Neurocomputing 74 (2011) 2511–2519

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Application of error minimized extreme learning machine for simultaneous learning of a function and its derivatives

S. Balasundaram *, Kapil

School of Computer and Systems Sciences, Jawaharlal Nehru University, New Delhi 110067, India

ARTICLE INFO

Available online 8 May 2011

Keywords:
Derivatives approximation
Extreme learning machine
Feedforward neural networks
Function approximation
Incremental learning

ABSTRACT

In this paper a new learning algorithm is proposed for the problem of simultaneous learning of a function and its derivatives as an extension of the study of error minimized extreme learning machine for single hidden layer feedforward neural networks. Our formulation leads to solving a system of linear equations and its solution is obtained by Moore–Penrose generalized pseudo-inverse. In this approach the number of hidden nodes is automatically determined by repeatedly adding new hidden nodes to the network either one by one or group by group and updating the output weights incrementally in an efficient manner until the network output error is less than the given expected learning accuracy. For the verification of the efficiency of the proposed method a number of interesting examples are considered and the results obtained with the proposed method are compared with that of other two popular methods. It is observed that the proposed method is fast and produces similar or better generalization performance on the test data.

1. Introduction

Prediction by regression is an important method of solution for forecasting. In regression by analyzing the given input and their corresponding output values an approximation function that describes the underlying relationship between them is determined. For any unseen input example its output is predicted using this relationship.

The problem of regression estimation for a set of input examples given arises in a number of applications of practical importance like drug discovery [3], time series prediction [14,20], blind identification [18]. However in certain applications like linear circuits [17], Kalman filtering [2] it is necessary to estimate both the function and its derivatives. Recently Lazaro et al. [12] proposed a support vector regression (SVR) based approach for the simultaneous learning of a function and its derivatives that leads to solving a quadratic minimization problem. In this approach it is assumed that at each input example both the function and its derivative values are given together and to obtain the resulting optimal solution an iterative reweighted least squares (IRWLS) procedure is applied. In [10], the problem of simultaneous learning of a function and its derivatives is formulated using a regularized least squares SVR. This formulation allows the set of input examples at which the function values and the set where the derivatives to be given independent of each other. The main advantage of this formulation is that the solution is obtained by inverting a single matrix of order equals to the sum of the number of input examples at which the function and the derivative values are given. Finally for the study of simultaneous learning of a function and its derivatives using neural networks and/or its applications see [1,11,13,15].

In [6], Huang et al. proposed a new non-iterative learning algorithm for single hidden layer feedforward networks (SLFNs) architecture called extreme learning machine (ELM) which overcomes the problems caused by gradient decent based algorithms such as back propagation. In this algorithm the input weights and bias are randomly chosen. The ELM formulation leads to solving a system of linear equations in terms of the unknown weights connecting the hidden layer to the output layer and its solution is obtained using Moore–Penrose generalized pseudo-inverse [16]. Although ELM is a simple and an efficient learning algorithm, the number of hidden nodes of the SLFN is a parameter and its value is to be given at the beginning of the algorithm. Recently an error minimization based approach has been proposed in [4] to automatically determine the number of hidden nodes of the SLFN. In this approach the hidden nodes are allowed to grow one by one or group by group and once new hidden nodes are added the output weights are incrementally updated in an efficient manner. This process of adding more number of hidden nodes and determining its output weights is continued until the desired learning accuracy is achieved. Finally for the interesting work of an extension

* Corresponding author. Tel.: +91 11 26704724; fax: +91 11 26741586.
E-mail addresses: bala_jnu@hotmail.com, balajnu@gmail.com (S. Balasundaram), navkapil@gmail.com (Kapil).

0925-2312/$ - see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.neucom.2010.12.033

of ELM to support vector networks and the application of ELM for time series the interested reader is referred to [9,19].

In this paper, we extend the study of error minimized ELM (EM-ELM) [4] proposed for estimation of a function to the problem of simultaneous learning of a function and its derivatives. Though our method of solution is applicable to the problem of simultaneous learning of a function and its derivatives of higher order, for reason of simplicity, we consider only the problem of simultaneous learning of a function and its first-order derivatives. The main advantage of our approach is that our formulation will lead to solving a rectangular system of linear equations and its solution is obtained by Moore–Penrose generalized pseudo-inverse [16]. Finally to verify the effectiveness of our method a number of examples are considered. It is observed that the proposed method is fast and produces similar or better generalization performance clearly demonstrates its practical use.

The rest of the paper is organized as follows: in Section 2 we briefly discuss the ELM initially proposed by Huang et al. [6] for SLFNs. For the problem of simultaneous learning of a function and its derivatives, the approach proposed by Lazaro et al. [12] in formulating the problem as an extended ε-insensitive SVR problem and the regularized least squares approach of Jayadeva et al. [10] are briefly discussed in Section 3. Motivated by their work [10,12] we propose in Section 4 the extension of the study of EM-ELM for the problem of simultaneous learning of a function and its derivatives. Experimental results are reported in Section 5 and finally we conclude our paper in Section 6.

Throughout in this work all vectors are assumed as column vectors. For any two vectors x, y in R^n the inner product of the vectors is denoted by $x^t y$ where x^t is the transpose of the vector x. The 2-norm of a vector x is denoted by $\|x\|$.

2. Extreme learning machine method

Let $\{(x_i,y_i)\}_{i=1,2,\ldots,p}$ be a set of training examples given where for the input example $x_i=(x_{i1},\ldots,x_{in})^t\in R^n$ its corresponding observed value of the function being $y_i\in R$. Then for the randomly assigned values for the weight vector $a_s=(a_{s1},\ldots,a_{sn})^t\in R^n$ and the bias $b_s\in R$ connecting the input layer to the sth hidden node, the standard SLFNs with ℓ number of hidden nodes approximate the input examples with zero error if and only if there exists an output vector $w=(w_1,\ldots,w_\ell)^t\in R^\ell$ in which w_s is the weight connecting the sth hidden node to the output node such that the following condition:

$$y_i=\sum_{s=1}^{\ell} w_s G(a_s,b_s,x_i) \quad \text{for } i=1,\ldots,p$$

holds, where $G(a_s, b_s, x_i)$ is the output of the sth hidden node for the input x_i. This set of equations can be written in matrix form as

$$Hw=y, \tag{1}$$

where

$$H=\begin{bmatrix} G(a_1,b_1,x_1) & \ldots & G(a_\ell,b_\ell,x_1) \\ \cdot & \ldots & \cdot \\ G(a_1,b_1,x_p) & \ldots & G(a_\ell,b_\ell,x_p) \end{bmatrix}_{p\times\ell} \tag{2}$$

and

$$y=(y_1,\ldots,y_p)^t\in R^p. \tag{3}$$

For additive hidden node with activation function g: $R\to R$, $G(a_s, b_s, x)$ is given by

$$G(a_s,b_s,x)=g(a_s^t x+b_s),$$

where a_s and b_s are the weight vector and bias connecting the input layer to the sth hidden node. Similarly for radial basis function (RBF) hidden node with activation function g: $R\to R$, $G(a_s,b_s,x)$ is given by

$$G(a_s,b_s,x)=g(b_s\|x-a_s\|),$$

where a_s and $b_s>0$ are the center and impact factor of the sth RBF node.

For a given SLFN for which the activation function $g(\cdot)$ in any interval is infinitely differentiable and the p training examples are distinct with the number of hidden nodes ℓ equals to p, for any randomly chosen $a_s\in R^n$ and $b_s\in R$ from any intervals of R^n and R, respectively, according to any continuous probability distribution, it has been shown in [6] that with probability one the hidden layer output matrix H of the SLFN defined by (2) is invertible and $\|Hw-y\|=0$. However, in real applications $\ell<p$ is true and in this case for the randomly assigned values of the parameters $a_s\in R^n$ and $b_s\in R$, training the SLFN is equivalent to obtaining a least squares solution w of the linear system (1). In fact, w is determined to be the minimum norm least squares solution of (1) which can be explicitly obtained by [6]

$$w=H^\dagger y,$$

where $H^\dagger$ is the Moore–Penrose generalized inverse [16] of the matrix H. Also when $\mathrm{rank}(H)=\ell$, we can write

$$H^\dagger=(H^tH)^{-1}H^t, \tag{4}$$

where H^t is the transpose of H. Finally by obtaining the solution $w\in R^\ell$, the regression estimation function $f(\cdot)$ for any input example $x\in R^n$ is determined to be

$$f(x)=\sum_{s=1}^{\ell} w_s G(a_s,b_s,x). \tag{5}$$

Remark 1. Once the values of the weight vector $a_s\in R^n$ and the bias $b_s\in R$ are randomly assigned at the beginning of the learning algorithm they remain fixed and therefore the matrix H is unchanged.

Remark 2. Since the sigmoidal, radial basis, sine, cosine and exponential functions are infinitely differentiable in any interval of definition they can be chosen as activation functions.

Finally, it is important to note that, according to ELM theory, ELM works for generalized feedforward networks which may not be neuron alike [5,7,8].

3. The SVR approach of Lazaro et al. [12] and Jayadeva et al. [10] with derivatives

3.1. Support vector regression with derivatives (SVRD)

In [12], Lazaro et al. proposed the regression approximation problem for the simultaneous learning of a function and its derivatives formulated as an extended ε-insensitive loss function based SVR problem. However in order to optimize the regression estimation problem they employed an IRWLS procedure. They studied initially the method for solving one-dimensional input space problems and later extended their work to the general case.

In this subsection we briefly discuss the model of Lazaro et al. [12] for one-dimensional input space problems and for the general case of multidimensional input spaces the interested reader is referred to [12].

Assume that a set of input examples $\{(x_i,y_i,y_i')\}_{i=1,2,\ldots,p}$ is given where x_i denotes the ith input example for which the observed values of the function and its derivative being $y_i\in R$ and $y_i'\in R$, respectively. Consider the problem of finding the regression

estimation function $f: R \to R$ such that

$$f(x) = w^t \phi(x) + b,$$

where ϕ: $R \to R^d$ $(d > 1)$ maps x into a higher-dimensional feature space, which is introduced to achieve the nonlinearity of the algorithm, and the unknowns $w \in R^d$, $b \in R$ are obtained as the solution of the following SVR problem defined by [12]

$$\text{Min } L_P(w,b,\xi,\xi^*,\eta,\eta^*) = \frac{1}{2} w^t w + C_1 \sum_{i=1}^{p} (\xi_i + \xi_i^*) + C_2 \sum_{i=1}^{p} (\eta_i + \eta_i^*) \quad (6)$$

subject to

$$w^t \phi(x_i) + b - y_i \le \varepsilon + \xi_i,$$

$$y_i - w^t \phi(x_i) - b \le \varepsilon + \xi_i^*,$$

$$w^t \phi'(x_i) - y_i' \le \varepsilon_1 + \eta_i,$$

$$y_i' - w^t \phi'(x_i) \le \varepsilon_1 + \eta_i^*,$$

$$\xi_i, \xi_i^*, \eta_i, \eta_i^* \ge 0 \quad \text{for } i = 1,2,\ldots,p,$$

where ξ_i, ξ_i^*, η_i, η_i^* are slack variables; ε, $\varepsilon_1 > 0$; C_1, C_2 are parameters and $\phi'(x)$ is the derivative of $\phi(x)$.

As in the classical SVR formulation [21], the dual of the above problem (6) can be formulated as a quadratic programming problem (QPP). However, when the problem having a large number of input examples is considered solving the QPP is computationally expensive and therefore to reduce the computational cost an IRWLS procedure has been developed in [12]. For this, the above problem (6) is re-written as an unconstrained optimization problem of the following form [12]:

$$\text{Min } L_P(w,b) = \frac{1}{2} w^t w + C_1 \sum_{i=1}^{p} (L(e_i) + L(e_i^*)) + C_2 \sum_{i=1}^{p} (L(d_i) + L(d_i^*)), \quad (7)$$

where

$$e_i = w^t \phi(x_i) + b - y_i - \varepsilon,$$

$$e_i^* = y_i - w^t \phi(x_i) - b - \varepsilon,$$

$$d_i = w^t \phi'(x_i) - y_i' - \varepsilon_1,$$

$$d_i^* = y_i' - w^t \phi'(x_i) - \varepsilon_1 \quad \text{for } i = 1,2,\ldots,p$$

and $L(u) = \max\{u, 0\}$.

By introducing a new differentiable approximation function $L(u)$ to the non-differentiable function $\max\{u,0\}$, given by

$$L(u) = \begin{cases} 0, & u \le 0, \\ \frac{Ku^2}{2}, & 0 \le u \le \frac{1}{K}, \\ u - \frac{1}{2K}, & u \ge \frac{1}{K}, \end{cases}$$

the IRWLS iterative procedure is constructed by modifying (7) using a first-order Taylor expansion of $L(u)$ over the previous iterative solution, where $K > 0$ is taken to be a large value. For more details see [12].

3.2. Regularized least squares SVR with derivatives (RLSVRD)

In [10], Jayadeva et al. proposed the regression approximation problem for the simultaneous learning of a function and its derivatives formulated as a regularized least squares SVR problem. The main advantage of this formulation is that this will lead to solving a system of linear equations rather than solving a QPP. In this subsection we briefly discuss the RLSVRD formulation of Jayadeva et al. [10].

Assume that $\{(x_i, y_i)\}_{i=1,2,\ldots,p}$ and $\{(x_j', y_j^{(k)})\}_{j=1,2,\ldots,q}$ be two sets of input samples given where for the vector $x_i \in R^n$, the observed value of the function at x_i being $y_i \in R$ and x_j' being the vector at which $y_j^{(k)}$ = the partial derivative of the function with respect to the k-th variable x_k is prescribed. Then for the problem of determining the regression estimation function f: $R^n \to R$, consider

$$f(x) = w^t \phi(x) + b$$

and its derivatives given by

$$\frac{\partial f(x)}{\partial x_k} = w^t \phi^{(k)}(x) \quad \text{for } k = 1,2,\ldots,n,$$

where $\phi : R^n \to R^d$ $(d > n)$ maps any vector x in R^n into a higher-dimensional feature space and the unknowns $w \in R^d$, $b \in R$ are obtained as the solution of the following problem of the regularized least squares SVR [10]:

$$\text{Min } \frac{1}{2}(w^t w + b^2) + \frac{C_1}{2} \xi^t \xi + \sum_{k=1}^{n} \frac{C_2^{(k)}}{2} \eta^{(k)t} \eta^{(k)} \quad (8)$$

subject to

$$w^t \phi(x_i) + b - y_i = \xi_i \quad \text{for } i = 1,2,\ldots,p,$$

$$w^t \phi^{(k)}(x_j') - y_j^{(k)} = \eta_j^{(k)} \quad \text{for } j = 1,2,\ldots,q,$$

where for any vector $x \in R^n$ and $k = 1,2,\ldots,n$,

$$\phi(x) = (\phi_1(x),\ldots,\phi_d(x))^t \in R^d;$$

$$\phi^{(k)}(x) = \left(\frac{\partial \phi_1(x)}{\partial x_k}, \ldots, \frac{\partial \phi_d(x)}{\partial x_k} \right)^t \in R^d$$

and $C_1 > 0$, $C_2^{(k)} > 0$ are parameters.

By introducing Lagrange multipliers, the dual of the above problem (8) can be constructed which will lead to solving a system of linear equations in dual variables. In comparison to SVRD [12], it is claimed [10] that this algorithm is simple and fast as it involves solving a system of linear equations rather than solving a QPP.

4. Proposed error minimized ELM with derivatives (EM-ELMD)

In this section we extend the study of EM-ELM [4] for the problem of simultaneous learning of a function and its derivatives. Throughout in our discussion the problem of simultaneous learning of a function and its first-order derivatives is considered. However, our method of solution can be easily extended to the problem of simultaneous learning of a function and its derivatives of higher order.

Let $\{(x_i, y_i)\}_{i=1,2,\ldots,p}$ and $\{(x_j', y_j', k_j)\}_{j=1,2,\ldots,q}$ be two sets of input examples given where for the vector $x_i = (x_{i1},\ldots,x_{in})^t \in R^n$ its observed value of the function being $y_i \in R$ and for $k_j \in \{1,2,\ldots,n\}$, let $y_j' \in R$ be the observed value of the derivative of the function with respect to the k_j-th variable x_{kj} at the vector $x_j' = (x_{j1}',\ldots,x_{jn}')^t \in R^n$. Assuming G is differentiable, for a given SLFN having ℓ number of hidden nodes and for the randomly assigned values of the weight vector $a_s = (a_{s1},\ldots,a_{sn})^t \in R^n$ connecting the input layer to the sth hidden node and the bias $b_s \in R$, we formulate the problem of the simultaneous learning of a function and its derivatives as the problem of determining the output vector $w = (w_1,\ldots,w_\ell)^t \in R^\ell$ in which w_s is the weight connecting the sth hidden node to the output node satisfying the following system of equations simultaneously:

$$\sum_{s=1}^{\ell} w_s G(a_s, b_s, x_i) = y_i \quad \text{for } i = 1,2,\ldots,p$$

and

$$\sum_{s=1}^{\ell} w_s \frac{\partial G}{\partial x_{k_j}}(a_s,b_s,x'_j) = y'_j \quad \text{for } j=1,2,\ldots,q.$$

The above system of equations can be re-written in matrix form as

$$\mathcal{H}w = Y, \tag{9}$$

where

$$\mathcal{H} = \begin{bmatrix} G(a_1,b_1,x_1) & \ldots & G(a_\ell,b_\ell,x_1) \\ \cdot & \ldots & \cdot \\ G(a_1,b_1,x_p) & \ldots & G(a_\ell,b_\ell,x_p) \\ \frac{\partial G(a_1,b_1,x'_1)}{\partial x_{k_1}} & \ldots & \frac{\partial G(a_\ell,b_\ell,x'_1)}{\partial x_{k_1}} \\ \cdot & \ldots & \cdot \\ \frac{\partial G(a_1,b_1,x'_q)}{\partial x_{k_q}} & \ldots & \frac{\partial G(a_\ell,b_\ell,x'_q)}{\partial x_{k_q}} \end{bmatrix}$$

is a matrix of size $(p+q)\times\ell$ and $w=(w_1,\ldots,w_\ell)^t\in R^\ell$, $y=(y_1,\ldots,y_p)^t\in R^p$, $y'=(y'_1,\ldots,y'_q)^t\in R^q$, $Y=\begin{bmatrix} y \\ y' \end{bmatrix}=(y_1,\ldots,y_p,y'_1,\ldots,y'_q)^t\in R^{p+q}$ are column vectors.

Define the network output error as

$$E(\mathcal{H}) = \min_{w\in R^\ell} \|\mathcal{H}w - Y\|. \tag{10}$$

Following the method of proof of [6] it can be verified that when the p number of input examples for the function values and similarly the q number of input examples for the derivative values are distinct and further $\ell\le(p+q)$, the hidden layer output matrix $\mathcal{H}$ of the neural network is column full rank with probability one. In fact one can prove the following theorem.

Theorem 1. *Assume that a set of $(p+q)$ distinct input examples $\{(x_i,y_i)\}_{i=1,2,\ldots,p}\cup\{(x_j',y_j',k_j)\}_{j=1,2,\ldots,q}$ be given with $k_j\in\{1,2,\ldots,n\}$. Then for SLFNs having $(p+q)$ number of hidden nodes with activation function g: $R\to R$ that is infinitely differentiable in any interval considered, for any randomly chosen $a_s\in R^n$ and $b_s\in R$ from any intervals of R^n and R, respectively, where $s=1,2,\ldots,(p+q)$, according to any continuous probability distribution, with probability one the matrix $\mathcal{H}$ is invertible and $\|\mathcal{H}w-Y\|=0$.*

Proof. For any input vector a_s and bias value b_s randomly chosen from any intervals of R^n and R, respectively, we will prove that, according to any continuous probability distribution with probability one, the matrix $\mathcal{H}$ can be made column full rank.

Following the method of proof by Huang et al. [6], we will prove our result by contradiction. Consider the ith column of $\mathcal{H}$ given by

$$c(b_i) = (G(a_i,b_i,x_1),\ldots,G(a_i,b_i,x_p),\quad \frac{\partial G}{\partial x_{k_1}}(a_i,b_i,x'_1),\ldots,\frac{\partial G}{\partial x_{k_q}}(a_i,b_i,x'_q))^t,$$

where $b_i\in(u,v)$ and (u,v) is any interval of R.

Since the input vectors a_s are randomly chosen based on a continuous probability distribution we can assume that when $i\ne j$, $a_s^t x_i\ne a_s^t x_j$ and similarly $a_s^t x'_i\ne a_s^t x'_j$ are satisfied. Suppose that the vector $c(b_i)$ belongs to a column subspace of dimension $(p+q-1)$. Then there exists a non-zero vector $\alpha\in R^{p+q}$ which is orthogonal to this subspace, i.e.

$$\begin{aligned}\alpha^t(c(b_i)-c(u)) &= \alpha_1 G(a_i,b_i,x_1)+\ldots+\alpha_p G(a_i,b_i,x_p) \\ &\quad+\alpha_{p+1}\frac{\partial G}{\partial x_{k_1}}(a_i,b_i,x'_1)+\ldots+\alpha_{p+q}\frac{\partial G}{\partial x_{k_q}}(a_i,b_i,x'_q)-\alpha^t c(u) \\ &= 0\end{aligned}$$

Now $g(x)$ is infinitely differentiable in any interval and suppose $\alpha_j\ne 0$ implies that for any $m=1,\ldots,(p+q),\ldots$

$$\begin{aligned}&\gamma_1(G(a_i,b_i,x_1))^{(m)}+\cdots+\gamma_p(G(a_i,b_i,x_p))^{(m)} \\ &\quad+\gamma_{p+1}\left(\frac{\partial G}{\partial x_{k_1}}(a_i,b_i,x'_1)\right)^{(m)}+\cdots+\gamma_{p+q}\left(\frac{\partial G}{\partial x_{k_q}}(a_i,b_i,x'_q)\right)^{(m)}=0\end{aligned}$$

is satisfied where $(\cdot)^{(m)}$ is the mth derivative with respect to the variable b_i and $\gamma_i=\alpha_i/\alpha_j$.

This is a contradiction because we have $(p+q-1)$ free coefficients with more number of equations. Thus $c(b_i)$ does not belong to any subspace whose dimension is less than $(p+q)$.

This will imply that for $a_s\in R^n$ and $b_s\in R$ chosen from any intervals of R^n and R, respectively, according to any continuous probability distribution, with probability one the column vectors of the matrix $\mathcal{H}$ can be made full rank. □

Theorem 2. *For $\varepsilon>0$ and the activation function g: $R\to R$ be infinitely differentiable in any interval given, there exists $\tilde{\ell}\le(p+q)$ such that $\|\mathcal{H}w-Y\|<\varepsilon$ is satisfied with probability one for $(p+q)$ number of distinct samples where the weights $a_s\in R^n$ and bias $b_s\in R$ are chosen randomly from any intervals of R^n and R, respectively, according to any continuous probability distribution.*

For fixed $a_s\in R^n$ and $b_s\in R$ where $s=1,2,\ldots,\ell$, training the given SLFN is equivalent to obtaining a least squares solution w of the linear system (9)*. In fact, for $\ell<(p+q)$, w is determined to be the minimum norm least squares solution of* (9) *which can be explicitly obtained by*

$$w = \mathcal{H}^\dagger Y, \tag{11}$$

where $\mathcal{H}^\dagger$ is the Moore–Penrose generalized inverse of the matrix $\mathcal{H}$ [6,16]*. Now, using the solution $w\in R^\ell$ the regression estimation function $f(\cdot)$ is determined by* (5) *and for any input vector $x\in R^n$ its derivative is given by*

$$\frac{\partial f(x)}{\partial x_k} = \sum_{s=1}^{\ell} w_s \frac{\partial G}{\partial x_k}(a_s,b_s,x) \quad \text{for } k=1,2,\ldots,n.$$

Let us recall that the only parameter in the SLFN architecture is the number of hidden nodes. Now we discuss the method of incrementally updating the output weights when the number of hidden nodes is increased. Our approach is an extension of Feng et al. [4] to automatically determine the optimal number of hidden nodes.

Assume that initially we are given an SLFN with ℓ_1 number of hidden nodes. Then the hidden layer output matrix $\mathcal{H}_1$ of size $(p+q)\times\ell_1$ is given by

$$\mathcal{H}_1 = \begin{bmatrix} G(a_1,b_1,x_1) & \ldots & G(a_{\ell_1},b_{\ell_1},x_1) \\ \cdot & \ldots & \cdot \\ G(a_1,b_1,x_p) & \ldots & G(a_{\ell_1},b_{\ell_1},x_p) \\ \frac{\partial G(a_1,b_1,x'_1)}{\partial x_{k_1}} & \ldots & \frac{\partial G(a_{\ell_1},b_{\ell_1},x'_1)}{\partial x_{k_1}} \\ \cdot & \ldots & \cdot \\ \frac{\partial G(a_1,b_1,x'_q)}{\partial x_{k_q}} & \ldots & \frac{\partial G(a_{\ell_1},b_{\ell_1},x'_q)}{\partial x_{k_q}} \end{bmatrix} \tag{12}$$

such that $\mathcal{H}_1 w=Y$ is satisfied where $w=(w_1,\ldots,w_{\ell_1})^t\in R^{\ell_1}$ and w_s is the weight connecting the sth hidden node to the output node.

For a given SLFN having ℓ_1 number of hidden nodes with the hidden layer output matrix $\mathcal{H}_1$ and the learning accuracy ε, a small positive number, if $E(\mathcal{H}_1)<\varepsilon$ implies the learning procedure is completed. However, if $E(\mathcal{H}_1)\ge\varepsilon$ is true then we append $\delta\ell_1$ number of hidden nodes in the current SLFN and therefore the total number of hidden nodes for the new SLFN becomes $\ell_2=\ell_1+\delta\ell_1$. In this case the problem (9) can be written in matrix form as

$$\mathcal{H}_2 w = Y, \tag{13}$$

where $w=(w_1,\ldots,w_{\ell_1},w_{\ell_1+1},\ldots,w_{\ell_2})^t\in R^{\ell_2}$ and the augmented matrix $\mathcal{H}_2=[\mathcal{H}_1 \quad \delta\mathcal{H}_1]$ is such that

$$\delta\mathcal{H}_1=\begin{bmatrix} G(a_{\ell_1+1},b_{\ell_1+1},x_1) & \ldots & G(a_{\ell_2},b_{\ell_2},x_1)\\ \vdots & \ldots & \vdots\\ G(a_{\ell_1+1},b_{\ell_1+1},x_p) & \ldots & G(a_{\ell_2},b_{\ell_2},x_p)\\ \frac{\partial G(a_{\ell_1+1},b_{\ell_1+1},x'_1)}{\partial x_{k_1}} & \ldots & \frac{\partial G(a_{\ell_2},b_{\ell_2},x'_1)}{\partial x_{k_1}}\\ \vdots & \ldots & \vdots\\ \frac{\partial G(a_{\ell_1+1},b_{\ell_1+1},x'_q)}{\partial x_{k_q}} & \ldots & \frac{\partial G(a_{\ell_2},b_{\ell_2},x'_q)}{\partial x_{k_q}} \end{bmatrix}$$

Using Eq. (4) we obtain

$$\mathcal{H}_2^{\dagger}=\left(\begin{bmatrix}\mathcal{H}_1 & \delta\mathcal{H}_1\end{bmatrix}^t\begin{bmatrix}\mathcal{H}_1 & \delta\mathcal{H}_1\end{bmatrix}\right)^{-1}\begin{bmatrix}\mathcal{H}_1 & \delta\mathcal{H}_1\end{bmatrix}^t$$
$$=\begin{bmatrix}\mathcal{H}_1^t\mathcal{H}_1 & \mathcal{H}_1^t\delta\mathcal{H}_1\\ \delta\mathcal{H}_1^t\mathcal{H}_1 & \delta\mathcal{H}_1^t\delta\mathcal{H}_1\end{bmatrix}^{-1}\begin{bmatrix}\mathcal{H}_1 & \delta\mathcal{H}_1\end{bmatrix}^t=\begin{bmatrix}U\\ D\end{bmatrix}$$

Proceeding in the same manner of [4], we obtain

$$D=((I-\mathcal{H}_1\mathcal{H}_1^{\dagger})\delta\mathcal{H}_1)^{\dagger}$$

and

$$U=\mathcal{H}_1^{\dagger}(I-\delta\mathcal{H}_1D).$$

This implies one may determine the solution of the system (13) by the following:

$$w=\mathcal{H}_2^{\dagger}Y=\begin{bmatrix}U\\ D\end{bmatrix}Y,$$

where

$w=(w_1,\ldots,w_{\ell_2})^t\in R^{\ell_2}$.

Lemma 1. (Feng et al. [4]) *For a given SLFN with ℓ_1 number of hidden nodes, let $\mathcal{H}_1$ be its hidden layer output matrix. Suppose $\delta\ell_1$ number of hidden nodes is added to the SLFN and let $\mathcal{H}_2=\begin{bmatrix}\mathcal{H}_1 & \delta\mathcal{H}_1\end{bmatrix}$ be its corresponding hidden layer output matrix with $\ell_2=\ell_1+\delta\ell_1$ number of hidden nodes. Then*

$$E(\mathcal{H}_2)\le E(\mathcal{H}_1)$$

is satisfied where the network output error $E(\cdot)$ is defined by (10).

Now we summarize the incrementally updating output weights corresponding to growing hidden nodes error minimized ELM with derivatives (EM-ELMD) algorithm as below:

Input:

- $\{(x_i,y_i)\}_{i=1,2,\ldots,p}$ and $\{(x_j',y_j',k_j)\}_{j=1,2,\ldots,q}$ being the input examples, $k_j\in\{1,2,\ldots,n\}$;
- $\ell_{\max}$=the maximum number of hidden nodes;
- ε=the expected learning accuracy;
- ℓ_1=the initial number of hidden nodes.

Step 1: *Initialization*:

- For $s=1,\ldots,\ell_1$, randomly generate the weight vector $a_s=(a_{s1},\ldots,a_{sn})^t\in R^n$ connecting the sth hidden node with the input nodes and the bias b_s;
- Compute the hidden layer output matrix $\mathcal{H}_1$ given by (12) and the right hand side vector $Y=\begin{bmatrix}y\\ y'\end{bmatrix}=(y_1,\ldots,y_p,y'_1,\ldots,y'_q)^t\in R^{p+q}$ where $y=(y_1,\ldots,y_p)^t\in R^p$ and $y'=(y'_1,\ldots,y'_q)^t\in R^q$;
- Calculate the output error $E(\mathcal{H}_1)=\min_w\|\mathcal{H}_1w-Y\|=\|\mathcal{H}_1\mathcal{H}_1^{\dagger}Y-Y\|$.

Step 2: *Recursively growing the hidden nodes*:

(i) Take $k=1$
(ii) while $\ell_k<\ell_{\max}$ and $E(\mathcal{H}_k)>\varepsilon$

- Randomly add $\delta\ell_k$ number of hidden nodes to the existing SLFN and therefore the total number of hidden nodes becomes

$$\ell_{k+1}=\ell_k+\delta\ell_k$$

- Compute the hidden layer output augmented matrix $\mathcal{H}_{k+1}$ to be

$$\mathcal{H}_{k+1}=\begin{bmatrix}\mathcal{H}_k & \delta\mathcal{H}_k\end{bmatrix},\text{where}$$

$$\delta\mathcal{H}_k=\begin{bmatrix} G(a_{\ell_k+1},b_{\ell_k+1},x_1) & \ldots & G(a_{\ell_{k+1}},b_{\ell_{k+1}},x_1)\\ \vdots & \ldots & \vdots\\ G(a_{\ell_k+1},b_{\ell_k+1},x_p) & \ldots & G(a_{\ell_{k+1}},b_{\ell_{k+1}},x_p)\\ \frac{\partial G(a_{\ell_k+1},b_{\ell_k+1},x'_1)}{\partial x_{k_1}} & \ldots & \frac{\partial G(a_{\ell_{k+1}},b_{\ell_{k+1}},x'_1)}{\partial x_{k_1}}\\ \vdots & \ldots & \vdots\\ \frac{\partial G(a_{\ell_k+1},b_{\ell_k+1},x'_q)}{\partial x_{k_q}} & \ldots & \frac{\partial G(a_{\ell_{k+1}},b_{\ell_{k+1}},x'_q)}{\partial x_{k_q}} \end{bmatrix}$$

- By calculating the matrices

$$D_k=((I-\mathcal{H}_k\mathcal{H}_k^{\dagger})\delta\mathcal{H}_k)^{\dagger}\quad\text{and}\quad U_k=\mathcal{H}_k^{\dagger}(I-\delta\mathcal{H}_kD_k)$$

obtain the solution of the following system

$$\mathcal{H}_{k+1}w=Y$$

as

$$w=\mathcal{H}_{k+1}^{\dagger}Y=\begin{bmatrix}U_k\\ D_k\end{bmatrix}Y$$

- Compute

$$E(\mathcal{H}_{k+1})=\|\mathcal{H}_{k+1}\mathcal{H}_{k+1}^{\dagger}Y-Y\|$$

- Replace k by $k+1$

end while

Remark 3. In general, the new hidden nodes added at each iteration can be either a single node or a group of nodes. This implies that the number of new hidden nodes added at each iteration need not be the same, i.e. $\delta\ell_k\neq\delta\ell_{k+1}$ may be assumed for some positive integer k.

Remark 4. Suppose new hidden nodes are added one by one in each iteration, i.e. $\delta\ell_1=\delta\ell_2=\cdots=1$. Then for any positive integer k, we have

$$\delta\mathcal{H}_k=(G(a_{\ell_k+1},b_{\ell_k+1},x_1),\ldots,G(a_{\ell_k+1},b_{\ell_k+1},x_p),$$
$$\frac{\partial G}{\partial x_{k_1}}(a_{\ell_k+1},b_{\ell_k+1},x'_1),\ldots,\frac{\partial G}{\partial x_{k_q}}(a_{\ell_k+1},b_{\ell_k+1},x'_q))^t$$

becomes a vector and accordingly the matrices D_k and U_k can be written as

$$D_k=\frac{\delta\mathcal{H}_k^t(I-\mathcal{H}_k\mathcal{H}_k^{\dagger})^t}{\delta\mathcal{H}_k^t(I-\mathcal{H}_k\mathcal{H}_k^{\dagger})\delta\mathcal{H}_k}$$

and

$$U_k=\mathcal{H}_k^{\dagger}(I-\delta\mathcal{H}_kD_k).$$

Following the same arguments of [4] the convergence of the EM-ELMD can be easily verified. In fact we have the following theorem.

Theorem 3. *For the given set of p number of distinct input examples for the function values $\{(x_i,y_i)\}_{i=1,2,\ldots,p}$, the set of q number of distinct inputs for the derivative values $\{(x_j',y_j',k_j)\}_{j=1,2,\ldots,q}$ in which $k_j\in\{1,2,\ldots,n\}$ and the expected learning accuracy $\varepsilon>0$, there exists*

a positive integer k such that

$E(\mathcal{H}_k) = \min_w \|\mathcal{H}_k w - Y\| < \varepsilon.$

5. Experimental results

In order to demonstrate the effectiveness of the proposed algorithm the performance of EM-ELMD is compared with the other methods RLSVRD [10] and SVRD [12] for the problem of simultaneous learning of a function and its derivatives for the function defined by

$$f(x_1,x_2) = x_1{}^2 x_2 + \sin(x_2) \quad \text{for } -5 \le x_1, x_2 \le 5 \tag{14}$$

and the eight functions considered in [12] sampled over a two-dimensional input space. The analytical expressions of these eight functions are given in Table 1. Experiments are carried out in the MATLAB environment. The Root Mean Square Error (RMSE) given by the following formula:

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \tilde{y}_i)^2}$$

is used to evaluate the accuracy of the results obtained, where N is the number of test samples, y_i and $\tilde{y}_i$ are the observed and its corresponding predicted values, respectively.

In all our experiments, the number of hidden nodes is initially assumed to be 20 and the sigmoid activation function $g(\cdot)$ given by

$$g(a^t x + b) = 1/(1+\exp(-(a^t x + b)))$$

is considered. At each iteration of our proposed EM-ELMD algorithm, new hidden nodes are added either one by one or two by two or five by five. The last two cases are considered as examples of hidden nodes growing group by group. The maximum number of hidden nodes is taken as 200. For the function defined by (14) with growing hidden nodes $\delta\ell = 1$, the learning accuracy ε is taken as 10^{-5} and for the remaining eight functions defined in Table 1 it is set as: 10^{-8}, 10^{-5} and 10^{-3} for the case of growing hidden nodes one by one, two by two and five by five, respectively. The average results are obtained over 10 trials.

By considering the same training and test samples the performance of EM-ELMD is compared with that of RLSVRD and SVRD. In RLSVRD and SVRD, the Gaussian kernel is used and is defined

Table 1
Two-dimensional test functions ($y = f(x_1,x_2)$) used.

Name	Function	Domain
Function1	$y = \sin(x_1 x_2)$	$[-2,2]$
Function2	$y = \exp(x_1 \sin(\pi x_2))$	$[-1,1]$
Function3	$y = \dfrac{40\exp((8((x_1-0.5)^2+(x_2-0.5)^2)))}{\exp(8((x_1-0.2)^2+(x_2-0.7)^2))+\exp(8((x_1-0.7)^2+(x_2-0.2)^2))}$	$[0,1]$
Function4	$y = \dfrac{1+\sin(2x_1+3x_2)}{3.5+\sin(x_1-x_2)}$	$[-2,2]$
Function5	$y = 42.659(0.1 + x_1(0.05 + x_1{}^4 - 10x_1{}^2x_2{}^2 + 5x_2{}^4))$	$[-0.5,0.5]$
Function6	$y = 1.3356(\exp(3(x_2-0.5))\sin(4\pi(x_2-0.9)^2) + 1.5(1-x_1) + \exp(2x_1-1)\sin(3\pi(x_1-0.6)^2))$	$[0,1]$
Function7	$y = 1.9(1.35 + \exp(x_1)\sin(13(x_1-0.6)^2) + \exp(3(x_2-0.5))\sin(4\pi(x_2-0.9)^2))$	$[0,1]$
Function8	$y = \sin(2\pi\sqrt{x_1^2+x_2^2})$	$[-1,1]$

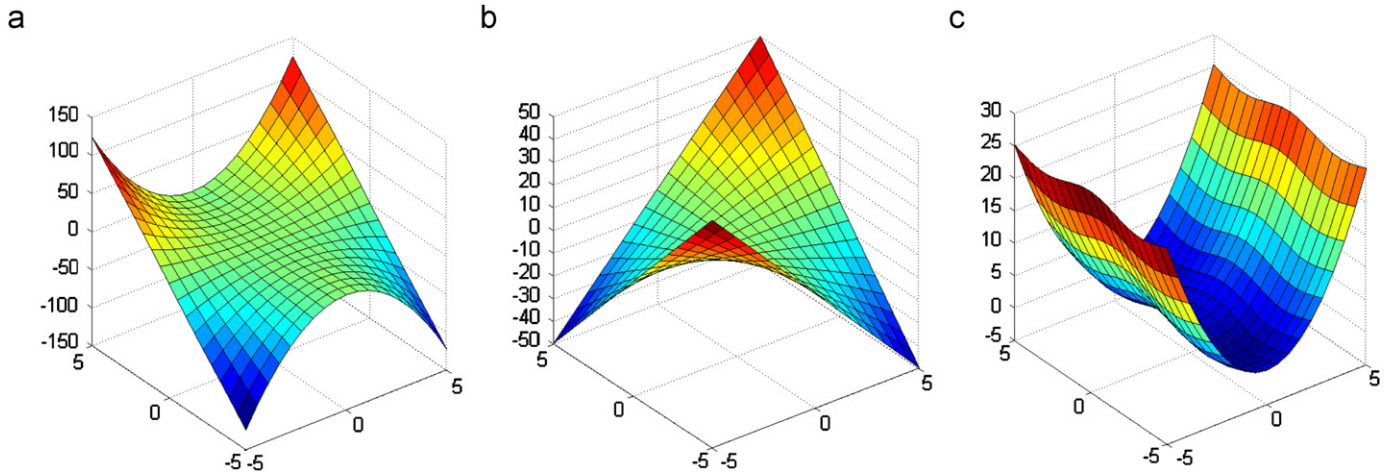

Fig. 1. Plots of the function $f(x_1,x_2) = x_1{}^2 x_2 + \sin(x_2)$ defined in the region $-5 \le x_1, x_2 \le 5$ and its derivatives: (a) $f(x_1,x_2)$, (b) $\partial f(x_1,x_2)/\partial x_1$ and (c) $\partial f(x_1,x_2)/\partial x_2$.

by: For $x_1, x_2 \in R^n$,

$$k(x_1,x_2) = \exp\left(-\frac{1}{2\sigma^2}\|x_1 - x_2\|^2\right)$$

where $\sigma > 0$ is a parameter. The best prediction performance on the training sets were obtained by varying the regularization and the kernel parameters from the sets $\{10^{-5},10^{-4},\ldots,10^{5}\}$ and $\{2^{-10},2^{-9},\ldots,2^{10}\}$, respectively, using 10-fold cross-validation.

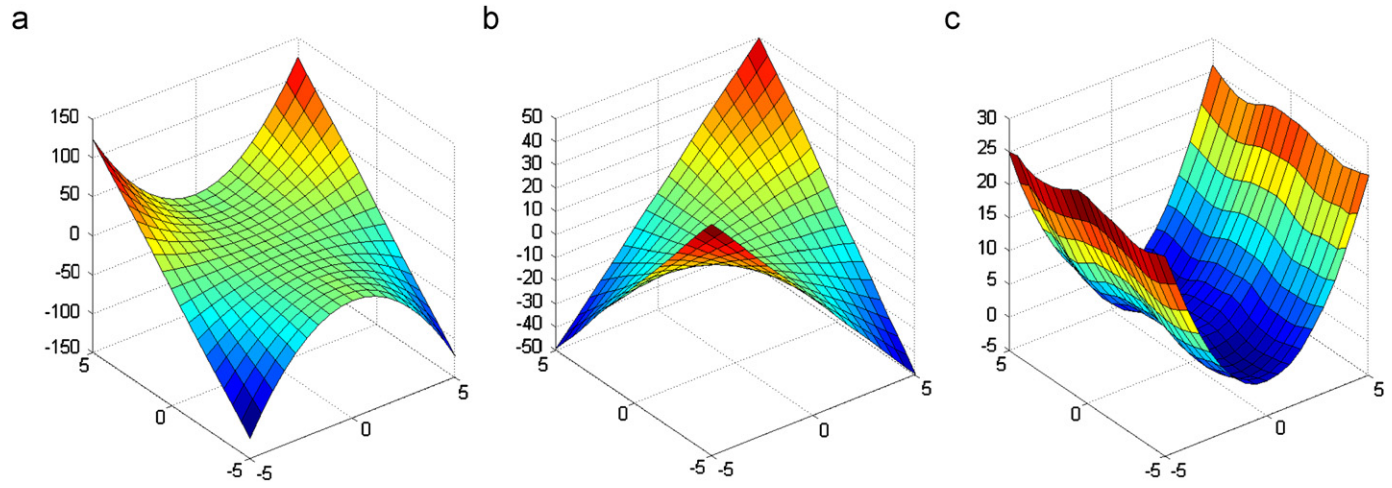

Fig. 2. Plots of the proposed EM-ELMD regressor for the function $f(x_1,x_2)$ defined in Fig. 1 and its derivatives with the number of growing hidden nodes $\delta\ell = 1$, the accuracy used to terminate adding the hidden nodes $\varepsilon = 10^{-5}$ and the average number of hidden nodes = 81.6: (a) $f(x_1,x_2)$, (b) $\partial f(x_1,x_2)/\partial x_1$ and (c) $\partial f(x_1,x_2)/\partial x_2$.

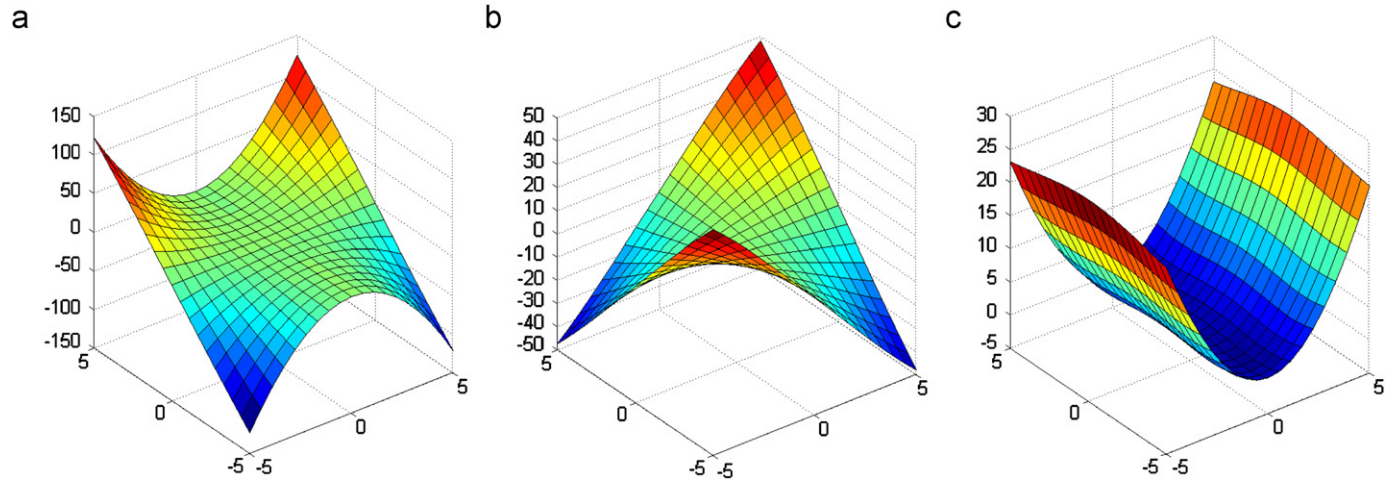

Fig. 3. Plots of the RLSVRD [10] regressor for the function $f(x_1,x_2)$ defined in Fig. 1 and its derivatives with the regularization parameter $C = 10^4$ and the Gaussian kernel parameter $\sigma = 2^3$: (a) $f(x_1,x_2)$, (b) $\partial f(x_1,x_2)/\partial x_1$ and (c) $\partial f(x_1,x_2)/\partial x_2$.

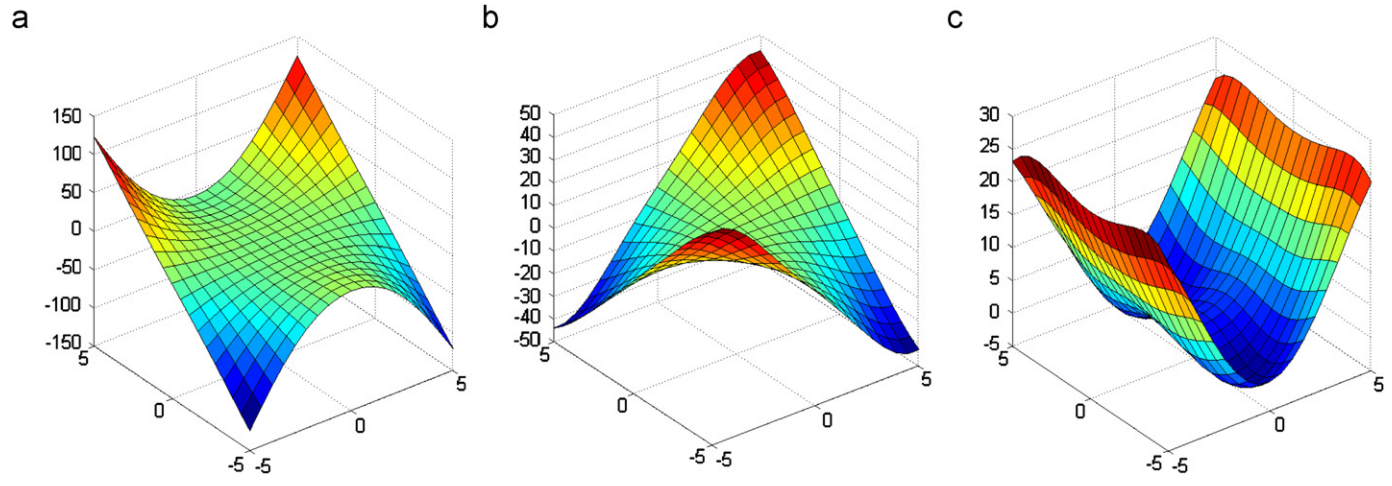

Fig. 4. Plots of the SVRD [12] regressor for the function $f(x_1,x_2)$ defined in Fig.1 and its derivatives with the regularization parameter $C = 10^5$ and the Gaussian kernel parameter $\sigma = 2^2$: (a) $f(x_1,x_2)$, (b) $\partial f(x_1,x_2)/\partial x_1$ and (c) $\partial f(x_1,x_2)/\partial x_2$.

Table 2
Comparison of training time (s) of the proposed EM-ELMD, RLSVRD [10] and SVRD [2]. ($\delta\ell$=number of growing hidden nodes; ε=the accuracy used to terminate adding of hidden nodes).

Functions	Average number of hidden nodes used for training			Training time				
	EM-ELMD			EM-ELMD			RLSVRD	SVRD
	$\delta\ell=1$, $\varepsilon=10^{-8}$	$\delta\ell=2$, $\varepsilon=10^{-5}$	$\delta\ell=5$, $\varepsilon=10^{-3}$	$\delta\ell=1$, $\varepsilon=10^{-8}$	$\delta\ell=2$, $\varepsilon=10^{-5}$	$\delta\ell=5$, $\varepsilon=10^{-3}$		
1	111.6	125.4	98	15.3037	11.6065	2.3400	101.135	2450.32
2	108.6	132.6	98	14.0869	11.6845	2.1372	92.3058	2046.02
3	119	191.8	122.5	15.7873	35.4902	5.4756	89.4354	1629.15
4	102.9	116	88	13.3225	8.72046	1.9812	97.1886	2588.4
5	119	89	55	15.7717	4.1652	0.468	90.8082	1755.79
6	119	124.8	82.5	15.6625	11.3569	1.3728	90.2778	1839.41
7	119	185.2	117	15.7561	33.7274	3.7908	89.7942	1847.08
8	119	199.4	114.5	16.4893	35.2562	3.2448	90.2154	1744.06

Table 3
Performance comparison using RMSE for functions listed in Table 1 and their derivatives. ($\delta\ell$=number of growing hidden nodes; ε=the accuracy used to terminate adding of hidden nodes).

(a) RMSE for function estimation using the proposed EM-ELMD, RLSVRD [10] and SVRD [12]

Functions	$f(x_1,x_2)$				
	EM-ELMD			RLSVRD	SVRD
	$\delta\ell=1$, $\varepsilon=10^{-8}$	$\delta\ell=2$, $\varepsilon=10^{-5}$	$\delta\ell=5$, $\varepsilon=10^{-3}$		
Function 1	0.001871	0.001369	0.002494	0.00712636	0.00725263
Function 2	0.006454	0.004746	0.007778	0.00653823	0.00596265
Function 3	0.126554	0.120608	0.128681	0.0318494	0.0444544
Function 4	0.002229	0.001047	0.002443	0.00401758	0.0032606
Function 5	0.000473	0.000125	0.000322	0.0255504	0.0287183
Function 6	0.070753	0.070975	0.072052	0.0185586	0.0179874
Function 7	0.044296	0.03961	0.046435	0.0356726	0.0174734
Function 8	0.135575	0.175677	0.13337	0.0944004	0.0630885

(b) RMSE for the estimation of $(\partial f/\partial x_1)$ using the proposed EM-ELMD, RLSVRD [10] and SVRD [12]

Functions	$\left(\frac{\partial f}{\partial x_1}\right)$				
	EM-ELMD			RLSVRD	SVRD
	$\delta\ell=1$, $\varepsilon=10^{-8}$	$\delta\ell=2$, $\varepsilon=10^{-5}$	$\delta\ell=5$, $\varepsilon=10^{-3}$		
Function 1	0.007342	0.00566	0.009514	0.0151841	0.0160762
Function 2	0.024712	0.01779	0.024602	0.0287481	0.0259537
Function 3	1.61877	1.57258	1.61099	0.383636	0.553336
Function 4	0.006822	0.003221	0.006091	0.00910648	0.0077035
Function 5	8.94E−05	8.60E−05	0.000723	0.211173	0.440658
Function 6	0.065724	0.076386	0.086682	0.192414	0.186178
Function 7	0.710618	0.663461	0.774763	0.309059	0.251864
Function 8	0.826929	0.87029	0.833759	0.56477	0.524524

(c) RMSE for the estimation of $(\partial f/\partial x_2)$ using the proposed EM-ELMD, RLSVRD [10] and SVRD [12]

Functions	$\left(\frac{\partial f}{\partial x_2}\right)$				
	EM-ELMD			RLSVRD	SVRD
	$\delta\ell=1$, $\varepsilon=10^{-8}$	$\delta\ell=2$, $\varepsilon=10^{-5}$	$\delta\ell=5$, $\varepsilon=10^{-3}$		
Function 1	0.007367	0.005241	0.009978	0.0157907	0.0166526
Function 2	0.055271	0.03944	0.070648	0.0459112	0.0442606
Function 3	1.56908	1.50232	1.62731	0.379402	0.541475
Function 4	0.009929	0.004739	0.01155	0.0117162	0.00979171
Function 5	8.00E−05	9.90E−05	0.000523	0.140911	0.288248
Function 6	1.23819	1.24099	1.25918	0.209238	0.264697
Function 7	0.526276	0.41851	0.535784	0.411482	0.176646
Function 8	0.864982	1.13601	0.851754	0.565837	0.523069

In all the experiments, the training set is obtained by prescribing the value of the function and its derivatives at the 11×11 uniform grid points of the interval of definition of the function. Further the test data consisting of 19×19 uniform grid points of the interval of definition is considered. The actual and the predicted values for the function defined by (14) and its derivatives obtained using EM-ELMD, RLSVRD, SVRD along with the values of the parameters are shown in Figs. 1, 2, 3 and 4, respectively. For the eight functions listed in Table 1, the average number of hidden nodes used by EM-ELMD and the time taken for training (in seconds) by all the methods considered are reported in Table 2. For the EM-ELMD the training time includes the 10 trials and for other two methods the time taken for cross-validation is included. The comparative RMSE results on the test samples for the functions listed in Table 1, are summarized in Table 3. From the experiments it is observed that solutions with accuracy similar to or better than other methods [10,12] are obtained by the proposed method at much faster learning speed. In real applications one may not consider the termination condition to be $|E(\mathcal{H}_{k+1})| < \varepsilon$. In fact in all our experiments adding of new hidden nodes is terminated whenever $|E(\mathcal{H}_{k+1}) - E(\mathcal{H}_k)| < \varepsilon$. Our results clearly show that EM-ELMD formulation can obtain solutions with accuracy comparable to the formulations of [10,12] while being much faster to solve.

6. Conclusions

A new learning algorithm for the problem of simultaneous learning of a function and its derivatives is proposed as an extension of the study of error minimized extreme learning machine for single hidden layer feedforward neural networks. In this approach the number of hidden nodes is automatically determined by adding new hidden nodes to the network either one by one or group by group. The proposed method is fast and produces similar or better generalization error in comparison to other popular methods. Future work includes the application of the proposed method for solving ordinary and partial differential equations.

Acknowledgments

The authors would like to thank Prof. S. Chandra and Prof. Lazaro for providing the codes of RLSVRD [10] and SVRD [12], respectively. Also they are extremely thankful to the referees for their comments and suggestions which greatly helped in improving the presentation of the paper.

References

[1] L.P. Aarts, P.V.D. Veer, Neural network method for solving partial differential equations, Neural Processing Letters 14 (3) (2001) 261–271.
[2] P.M.T. Broersen, How to select polynomial models with an accurate derivative, IEEE Transactions on Instrumentation and Measurement 49 (5) (2000) 910–914.
[3] A. Demiriz, K. Bennett, C. Breneman, M. Embrechts, Support vector machine regression in chemometrics, Computing Science & Statistics (2001).
[4] G. Feng, G.-B. Huang, Q. Lin, R. Gay, Error minimized extreme learning machine with growth of hidden nodes and incremental learning, IEEE Transactions on Neural Networks 20 (8) (2009) 1352–1357.
[5] G.-B. Huang, L. Chen, C.-K. Siew, Universal approximation using incremental constructive feedforward networks with random hidden nodes, IEEE Transactions on Neural Networks 17 (4) (2006) 879–892.
[6] G.-B. Huang, Q.-Y. Zhu, C.-K. Siew, Extreme learning machine: Theory and applications, Neurocomputing 70 (2006) 489–501.
[7] G.-B. Huang, L. Chen, Convex incremental extreme learning machine, Neurocomputing 70 (2007) 3056–3062.
[8] G.-B. Huang, L. Chen, Enhanced random search based incremental extreme learning machine, Neurocomputing 71 (2008) 3460–3468.
[9] G.-B. Huang, X. Ding, H. Zhou, Optimization method based extreme learning machine for classification, Neurocomputing 74 (1–3) (2010) 155–163.
[10] Jayadeva, R. Khemchandani, S. Chandra, Regularized least squares support vector regression for the simultaneous learning of a function and its derivatives, Information Sciences 178 (2008) 3402–3414.
[11] I.E. Lagaris, A. Likas, D.I. Fotiadis, Artificial neural networks for solving ordinary and partial differential equations, IEEE Transactions on Neural Networks 9 (5) (1998) 987–1000.
[12] M. Lazaro, I. Santamaria, F. Perez-Cruz, A. Artes-Rodriguez, Support vector regression for the simultaneous learning of a multivariate function and its derivatives, Neurocomputing 69 (2005) 42–61.
[13] X. Li, Simultaneous approximations of multivariate functions and their derivatives by neural networks with one hidden layer, Neurocomputing 12 (1996) 327–343.
[14] S. Mukherjee, E. Osuna, F. Girosi, Nonlinear prediction of chaotic time series using support vector machines, in: NNSP'97: Neural Networks for Signal Processing VII: Proceedings of the IEEE Signal Processing Society Workshop, Amelia Island, FL, USA, 1997, pp. 511–520.
[15] T. Nguyen-Thien, T. Tran-Cong, Approximation of functions and their derivatives: a neural network implementation with applications, Neurocomputing 23 (1999) 687–704.
[16] C.R. Rao, S.K. Mitra, Generalized inverse of matrices and its applications, John Wiley, New York, 1971.
[17] L. Rutkowski, A general approach for nonparametric fitting of functions and their derivatives with applications to linear circuits identification, IEEE Transactions on Circuits and Systems 33 (8) (1986) 812–818.
[18] I. Santamaria, J. Via, C.C. Gaudes, Robust blind identification of SIMO channels: a support vector regression approach, in: Proceedings of the 24th IEEE International Conference on Acoustics, Speech and Signal Processing, vol. 5, 2004, pp. 673–676.
[19] R. Singh, S. Balasundaram, Application of extreme learning machine for time series analysis, International Journal of Intelligent Technology 2 (2007) 256–262.
[20] F.E.H. Tay, L.J. Cao, Application of support vector machines in financial time series with forecasting, Omega 29 (4) (2001) 309–317.
[21] V.N. Vapnik, The Nature of Statistical Learning Theory, 2nd ed., Springer, New York, 2000.

S. Balasundaram is a Professor of Jawaharlal Nehru University, India. He received his Ph.D. from Indian Institute of Technology, New Delhi in 1983. From 1983 to 85 he was a post doctoral fellow in INRIA, Rocquencourt, France. He joined as an Assistant Professor in Jawaharlal Nehru University in 1986. During 2003–05 he was a visiting faculty in Eastern Mediterranean University, North Cyprus. His main research includes support vector machine and extreme learning machine methods for classification and regression problems, fuzzy regression and applied optimization.

Kapil is a Ph.D. student of Jawaharlal Nehru University, India. He received his Masters of Computer Applications and M.Tech. degrees from Jawaharlal Nehru University in 2005 and 2010, respectively. His research interests include support vector machine and other data mining techniques.