World Wide Web
DOI 10.1007/s11280-013-0220-x

Extreme learning machine with errors in variables

Jianwei Zhao · Zhihui Wang · Feilong Cao

Received: 6 September 2012 / Revised: 22 February 2013 / Accepted: 21 April 2013

Abstract Extreme learning machine (ELM) is widely used in training single-hidden layer feedforward neural networks (SLFNs) because of its good generalization and fast speed. However, most improved ELMs usually discuss the approximation problem for sample data with output noises, not for sample data with noises both in input and output values, i.e., error-in-variable (EIV) model. In this paper, a novel algorithm, called (regularized) TLS-ELM, is proposed to approximate the EIV model based on ELM and total least squares (TLS) method. The proposed TLS-ELM uses the idea of ELM to choose the hidden weights, and applies TLS method to determine the output weights. Furthermore, the perturbation quantities of hidden output matrix and observed values are given simultaneously. Comparison experiments of our proposed TLS-ELM with least square method, TLS method and ELM show that our proposed TLS-ELM has better accuracy and less training time.

Keywords Extreme learning machine · Errors-in-variables model · Total least square · Regularization

1 Introduction

With their universal approximation capability [2–4, 7, 26, 27], single-hidden layer feedforward neural networks (SLFNs) are often chosen to be the class of approximation functions for an unknown function. As one of the most state-of-the-art algorithms for training SLFNs, extreme learning machine (ELM) [11, 12] has attracted wide people's attention because of its fast learning speed and high generalization. So far, ELM has been widely applied in many fields, such as image processing [20, 21, 30, 33], surface reconstruction [31] and commerce [22].

J. Zhao · Z. Wang · F. Cao (✉)
Department of Information and Mathematics Sciences, China Jiliang University, Hangzhou 310018, Zhejiang Province, People's Republic of China
e-mail: feilongcao@gmail.com

Published online: 03 May 2013

Up to now, there have been all kinds of modified ELMs to enhance their performances and simplify the structures of neural networks [6, 15–17, 24, 28, 29, 32]. Some of them consider good methods for calculating the network weights [6, 15, 16, 24, 32], and the others think over the training algorithms for some special practical problems based on ELM [17, 28, 29], such as online sequential extreme learning machine (OS-ELM) [17] and OS-ELM with forgetting mechanism [29]. In any case, the basic idea contained in those modified ELMs is that all the hidden weights are randomly assigned and the output weights β are determined by the following linear system:

$$H\beta = T + \Delta T, \tag{1.1}$$

where H is the hidden layer output matrix, and ΔT is the unknown measurement error for the observed values of sample data. Therefore, how to calculate or approximate the output weights in some forms of optimization functions becomes the most important question. Until now, the output weights are often considered under the (weighted) least square error or regularized least square error [5, 19].

However, it is a common phenomena that sample data often have noises both in the input and output values in many practical areas, such as computer vision, image reconstruction, speech and audio processing and astronomy. Those noises come from many factors such as instrumental error and environmental interferences. In mathematics, the model that has measurement errors both in the independent variables and dependent variables is called errors-in-variables (EIV) model [9]. For this model, there are two important problems: (1) How to approximate the unknown function from some sample data with noises in both independent variables and dependent variables? (2) How to revise those sample data with noises? There have been some research works for linear EIV model with total least squares (TLS) method [13, 14], that is, the unknown objective function is linear. How about the non-linear EIV model?

In this paper, SLFNs are still taken to be the class of approximation functions for sample data with noises. But we provide a new algorithm, called TLS-ELM, to train SLFNs for non-linear EIV model with TLS method and ELM. The proposed TLS-ELM utilizes the idea of ELM to determine the hidden weights, and the idea of TLS method to estimate the optimal output weight vector β, intermediate perturbation quantity for input variables and the measurement errors for output variables that are used to revise those noise sample data. Numerical experiments show that the Frobenius norm of measurement errors in input and output variables are extremely small compared with training errors of ELM that is just the minimization of residual errors for output variables.

The outline of this paper is as follows. Section 2 reviews ELM. A new algorithm, called (regularized) TLS-ELM, is proposed in Section 3 based on the TLS method and ELM. Section 4 gives some comparison experiments of our proposed method with some other state-in-the-art methods. Conclusions of this paper are highlighted in Section 5.

2 Brief review of ELM

This section briefly reviews ELM proposed by [11] to provide the necessary background for the development of TLS-ELM in Section 3.

The output of an SLFN with L hidden nodes (additive or RBF nodes) can be represented by

$$f_L(\mathbf{x}) = \sum_{i=1}^{L} \beta_i G(\mathbf{a}_i, b_i, \mathbf{x}), \quad \mathbf{x} \in \mathbb{R}^n, \tag{2.1}$$

where $\mathbf{a}_i \in \mathbb{R}^n$ and $b_i \in \mathbb{R}$ are the learning parameters of hidden nodes, $\beta_i \in \mathbb{R}^m$ is the output weight, and $G(\mathbf{a}_i, b_i, \mathbf{x})$ denotes the output of the i-th hidden node with respect to the input $\mathbf{x}$. Additive hidden node and RBF hidden node are often used in applications. For the additive hidden node, $G(\mathbf{a}_i, b_i, \mathbf{x})$ is given by

$$G(\mathbf{a}_i, b_i, \mathbf{x}) = g(\mathbf{a}_i \cdot \mathbf{x} + b_i), \tag{2.2}$$

where $g : \mathbb{R} \to \mathbb{R}$ is an activation function and $\mathbf{a}_i \cdot \mathbf{x}$ denotes the inner product of vectors $\mathbf{a}_i$ and $\mathbf{x}$ in $\mathbb{R}^n$. For the RBF hidden node, $G(\mathbf{a}_i, b_i, \mathbf{x})$ is given by

$$G(\mathbf{a}_i, b_i, \mathbf{x}) = g(b_i \parallel \mathbf{x} - \mathbf{a}_i \parallel), \tag{2.3}$$

where $g : \mathbb{R} \to \mathbb{R}$ is an activation function, $\mathbf{a}_i$ and $b_i (b_i > 0)$ are the center and impact factor of i-th RBF node, respectively.

In supervised batch learning, learning algorithms use finite input-output samples for training. For a given set of distinct training data $\{(\mathbf{x}_i, \mathbf{t}_i)\}_{i=1}^{N} \subset \mathbb{R}^n \times \mathbb{R}^m$, where $\mathbf{x}_i$ is an $n \times 1$ input vector and $\mathbf{t}_i$ is the corresponding $m \times 1$ observation vector, an SLFN with L hidden nodes approximating those N training data with zero error means that there exist $\mathbf{a}_i$, b_i and β_i such that

$$\sum_{i=1}^{L} \beta_i G(\mathbf{a}_i, b_i, \mathbf{x}_j) = \mathbf{t}_j + \Delta\mathbf{t}_j, \quad j = 1, 2, \ldots, N. \tag{2.4}$$

Equation (2.4) can be written compactly as

$$\mathbf{H}\beta = \mathbf{T} + \Delta\mathbf{T}, \tag{2.5}$$

where

$$\begin{aligned} &\mathbf{H}(\mathbf{a}_1, \ldots, \mathbf{a}_L, b_1, \ldots, b_L, \mathbf{x}_1, \ldots, \mathbf{x}_N) \\ &\quad = \begin{bmatrix} G(\mathbf{a}_1, b_1, \mathbf{x}_1) & \cdots & G(\mathbf{a}_L, b_L, \mathbf{x}_1) \\ \vdots & \ddots & \vdots \\ G(\mathbf{a}_1, b_1, \mathbf{x}_N) & \cdots & G(\mathbf{a}_L, b_L, \mathbf{x}_N) \end{bmatrix}_{N \times L}, \end{aligned} \tag{2.6}$$

$$\beta = \begin{bmatrix} \beta_1^\top \\ \vdots \\ \beta_L^\top \end{bmatrix}_{L \times m}, \quad \mathbf{T} = \begin{bmatrix} \mathbf{t}_1^\top \\ \vdots \\ \mathbf{t}_N^\top \end{bmatrix}_{N \times m} \quad \text{and} \quad \Delta\mathbf{T} = \begin{bmatrix} \Delta\mathbf{t}_1^\top \\ \vdots \\ \Delta\mathbf{t}_N^\top \end{bmatrix}_{N \times m} \tag{2.7}$$

Here $\beta^\top$ denotes the transpose of the vector β. As named by Huang et al. [12], $\mathbf{H}$ is called the hidden-layer output matrix of the neural network.

In the process of training SLFNs with ELM, the hidden weights are assigned randomly and the output weight vector is solved by a linear system. The main steps of ELM can be summarized as follows [12]:

Algorithm ELM 2.1:

Given a set of training data $\{(\mathbf{x}_i, \mathbf{t}_i)\}_{i=1}^N \subset \mathbb{R}^n \times \mathbb{R}^m$, choose the hidden-node output function $G(\mathbf{a}, b, \mathbf{x})$ and the number of hidden nodes L in prior.

1) Randomly assign hidden weights $(\mathbf{a}_i, b_i)$, $i = 1, \cdots, L$;
2) Calculate the hidden layer output matrix $\mathbf{H}$;
3) Compute the output weights vector with $\beta = \mathbf{H}^{\dagger}\mathbf{T}$.

3 Proposed TLS-ELM

Since sample data usually have noises both in input variables and output variables in practical applications, while traditional ELM only considers the case of noises in output variables, it is necessary to give a new algorithm for dealing with the EIV model. In this section, we propose a new algorithm, named TLS-ELM, to solve this problem based on TLS method and ELM.

Consider the set of N distinct sample data $\{(\mathbf{x}_i, \mathbf{t}_i)\}_{i=1}^N \subset \mathbb{R}^n \times \mathbb{R}^m$ with input noise $\Delta\mathbf{x}_i$ and output noise $\Delta\mathbf{t}_i$, respectively, $i = 1, 2, \ldots, N$. Then each true input value is $\mathbf{x}_j + \Delta\mathbf{x}_j$ and the corresponding true output value is $\mathbf{t}_j + \Delta\mathbf{t}_j$, $i = 1, 2, \ldots, N$. In order to determine the unknown function, we still take SLFNs with the form of (2.1) as the class of approximation tools for sample data with noises. So an SLFN with L hidden nodes approximating these N training data with zero error means that there exist $\mathbf{a}_i$, b_i and β_i such that

$$\sum_{i=1}^{L} \beta_i G(\mathbf{a}_i, b_i, \mathbf{x}_j + \Delta\mathbf{x}_j) = \mathbf{t}_j + \Delta\mathbf{t}_j, \quad j = 1, 2, \ldots, N. \tag{3.1}$$

Equation (3.1) can be written compactly as

$$(\mathbf{H} + \Delta\mathbf{H})\beta = \mathbf{T} + \Delta\mathbf{T}, \tag{3.2}$$

where $\mathbf{H}$ is the hidden output matrix in (2.6) with respect to input variables $\mathbf{x}_j$, $j = 1, 2, \ldots, N$ and $\Delta\mathbf{H}$ is the hidden perturbation matrix with respect to input noises $\Delta\mathbf{x}_j$, $j = 1, 2, \ldots, N$.

In our proposed algorithm, we also calculate the hidden weights $\mathbf{a}_i, b_i$ by the similar method in ELM, that is, we randomly assign the hidden weights $\mathbf{a}_i, b_i$, $i = 1, 2, \ldots, L$. Now the problem of (3.1) is turned into how to compute the output weights vector β under the linear system with the unknown hidden perturbation matrix $\Delta\mathbf{H}$ and $\Delta\mathbf{T}$. As we all known, ELM determines the output weights vector β by solving the linear system

$$\mathbf{H}\beta = \mathbf{T} + \Delta\mathbf{T} \tag{3.3}$$

with least square method that minimizes $\Delta\mathbf{T} = \mathbf{T} - \mathbf{H}\beta$ under 2-norm. But in the linear system (3.1), both $\mathbf{H}$ and $\mathbf{T}$ have perturbations, so just least square method can not reflect the generalization well. Fortunately, TLS method [8, 18, 25] can solve the

linear system (3.1) well. So we use TLS to seek the optimal output weights vector β, perturbation matrix ΔH and ΔT such that

$$\min_{\beta,\Delta H,\Delta T} \| [\Delta H \ \ \Delta T] \|_F \ \text{ subject to } \ (H + \Delta H)\beta = T + \Delta T, \tag{3.4}$$

where $\| \cdot \|_F$ is the Frobenius norm.

Let C be the matrix $[H \ \ T]$, then C has the singular value decomposition (SVD) $C = U\Sigma V^\top$, where $\Sigma = diag(\sigma_1, \cdots, \sigma_{L+m})$. Divide the matrix Σ and V as

$$\Sigma = \begin{matrix} L & m \\ \end{matrix} \begin{bmatrix} \Sigma_1 & 0 \\ 0 & \Sigma_2 \end{bmatrix} \begin{matrix} L \\ m \end{matrix} \text{ and } \quad V = \begin{matrix} L & m \\ \end{matrix} \begin{bmatrix} V_{11} & V_{12} \\ V_{21} & V_{22} \end{bmatrix} \begin{matrix} L \\ m, \end{matrix} \tag{3.5}$$

then if V_{22} is nonsingular, we have an optimal output weights vector

$$\hat{\beta} = -V_{12}V_{22}^{-1} \tag{3.6}$$

and the corresponding TLS correction matrix is

$$\Delta C := [\Delta H \ \ \Delta T] = -U \text{diag}(0, \Sigma_2) V^\top \tag{3.7}$$

for (3.4).

Now the proposed TLS-ELM can be given as follows:

Proposed TLS-ELM: 3.1

Given a training set $\{(\mathbf{x}_i, \mathbf{t}_i)\}_{i=1}^N \in \mathbb{R}^n \times \mathbb{R}^m$ with input noise $\Delta\mathbf{x}_i$ and output noise $\Delta\mathbf{t}_i, i = 1, 2, \ldots, N$, respectively, choose the hidden output function $G(\mathbf{a}, b, \mathbf{x})$ and the number L of hidden nodes in prior.

(1) Randomly assign hidden weights $(\mathbf{a}_i, b_i), i = 1, \cdots, L$.
(2) Calculate the hidden layer output matrix $\mathbf{H}$.
(3) Take the SVD for matrix $[\mathbf{HT}] = U\Sigma V^\top$, and divide the matrix Σ and V as
$$\Sigma = \begin{matrix} L & m \\ \end{matrix} \begin{bmatrix} \Sigma_1 & 0 \\ 0 & \Sigma_2 \end{bmatrix} \begin{matrix} L \\ m \end{matrix} \text{ and } \quad V = \begin{matrix} L & m \\ \end{matrix} \begin{bmatrix} V_{11} & V_{12} \\ V_{21} & V_{22} \end{bmatrix} \begin{matrix} L \\ m. \end{matrix}$$
(4) **If** the submatrix V_{22} of the matrix V is nonsingular, then estimate the output weights vector $\hat{\beta} = -V_{12}V_{22}^{-1}$, and the corresponding correction matrix $\Delta C = -U \text{diag}(0, \Sigma_2) V^\top$; **else** output a message that "Hidden weights are not chosen improper" and go back to step (1).

Now we begin to focus on the very ill-conditioned problem for (3.4), that is, the singular values of $[\mathbf{H} \ \ \mathbf{T}]$ decay gradually to zero. In this case, the output weights vector can be hopelessly contaminated by the noises in directions corresponding to the small singular values of $[\mathbf{H} \ \ \mathbf{T}]$. Therefore, it is necessary to compute a regularized solution in which the effect of such noises are filtered out. Here, we use the regularized TLS [8] that adopted Tikhonov regularization method in TLS to compute the solution of (3.4). Therefore, the corresponding problem to (3.4) in the process of Tikhonov regularization is

$$\min_{\beta,\Delta H,\Delta T} \| [\Delta H \ \Delta T] \|_F \ \text{ subject to } (\mathbf{H} + \Delta\mathbf{H})\beta = \mathbf{T} + \Delta\mathbf{T} \ \text{ and } \| \beta \|_2 \leq \delta, \tag{3.8}$$

where δ is a given positive constant. And the corresponding Language multiplier formulation is

$$\mathcal{L}(\mathbf{H}+\Delta\mathbf{H},\beta,\mu) = \| [\Delta\mathbf{H}\ \ \Delta\mathbf{T}] \|_F^2 + \mu(\| \beta \|_2^2 - \delta^2), \tag{3.9}$$

where μ is the Language multiplier and it is zero if the inequality constraint is inactive.

The output weights vector β is calculated by the similar method as TLS-ELM. Let $C' = [\mathbf{H}^\top\mathbf{H} + \lambda I, \quad \mathbf{H}^\top\mathbf{T}]$, where λ is the regularization parameter, then its SVD is $C' = U'\Sigma' V'^\top$, where $\Sigma' = diag(\sigma'_1, \cdots, \sigma'_{L+m})$. Divide Σ' and V' as

$$\Sigma' = \begin{matrix} L & m \\ \end{matrix} \begin{bmatrix} \Sigma'_1 & 0 \\ 0 & \Sigma'_2 \end{bmatrix} \begin{matrix} L \\ m \end{matrix} \quad \text{and} \quad V' = \begin{matrix} L & m \\ \end{matrix} \begin{bmatrix} V'_{11} & V'_{12} \\ V'_{21} & V'_{22} \end{bmatrix} \begin{matrix} L \\ m. \end{matrix} \tag{3.10}$$

If V'_{22} is nonsingular, then the optimal output weights vector is

$$\hat{\beta}' = -V'_{12}V'^{-1}_{22}. \tag{3.11}$$

So the output weights vector β is calculated with proper parameter $\lambda \in [-\sigma^2_{min}, \epsilon]$, where ϵ is a given positive constant, and the selection of λ has been discussed in [8].

Similar to TLS-ELM, we begin to compute the corresponding correction matrix $\Delta C' = [\Delta H\ \ \Delta T]$. For the corresponding Language multiplier formulation (3.9), compute its partial derivatives with respect to $\mathbf{H}+\Delta\mathbf{H}$ and β, respectively, and set them to zero, that is,

$$\begin{cases} \Delta\mathbf{H} + \Delta\mathbf{T}\beta^\top = 0; \\ (\mathbf{H}+\Delta\mathbf{H})^\top\Delta\mathbf{T} + \mu\beta = 0. \end{cases} \tag{3.12}$$

So

$$\Delta\mathbf{H} = -\Delta\mathbf{T}\beta^\top. \tag{3.13}$$

Then

$$\Delta\mathbf{T} = (\mathbf{H}+\Delta\mathbf{H})\beta - \mathbf{T} = (\mathbf{H} - \Delta\mathbf{T}\beta^\top)\beta - \mathbf{T} = \mathbf{H}\beta - \Delta\mathbf{T}\beta^\top\beta - \mathbf{T}. \tag{3.14}$$

From (3.13) and (3.14), we get

$$\begin{cases} \Delta\mathbf{T} = \dfrac{\mathbf{T} - \mathbf{H}\beta}{1 + \| \beta \|_2^2}; \\ \Delta\mathbf{H} = -\Delta\mathbf{T}\beta^\top. \end{cases} \tag{3.15}$$

With the output weights vector $\hat{\beta}'$ in (3.11), the corrected matrix $\Delta C' = [\Delta\mathbf{H}\ \Delta\mathbf{T}]$, where $\Delta\mathbf{H} = \Delta\mathbf{T}\hat{\beta}'^\top$ and $\Delta\mathbf{T} = \dfrac{\mathbf{T} - \mathbf{H}\hat{\beta}'}{1 + \| \hat{\beta}' \|_2^2}$.

As this method integrates TLS with the idea of ELM, above algorithm is called regularized TLS-ELM. Now the proposed regularized TLS-ELM can be described as follows:

Proposed regularized TLS-ELM 3.2:

Given a training set $\{(\mathbf{x}_i, \mathbf{t}_i)\}_{i=1}^{N} \in \mathbb{R}^n \times \mathbb{R}^m$ with input noise $\Delta\mathbf{x}_i$ and output noise $\Delta\mathbf{t}_i, i = 1, 2, \ldots, N$, respectively, choose the hidden output function $G(\mathbf{a}, b, \mathbf{x})$ and the number L of hidden nodes in prior.

(1) Randomly assign hidden weights $(\mathbf{a}_i, b_i), i = 1, \cdots, L$;

(2) Calculate the hidden-layer output matrix $\mathbf{H}$;

(3) Calculate the smallest singular value $\sigma_{\min}$ of augmented matrix $[\mathbf{H}\ \ \mathbf{T}]$ and determine the proper regularization parameter λ according to the value of $\sigma_{\min}$.

(4) Take the SVD for the matrix $\left[\mathbf{H}^\top\mathbf{H}+\lambda\mathbf{I},\ \ \mathbf{H}^\top\mathbf{T}\right] = U'\Sigma'V'^\top$. Divide Σ' and V' as $\Sigma' = \begin{matrix} L & m \\ \end{matrix}$ $\begin{bmatrix} \Sigma'_1 & 0 \\ 0 & \Sigma'_2 \end{bmatrix}$ $\begin{matrix} L \\ m \end{matrix}$ and $V' = \begin{matrix} L & m \\ \end{matrix}$ $\begin{bmatrix} V'_{11} & V'_{12} \\ V'_{21} & V'_{22} \end{bmatrix}$ $\begin{matrix} L \\ m. \end{matrix}$

(5) **If** V'_{22} is nonsingular, then estimate output weights $\hat{\beta}' = -V'_{12}V'^{-1}_{22}$, and the corresponding correction matrix $\Delta\mathbf{T} = \dfrac{\mathbf{T} - \mathbf{H}\hat{\beta}'}{1 + \| \hat{\beta}' \|_2^2}$, $\Delta\mathbf{H} = \Delta\mathbf{T}\hat{\beta}'^\top$; **else** output a message that "Hidden weights are not chosen improper" and go back to step (1).

4 Performance evaluation

This section presents the simulation results for EIV model, and the performance of our proposed regularized TLS-ELM is verified and compared with LS method, TLS, and ELM. Among those algorithms, LS and ELM merely minimize residual error $\Delta\mathbf{T}$ for output variables of those training data, while both TLS and regularized TLS-ELM minimize residual errors of outputs $\Delta\mathbf{T}$ and inputs $\Delta\mathbf{H}$. Similar to numerical data chosen in [1], seven groups of numerical data are random sampled from the domains according to uniform distribution and their analytical expressions are given in Table 1. For those gathered data, their inputs and outputs are disturbed with random noise according to uniform distribution on [0, 0.1].

In our experiments for ELM and the proposed regularized TLS-ELM, the hidden output function

$$G(\mathbf{x}, \mathbf{a}, b) = 1/(1 + \exp(-(\mathbf{a}\mathbf{x} + b))),$$

where the hidden weights $\mathbf{a}$ and b are randomly chosen from the interval $[-1, 1]$. Let

$$\#\,\mathbf{H} = \| \Delta\mathbf{H} \|_F \,/\, \| \mathbf{H} \|_F$$

and

$$\#\,\mathbf{T} = \| \Delta\mathbf{T} \|_F \,/\, \| \mathbf{T} \|_F,$$

Table 1 Analytical expressions of used test functions ($z = f(x, y)$)

Name	Function	Data number	Domain
Function 1	$z = \frac{sin(\sqrt{x^2+y^2})}{\sqrt{x^2+y^2}}$	1,000	$[-8, 8]\times[-8, 8]$
Function 2	$z = sin(xy)$	1,000	$[-8, 8]\times[-8, 8]$
Function 3	$z = sin(2\pi\sqrt{x^2+y^2})$	1,000	$[-8, 8]\times[-8, 8]$
Function 4	$z = (\sqrt{x^2+y^2}-6)^2+4$	800	$[-4, 4]\times[-4, 4]$
Function 5	$z = sin(x-y)$	1,000	$[-8, 8]\times[-8, 8]$
Function 6	$z = sin(x)cos(y)$	500	$[-2, 2]\times[-2, 2]$
Function 7	$z = \sqrt{\lvert x\rvert}y + x\sqrt{\lvert x\rvert}$	500	$[-2, 2]\times[-2, 2]$

then we use # **H** and # **T** to reflect the performance of above four algorithms. The numerical results of ELM and proposed regularized TLS-ELM are averaged over 10 trials for all cases.

All the experiments are carried out in MATLAB 7.10.0 environment running on a desktop with CPU 2.69 GHz and 1.96 GB RAM.

Experiment 4.1 In this experiment, the optimal number of hidden nodes for regularized TLS-ELM is selected. We take data of Function 2 as training samples. Then the relation of # H and # T with the number of hidden nodes are shown in Figure 1.

As observed from Figure 1, when the number of hidden nodes exceeds 40, the values of $\| \Delta\mathbf{H} \|_F / \| \mathbf{H} \|_F$ descend gradually to a stable level. Similarly, the optimal number of hidden nodes for $\| \Delta\mathbf{T} \|_F / \| \mathbf{T} \|_F$ is about 30. Therefore, we choose 40 hidden nodes for training samples of Function 2 in the following experiments.

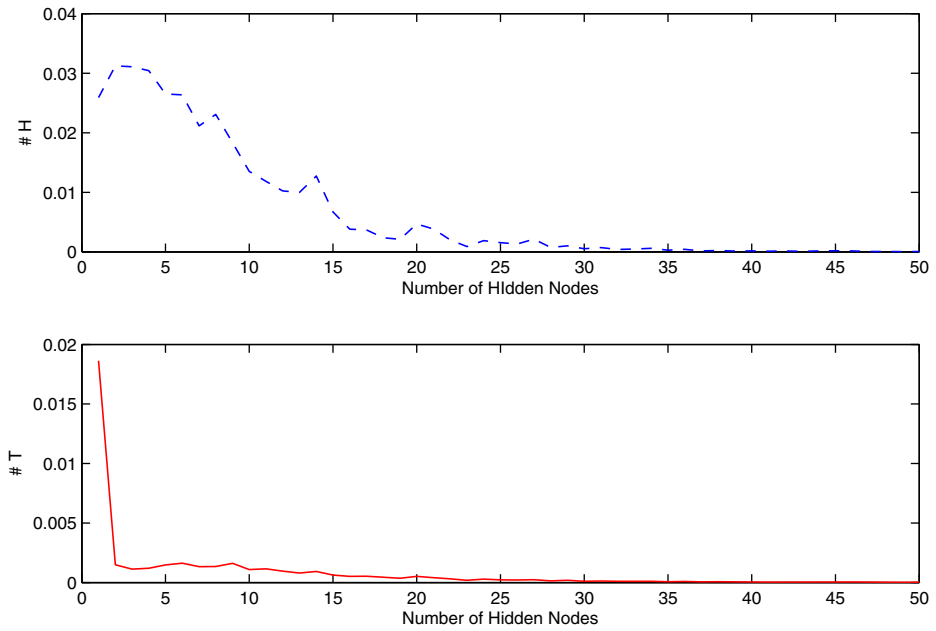

Figure 1 Learning evolution for samples of Function 2 with regularized TLS-ELM

Experiment 4.2 In this experiment, we compare the mean values and standard deviations of # H and # T for our proposed regularized TLS-ELM with LS, TLS, and ELM. The experimental results are shown in Table 2.

As observed from Table 2, LS almost fails to deal with those given samples, and TLS just has effects on those samples with smooth analytical expressions (e.g. Function 1,2,5,6). For each group of numerical samples, our proposed regularized TLS-ELM fits these numerical samples excellently and much better than ELM with same number of hidden nodes. The mean values and standard deviations of $\| \Delta\mathbf{H} \|_F / \| \mathbf{H} \|_F$ and $\| \Delta\mathbf{T} \|_F / \| \mathbf{T} \|_F$ over 10 trials are vitally small for all seven groups of numerical samples.

Experiment 4.3 In this section, we illustrate the relation of training times with different hidden nodes for our proposed regularized TLS-ELM. The training time contains the time of calculating $\sigma_{\min}$ and estimating optimal output weights $\hat{\beta}'$. We

Table 2 Comparison of the mean values and standard deviations of # H and # T for our proposed regularized TLS-ELM with LS, TLS, and ELM

Functions	Algorithm	# Nodes	$\| \Delta\mathbf{H} \|_F / \| \mathbf{H} \|_F$		$\| \Delta\mathbf{T} \|_F / \| \mathbf{T} \|_F$	
			Mean	Dev.	Mean	Dev.
Function 1	LS	–	0	–	1.0000	–
	TLS	–	9.1065e-006	–	0.0055	–
	ELM	40	0	0	0.3861	0.0374
	TLS-ELM	40	3.8514e-004	1.2953e-004	2.3500e-005	3.5845e-006
Function 2	LS	–	0	–	0.9965	–
	TLS	–	0.0014	–	0.0859	–
	ELM	40	0	0	0.9683	0.0022
	TLS-ELM	40	4.9223e-004	9.5953e-005	1.8387e-004	1.7445e-005
Function 3	LS	–	0	–	0.9992	–
	TLS	–	7.1455e-004	–	0.0421	–
	ELM	40	0	0	0.9729	0.0017
	TLS-ELM	40	4.8013e-004	9.5688e-005	2.2120e-004	1.9063e-005
Function 4	LS	–	0	–	0.9981	–
	TLS	–	0.6854	–	1.0000	–
	ELM	20	0	0	0.0570	0.0136
	TLS-ELM	20	1.2131e-007	1.6069e-007	3.6728e-005	1.9887e-005
Function 5	LS	–	0	–	0.9949	–
	TLS	–	0.0018	–	0.1032	–
	ELM	20	0	0	0.8249	0.0834
	TLS-ELM	20	1.3741e-004	1.3150e-004	5.6072e-005	3.4288e-005
Function 6	LS	–	0	–	0.7212	–
	TLS	–	0.0601	–	0.7524	–
	ELM	20	0	0	0.0782	0.0013
	TLS-ELM	20	6.0625e-005	4.4479e-005	2.1955e-005	9.4189e-006
Function 7	LS	–	0	–	0.8350	–
	TLS	–	0.5109	–	0.9839	–
	ELM	20	0	0	0.0917	0.0030
	TLS-ELM	20	1.4143e-006	1.6806e-005	1.1370e-006	7.2026e-006

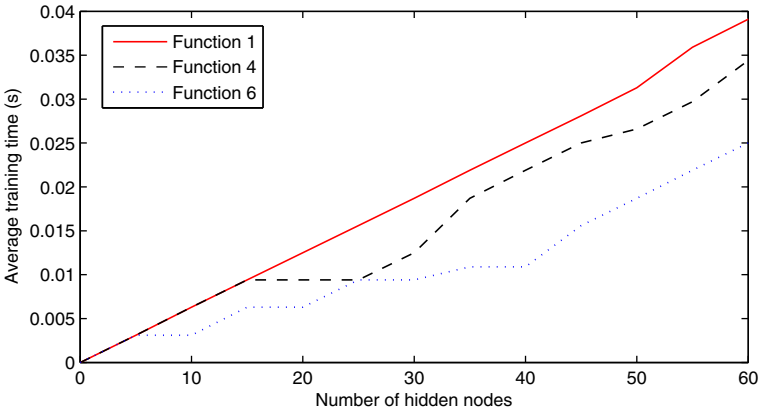

Figure 2 Relations of the average training time(s) of regularized TLS-ELM with different hidden nodes

choose 1000 samples for Function 1, 800 samples for Function 4, and 500 samples for Function 6. Then the average training time with the number of hidden nodes are shown in Figure 2.

From Figure 2, we find that even for Function 1, the training time by regularized TLS-ELM with 60 hidden nodes is less than 0.04 s. So the regularized TLS-ELM is still rather simple and efficient for EIV model.

5 Conclusions

Extreme learning machine (ELM) usually considers the approximation problem of sample data with output noises, not for error-in-variables (EIV) model that has noises both in input and output values. In this paper, we discuss the approximation problem of EIV model with a new algorithm, called (regularized) TLS-ELM. The proposed TLS-ELM uses the idea of ELM to choose the hidden weights, and applies the total least squares (TLS) method to determine the output weights vector. Furthermore, the perturbation quantities of hidden output matrix and output vector are computed simultaneously. Comparison experiments of our proposed TLS-ELM with some state-of-the-art methods show that our proposed TLS-ELM has better accuracy and less training time.

Acknowledgements The research was supported by the National Nature Science Foundation of China (No. 61101240, 61272023), the Zhejiang Provincial Natural Science Foundation of China (No. Y6110117) and the Science Foundation of Zhejiang Education Office (No. Y201122002).

References

1. Balasundaram, S., Kapil: Application of error minimized extreme learning machine for simultaneous learning of a function and its derivatives. Neurocomputing **74**, 2511–2519 (2011)
2. Cao, F.L., Lin, S.B., Xu, Z.B.: Approximation capability of interpolation neural networks. Neurocomputing **74**, 457–460 (2010)

3. Chen, T.P., Chen, H., Liu, R.W.: Approximation capability in by multiplayer feedforward networks and related problems. IEEE T. Neural Networ. **6**, 25–30 (1995)
4. Cybenko, G.: Approximation by superpositions of sigmoidal function. Math. Control Signal Systems **2**, 303–314 (1989)
5. Deng, W., Zheng, Q., Chen, L.: Regularized extreme learning machine. In: IEEE Symposium on Computational Intelligence and Data Mining (CIDM'09), pp. 389–395. Nashville, TN (2009)
6. Feng, G.R., Huang, G.-B., Lin, Q.P., Gay, R.: Error minimized extreme learning machine with growth of hidden nodes and incremental learning. IEEE T. Neural Networ. **20**(8), 1152–1157 (2009)
7. Funahashi, K.I.: On the approximate realization of continuous mappings by neural networks. Neural Netw. **2**, 183–192 (1989)
8. Golub, G.H., Christian, H.P., O'Leary, D.P.: Tikhonov regularization and total least squares. SIAM J. Matrix Anal. A **21**(1), 185–194 (1999)
9. Golub, G.H., Loan, C.F.V.: An analysis of the total least squares problem. SIAM J. Numer. Anal. **17**, 883–893 (1980)
10. Huang, G.-B., Wang, D.H., Lan, Y.: Extreme learning machines: a survey. Int. J. Mach. Learn. Cybern. **2**, 107–122 (2011)
11. Huang, G.-B., Zhu, Q.-Y., Siew, C.K.: Extreme learning machine: a new learning scheme of feedforward neural networks. In: Proceedings of the International Joint Conference on Neural Net works (IJCNN2004), pp. 25–29. Budapest, Hungary (2004)
12. Huang, G.-B., Zhu, Q.-Y., Siew, C.K.: Extreme learning machine: theory and applications. Neurocomputing **70**, 489–501 (2006)
13. Fromkorth, A., Kohler, M.: Analysis of least squares regression estimates in case of additional errors in the variables. J. Stat. Plan. Infer. **141**, 172–188 (2011)
14. Kukush, A., Markovsky, I., Van Huffel, S.: Consistency of the structured total least squares estimator in a multivariate errors-in-variables model. J. Stat. Plan. Infer. **133**, 315–358 (2005)
15. Lan, Y., Soh, Y.C., Huang, G.-B.: Constructive hidden nodes selection of extreme learning machine for regression. Neurocomputing **73**, 3191–3199 (2010)
16. Lan, Y., Soh, Y.C., Huang, G.-B.: Two-stage extreme learning machine for regression. Neurocomputing **73**, 3028–3038 (2010)
17. Liang, N.Y., Huang, G.-B., Saratchandran, P., Sundararajan, N.: A fast and accurate online sequential learning algorithm for feedforward networks. IEEE T. Neural Networ. **17**(6), 1411–1423 (2006)
18. Markovsky, I., Huffel, S.V.: Overview of total least squares methods. Signal Process. **87**, 2283–2302 (2007)
19. Martinez, J.M.M., Montero, P.E., Olivas, E.S., Guerrero, J.D.M., Benedito, R.M., Sanchis, J.G.: Regularized extreme learning machine for regression problems. Neurocomputing **74**(17), 3716–3721 (2011)
20. Minhas, R., Baradarani, A., Seifzadeh, S., Wu, Q.M.J.: Human action recognition using extreme learning machine based on visual vocabularies. Neurocomputing **73**, 1906–1917 (2010)
21. Mohammed, A.A., Minhas, R., JonathanWu, Q.M., Sid-Ahmed, M.A.: Human face recognition based on multidimensional PCA and extreme learning machine. Pattern Recogn. **44**, 2588–2597 (2011)
22. Sun, Z.L., Choi, T.M., Au, K.F., Yu, Y.: Sales forecasting using extreme learning machine with applications in fashion retailing. Decis. Support Syst. **46**, 411–419 (2008)
23. Wang, L., Huang, Y.P., Luo, X.Y., Wang, Z., Luo, S.W.: Image deblurring with filters learned by extreme learning machine. Neurocomputing **74**, 2464–2474 (2011)
24. Wang, Y.G., Cao, F.L., Yuan, Y.B.: A study on effectiveness of extreme learning machine. Neurocomputing **74**, 2483–2490 (2011)
25. Wei, Y.M., Min, N., Ng, M.K., Xu, W.: Tikhonov regularization for weighted total least squares problems. Appl. Math. Lett. **20**, 82–87 (2007)
26. Xu, Z.B., Cao, F.L.: The essential order of approximation for neural networks. Sci. China Ser. F **47**, 97–112 (2004)
27. Yeung, D.S., Ng, W.W.Y., Wang, D.F., Tsang, E.C.C., Wang, X.Z.: Localized generalization error model and its application to architecture selection for radial basis function neural network. IEEE T. Neural Networ. **18**(5), 1294–1305 (2007)
28. Zhao, J.W., Park, D.S., Lee, J.W., Cao, F.L.: Generalized extreme learning machine acting on a metric space. Soft Comput. **16**, 1503–1514 (2012)
29. Zhao, J.W., Wang, Z.H., Park, D.S.: Online sequential extreme learning machine with forgetting mechanism. Neurocomputing **87**, 79–89 (2012)

30. Zhao, J.W., Zhou, Z.H., Cao, F.L.: Human face recognition based on ensemble of polyharmonic extreme learning machine. Neural Comput. Appl. (2013). doi:10.1007/s00521-013-1356-4
31. Zhou, Z.H., Zhao, J.W., Cao, F.L.: Surface reconstruction based on extreme learning machine. Neural Comput. Appl. (2012). doi:10.1007/s00521-012-0891-8
32. Zhu, Q.-Y., Qin, A.K., Suganthan, P.N., Huang, G.-B.: Evolutionary extreme learning machine. Pattern Recogn. **38**, 1759–1763 (2005)
33. Zong, W.W., Huang, G.-B.: Face recognition based on extreme learning machine. Neurocomputing **74**, 2541–2551 (2011)