On-Line Sequential Extreme Learning Machine.

Conference Paper · January 2005

Source: DBLP

CITATIONS
139

READS
1,816

5 authors, including:

Guang-Bin Huang
Nanyang Technological University
246 PUBLICATIONS **42,995** CITATIONS
SEE PROFILE

Nanying Liang
Institute for Infocomm Research
11 PUBLICATIONS **1,949** CITATIONS
SEE PROFILE

Hai-Jun Rong
Xi'an Jiaotong University
55 PUBLICATIONS **1,534** CITATIONS
SEE PROFILE

Narasimhan Sundararajan
Nanyang Technological University
259 PUBLICATIONS **9,409** CITATIONS
SEE PROFILE

the IASTED International Conference on Computational Intelligence (CI 2005), Calgary, Canada, July 4-6, 2005

On-Line Sequential Extreme Learning Machine

Guang-Bin Huang, Nan-Ying Liang, Hai-Jun Rong, P. Saratchandran, and N. Sundararajan
School of Electrical and Electronic Engineering
Nanyang Technological University
Nanyang Avenue, Singapore 639798
E-mail: egbhuang@ntu.edu.sg

Abstract

The primitive **E**xtreme **L**earning **M**achine (ELM) [1, 2, 3] with *additive neurons and RBF kernels* was implemented in batch mode. In this paper, its sequential modification based on recursive least-squares (RLS) algorithm, which referred as **O**nline **S**equential **E**xtreme **L**earning **M**achine (OS-ELM), is introduced. Based on OS-ELM, Online Sequential **Fuzzy E**xtreme **L**earning **M**achine (Fuzzy-ELM) is also introduced to implement zero order TSK model and first order TSK model. The performance of OS-ELM and Fuzzy-ELM are evaluated and compared with other popular sequential learning algorithms, and experimental results on some real benchmark regression problems show that the proposed Online Sequential Extreme Learning Machine (OS-ELM) produces better generalization performance at very fast learning speed.

Index Terms - Online Sequential Extreme Learning Machine (OS-ELM), Online Sequential Fuzzy Extreme Learning Machine (Fuzzy-ELM), GAP-RBF, MRAN.

1 Introduction

It is well-known that conventional neural networks usually complete learning procedure very slowly. It is not surprising to see that it may take several minutes and several hours to train neural networks in most applications and control parameters (i.e, learning rate, learning epochs, stopping criteria and other pre-defined parameters) must be appropriately selected in advance. Without appropriate values for these control parameters, a neural network may not be trained successfully and the trained neural network may be either overfitting or incapable of learning. Unlike traditional popular implementations which normally tune all the parameters (weights and hidden neuron biases for **s**ingle-hidden **l**ayer **f**eedforward neural **n**etworks (SLFNs) with additive neurons or centers and impact widths for SLFNs with RBF kernels), Huang, et al [1, 2, 3] have recently proposed a new batch learning algorithm called **e**xtreme **l**earning **m**achine (ELM) for SLFNs which randomly chooses the input weights and the hidden neurons' biases for SLFN with addtive neurons or centers and impact widths for SLFNs with RBF kernels, and then *analytically determines* instead of tuning/adjusting the output weights of SLFNs. Input weights are the weights of the connections between input neurons and hidden neurons and output weights are the weights of the connections between hidden neurons and output neurons. In theory, it has been shown [4] that SLFNs' input weights and hidden neurons' biases need not be adjusted during training and one may simply randomly assign values to them. The experimental results based on a few artificial and real benchmark function regression and classification problems have shown that compared with other gradient-descent based learning algorithms (such as backpropagation algorithms (BP)) for feedforward networks this ELM algorithm tends to provide better generalization performance at extremely fast learning speed and the learning phase of many applications can now be completed within seconds[1]. It should be noted that the conventional tuning-based training methods such as backpropagation learning algorithms may not prevent the troubling issues such as stopping criteria, learning rate, learning epochs, and local minima. However, this ELM learning algorithm (in batch mode) can fully avoid these difficulties very well. In some online industrial applications, sequential learning algorithms may be preferred over batch learning algorithms as they do not require retraining whenever a new data is received. In this paper, its sequential modification based on recursive least-squares (RLS) algorithm, which referred as **O**nline **S**equential Extreme Learning Machine (OS-ELM) and Online Sequential Fuzzy Extreme Learning Machine (Fuzzy-ELM), are first introduced. OS-ELM and Fuzzy-ELM can learn the training data one-by-one or chunk by chunk (with fixed or varying size) and discard the training data as long as the training procedure for those data is completed.

The paper is organized as follows. Section 2 gives a brief review of the primitive batch ELM. Section 3 presents the derivation of the OS-ELM in certain details. Section 4 introduces the structure of fuzzy neural networks based on RBF networks. Section 5 proposes an online sequential fuzzy ELM. The comparison on several real benchmark regression problems between the proposed online sequential learning algorithms and other sequential learning algorithms is conducted in Section 6. Discussions and conclusions are summarized in Session 7.

2 Review of Extreme Learning Machine (ELM)

This section will briefly review the primitive batch mode ELM proposed by Huang, et al [1, 2, 3].

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 with activation function $g(x)$ can approximate these N samples with zero error means that there exist β_i, $\mathbf{w}_i$ and b_i such that

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

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

The above N equations can be written compactly as:

$$\mathbf{H}\beta = \mathbf{T} \tag{2}$$

where

$$\mathbf{H}(\mathbf{w}_1, \cdots, \mathbf{w}_{\tilde{N}}, b_1, \cdots, b_{\tilde{N}}, \mathbf{x}_1, \cdots, \mathbf{x}_N) = \begin{bmatrix} g(\mathbf{w}_1 \cdot \mathbf{x}_1 + b_1) & \cdots & g(\mathbf{w}_{\tilde{N}} \cdot \mathbf{x}_1 + b_{\tilde{N}}) \\ \vdots & \cdots & \vdots \\ g(\mathbf{w}_1 \cdot \mathbf{x}_N + b_1) & \cdots & g(\mathbf{w}_{\tilde{N}} \cdot \mathbf{x}_N + b_{\tilde{N}}) \end{bmatrix}_{N \times \tilde{N}} \tag{3}$$

$$\beta = \begin{bmatrix} \beta_1^T \\ \vdots \\ \beta_{\tilde{N}}^T \end{bmatrix}_{\tilde{N} \times m} \quad \text{and} \quad \mathbf{T} = \begin{bmatrix} \mathbf{t}_1^T \\ \vdots \\ \mathbf{t}_N^T \end{bmatrix}_{N \times m} \tag{4}$$

$\mathbf{H}$ 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 $\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_N$.

Similar to SLFNs with additive neurons, for SLFNs with RBF kernel function $\phi\left(\frac{\|\mathbf{x}-\mu\|}{\sigma}\right)$ we have

$$\mathbf{H}\beta = \mathbf{T} \tag{5}$$

where $\mathbf{H}$ is the hidden layer output matrix of the RBF network (SLFNs with RBF kernels);

$$\mathbf{H}(\mu_1, \cdots, \mu_{\tilde{N}}, \sigma_1, \cdots, \sigma_{\tilde{N}}, \mathbf{x}_1, \cdots, \mathbf{x}_N) = \begin{bmatrix} \phi\left(\frac{\|\mathbf{x}_1-\mu_1\|}{\sigma_1}\right) & \cdots & \phi\left(\frac{\|\mathbf{x}_1-\mu_{\tilde{N}}\|}{\sigma_{\tilde{N}}}\right) \\ \vdots & \cdots & \vdots \\ \phi\left(\frac{\|\mathbf{x}_N-\mu_1\|}{\sigma_1}\right) & \cdots & \phi\left(\frac{\|\mathbf{x}_N-\mu_{\tilde{N}}\|}{\sigma_{\tilde{N}}}\right) \end{bmatrix}_{N \times \tilde{N}} \tag{6}$$

the ith column of $\mathbf{H}$ is the output of the ith kernel with respect to inputs $\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_N$.

As proved in some previous works [4] that SLFNs' input weights $\mathbf{w}_i$ and hidden neurons' biases b_i or centers and impacts of RBF kernels need not be adjusted during training and one may simply randomly assign values to them. In the batch ELM, the input weights and hidden biases are randomly assigned and then the output weights β are estimated as:

$$\hat{\beta} = \mathbf{H}^{\dagger}\mathbf{T} \tag{7}$$

where $\mathbf{H}^{\dagger}$ is the Moore-Penrose generalized inverse (pseudo inverse) [6] of the hidden layer output matrix $\mathbf{H}$. As thus, the three-step learning algorithm can be summarized as follows:

Batch ELM Algorithm[1, 2, 3]: 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 or kernel function ϕ, and hidden neuron or kernel number $\tilde{N}$,

step 1 Assign arbitrary input weight $\mathbf{w}_i$ and bias b_i or center μ_i and impact width σ_i, $i = 1, \cdots, \tilde{N}$.

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

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

As theoretically analyzed and experimentally demonstrated by Huang, et al [1, 2, 3] this ELM algorithm tends to provide better generalization performance at extremely fast learning speed. There may have several implementation of of pseudo inverse of $\mathbf{H}$, for example one may simply use SVD-based pseudo inverse of $\mathbf{H}$. Huang, et al [7] also extends ELM from real domain to complex domain where fully complex activation functions can be used in ELM.

3 Proposed Online Sequential Extreme Learning Machine (OS-ELM)

In the derivation of sequential ELM, only the specific matrix $\mathbf{H}$ is considered, where $\mathbf{H} \in \mathbf{R}^{N \times \tilde{N}}$, $N \geq \tilde{N}$, and $rank(\mathbf{H}) = \tilde{N}$ the number of hidden neurons. Under this condition, the following implementation of pseudo inverse of $\mathbf{H}$ is easily derived and given by:

$$\mathbf{H}^{\dagger} = (\mathbf{H}^{\mathbf{T}}\mathbf{H})^{-\mathbf{1}}\mathbf{H}^{\mathbf{T}} \tag{8}$$

which is often called the *left pseudo inverse* of $\mathbf{H}$ from the fact that $\mathbf{H}^{\dagger}\mathbf{H} = \mathbf{I}_{\tilde{\mathbf{N}}}$. Substitute equation (8) into equation (7), the corresponding estimation of β is given by:

$$\hat{\beta} = (\mathbf{H}^{\mathbf{T}}\mathbf{H})^{-\mathbf{1}}\mathbf{H}^{\mathbf{T}}\mathbf{T} \tag{9}$$

which is called the *least-squares solution* to $\mathbf{H}\beta = \mathbf{T}$. Sequential implementation of *least-squares solution* of equation (9) results is our sequential ELM. Hereafter we talk about the sequential implementation of *least-squares solution* of equation (9) followed by the derivation of sequential

ELM algorithm. The sequential implementation of equation (9) can be derived and referred as the *recursive least-squares* (RLS) algorithm. The integrated proof of the RLS algorithm can be found in Chong and Żak [8]. A summarization of the RLS algorithm is given here:

RLS Algorithm: Given $\mathbf{H}\beta = \mathbf{T}$, we can define $\mathbf{H}_0 = [\mathbf{h}_1 \cdots \mathbf{h}_{\tilde{N}}]^T$ which is the sub-matrix comprising of the first $\tilde{N}$ rows of $\mathbf{H}$, where $rank(\mathbf{H}_0) = rank(\mathbf{H}) = \tilde{N}$, and $\mathbf{T}_0 = [\mathbf{t}_1 \cdots \mathbf{t}_{\tilde{N}}]^T$.

step 1 Initialize $\mathbf{M}_0 = (\mathbf{H}_0^T\mathbf{H}_0)^{-1}$ and $\beta^{(0)} = \mathbf{M}_0\mathbf{H}_0^T\mathbf{T}_0$.

step 2 For each $\mathbf{h}_{k+1}$ ($(\tilde{N}+k+1)$-th row of $\mathbf{H}$) and $\mathbf{t}_{k+1} = \mathbf{T}^T_{(\tilde{N}+k+1)}$, estimate

$$\begin{aligned}\mathbf{M}_{k+1} &= \mathbf{M}_k - \frac{\mathbf{M}_k\mathbf{h}_{k+1}\mathbf{h}_{k+1}^T\mathbf{M}_k}{1+\mathbf{h}_{k+1}^T\mathbf{M}_k\mathbf{h}_{k+1}} \\ \beta^{(k+1)} &= \beta^{(k)} + \mathbf{M}_{k+1}\mathbf{h}_{k+1}(\mathbf{t}_{k+1}^T - \mathbf{h}_{k+1}^T\beta^{(k)})\end{aligned} \tag{10}$$

where $k = 0, 1, 2, \cdots, N - \tilde{N} - 1$.

Based on the RLS algorithm, the **O**nline **S**equential **E**xtreme **L**earning **M**achine (OS-ELM) is derived as the following:

Proposed Online Sequential ELM (OS-ELM) Algorithm: Given an activation function g (which may or may not be a sigmoid function) or RBF kernel ϕ, and hidden neuron or RBF kernel number $\tilde{N}$ for a specific application,

step 1 **Boosting Phase:** Given a small initial training set $\aleph = \{(\mathbf{x}_i, \mathbf{t}_i) | \mathbf{x}_i \in \mathbf{R}^n, \mathbf{t}_i \in \mathbf{R}^m, i = 1, \cdots, \tilde{N}\}$ to boost the learning algorithm first through the following boosting procedure:

(a) Assign arbitrary input weight $\mathbf{w}_i$ and bias b_i or center μ_i and impact width σ_i, $i = 1, \cdots, \tilde{N}$.

(b) Calculate the initial hidden layer output matrix $\mathbf{H}_0 = [\mathbf{h}_1, \cdots, \mathbf{h}_{\tilde{N}}]^T$, where $\mathbf{h}_i = [g(\mathbf{w}_1 \cdot \mathbf{x}_i + b_1), \cdots, g(\mathbf{w}_{\tilde{N}} \cdot \mathbf{x}_i + b_{\tilde{N}})]^T$, $i = 1, \cdots, \tilde{N}$.

(c) Estimate the initial output weight $\beta^{(0)} = \mathbf{M}_0\mathbf{H}_0^T\mathbf{T}_0$, where $\mathbf{M}_0 = (\mathbf{H}_0^T\mathbf{H}_0)^{-1}$ and $\mathbf{T}_0 = [\mathbf{t}_1, \cdots, \mathbf{t}_{\tilde{N}}]^T$.

(d) Set $k = 0$.

step 2 **Sequential Learning Phase:** for *each* further coming observation $(\mathbf{x}_i, \mathbf{t}_i)$, where, $\mathbf{x}_i \in \mathbf{R}^n, \mathbf{t}_i \in \mathbf{R}^m$ and $i = \tilde{N}+1, \tilde{N}+2, \tilde{N}+3, \cdots$, do

(a) Calculate the hidden layer output vector $\mathbf{h}_{(k+1)} = [g(\mathbf{w}_1 \cdot \mathbf{x}_i + b_1), \cdots, g(\mathbf{w}_{\tilde{N}} \cdot \mathbf{x}_i + b_{\tilde{N}})]^T$.

(b) Calculate latest output weight $\beta^{(k+1)}$ based on RLS algorithm:

$$\begin{aligned}\mathbf{M}_{k+1} &= \mathbf{M}_k - \frac{\mathbf{M}_k\mathbf{h}_{k+1}\mathbf{h}_{k+1}^T\mathbf{M}_k}{1+\mathbf{h}_{k+1}^T\mathbf{M}_k\mathbf{h}_{k+1}} \\ \beta^{(k+1)} &= \beta^{(k)} + \mathbf{M}_{k+1}\mathbf{h}_{k+1}(\mathbf{t}_i^T - \mathbf{h}_{k+1}^T\beta^{(k)})\end{aligned} \tag{11}$$

(c) Set $k = k + 1$.

The OS-ELM consists of two main phases. Boosting phase is to train the SLFNs using the primitive ELM method with some batch of training data in the initialization stage and these boosting training data will be discarded as soon as boosting phase is completed. The required batch of training data are very small, which can be equal to the number of hidden neurons. For example, if there are 10 neurons, we may need 10 training samples to boost the learning. Similar to the previous work [9, 10] it can be easily proved that $rank(\mathbf{H}_0) = \tilde{N}$ if the first $\tilde{N}$ are distinct. After boosting phase, the OS-ELM will then learn the train data one-by-one or chunk-by-chunk and all the training data will be discarded once the learning procedure on these data is completed.

4 Structure of Fuzzy Neural Networks Based on RBF Network

The structure of fuzzy neural networks based on RBF neural networks illustrated by Figure 1 consists of five layers to realizes the following TSK fuzzy model:

Rule k : if $(x_1$ is $A_{1k}) \cdots (x_u$ is $A_{uk})$, then $(y_1$ is $\omega_{1k}) \cdots (y_m$ is $\omega_{mk})$

where $\omega_{jk} (j = 1, 2, \cdots, m, k = 1, 2, \cdots, \tilde{N})$ is the consequent terms in the kth rule. When the consequent terms are considered as the linear function about input variables $\omega_{jk} = a_{jk0} + a_{jk1}x_1 + \cdots + a_{jku}x_u$, the TSK model is called as first order TSK fuzzy model. When the consequent parts are considered as real constants $\omega_{jk} = c_{jk}$, the TSK model is called as zero order TSK fuzzy model. $A_{ik} (i = 1, 2, \cdots, u)$ is the membership value of the ith input variable x_i in rule k, u is the dimension of the input vector $\mathbf{x} (\mathbf{x} = [x_1, \cdots, x_u]^T)$, $\tilde{N}$ is the number of fuzzy rules, m is the dimension of the output vector $\mathbf{y} (\mathbf{y} = [y_1, \cdots, y_m]^T)$.

Layer 1: In layer 1, each node represents an input variable and directly transmits the input signal to layer 2.

Layer 2: In the layer each node represents the membership value of each input variable. According to the function equivalence between a RBF network and a FIS and thus its antecedent part (if part) in fuzzy rules is achieved by Gaussian functions of the RBF network. The membership value $A_{ik}(x_i)$ of the ith input variable x_i in the kth Gaussian function is given by

$$A_{ik}(x_i) = \exp\left(-\frac{(x_i - \mu_{ik})^2}{\sigma_k^2}\right), k = 1, 2, \cdots, \tilde{N} \tag{12}$$

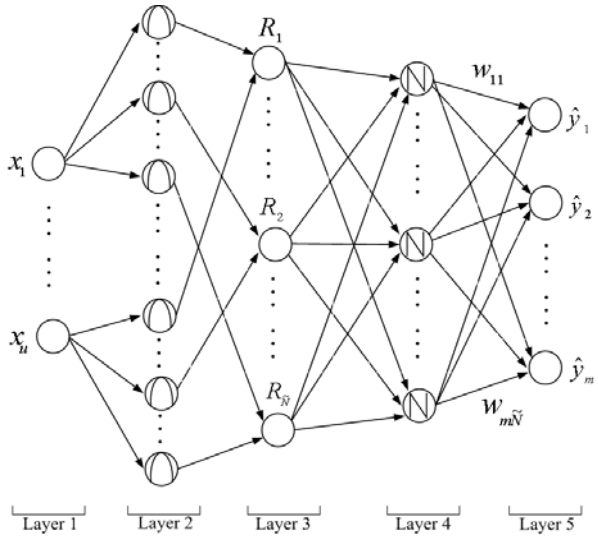

Figure 1: Structure of fuzzy neural networks based on RBF network

where $\tilde{N}$ is the number of the Gaussian functions, μ_{ik} is the center of the kth Gaussian function for the ith input variable, σ_k is the width of the kth Gaussian function.

Layer 3: Each node in the layer represents the if part of if-then rules obtained by the sum-product composition. The firing strength (if part) of the kth rule is given by

$$R_k(X) = \prod_{i=1}^{u} A_{ik}(x_i) = \exp\left(-\sum_{i=1}^{u} \frac{(x_i - \mu_{ik})^2}{\sigma_k^2}\right) = \exp\left(-\frac{\|\mathbf{x} - \mu_k\|^2}{\sigma_k^2}\right) \tag{13}$$

Layer 4: The nodes in the layer are named as normalized nodes whose number is equal to the number of the nodes in third layer. The kth normalized node is equal to the following equation:

$$\bar{R}_k = \frac{R_k(\mathbf{x})}{\sum_{k=1}^{\tilde{N}} R_k(\mathbf{x})} \tag{14}$$

Layer 5: Each node in the layer corresponds to an output variable, which is achieved by the weighted sum of the output of each normalized rule. The system output is calculated by

$$\mathbf{y} = \frac{\sum_{k=1}^{\tilde{N}} R_k(\mathbf{x})\omega_k}{\sum_{k=1}^{\tilde{N}} R_k(\mathbf{x})} \tag{15}$$

where

$$\mathbf{y} = [y_1, y_2, \cdots, y_m]^T, \omega_k = [\omega_{k1}, \omega_{k2}, \cdots, \omega_{km}]^T.$$

5 Proposed Online Sequential Fuzzy Extreme Learning Machine (Fuzzy-ELM)

For N arbitrary distinct samples $(\mathbf{x}_n, \mathbf{t}_n)$, where $\mathbf{x}_n = [x_{n1}, x_{n2}, \cdots, x_{nu}]^T \in \mathbf{R}^u$ and $\mathbf{t}_n = [t_{n1}, t_{n2}, \cdots, t_{nm}]^T \in \mathbf{R}^m$, fuzzy neural networks based on RBF network with $\tilde{N}$ fuzzy rules are mathematically modeled as

$$\mathbf{H}\beta = \mathbf{T} \tag{16}$$

where $\mathbf{H}$ is the normalization node output matrix; the kth column of $\mathbf{H}$ is the kth rule's normalization output vector with respect to inputs $\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_N$; β is the parameters existing in the consequent terms.

For zero order TSK model,

$$\mathbf{H}(\mu_1, \cdots, \mu_{\tilde{N}}, \sigma_1, \cdots, \sigma_{\tilde{N}}, \mathbf{x}_1, \cdots, \mathbf{x}_N) = \begin{bmatrix} \bar{R}(\mu_1, \mathbf{x}_1, \sigma_1) & \cdots & \bar{R}(\mu_{\tilde{N}}, \mathbf{x}_1, \sigma_{\tilde{N}}) \\ \vdots & \cdots & \vdots \\ \bar{R}(\mu_1, \mathbf{x}_N, \sigma_1) & \cdots & \bar{R}(\mu_{\tilde{N}}, \mathbf{x}_N, \sigma_{\tilde{N}}) \end{bmatrix}_{N \times \tilde{N}}$$
$$\beta = \begin{bmatrix} c_{11} \cdots c_{m1} \\ \vdots \\ c_{1\tilde{N}} \cdots c_{m\tilde{N}} \end{bmatrix}_{\tilde{N} \times m}, \quad \mathbf{T} = \begin{bmatrix} \mathbf{t}_1^T \\ \vdots \\ \mathbf{t}_N^T \end{bmatrix}_{N \times m} \tag{17}$$

For first order TSK model,

$$\mathbf{H}(\mu_1, \cdots, \mu_{\tilde{N}}, \sigma_1, \cdots, \sigma_{\tilde{N}}, \mathbf{x}_1, \cdots, \mathbf{x}_N) = \begin{bmatrix} \bar{R}(\mu_1, \mathbf{x}_1, \sigma_1) & \cdots & \bar{R}(\mu_1, \mathbf{x}_N, \sigma_1) \\ \vdots & \cdots & \vdots \\ \bar{R}(\mu_{\tilde{N}}, \mathbf{x}_1, \sigma_{\tilde{N}}) & \cdots & \bar{R}(\mu_{\tilde{N}}, \mathbf{x}_N, \sigma_{\tilde{N}}) \\ \bar{R}(\mu_1, \mathbf{x}_1, \sigma_1)x_{11} & \cdots & \bar{R}(\mu_1, \mathbf{x}_N, \sigma_1)x_{1N} \\ \vdots & \cdots & \vdots \\ \bar{R}(\mu_{\tilde{N}}, \mathbf{x}_1, \sigma_{\tilde{N}})x_{11} & \cdots & \bar{R}(\mu_{\tilde{N}}, \mathbf{x}_N, \sigma_{\tilde{N}})x_{1N} \\ \vdots & \cdots & \vdots \\ \bar{R}(\mu_1, \mathbf{x}_1, \sigma_1)x_{u1} & \cdots & \bar{R}(\mu_1, \mathbf{x}_N, \sigma_1)x_{uN} \\ \vdots & \cdots & \vdots \\ \bar{R}(\mu_{\tilde{N}}, \mathbf{x}_1, \sigma_{\tilde{N}})x_{u1} & \cdots & \bar{R}(\mu_{\tilde{N}}, \mathbf{x}_N, \sigma_{\tilde{N}})x_{uN} \end{bmatrix}^T_{\tilde{N}(u+1) \times N} \tag{18}$$

$$\beta = \begin{bmatrix} a_{110} & \cdots & a_{m10} \\ \vdots & \vdots & \vdots \\ a_{1\tilde{N}0} & \cdots & a_{m\tilde{N}0} \\ a_{111} & \cdots & a_{m11} \\ \vdots & \vdots & \vdots \\ a_{1\tilde{N}1} & \cdots & a_{m\tilde{N}1} \\ \vdots & \cdots & \vdots \\ a_{11u} & \cdots & a_{m1u} \\ \vdots & \vdots & \vdots \\ a_{1\tilde{N}u} & \cdots & a_{m\tilde{N}u} \end{bmatrix}_{\tilde{N}(u+1) \times m}, \quad \mathbf{T} = \begin{bmatrix} \mathbf{t}_1^T \\ \vdots \\ \mathbf{t}_N^T \end{bmatrix}_{N \times m} \tag{19}$$

Simiar to Online Sequential ELM (OS-ELM), in Fuzzy-ELM the centers and widths of RBF kernels need

Table 1: Performance comparison between OS-ELM and other sequential algorithms on regression application.

Databases	Algorithms	Time (s)	Testing RMSE	# Neurons
Auto-MPG	**OS-ELM-FF**	0.0326	0.0696	25
	OS-ELM-RBF	0.0987	0.0695	25
	Stochastic BP	0.0875	0.1028	13
	GAP-RBF[5]	0.4520	0.1404	3.12
	MRAN[5]	1.4644	0.1376	4.46
Abalone	**OS-ELM-FF**	0.3154	0.0775	25
	OS-ELM-RBF	0.9776	0.0773	25
	Stochastic BP	0.7472	0.0972	11
	GAP-RBF[5]	83.784	0.0966	23.62
	MRAN[5]	1500.4	0.0837	87.571

not be adjusted during training and one may simply randomly assign values to them. Thus, we have

Proposed Online Sequential Fuzzy ELM (Fuzzy-ELM) Algorithm: Given Gaussian membership function and rule number $\tilde{N}$ for a specific application,

step 1 **Boosting Phase:** Given a small initial training set $\aleph = \{(\mathbf{x}_n, \mathbf{t}_n) | \mathbf{x}_n \in \mathbf{R}^u, \mathbf{t}_n \in \mathbf{R}^m, n = 1, \cdots, \tilde{N}\}$ to boost the learning algorithm first through the following boosting procedure:

(a) Assign random centers μ_n and widths σ_n, $n = 1, \cdots, \tilde{N}$.

(b) Calculate the normalization node output matrix $\mathbf{H}_0 = [\mathbf{h}_1, \cdots, \mathbf{h}_{\tilde{N}}]^T$.

(c) Estimate the initial parameters $\beta^{(0)} = \mathbf{M}_0 \mathbf{H}_0^T \mathbf{T}_0$, where $\mathbf{M}_0 = (\mathbf{H}_0^T \mathbf{H}_0)^{-1}$ and $\mathbf{T}_0 = [\mathbf{t}_1, \cdots, \mathbf{t}_{\tilde{N}}]^T$.

(d) Set $l = 0$.

step 2 **Sequential Learning Phase:** for *each* further coming observation $(\mathbf{x}_n, \mathbf{t}_n)$, where, $\mathbf{x}_n \in \mathbf{R}^u, \mathbf{t}_n \in \mathbf{R}^m$ and $n = \tilde{N}+1, \tilde{N}+2, \tilde{N}+3, \cdots$, do

(a) Calculate the normalization node output vector $\mathbf{h}_{l+1}$; For zero order TSK model

$$\mathbf{h}_{(l+1)} = [\bar{R}(\mu_n, \mathbf{x}_n, \sigma_1), \cdots, \bar{R}(\mu_{\tilde{N}}, \mathbf{x}_n, \sigma_{\tilde{N}})]^T$$

For first order TSK model

$$\mathbf{h}_{(l+1)} = [\bar{R}(\mu_1, \mathbf{x}_n, \sigma_1), \cdots, \bar{R}(\mu_{\tilde{N}}, \mathbf{x}_n, \sigma_{\tilde{N}}), \cdots, \bar{R}(\mu_1, \mathbf{x}_n, \sigma_1) x_{un}, \cdots, \bar{R}(\mu_{\tilde{N}}, \mathbf{x}_n, \sigma_{\tilde{N}}) x_{un}]^T$$

(b) Calculate latest parameters $\beta^{(l+1)}$ based on RLS algorithm:

$$\mathbf{M}_{l+1} = \mathbf{M}_l - \frac{\mathbf{M}_l \mathbf{h}_{l+1} \mathbf{h}_{l+1}^T \mathbf{M}_l}{1 + \mathbf{h}_{l+1}^T \mathbf{M}_l \mathbf{h}_{l+1}}$$
$$\beta^{(l+1)} = \beta^{(l)} + \mathbf{M}_{l+1} \mathbf{h}_{l+1} (\mathbf{t}_n^T - \mathbf{h}_{l+1}^T \beta^{(l)}) \quad (20)$$

(c) Set $l = l + 1$.

6 Performance Evaluation of Proposed Online Sequential ELM (OS-ELM)

In this section, the performance of the proposed Online Sequential Extreme Learning Machine (OS-ELM) is compared with the popular sequential learning algorithms such as stochastic gradient back-propagation (BP) algorithm, MRAN, GAP-RBF[5] and GGAP-RBF on some real benchmark regression problems such as Auto-MPG and Abalone. There are 320 training and 72 testing data for Auto-MPG and 3000 training data and 1177 testing data for Abalone. All the simulations are carried out in MATLAB 6.5 environment running in an ordinary PC with 1.9 GHZ CPU. The activation function used in the OS-ELM and stochastic gradient BP algorithm is a simple sigmoidal function $g(x) = 1/(1 + \exp(-x))$. The RBF kernel function used in OS-ELM is $\phi(x) = \exp(-\frac{\|\mathbf{x}-\mu\|^2}{\sigma^2})$. 50 trials have been conducted for OS-ELM[1] and stochastic gradient BP and the average results are shown in Table 1. As observed from Table 1, OS-ELM can obtain better generalization performance than other popular sequential learning algorithms in these real cases and complete learning at very fast speed.

The performance of the proposed Online Sequential Fuzzy Extreme Learning Machine (Fuzzy-ELM) is compared with other learning algorithms such as ESOM[11], EFuNN[12] and DENFIS[13]. Fuzzy neural networks based on RBF network is trained for 50 trials by means of Fuzzy-ELM to implement zero order TSK model (Zero-Fuzzy-ELM) and first order TSK model (First-Fuzzy-ELM). Average results are illustrated by Table 2.

The Mackey-Glass time series prediction is generated from the following differential equation[13]:

$$\frac{dx(t)}{dt} = \frac{0.2x(t-\tau)}{1 + x^{10}(t-\tau)} - 0.1x(t) \quad (21)$$

where $\tau = 17$ and $x(0) = 1.2$. For the purpose of training and testing, 6000 samples are produced by means of the fourth-order Runge-Kutta method. The prediction task is to predict the value $x(t+85)$ from the input vector $[x(t-18)\ x(t-12)\ x(t-6)\ x(t)]$ for any value of the time t. To compare with the algorithms shown in Kasabov and Song[13], the observations between $t = 201$ and $t = 3200$ and the observations between $t = 5001$ and $t = 5500$ are extracted from the series and used as training and testing data. To compare conveniently, the testing accuracy is based on the Non-Dimensional Error Index (NDEI) defined as the RMSE divided by the standard deviation of the true output values

Performance comparison of Zero-Fuzzy-ELM and First-Fuzzy-ELM with ESOM, EFuNN and DENFIS is shown in Table 2. From the table it can be seen that

[1]ELM Source Codes are available at: http://www.ntu.edu.sg/home/egbhuang/.

First-Fuzzy-ELM acquires better testing accuracy compared with other algorithms with the fewest fuzzy rule number, and also Zero-Fuzzy-ELM obtains better testing accuracy compared with these algorithms with fewer fuzzy rule number than ESOM and EFUNN and with more fuzzy rule number than DENFIS.

Table 2: Performance comparison for Mackey-Glass time series prediction

Methods	Network Size	Testing NDEI
Zero-Fuzzy-ELM	70	0.270
First-Fuzzy-ELM	15	0.266
ESOM[11]	114	0.320
EFuNN[12]	193	0.401
DENFIS[13]	58	0.276

7 Conclusions

In this paper, online sequential versions of the extreme learning machine (OS-ELM and Fuzzy-ELM) have bee proposed. The newly proposed OS-ELM works for both additive neurons and RBF kernels, and has no other control parameters for users to mannually tuning when given OS-ELM architecture. Fuzzy-ELM has been proposed to train the fuzzy neural networks based on RBF network to implement the zero order TSK model and first order TSK model. The newly proposed OS-ELM and Fuzzy-ELM can learn the train data one-by-one without retraining all the historic data. In fact, all the latest coming training observations can be discarded as long as the learning procedure on them is completed. The algorithm is compared with other well known sequential learning algorithms on some real world benchmark regression and time series problems and the simulation results indicate that the proposed OS-ELM and Fuzzy-ELM produce higher generalization performance with much less training time. The OS-ELM and Fuzzy-ELM proposed in this paper indeed also works for online sequential learning with varying length of coming chunk data and under certain mild condition the generalization performance of the proposed OS-ELM and Fuzzy-ELM can be proved similar to the batch ELM and these aspects will be rigorously investigated and proved in our near future work. The original ELM now becomes only a specific case of OS-ELM when all the training data are used at boost stage.

References

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

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

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

[4] G.-B. Huang, L. Chen, and C.-K. Siew, "Universal approximation using incremental constructive networks," *revised and resubmitted to IEEE Transactions on Neural Networks*, 2005.

[5] G.-B. Huang, P. Saratchandran, and N. Sundararajan, "An efficient sequential learning algorithm for growing and pruning RBF (GAP-RBF) networks," *IEEE Transactions on Systems, Man and Cybernetics, Part B*, vol. 34, no. 6, pp. 2284–2292, 2004.

[6] D. Serre, *Matrices: Theory and applications*. New York: Springer-Verlag, 2002.

[7] M.-B. Li, G.-B. Huang, P. Saratchandran, and N. Sundararajan, "Fully complex extreme learning machine," *to appear in Neurocomputing*, 2005.

[8] E. K. P. Chong and S. H. Żak, *An introduction to optimization*. New York: John Wiley, 2001.

[9] 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.

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

[11] D. Deng and N. Kasabov, "Evolving self-organizing maps for online learning, data analysis and modeling," in *Proceedings of the IJCNN'*2000 *on Neural Networks Neural Computing: New Challenges Perspectives New Millennium* (S.-I. Amari, C. L. Giles, M. Gori, and V. Piuri, eds.), vol. VI, pp. 3–8, IEEE Press, 2000.

[12] N. Kasabov, "Evolving fuzzy neural networks for supervised/unsupervised online knowlege-based learning," *IEEE Transactions on Systems, Man, and Cybernetic-Part B: Cybernetics*, vol. 31, no. 6, pp. 902–918, 2001.

[13] N. K. Kasabov and Q. Song, "DENFIS: Dynamic evolving neural-fuzzy inference system and its application for time series prediction," *IEEE Transactions on Fuzzy Systems*, vol. 10, no. 2, pp. 144–154, 2002.