case of OS-eELM-left while batch mode eELM-right is the special case of OS-eELM-right, when $N_0 = N$.

5 PERFORMANCE EVALUATION

Since OS-ELM has been verified to have better performance than other well-known sequential learning algorithms (Liang et al., 2006), in this paper, the proposed OS-eELM-right is compared with OS-ELM and OS-eELM-left. For sequential learnings, the number of initial training samples is set the same as in (Liang et al., 2006). All simulations are conducted on a laptop with Pentium Dual CPU 1.86GHz and 2GB memory.

Table 1: Specification of Benchmark Data Sets.

Dataset	#Attributes	#Classes	#Train Data	#Test Data
Auto-MPG	7	-	320	72
Abalone	8	-	3,000	1,177
California Housing	8	-	8,000	12,640
Image Segment	19	7	1,500	810
Satellite Image	36	6	4,435	2,000
DNA	180	3	2,000	1,186

5.1 Data Sets Specification

As shown in Table 1, six benchmarking data sets (Blake and Merz, 1998) have been studied in the simulations, including three regression problems (auto-MPG and abalone), and three multi-class classification problems (image segmentation and satellite image). Same as in (Liang et al., 2006), the input attributes and output attributes of regression problems are normalized into the range of [0,1]; the input attributes of classification problems are normalized into the range of [-1,1].

Table 2: Performance comparison on regression problems.

Regression		Auto-MPG					Abalone					California Housing				
	ELM-Based	Time	RMSE		Parameter		Time	RMSE		Parameter		Time	RMSE		Parameter	
	Algorithms	(s)	Training	Testing	C	L	(s)	Training	Testing	C	L	(s)	Training	Testing	C	L
Sequential 20-by-20	OS-ELM	0.0103	0.0686	0.0759	-	25	0.0437	0.0753	0.0779	-	25	0.3204	0.1304	0.1331	-	50
	OS-eELM-left	3.9827	0.0607	0.0724	2^8	1000	23.9346	0.0719	0.0774	2^{12}	1000	75.1894	0.1246	0.1282	2^{10}	1000
	OS-eELM-right	2.1338	0.0610	**0.0709**	2^8	1000	2.1338	0.0610	**0.0709**	2^8	1000	78.7415	0.1247	0.1282	2^{10}	1000
Sequential [10,30]	OS-ELM	0.0115	0.0680	0.0781	-	25	0.0711	0.0752	0.0783	-	25	0.4842	0.1303	0.1322	-	50
	OS-eELM-left	3.8794	0.0612	0.0706	2^8	1000	26.3018	0.0724	0.0771	2^{10}	1000	78.7824	0.1200	**0.1294**	2^{14}	1000
	OS-eELM-right	2.6289	0.0578	**0.0701**	2^{10}	1000	15.8310	0.0752	0.0769	2^4	1000	71.4949	0.1203	0.1300	2^{14}	1000
Sequential 1-by-1	OS-ELM	0.0365	0.0690	0.0739	-	25	0.2315	0.0755	0.0780	-	25	3.4161	0.1300	0.1338	-	50
	OS-eELM-left	17.3220	0.0643	0.0714	2^6	1000	131.1391	0.0736	0.0766	2^8	1000	537.7711	0.1185	**0.1285**	2^{16}	1000
	OS-eELM-right	14.1596	0.0644	**0.0708**	2^6	1000	106.4070	0.0724	0.0769	2^{10}	1000	497.1973	0.1203	0.1300	2^{14}	1000

Table 3: Performance comparison on multi-class classification problems.

Classification		Image Segment					Satellite Image					DNA				
	ELM-Based	Time	Rates(%)		Parameter		Time	Rates(%)		Parameter		Time	Rates(%)		Parameter	
	Algorithms	(s)	Training	Testing	C	L	(s)	Training	Testing	C	L	(s)	Training	Testing	C	L
Sequential 20-by-20	OS-ELM	0.7806	97.02	94.88	-	180	19.9366	91.96	88.95	-	400	1.9425	92.67	87.94	-	200
	OS-eELM-left	17.7426	98.19	95.91	2^8	1000	55.5763	94.62	89.95	2^6	1000	20.5104	97.26	93.75	2^{-8}	1000
	OS-eELM-right	11.9593	98.11	**95.99**	2^8	1000	52.0285	93.92	**90.04**	2^4	1000	20.1678	97.25	93.73	2^{-8}	1000
Sequential [10,30]	OS-ELM	1.1691	97.05	94.89	-	180	28.7451	91.97	88.98	-	400	1.8754	92.87	88.06	-	200
	OS-eELM-left	15.6828	98.54	95.94	2^{10}	1000	55.2733	94.64	**89.91**	2^6	1000	20.9615	97.30	**93.66**	2^{-8}	1000
	OS-eELM-right	9.4103	98.56	**96.00**	2^{10}	1000	51.5571	95.03	89.67	2^8	1000	21.3806	97.32	**93.77**	2^{-8}	1000
Sequential 1-by-1	OS-ELM	10.3722	97.05	94.75	-	180	377.1901	92.00	89.00	-	400	24.2925	92.57	87.91	-	200
	OS-eELM-left	93.6714	97.58	95.88	2^6	1000	284.5552	93.93	89.99	2^4	1000	120.4346	97.28	**93.74**	2^{-8}	1000
	OS-eELM-right	88.2663	98.17	**95.92**	2^8	1000	280.7665	93.94	89.98	2^4	1000	132.8770	97.24	93.65	2^{-8}	1000

5.2 Parameter Settings

Sigmoidal additive hidden nodes are selected for all the algorithms. Other hidden nodes, such as RBF nodes, can be studied in the future work. Similar to (Huang et al., 2011; Huang et al., 2010), good performance is achieved usually when the number of hidden nodes L is large. The performances of our proposed algorithm and OS-eELM-left are insensitive to L as well (Figure 1). Therefore, it is convenient that L is fixed as 1000 for both algorithms. Another parameter C which determines the positive value added to the diagonal needs to be specified by users. A wide rage of C $\{2^{-24}, 2^{-23}, \cdots, 2^{24}, 2^{25}\}$ is validated for the optimal testing rate.

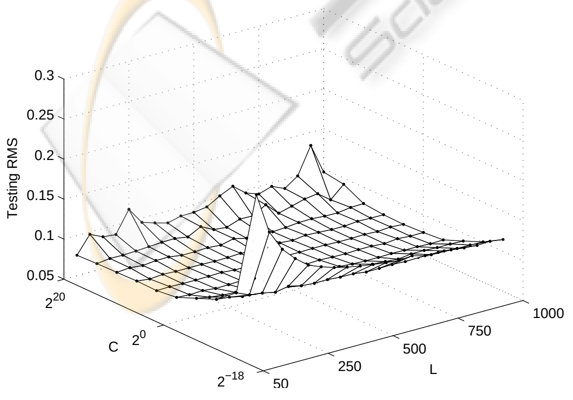

Figure 1: Testing RMSE of abalone with respect to C and L.

5.3 Comparison of Average Testing Accuracy

The average testing RMSE for regression problems and average testing rate for classification problems are obtained over 50 trials.

It can be observed from Table 2 and Table 3 that for online sequential learning, the accuracy on testing data is improved when a positive value is added to the diagonal of $\mathbf{H}^T\mathbf{H}$ for left pseudo-inverse or $\mathbf{H}\mathbf{H}^T$ for right pseudo-inverse. It is also shown that the sequential algorithms using right or left pseudo-inverse obtain similar testing accuracy.

5.4 Comparison of Training Time

Different from the original ELM, the number of hidden nodes is fixed as a large value (1000) in the enhanced ELM. That is the reason why enhanced ELM tends to be slower than the original ELM in both batch and sequential learning mode.

It is not difficult to find out from (12) that the computational complexity of the incremental learning phases for both OS-eELM-left and OS-eELM-right are similar. Therefore, Main focus on analysis of computational complexity is given to the initial phases for both formulas. For right pseudo-inverse of the form $\mathbf{H}^\dagger = \mathbf{H}^T(\frac{\mathbf{I}}{C} + \mathbf{H}\mathbf{H}^T)^{-1}$, matrix inversion is carried out on a $N \times N$ matrix. While it is $L \times L$ in the case of left pseudo-inverse. Therefore, in terms of the training speed, right pseudo-inverse is preferred when the size of training data is small; otherwise left pseudo-inverse is more appropriate. Hence, it can be observed from Table 2 and Table 3 that OS-eELM-right runs relatively faster than OS-eELM-left since the size of initial training data is chosen much smaller than the number of hidden nodes.

6 CONCLUSIONS

In this paper, an online sequential learning algorithm for SLFNS and other regularization networks based on the enhanced ELM is proposed, which is capable of learning data on a one-by-one basis or chunk-by-chunk basis. Simulations on six benchmarking datasets have shown that, by adding a positive value to the diagonal of $\mathbf{HH}^T$ and $\mathbf{H}^T\mathbf{H}$, the generalization performance of our proposed methods outperform the original OS-ELM. In addition, in the simulations, OS-eELM-right is more suitable for sequential learning than OS-eELM-left concerning the issue of training speed since there are less than 1,000 observations during the initial training phase. Different hidden nodes, such as RBF nodes, can be implemented in the future work.

REFERENCES

Asirvadam, V. S., McLoone, S. F., and Irwin, G. W. (2002). Parallel and separable recursive levenberg-marquardt training algorithm. In *12th IEEE Workshop on Neural Networks for Signal Processing*, pages 129–138. IEEE.

Blake and Merz, C. J. (1998). UCI repository of machine learning databases.

Boyd, S., Ghaoui, L. E., Feron, E., and Balakrishnan, V. (1994). *Linear Matrix Inequalities in System and Control Theory*. Society for Industrial and Applied Mathematic.

Deng, W., Zheng, Q., and Chen, L. (2009). Proximal support vector machine classifiers. In *IEEE Symposium on Computational Intelligence and Data Mining (CIDM 09)*, pages 389–395.

Feng, G., Huang, G.-B., Lin, Q., and Gay, R. (2009). Error minimized extreme learning machine with growth of hidden nodes and incremental learning. *IEEE Transactions on Neural Networks*, 20(8):1352–1357.

Hoerl, A. E. and Kennard, R. W. (1970). Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67.

Huang, G.-B. and Chen, L. (2007). Convex incremental extreme learning machine. *Neurocomputing*, 70:3056–3062.

Huang, G.-B. and Chen, L. (2008). Enhanced random search based incremental extreme learning machine. *Neurocomputing*, 71:3460–3468.

Huang, G.-B., Chen, L., and Siew, C.-K. (2006a). Universal approximation using incremental constructive feedforward networks with random hidden nodes. *IEEE Transactions on Neural Networks*, 17(4):879–892.

Huang, G.-B., Ding, X., and Zhou, H. (2010). Optimization method based extreme learning machine for classification. *Neurocomputing*, 74:155–163.

Huang, G.-B., Saratchandran, P., and Sundararajan, N. (2004). An efficient sequential learning algorithm for growing and pruning rbf (gap-rbf) networks. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 34.

Huang, G.-B., Saratchandran, P., and Sundararajan, N. (2005). A generalized growing and pruning rbf (ggap-rbf) neural network for function approximation. *IEEE Transactions on Neural Networks*, 16(1):57–67.

Huang, G.-B., Zhou, H., Ding, X., and Zhang, R. (2011). Extreme learning machine for regression and multiclass classification. *(in press) IEEE Transactions on Systems, Man, and Cybernetics*.

Huang, G.-B., Zhu, Q.-Y., and Siew, C.-K. (2006b). Extreme learning machine: Theory and applications. *Neurocomputing*, 70(1-3):489 – 501.

Liang, N.-Y., Huang, G.-B., Saratchandran, P., and Sundararajan, N. (2006). A fast and accurate online sequential learning algorithm for feedforward networks. *IEEE Transactions on Neural Networks*, 17(6):1411–1423.

Ngia, L. S., Sjöberg, J., and Viberg, M. (1998). Adaptive neural nets filter using a recursive levenberg-marquardt search direction. In *the 32nd Asilomar Conference on Signals, Systems and Computers*, CA, USA.

Rao, C. R. and Mitra, S. K. (1971). *Generalized Inverse of Matrices and its Applications*. John Wiley & Sons, Inc, New York.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning representations by back-propagation errors. *Nature*, 323:533–536.

Serre, D. (2002). *Matrices: Theory and Applications*. Springer-Verlag New York, Inc.

Toh, K.-A. (2008). Deterministic neural classification. *Neural Computation*, 20(6):1565–1595.