Int. J. Mach. Learn. & Cyber. (2016) 7:707–728
DOI 10.1007/s13042-014-0283-8

ORIGINAL ARTICLE

On optimization based extreme learning machine in primal for regression and classification by functional iterative method

S. Balasundaram · Deepak Gupta

Received: 17 November 2013 / Accepted: 26 June 2014 / Published online: 1 August 2014

Abstract In this paper, the recently proposed extreme learning machine in the aspect of optimization method by Huang et al. (Neurocomputing, 74: 155–163, 2010) has been considered in its primal form whose solution is obtained by solving an absolute value equation problem by a simple, functional iterative algorithm. It has been proved under sufficient conditions that the algorithm converges linearly. The pseudo codes of the algorithm for regression and classification are given and they can be easily implemented in MATLAB. Experiments were performed on a number of real-world datasets using additive and radial basis function hidden nodes. Similar or better generalization performance of the proposed method in comparison to support vector machine (SVM), extreme learning machine (ELM), optimally pruned extreme learning machine (OP-ELM) and optimization based extreme learning machine (OB-ELM) methods with faster learning speed than SVM and OB-ELM demonstrates its effectiveness and usefulness.

Keywords Extreme learning machine · Single hidden layer feedforward neural networks · Functional iterative method · Support vector machine

S. Balasundaram (✉) · D. Gupta
School of Computer and Systems Sciences, Jawaharlal Nehru University, New Delhi 110067, India
e-mail: bala_jnu@hotmail.com; balajnu@gmail.com

D. Gupta
e-mail: deepakjnu85@gmail.com

1 Introduction

Recently, a new learning algorithm for single hidden layer feedforward neural networks (SLFNs) architecture called extreme learning machine (ELM) method has been proposed in [21] to overcome many of the problems of traditional feedforward neural network learning algorithms such as the presence of local minima, imprecise learning rate, over fitting and slow rate of convergence. Once the input weights and hidden layer biases have been chosen randomly, ELM determines the unknown output weight vector of the network having the smallest norm by solving a system of linear equations. ELM is a simple unified algorithm for regression, binary and multiclass problems and it has been successfully tested on benchmark problems of practical importance. It was initially proposed for SLFNs and later extended to "generalized" SLFNs which may not neuron alike [15, 16]. The essence of ELM is that there is no need of tuning the hidden layer of SLFNs. The growing popularity of ELM [3, 4, 10, 14, 22, 27, 28, 33, 34, 38, 39, 42] is because of its better generalization performance with much faster learning speed in comparison to traditional computational intelligence techniques [19].

The main problem with ELM is the stochastic nature of the hidden layer output matrix which in practice may lower its learning accuracy [6]. Further, it was observed that to achieve an acceptable level of performance a large number of hidden nodes might be required [23, 44] which implies increase in problem size and therefore computational cost. This suggests to look for compact networks having the ability to achieve good generalization performance [23, 41, 44]. The other issue with ELM is that the number of hidden nodes is a parameter and is often chosen by trial and error method. Two heuristic approaches namely constructive [12, 15–17] and pruning methods [26] have been proposed in the literature to address this problem. In fact, the optimally pruned ELM (OP-ELM) method proposed in [26] selects the hidden neurons by applying 1-norm for the outputs and then computes their weights using the classical least squares method. For an interesting study of ELM in 1-norm resulting in a sparse model representation, see [1].

By applying ELM kernels in the SVM formulation [7, 37] instead of support vector machine (SVM) kernels, it was shown in [13, 24] that better generalization can be achieved. In [18], optimization based ELMs (OB-ELMs) for classification were proposed as an extension to support vector networks in which as in SVM the training error and the norm of the output weight vector were minimized. Further it was observed that the proposed method tends to achieve better generalization performance than SVM and is less sensitive to the input parameters. For an interesting study on an extension of ELM to least squares SVM (LS-SVM) and proximal SVM (PSVM) as a learning algorithm providing a unified solution, see the work of Huang et al. [20]. Finally, for an excellent survey on ELM, we refer the reader to [19].

Motivated by the works of [18, 20], an enhanced optimization based ELM is proposed in its primal whose solution is obtained by solving an absolute value equation using a simple, functional iterative method. Our formulation directly finds the approximate optimal solution in the primal space rather than finding in its dual space. The effectiveness of the proposed method for regression and binary classification problems is demonstrated by performing numerical experiments on a number of real-world datasets and comparing their results with SVM, ELM, OP-ELM and OB-ELM.

The paper is organized as follows. Section 2 dwells briefly SVM for regression and classification. In Sect. 3, a short introduction to ELM is given. The OB-ELMs for regression and classification are introduced in Sect. 4. The proposed optimization based ELM formulation whose solution is obtained by applying a functional iterative method, is described in Sect. 5. Numerical experiments were performed on a number of benchmark real-world datasets and their results with additive and radial basis function (RBF) hidden nodes are compared with that of SVM, ELM, OP-ELM and OB-ELM in Sect. 6 and we conclude our work in Sect. 7.

Throughout in this paper, all vectors are taken as column vectors. For any two vectors $\mathbf{x}$,$\mathbf{y}$ in R^n, we denote their inner product by $\mathbf{x}^t\mathbf{y}$ where $\mathbf{x}^t$ is the transpose of the vector $\mathbf{x}$. The 2-norm of a vector $\mathbf{x}$ will be denoted by $\|\mathbf{x}\|$. For any vector $\mathbf{x} = (x_1, \ldots, x_n)^t \in R^n$, let $\mathbf{x}_+$ be a vector in R^n defined by setting all the negative components of $\mathbf{x}$ to zero and further let $|\mathbf{x}| = (|x_1|, \ldots, |x_n|)^t \in R^n$. The zero vector and the column vector of ones of dimension m are denoted by $\mathbf{0}$ and $\mathbf{e}$ respectively. For any real matrix $\mathcal{H} \in R^{m\times\ell}$, its transpose is denoted by $\mathcal{H}^t$. For any symmetric matrix $\mathcal{A}$, let its maximum eigenvalue be $\lambda_{\max}(\mathcal{A})$. The identity matrix of appropriate size is denoted by I. Further, *diag*($\mathbf{x}$) means a diagonal matrix of order n whose diagonal elements being the components of the vector $\mathbf{x} \in R^n$. If f is a real valued function of the variable $\mathbf{x} = (x_1, \ldots, x_n)^t \in R^n$, we denote its gradient by $\nabla f = (\partial f/\partial x_1, \ldots, \partial f/\partial x_n))^t$.

2 Support vector machines for regression and classification

In this section, we briefly describe the standard support vector machines for regression and binary classification problems.

2.1 Regression formulation

Assume that a set of training examples $\{(\mathbf{x}_i, y_i)\}_{i=1,\ldots,m}$ is given such that for each input example $\mathbf{x}_i = (x_{i1}, \ldots, x_{in})^t \in R^n$, let its corresponding observed value be $y_i \in R$. By mapping the input examples $\mathbf{x}_i \in R^n$ into a higher dimensional feature space via a nonlinear map $\varphi(.)$, the linear regression model $f(.)$ of the following form is fitted in the feature space

$$f(\mathbf{x}) = \mathbf{w}^t\varphi(\mathbf{x}) + b. \tag{1}$$

The goal of nonlinear SVR is in determining the unknowns $\mathbf{w}$ and b of (1) as the solution of the constrained minimization problem [7, 37]

$$\min_{\mathbf{w},b,\xi_1,\xi_2} \frac{1}{2}\mathbf{w}^t\mathbf{w} + C(\mathbf{e}^t\xi_1 + \mathbf{e}^t\xi_2)$$

subject to

$$\begin{aligned} y_i - \mathbf{w}^t\varphi(\mathbf{x}_i) - b &\le \varepsilon + \xi_{1i}, \\ \mathbf{w}^t\varphi(\mathbf{x}_i) + b - y_i &\le \varepsilon + \xi_{2i}, \\ \xi_{1i} \ge 0 \text{ and } \xi_{2i} \ge 0 &\quad \text{for } i = 1, 2, \ldots, m, \end{aligned} \tag{2}$$

where $C > 0$, $\varepsilon > 0$ are input parameters and $\xi_1 = (\xi_{11}, \ldots, \xi_{1m})^t, \xi_2 = (\xi_{21}, \ldots, \xi_{2m})^t \in R^m$ are vectors of slack variables.

Instead of solving the primal problem (2), by introducing Lagrange multipliers $\boldsymbol{\alpha}_1 = (\alpha_{11}, \ldots, \alpha_{1m})^t$, $\boldsymbol{\alpha}_2 = (\alpha_{21}, \ldots, \alpha_{2m})^t \in R^m$ and applying the kernel trick, its dual problem of the following form is solved [7, 37]

$$\min_{\boldsymbol{\alpha}_1,\boldsymbol{\alpha}_2} \frac{1}{2}\sum_{i,j=1}^{m}(\alpha_{1i} - \alpha_{2i})k(\mathbf{x}_i, \mathbf{x}_j)(\alpha_{1j} - \alpha_{2j}) + \varepsilon\sum_{i=1}^{m}(\alpha_{1i} + \alpha_{2i}) - \sum_{i=1}^{m} y_i(\alpha_{1i} - \alpha_{2i})$$

subject to

$$\sum_{i=1}^{m}(\alpha_{1i}-\alpha_{2i})=0 \text{ and } \mathbf{0}\le\boldsymbol{\alpha}_1,\boldsymbol{\alpha}_2\le C\,\mathbf{e},$$

where $k(\mathbf{x}_i, \mathbf{x}_j)$ is a kernel function that replaces the term $\varphi(\mathbf{x}_i)^t\varphi(\mathbf{x}_j)$.

Finally, for any $\mathbf{x}\in R^n$ its prediction by the fitted regression function $f(.)$ is given by [7, 37]

$$f(\mathbf{x})=\sum_{i=1}^{m}(\alpha_{1i}-\alpha_{2i}))k(\mathbf{x},\mathbf{x}_i)+b.$$

2.2 Classification formulation

Let $\{(\mathbf{x}_i,y_i)\}_{i=1,\ldots,m}$ be a set of training examples given in which for each input point $\mathbf{x}_i=(x_{i1},\ldots,x_{in})^t\in R^n$, let $y_i=\pm1$ be its corresponding class label. It is well known that SVM constructs a hyperplane of the form $\mathbf{w}^t\varphi(\mathbf{x})+b$, by maximizing the margin between the input points of the positive and negative classes in the feature space and at the same time minimizing the sum of the training errors. In fact, the unknowns $\mathbf{w}$ and b will be obtained as the solution of the following constrained minimization problem [7, 37]

$$\min_{\mathbf{w},b,\xi}\frac{1}{2}\mathbf{w}^t\mathbf{w}+C\mathbf{e}^t\boldsymbol{\xi}$$

subject to

$$\begin{aligned}&y_i(\mathbf{w}^t\varphi(\mathbf{x}_i)+b)\ge1-\xi_i,\\&\xi_i\ge0\quad\text{for } i=1,2,\ldots,m,\end{aligned}\tag{3}$$

where $C>0$ is the regularization parameter and $\boldsymbol{\xi}=(\xi_1,\ldots,\xi_m)^t$ in R^m is a vector of slack variables. Its dual can be written of the form [7, 37]

$$\min_{\boldsymbol{\alpha}\in R^m}\frac{1}{2}\sum_{i,j=1}^{m}\alpha_i\alpha_jy_iy_jk(\mathbf{x}_i,\mathbf{x}_j)-\sum_{i=1}^{m}\alpha_i$$

subject to

$$\sum_{i=1}^{m}\alpha_iy_i=0 \text{ and } \mathbf{0}\le\boldsymbol{\alpha}\le C\,\mathbf{e},$$

where $k(.,.)$ is a kernel function and $\boldsymbol{\alpha}=(\alpha_1,\ldots,\alpha_m)^t\in R^m$ is the Lagrange multiplier vector. In this case, the decision function $f(.)$ will become

$$f(\mathbf{x})=\text{sign}\Big(\sum_{j=1}^{m}\alpha_jy_jk(\mathbf{x},\mathbf{x}_j)+b\Big) \text{ for } \mathbf{x}\in R^n.$$

3 Extreme learning machine method

ELM is a unified learning approach for regression and classification problems proposed for training SLFNs. It is based on least squares solution having the ability to achieve comparable generalization performance at much faster learning speed in accordance with the traditional SVM.

Assume that a set of training examples $\{(\mathbf{x}_i,y_i)\}_{i=1,\ldots,m}$ is given such that for each input example $\mathbf{x}_i=(x_{i1},\ldots x_{in})^t\in R^n$, let $y_i\in R$ be its corresponding target value. For the randomly assigned values for the learning parameters $\mathbf{a}_s=(a_{s1},\ldots,a_{sn})^t\in R^n$ and $b_s\in R$ of the hidden nodes, ELM determines its output function $f(.)$ such that

$$f(\mathbf{x}_i)=\sum_{s=1}^{\ell}w_sG(\mathbf{a}_s,b_s,\mathbf{x}_i)=y_i\quad\text{for } i=1,\ldots,m,\tag{4}$$

where the hidden layer output function $G(\mathbf{a},b,\mathbf{x})$ is a nonlinear piecewise continuous function satisfying the conditions of the universal approximation capability theorems [17] and $\mathbf{w}=(w_1,\ldots,w_\ell)^t\in R^\ell$ is the unknown output weight vector connecting the hidden layer of ℓ nodes with the output node. The above condition (4) can be written in matrix form as

$$\mathcal{H}\mathbf{w}=\mathbf{y},\tag{5}$$

where

$$\mathcal{H}=\begin{bmatrix}G(\mathbf{a}_1,b_1,\mathbf{x}_1)&\cdots&G(\mathbf{a}_\ell,b_\ell,\mathbf{x}_1)\\ .&\cdots&.\\ G(\mathbf{a}_1,b_1,\mathbf{x}_m)&\cdots&G(\mathbf{a}_\ell,b_\ell,\mathbf{x}_m)\end{bmatrix}_{m\times\ell}$$

is the hidden layer output matrix of the network and $\mathbf{y}=(y_1,\ldots,y_m)^t\in R^m$ is the vector of observed values.

Some of the well-known hidden layer output functions are

a) Sigmoid function : $G(\mathbf{a},b,\mathbf{x})=\frac{1}{1+\exp(-(\mathbf{a}^t\mathbf{x}+b))}$,

b) Multiquadric function:

$$G(\mathbf{a},b,\mathbf{x})=\left(\|\mathbf{x}-\mathbf{a}\|^2+b^2\right)^{1/2},$$

c) Gaussian function: $G(\mathbf{a},b,\mathbf{x})=\exp\left(-b\|\mathbf{x}-\mathbf{a}\|^2\right)$.

Clearly, for the randomly assigned values for the parameters $\mathbf{a}_s\in R^n, b_s\in R$ and the predefined hidden layer output function $G(\mathbf{a},b,\mathbf{x})$, training the SLFN is equivalent to obtaining a least squares solution $\mathbf{w}\in R^\ell$ of the rectangular linear system (5). In fact, $\mathbf{w}\in R^\ell$ can be explicitly obtained as the smallest norm least squares solution of (5) [21]: $\mathbf{w}=\mathcal{H}^\dagger\mathbf{y}$, where $\mathcal{H}^\dagger$ is the Moore–Penrose generalized inverse [32] of the matrix $\mathcal{H}$.

Finally, with the solution $\mathbf{w}\in R^\ell$ obtained, the fitted model $f(.)$ for ELM regression is taken as

$$f(\mathbf{x})=\sum_{s=1}^{\ell}w_sG(\mathbf{a}_s,b_s,\mathbf{x}).\tag{6a}$$

For binary classification problems, the ELM decision function is, however, defined by

$$f(\mathbf{x}) = \operatorname{sign}\left(\sum_{s=1}^{\ell} w_s G(\mathbf{a}_s, b_s, \mathbf{x})\right). \tag{6b}$$

Note that, once the values of the weight vector $\mathbf{a}_s \in R^n$ and the bias $b_s \in R$ are randomly assigned at the beginning of the learning algorithm they remain fixed and therefore the matrix $\mathcal{H}$ remains unchanged.

4 Optimization method based ELM

Recently, Huang et al. [18] studied ELM for classification in the aspect of optimization method by extending ELM to support vector network model and showing that the minimum norm property of ELM and the maximum margin between classes in SVM are in fact consistent.

In this section, we briefly summarize the formulations for optimization method based ELM (OB-ELM) for regression and classification problems.

The mathematical formulation of the optimization method based ε–ELM for regression can be stated as a minimization problem

$$\min_{\mathbf{w},\xi_1,\xi_2} \frac{1}{2}\mathbf{w}^t\mathbf{w} + C\mathbf{e}^t(\xi_1 + \xi_2)$$

subject to

$$\begin{aligned} \mathcal{H}\mathbf{w} - \mathbf{y} &\le \varepsilon\, \mathbf{e} + \xi_1, \\ \mathbf{y} - \mathcal{H}\mathbf{w} &\le \varepsilon\, \mathbf{e} + \xi_2, \\ \xi_1,\ \xi_2 &\ge \mathbf{0}, \end{aligned} \tag{7}$$

where $C > 0, \varepsilon > 0$ are input parameters and $\xi_1, \xi_2 \in R^m$ are vectors of slack variables.

By introducing Lagrange multipliers $\alpha_1, \alpha_2 \in R^m$, the dual of (7) can be constructed as a minimization problem of the form

$$\min_{\alpha_1,\alpha_2 \in R^m} \frac{1}{2}(\alpha_1 - \alpha_2)^t K_{ELM}(\alpha_1 - \alpha_2) + \varepsilon\, \mathbf{e}^t(\alpha_1 + \alpha_2) - \mathbf{y}^t(\alpha_1 - \alpha_2)$$

subject to

$$\mathbf{0} \le \alpha_1, \alpha_2 \le C\,\mathbf{e}, \tag{8}$$

where $(K_{ELM})_{ij} = K_{ELM}(\mathbf{x}_i, \mathbf{x}_j) = (G(\mathbf{a}_1, b_1, \mathbf{x}_i), \ldots, G(\mathbf{a}_s, b_s, \mathbf{x}_i))\ (G(\mathbf{a}_1, b_1, \mathbf{x}_j), \ldots, G(\mathbf{a}_s, b_s, \mathbf{x}_j))^t$ is the ij-th coefficient of the matrix K_{ELM}. With its solution $\alpha_1, \alpha_2 \in R^m$, the regression estimation function (6a) becomes

$$f(\mathbf{x}) = (K_{ELM}(\mathbf{x}, \mathbf{x}_1), \ldots, K_{ELM}(\mathbf{x}, \mathbf{x}_m))\ (\alpha_1 - \alpha_2). \tag{9}$$

Note that, ε–ELM problem (8) has less number of constraints than SVR problem in its dual form and the decision function (9) is similar to the estimation function of SVR but without bias b.

For the purpose of separating the training data with acceptable minimum training error rather than zero training error, OB-ELM for classification can be formulated as

$$\min_{\mathbf{w},\,\xi} \frac{1}{2}\mathbf{w}^t\mathbf{w} + C\mathbf{e}^t\xi$$

subject to

$$\begin{aligned} \mathcal{D}\mathcal{H}\mathbf{w} &\ge \mathbf{e} - \xi, \\ \xi &\ge 0, \end{aligned} \tag{10}$$

where $\mathcal{D} = diag(\mathbf{y})$ is a diagonal matrix of order m whose diagonal elements become the components of the output vector $\mathbf{y} \in R^m$ of class labels. Since the separating ELM hyperplane tends to pass through the origin in the feature space with probability one [18], the bias term ''b'' is not included in (10).

Recall that the aim of ELM is to obtain an output weight vector $\mathbf{w} \in R^\ell$ with smallest training error and further $||\mathbf{w}||$ is minimum. However, since the distance between the separating boundaries of the two classes in the ELM feature will be: $2/||\mathbf{w}||$, minimizing the regularization term in (10) is equivalent to maximizing the margin between the boundaries in the ELM feature space.

The solution of the primal problem (10) will be obtained by solving its dual optimization problem

$$\min_{\alpha \in R^m} \frac{1}{2}\alpha^t\mathcal{D}K_{ELM}\mathcal{D}\alpha - \mathbf{e}^t\alpha$$

subject to

$$\mathbf{0} \le \alpha \le C\,\mathbf{e},$$

where α is the vector of Lagrange multipliers.

In this case, the decision function $f(.)$ is given by

$$f(\mathbf{x}) = \operatorname{sign}((K_{ELM}(\mathbf{x}, \mathbf{x}_1), \ldots, K_{ELM}(\mathbf{x}, \mathbf{x}_m))\ \mathcal{D}\alpha).$$

5 Proposed functional iterative extreme learning machine

In this section, we propose the extension of the study of optimization method based ELM learning approach for classification problems [18] to both regression and classification problems by formulating their primal problems as unconstrained minimization problems and solving them by a simple functional iterative method.

5.1 ε–ELM for regression

The ELM for regression in 2-norm with ε–insensitive error loss function having input parameters $C > 0$ and $\varepsilon > 0$ can be formulated as a constrained minimization problem of the following form [2, 18]

$$\min_{\mathbf{w},\xi_1,\xi_2} \frac{1}{2}\mathbf{w}^t\mathbf{w} + \frac{C}{2}(\xi_1^t\xi_1 + \xi_2^t\xi_2)$$

subject to

$$\begin{aligned}&\mathcal{H}\mathbf{w} - \mathbf{y} \le \varepsilon\,\mathbf{e} + \xi_1,\\ &\mathbf{y} - \mathcal{H}\mathbf{w} \le \varepsilon\,\mathbf{e} + \xi_2\end{aligned} \tag{11}$$

where $\xi_1, \xi_2 \in R^m$ are vectors of slack variables. Using its solution $\mathbf{w} \in R^\ell$, the regression estimation function $f(.)$ defined by (6a) is obtained. The non-negativity constraints on the components of the vectors $\xi_1, \xi_2 \in R^m$ have been dropped in the formulation (11) since none of their components will be negative at optimality. Note that adding the regularization term $\frac{1}{2}\mathbf{w}^t\mathbf{w}$ in the objective function of (11) leads to a stable solution having better generalization performance.

One can easily verify that the constrained primal problem (11) can be formulated into an equivalent unconstrained minimization problem as given below [25]

$$\min_{\mathbf{w}\in R^\ell} L_R(\mathbf{w}) = \frac{1}{2}\mathbf{w}^t\mathbf{w} + \frac{C}{2}(\|(\mathcal{H}\mathbf{w} - (\mathbf{y} + \varepsilon\,\mathbf{e}))_+\|^2 + \|(\mathbf{y} - \varepsilon\mathbf{e} - \mathcal{H}\mathbf{w})_+\|^2). \tag{12}$$

In this subsection, we solve the problem (12) by computing its critical point. For this purpose, consider the problem of solving the system of non-linear equations

$$\nabla L_R(\mathbf{w}) = \mathbf{0},$$

i.e. solve for $\mathbf{w} \in R^\ell$ such that

$$\mathbf{w} + C\mathcal{H}^t((\mathcal{H}\mathbf{w} - \mathbf{y} - \varepsilon\,\mathbf{e})_+ - (\mathbf{y} - \varepsilon\,\mathbf{e} - \mathcal{H}\mathbf{w})_+) = \mathbf{0}. \tag{13}$$

Now, using the identity $\mathbf{u}_+ = \frac{\mathbf{u}+|\mathbf{u}|}{2}$, for any vector $\mathbf{u} \in R^\ell$, the condition (13) can be written in the following equivalent form

$$\mathbf{w} + C\mathcal{H}^t\left[\frac{(\mathcal{H}\mathbf{w} - \mathbf{y} - \varepsilon\,\mathbf{e}) + |\mathcal{H}\mathbf{w} - \mathbf{y} - \varepsilon\,\mathbf{e}|}{2} - \frac{(\mathbf{y} - \varepsilon\,\mathbf{e} - \mathcal{H}\mathbf{w}) + |\mathbf{y} - \varepsilon\mathbf{e} - \mathcal{H}\mathbf{w}|}{2}\right] = \mathbf{0}$$

and from which we get

$$\left(\frac{I}{C} + \mathcal{H}^t\mathcal{H}\right)\mathbf{w} = \mathcal{H}^t\left(\mathbf{y} + \frac{|\mathbf{y} - \varepsilon\,\mathbf{e} - \mathcal{H}\mathbf{w}| - |\mathcal{H}\mathbf{w} - \mathbf{y} - \varepsilon\,\mathbf{e}|}{2}\right). \tag{14}$$

This leads to the following simple iterative scheme which will be our functional iterative ELM algorithm for regression (FELMR)

$$\mathbf{w}^{i+1} = \left(\frac{I}{C} + \mathcal{H}^t\mathcal{H}\right)^{-1} \mathcal{H}^t\left(\mathbf{y} + \frac{|\mathbf{y} - \varepsilon\mathbf{e} - \mathcal{H}\mathbf{w}^i| - |\mathcal{H}\mathbf{w}^i - \mathbf{y} - \varepsilon\mathbf{e}|}{2}\right) \quad \text{for} \quad i = 0, 1, 2, \ldots \tag{15}$$

The iterative algorithm (15) for solving the regression problem (12) is summarized below.

Algorithm 1 (FELMR).

```
Input.
• Parameter values $C > 0$ and $\varepsilon > 0$
• $tol$ = tolerance value for learning accuracy, $imax$ = maximum number of iterations
Step 1.
• Set $i = 0$ and the initial vector $\mathbf{w} = \mathbf{w}^0$ in $R^\ell$
Step 2.
• Compute the vectors $\mathbf{ym} = \mathbf{y} - \varepsilon\,\mathbf{e}$ and $\mathbf{yp} = \mathbf{y} + \varepsilon\,\mathbf{e}$, and the matrix $\mathcal{G} = (\frac{\mathbf{I}}{C} + \mathcal{H}^t\mathcal{H})^{-1}\mathcal{H}^{\mathbf{t}}$
• Compute $\mathbf{w}^1 = \mathcal{G}\left(\mathbf{y} + \frac{|\mathbf{ym} - \mathcal{H}\,\mathbf{w}^0| - |\mathcal{H}\,\mathbf{w}^0 - \mathbf{yp}|}{2}\right)$
Step 3.
• While ( $||\mathbf{w}^{i+1} - \mathbf{w}^i|| > tol$ & $i < imax$)
    $i = i + 1$
    $\mathbf{w}^{i+1} = \mathcal{G}\left(\mathbf{y} + \frac{|\mathbf{ym} - \mathcal{H}\,\mathbf{w}^i| - |\mathcal{H}\,\mathbf{w}^i - \mathbf{yp}|}{2}\right)$
end while
•
    $\mathbf{w} = \mathbf{w}^{i+1}$
```

Now we show the convergence of the proposed FELMR algorithm in the next theorem.

Theorem 1 Assume that the following condition holds:

$$\sqrt{\lambda_{\max}(\mathcal{H}\mathcal{H}^t)\lambda_{\max}(\mathcal{H}^t\mathcal{H})} < \frac{1}{C}.$$

Then, for any starting vector $\mathbf{w}^0$ in R^ℓ, the sequence of iterates $\{\mathbf{w}^i\}$ by the FELMR iterative step (15) converges to the unique solution $\mathbf{w}$ of (12) at the linear rate.

Proof Since $\mathbf{w}$ is the solution of (12), it satisfies (14). Therefore, using (14) and (15), we get

$$\mathbf{w}-\mathbf{w}^{i+1}=\left(\frac{I}{C}+\mathcal{H}^t\mathcal{H}\right)^{-1}$$

$$\mathcal{H}^t\begin{bmatrix}\dfrac{|\mathbf{y}-\varepsilon\,\mathbf{e}-\mathcal{H}\mathbf{w}|-|\mathbf{y}-\varepsilon\,\mathbf{e}-\mathcal{H}\mathbf{w}^i|}{2}\\ +\dfrac{|\mathcal{H}\mathbf{w}^i-\mathbf{y}-\varepsilon\,\mathbf{e}|-|\mathcal{H}\mathbf{w}-\mathbf{y}-\varepsilon\,\mathbf{e}|}{2}\end{bmatrix}$$

But,

$$|\mathbf{y}-\varepsilon\,\mathbf{e}-\mathcal{H}\mathbf{w}|-|\mathbf{y}-\varepsilon\,\mathbf{e}-\mathcal{H}\mathbf{w}^i|\le|\mathcal{H}(\mathbf{w}-\mathbf{w}^i)|$$

and similarly

$$|\mathcal{H}\mathbf{w}^i-\mathbf{y}-\varepsilon\,\mathbf{e}|-|\mathcal{H}\mathbf{w}-\mathbf{y}-\varepsilon\,\mathbf{e}|\le|\mathcal{H}(\mathbf{w}-\mathbf{w}^i)|$$

hold. Thus, we have

$$\left\|\mathbf{w}-\mathbf{w}^{i+1}\right\|\le||\left(\frac{I}{C}+\mathcal{H}^t\mathcal{H}\right)^{-1}\mathcal{H}^t||\left\|\mathcal{H}(\mathbf{w}-\mathbf{w}^i)\right\|$$

$$\le||\left(\frac{I}{C}+\mathcal{H}^t\mathcal{H}\right)^{-1}||\;||\mathcal{H}^t||\quad||\mathcal{H}||\left\|(\mathbf{w}-\mathbf{w}^i)\right\|.$$

Since $||\left(\frac{I}{C}+\mathcal{H}^t\mathcal{H}\right)^{-1}||\le C$ [43] and $\|\mathcal{H}\|=\sqrt{\lambda_{\max}(\mathcal{H}^t\mathcal{H})}$, we get

$$\left\|\mathbf{w}-\mathbf{w}^{i+1}\right\|\le C\sqrt{\lambda_{\max}(\mathcal{H}\mathcal{H}^t)\lambda_{\max}(\mathcal{H}^t\mathcal{H})}\left\|\mathbf{w}-\mathbf{w}^i\right\|$$

and the result follows by our assumption.

5.2 ELM for binary classification

Instead of taking zero training error condition, one may train ELM by allowing acceptable minimum training error. In this case, the ELM in 2-norm can be formulated as a minimization problem in primal of the form [18]

$$\min_{\mathbf{w},\boldsymbol{\xi}}\frac{1}{2}\mathbf{w}^t\mathbf{w}+\frac{C}{2}\boldsymbol{\xi}^t\boldsymbol{\xi}$$

subject to

$$\mathcal{D}\mathcal{H}\mathbf{w}\ge\mathbf{e}-\boldsymbol{\xi},\qquad(16)$$

which is very much similar to the conventional SVM formulation (3) but without the bias term. Here $\boldsymbol{\xi}\in R^m$ is a vector of slack variables and $\mathcal{D}=diag(y_1,\ldots,y_m)$ is a diagonal matrix whose i-th diagonal element is y_i.

Since at the solution of the above problem (16), the condition

$$\boldsymbol{\xi}=(\mathbf{e}-\mathcal{D}\mathcal{H}\mathbf{w})_+\qquad(17)$$

will be satisfied [25] and therefore using (17), the original ELM classification problem in 2-norm can be written as an unconstrained minimization of the following form

$$\min_{\mathbf{w}}L_C(\mathbf{w})=\frac{1}{2}\mathbf{w}^t\mathbf{w}+\frac{C}{2}\left\|(\mathbf{e}-\mathcal{D}\mathcal{H}\mathbf{w})_+\right\|^2.\qquad(18)$$

Consider the problem in solving for $\mathbf{w}\in R^\ell$ such that $\nabla L_C(\mathbf{w})=\mathbf{0}$. Since $\nabla L_C(\mathbf{w})=\mathbf{w}-C\mathcal{H}^t\mathcal{D}(\mathbf{e}-\mathcal{D}\mathcal{H}\mathbf{w})_+$, using the identity $\mathbf{u}_+=\frac{\mathbf{u}+|\mathbf{u}|}{2}$ for any $\mathbf{u}\in R^\ell$, and proceeding as in the regression case, we get the following condition to be satisfied by $\mathbf{w}\in R^\ell$

$$\left(\frac{I}{(C/2)}+\mathcal{H}^t\mathcal{H}\right)\mathbf{w}=\mathcal{H}^t\mathcal{D}(\mathbf{e}+|\mathbf{e}-\mathcal{D}\mathcal{H}\mathbf{w}|).$$

This leads to the following simple iterative scheme which will be our functional iterative ELM algorithm for classification (FELMC)

$$\mathbf{w}^{i+1}=\left(\frac{I}{(C/2)}+\mathcal{H}^t\mathcal{H}\right)^{-1}\mathcal{H}^t\mathcal{D}(\mathbf{e}+|\mathbf{e}-\mathcal{D}\mathcal{H}\mathbf{w}^i|)\qquad(19)$$

for $i=0,1,2,\ldots$

We summarize below the pseudo code of the iterative method (19) applied for solving the classification problem (18).

Algorithm 2 (FELMC).

```
Input.
• Parameter value $C > 0$
• $tol$ = tolerance value for learning accuracy, $imax$ = maximum number of iterations
Step 1.
• Set $i = 0$ and the initial vector $\mathbf{w} = \mathbf{w}^0$ in $R^\ell$
Step 2.
• Compute the matrix $\mathcal{G} = \left(\frac{\mathbf{I}}{C/2} + \mathcal{H}^t\mathcal{H}\right)^{-1}\mathcal{H}^\mathbf{t}\mathcal{D}$
• Compute $\mathbf{w}^1 = \mathcal{G}(\mathbf{e} + |\mathbf{e} - \mathcal{D}\mathcal{H}\mathbf{w}^0|)$
Step 3.
• While ($||\mathbf{w}^{i+1} - \mathbf{w}^i|| > tol$ & $i < imax$)
  $i = i + 1$
  $\mathbf{w}^{i+1} = \mathcal{G}(\mathbf{e} + |\mathbf{e} - \mathcal{D}\mathcal{H}\mathbf{w}^i|)$
  end while
•
  $\mathbf{w} = \mathbf{w}^{i+1}$
```

Following the steps of the proof of Theorem 1, the convergence of the proposed FELMC algorithm can be easily verified.

Theorem 2 Assume that the following condition holds:

$$\sqrt{\lambda_{\max}(\mathcal{H}\mathcal{H}^t)\lambda_{\max}(\mathcal{H}^t\mathcal{H})}<\frac{2}{C}.$$

Then, for any starting vector $\mathbf{w}^0$ in R^ℓ, the sequence of iterates $\{\mathbf{w}^i\}$ by the FELMC iterative step (19) converges to the unique solution $\mathbf{w}$ of (18) at the linear rate.

Finally, for any test data $\mathbf{x} \in R^m$, its class label is predicted using the decision function (6b).

6 Numerical experiments and comparison of results

In this section, the performance of the proposed optimization method based ELM for regression and classification in primal solved by functional iterative methods were compared with SVM, ELM, OP-ELM and OB-ELM on interesting real-world benchmark datasets.

All experiments were carried-out on MATLAB R2008a environment on a PC running on Windows XP OS with 64 bit, 3.20 GHz Intel(R) core (TM) 2 Duo processor having 4 GB of RAM.

The standard SVM and OB-ELM formulations were solved by MOSEK optimization toolbox for MATLAB available at http://www.mosek.com and OP-ELM by the toolbox for OP-ELM [26]. However, no external optimizer was assumed for solving ELM and the proposed FELMR and FELMC.

The performance of the proposed formulation has been tested on additive and RBF hidden nodes. In fact, the activation function $G(\boldsymbol{a}, b, \boldsymbol{x})$ was chosen as the sigmoid function for additive nodes whereas both multiquadric and Gaussian functions were considered for RBF hidden nodes. In case of sigmoid and multiquadric activation functions, the hidden node parameters were chosen randomly with uniform distribution in [−0.5, 0.5] and for Gaussian function, however, they were chosen randomly from [0, 1]. Note that the input weights and biases of the hidden nodes were selected randomly at the beginning of the algorithm and they remain fixed in each trial of simulation.

All the numerical experiments on regression and classification datasets were performed after normalizing the original data by taking: $\bar{x}_{ij} = \frac{x_{ij} - x_j^{\min}}{x_j^{\max} - x_j^{\min}}$, where $x_j^{\min} = \min_{i=1,\cdots,m}(x_{ij})$ and $x_j^{\max} = \max_{i=1,\cdots,m}(x_{ij})$ denote the minimum and maximum values, respectively, of the j-th attribute over all the input examples $\boldsymbol{x}_i$ and $\bar{x}_{ij}$ is the normalized value corresponding to x_{ij}.

6.1 Regression

In this sub-section, we demonstrate the effectiveness of the proposed FELMR for regression in comparison to SVR, ELM, OP-ELM and OB-ELM by performing experiments nonlinearly on a number of benchmark datasets. They include: the inverse dynamics of a flexible robot arm from http://homes.esat.kuleuven.be/∼smc/daisy/daisydata.html; Box and Jenkins gas furnace dataset [5]; Servo, Auto-MPG, Machine CPU, Concrete CS, Wine-quality red, Wine-quality white, Abalone and Parkinson datasets from UCI repository [31]; Pollen grains, Bodyfat and Space_ga from the Statlib collection http://lib.stat.cmu.edu/datasets; Sunspots and SantaFeA from http://www.bme.ogi.edu/∼ericwan/data.html; the financial time series datasets: Citigroup, Intel, Microsoft, RedHat and Standard & Poor 500 (SNP500) from http://finance.yahoo.com; hydraulic actuator [11, 35] and, NO2, Bank-32fth, Demo and Kin-32fh from [8].

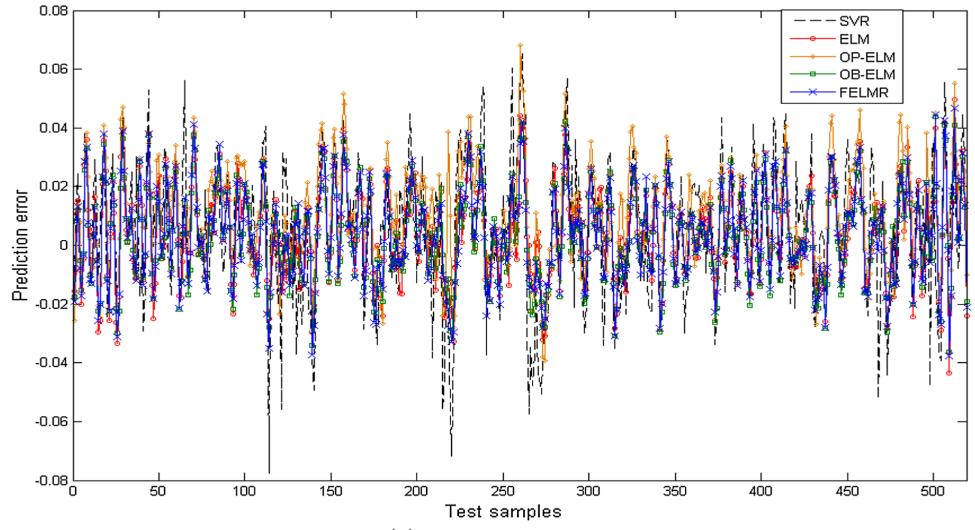
The 2-norm root mean square error (RMSE) was selected as the measure of prediction performance and it was calculated using the following formula: $RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \tilde{y}_i)^2}$, where y_i and $\tilde{y}_i$ are the observed and its corresponding predicted values respectively, and N is the number of test samples.

As the first example, the inverse dynamics of a flexible robot arm is estimated [36] as an interesting example of inverse system identification. It is assumed that the dynamics of the robot arm is a function of the measured values of the reaction torque of the structure $u(t)$ whose output $y(t)$ is its corresponding acceleration. Like in [36], by setting $\mathbf{x}(t) = (u(t-1), \ldots, u(t-5), \; y(t-1), \ldots, y(t-4))^t$ and $x^{\text{out}}(t) = u(t)$, the inverse identification problem was trained. The first 500 samples of the form: $(\mathbf{x}(t), x^{\text{out}}(t))$ were used for training and the remaining 519 samples for testing. Prediction errors corresponding to sigmoid, multiquadric and Gaussian functions on the test dataset were shown in Fig. 1a, b and c respectively.

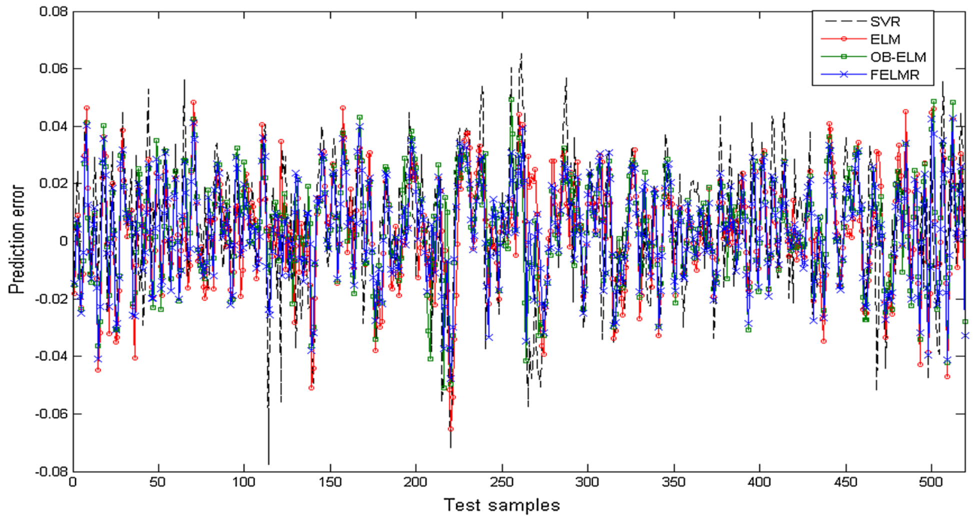
As the next practical example, the Box and Jenkins gas furnace dataset [5] was chosen. It consists of 296 pair of values of the form: $(u(t), y(t))$ where $u(t)$ is the flow rate of the input gas and its output $y(t)$ is the CO_2 concentration from the gas furnace. Assume that the output $y(t)$ is predicted based on six attributes taken to be of the form: $\mathbf{x}(t) = (y(t-1), y(t-2), \; y(t-3), u(t-1), u(t-2), u(t-3))$ [40]. Among the total of 293 samples $(\mathbf{x}(t), y(t))$ the first 100 samples were taken for training and the remaining samples for testing.

As an important study, the problem of time series prediction was considered. Along with the popular Sunspots and SantaFeA time series datasets, the stock index of: Citigroup, Intel, Microsoft, RedHat and SNP500 were further considered as examples of financial time series datasets. 755 closing stock prices starting from 01-01-2006 to 31-12-2008 were selected. Assuming that the current stock index is predicted using its previous five values, we

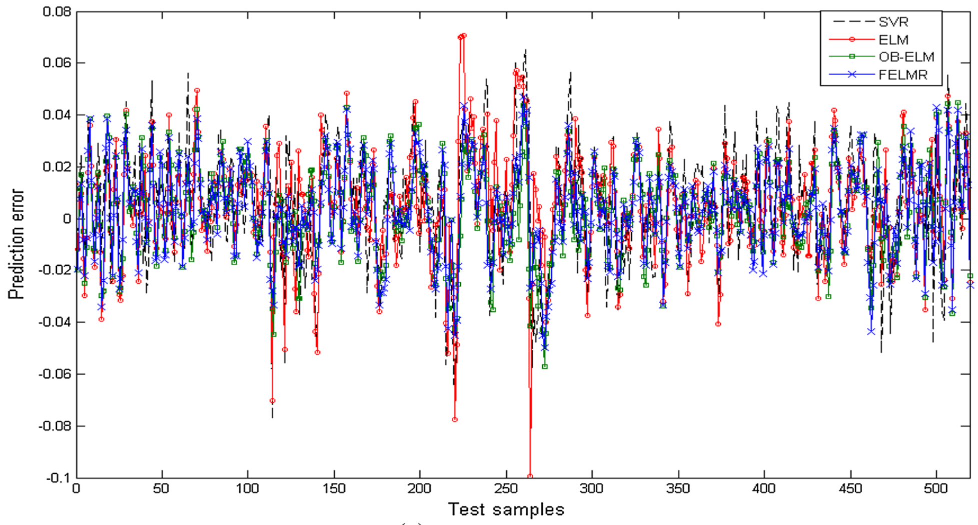
Fig. 1 Prediction error over the test data for flexible robot arm dataset



(a) Sigmoid additive node



(b) Multiquadric RBF node



(c) Gaussian RBF node

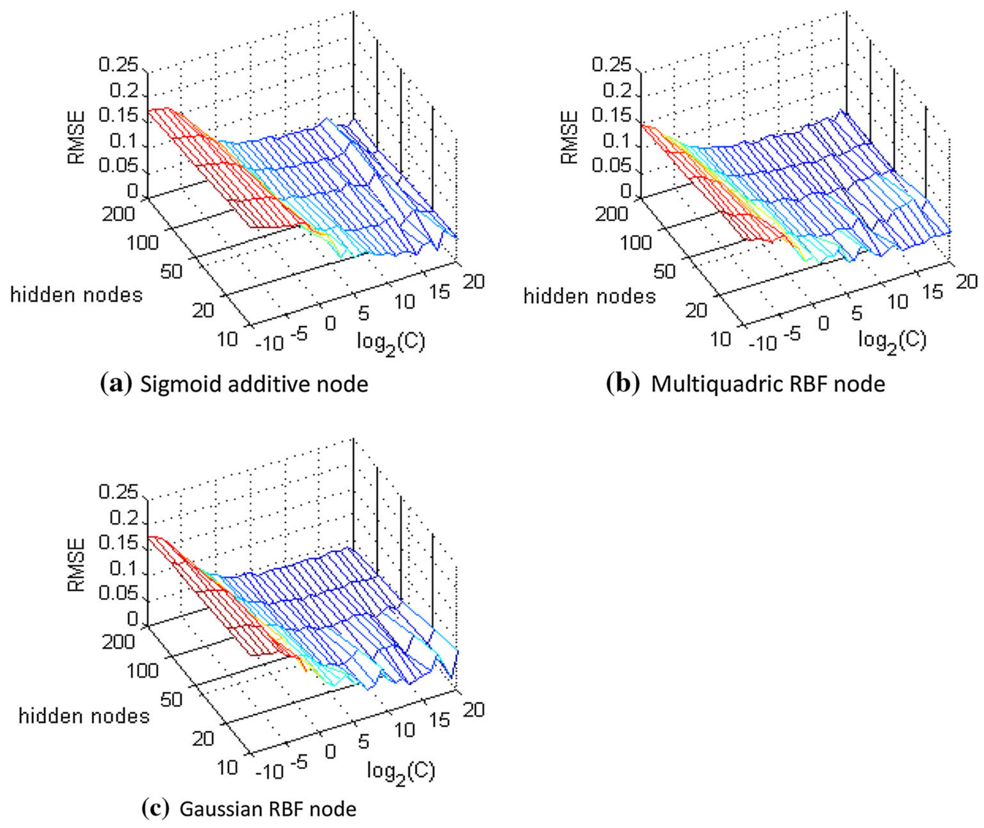


Fig. 2 Insensitivity performance of FELMR for regression to the user specified parameters (C, ℓ) on machine CPU dataset

get 750 samples, in total. The first 200 were taken for training and the rest 550 for testing.

Assuming different sampling rates $\tau = 0.05$ and $\tau = 0.20$ with parameter values $\rho = 10$, $r = 28$ and $b = 8/3$, two time series datasets $Lorenz_{0.05}$ and $Lorenz_{0.20}$ were generated as time series values corresponding to the variable x of the Lorenz differential equation [29, 30]: $\dot{x} = \rho(y - x), \dot{y} = rx - y - xz$ and $\dot{z} = xy - bz$, using fourth-order Runge–Kutta method. After discarding the first 1,000 values to avoid initial transients, the next 3,000 values were considered for our experiment. For predicting the current value, five previous values were used. The first 400 samples were taken for training and the rest for testing.

Finally, experiments were conducted on two more time series datasets, denoted by MG_{17} and MG_{30} generated using the Mackey–Glass time delay differential equation [29, 30]: $\frac{dx(t)}{dt} = -0.1x(t) + \frac{0.2x(t-\tau)}{1+x(t-\tau)^{10}}$, corresponding to the time delay $\tau = 17$ and 30, respectively. As before, five previous values were used to predict the current value. Among the total of 1,495 samples obtained, the first 500 were considered for training and the rest for testing.

Also we considered the hydraulic actuator dataset, a benchmark dataset for nonlinear systems identification [11, 35]. It assumes 1,024 pair of values $(u(t), y(t))$where $u(t)$denotes the size of the valve through which oil flows into the actuator and $y(t)$ is the oil pressure. By considering the same multi-dimensional regression model as in [11, 35], the output $y(t) = f(\mathbf{x}(t))$ is predicted by setting $\mathbf{x}(t) = (y(t-1), y(t-2), y(t-3), u(t-1), u(t-2))^t$ From the total of 1,021 samples of the form: $(\mathbf{x}(t), y(t))$, the first 511 samples were chosen for training and the remaining 510 samples for testing.

In the implementation of SVR, $\varepsilon = 0.01$ is assumed. The Gaussian nonlinear kernel function, $k(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\mu\|\mathbf{x}_i - \mathbf{x}_j\|^2\right)$ for $i, j = 1,\ldots,m$ was taken where $\mu > 0$ is a parameter. The optimal values of the regularization parameter C and the kernel parameter μ were chosen by varying their values over the pre-defined sets: $\{2^{-10}, \ldots, 2^{10}\}$ and $\{2^{-5}, \ldots, 2^{5}\}$ respectively and applying the tenfold cross-validation methodology on the training dataset. In case of ELM, the optimal value of the single parameter ℓ was chosen from the set: $\{10, 20, 50, 100, 200\}$ and for OP-ELM it was selected, however, from the set $\{10, 20, 50, 80, 100\}$. For OB-ELM, the optimal values of C and ℓ were chosen from $\{2^{-15}, \ldots, 2^{20}\}$ and $\{10, 20, 50, 100, 200\}$ respectively. Finally using these optimal parameter values, RMSE on the test set was calculated.

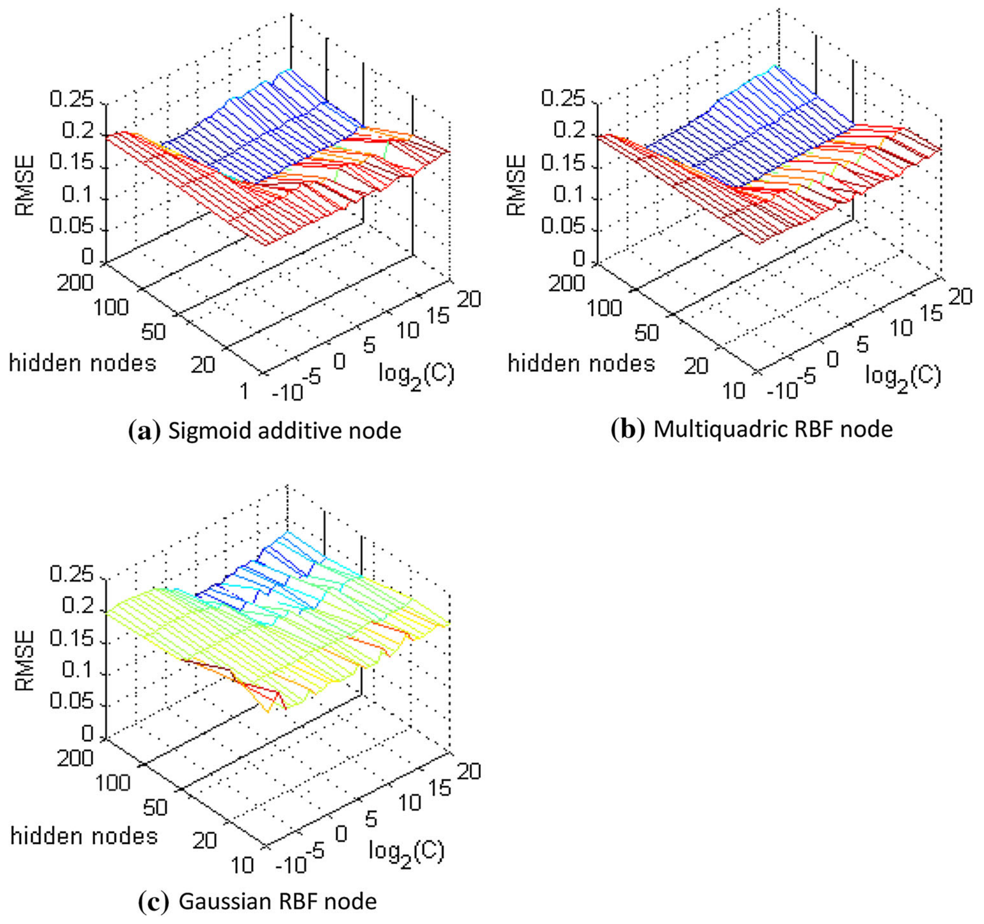
In case of FELMR, it was observed from experiments that better generalization performance could be achieved for small/moderate values of ℓ. Further, since large value of ℓ will result in increase in computational time we assumed $\ell \in \{10, 20, 50, 100, 200\}$. Again, by varying the parameter C from $\{2^{-10}, \ldots, 2^{20}\}$, the optimal values of C and ℓ were obtained using tenfold cross-validation. Using the optimal values, the average test accuracy for each dataset was computed by performing 30 independent trials. As the proposed FELMR is an iterative method, the termination condition was chosen to be: $||\mathbf{w}^{i+1} - \mathbf{w}^{i}|| < 10^{-6}$ and the maximum number of iterations was taken as 20.

Though the performance of SVR is sensitive to its parameters C and μ, it is known that ELM is not sensitive to the choice of the parameter ℓ [20]. After extensive simulations, it was found that FELMR is also not very sensitive to the user specified parameters. To illustrate this result, the performance of FELMR with sigmoid additive hidden node and, multiquadratic and Gaussian RBF hidden nodes on Machine CPU and Bank-32fh datasets was shown in Figs. 2 and 3, respectively. Also from the figures, note that better accuracy could be achieved for small/moderate values of ℓ but with medium/large values of C.

In order to test the convergence of the proposed FELMR algorithm emphericaly for both the additive and RBF hidden nodes in terms of the number of iterations, the values of $||\mathbf{w}^{i+1} - \mathbf{w}^{i}||$were computed as the error of convergence and their results were plotted for Kin-32fh and Wine-quality white datasets in Fig. 6a and b respectively. One can observe from the figures that the rate of convergence of FELMR is impressively faster.

For all the regression datasets considered: the number of training and test samples chosen, the number of attributes, the optimal parameter values determined using tenfold cross-validation and the accuracies obtained by FELMR, OB-ELM, OP-ELM, ELM and SVR on test sets were summarized in Table 1. One can observe from the table that the training time of FELMR is very close to that of ELM and OP-ELM, and much superior to SVR and OB-ELM. The number of times the best accuracy obtained by SVR, ELM, OP-ELM, OB-ELM and FELMR becomes 5, 3, 7, 5 and 12 respectively indicates the possible effectiveness of FELMR.

To further analyze the comparative performance of FELMR with SVR, ELM, OP-ELM and OB-ELM, the average ranks of all the algorithms on RMSE values were computed and listed in Table 2. One can clearly observe



(a) Sigmoid additive node

(b) Multiquadric RBF node

(c) Gaussian RBF node

Fig. 3 Insensitivity performance of FELMR for regression to the user specified parameters (C, ℓ) on bank-32fh dataset

Table 1 Performance comparison of FELMR with ELM, OB-ELM having sigmoid, multiquadric and Gaussian functions, OP-ELM having sigmoid function and SVR using Gaussian kernel for regression

Datasets (train size, test size)	SVR	ELM			OP-ELM
		Sigmoid	Multiquadric	Gaussian	Sigmoid
	(C,μ)	(ℓ)	(ℓ)	(ℓ)	(ℓ)
	Time	Time	Time	Time	Time
Flexible robot arm (500×9,519×9)	0.0171 (2^{10},2^{-5}) 6.6541	0.0167 ± 0.0003 (50) 0.0031	0.0235 ± 0.0008 (50) 0.0021	0.0189 ± 0.0032 (50) 0.0036	0.0166 ± 0.0002 (100) 0.0040
Gas furnace (100×6,193×6)	0.0293 (2^{5},2^{-2}) 0.3272	0.0289 ± 0.0012 (10) 0.0001	0.0310 ± 0.0016 (20) 0.0003	0.0332 ± 0.0134 (20) 0.0003	**0.0287** ± 0.0015 (50) 0.0023
Servo (100×4,67×4)	0.1249 (2^{9},2^{-1}) 0.3448	0.1402 ± 0.0102 (50) 0.0018	0.1432 ± 0.1051 (50) 0.0019	0.1644 ± 0.0083 (20) 0.0005	**0.1078** ± 0.0075 (100) 0.0032
Auto-MPG (100×7,292×7)	0.1644 (2^{4},2^{-2}) 0.3486	0.1614 ± 0.0107 (20) 0.0012	0.1697 ± 0.0106 (20) 0.0003	0.1694 ± 0.0098 (20) 0.0003	0.2289 ± 0.0444 (10) 0.0005
Machine CPU (120×7,89×7)	0.0401 (2^{10},2^{-5}) 0.4938	0.0496 ± 0.0290 (50) 0.0012	0.0540 ± 0.0141 (20) 0.0003	0.0596 ± 0.0220 (200) 0.0111	0.0367 ± 0.0167 (20) 0.0005
Pollen grains (150×5,3698×5)	**0.0788** (2^{6},2^{-5}) 0.7806	0.0824 ± 0.0009 (10) 0.0002	0.0826 ± 0.0021 (20) 0.0007	0.0865 ± 0.0029 (10) 0.0002	0.0878 ± 0.0013 (100) 0.0039
Bodyfat (150×14,102×14)	**0.0156** (2^{5},2^{-5}) 0.7881	0.0620 ± 0.0232 (20) 0.0004	0.1595 ± 0.0144 (20) 0.0012	0.0700 ± 0.0537 (20) 0.0012	0.0275 ± 0.0036 (50) 0.0024
NO2 (200×7,300×7)	0.1494 (2^{6},2^{-5}) 1.3977	0.1512 ± 0.0029 (10) 0.0002	0.1517 ± 0.0045 (20) 0.0014	0.1482 ± 0.0090 (10) 0.0002	**0.1049** ± 0.003 (80) 0.0071
Concrete CS (200×8,830×8)	0.2077 (2^{3},2^{1}) 1.3911	0.2749 ± 0.0675 (20) 0.0013	0.1913 ± 0.0258 (10) 0.0002	0.1578 ± 0.0596 (10) 0.0002	0.1617 ± 0.0278 (20) 0.0005
Sunspots (150×5,140×5)	0.0861 (2^{1},2^{1}) 0.7889	0.0973 ± 0.0043 (20) 0.0015	0.0931 ± 0.0025 (20) 0.0014	0.0833 ± 0.0040 (20) 0.0012	0.1493 ± 0.0457 (20) 0.005
Citigroup (200×5,550×5)	0.0240 (2^{3},2^{-4}) 1.3927	**0.0227** ± 0.0010 (10) 0.0002	0.0244 ± 0.0023 (20) 0.0012	0.0269 ± 0.0238 (10) 0.0002	0.3731 ± 0.0473 (100) 0.0043
Intel (200×5,550×5)	0.0399 (2^{10},2^{-4}) 1.3978	0.0343 ± 0.0008 (10) 0.0002	0.0513 ± 0.0041 (10) 0.0002	0.0415 ± 0.0305 (10) 0.0002	0.1887 ± 0.0572 (10) 0.0003
Microsoft (200×5,550×5)	**0.0314** (2^{5},2^{-5}) 1.3981	0.0315 ± 0.0002 (10) 0.0002	0.0362 ± 0.0014 (20) 0.0002	0.0339 ± 0.0062 (10) 0.0002	0.2615 ± 0.1388 (10) 0.0003
RedHat (200×5,550×5)	**0.0342** (2^{4},2^{-4}) 1.4041	0.0348 ± 0.0003 (10) 0.0002	0.0414 ± 0.0008 (10) 0.0002	0.0371 ± 0.0067 (20) 0.0004	0.172 ± 0.0944 (10) 0.0004
SNP500 (200×5,550×5)	0.0266 (2^{7},2^{-5}) 1.4159	0.0286 ± 0.0019 (10) 0.0002	0.0298 ± 0.0012 (10) 0.0002	0.0304 ± 0.0111 (20) 0.0006	0.2378 ± 0.0223 (10) 0.0007
$\text{Lorenz}_{0.05}$ (400×5,2595×5)	**0.0049** (2^{10},2^{-5}) 6.0221	0.0052 ± 0.0001 (10) 0.0013	0.0054 ± 0.0007 (20) 0.0005	0.0053 ± 0.0052 (50) 0.0018	0.3882 ± 0.1330 (80) 0.0074
$\text{Lorenz}_{0.2}$ (400×5,2595×5)	0.0073 (2^{7},2^{-5}) 5.9671	**0.0013** ± 0.0007 (20) 0.0009	0.0016 ± 0.0146 (50) 0.0028	0.0054 ± 0.0506 (100) 0.0066	0.1316 ± 0.1195 (50) 0.0075
MG_{17} (500×5,995×5)	0.0093 (2^{5},2^{1}) 9.7193	0.0056 ± 0.0005 (200) 0.0333	0.0064 ± 0.0020 (200) 0.0294	0.0109 ± 0.0006 (200) 0.0316	**0.0038** ± 0.0006 (80) 0.0337
MG_{30} (500×5,995×5)	**0.0228** (2^{4},2^{2}) 9.6175	0.0244 ± 0.0006 (200) 0.0363	0.0278 ± 0.0028 (100) 0.0085	0.0443 ± 0.0007 (50) 0.0037	0.0239 ± 0.0009 (80) 0.0098
SantaFeA (500×5,495×5)	0.0479 (2^{7},2^{2}) 9.8100	0.0573 ± 0.0039 (50) 0.0022	0.055 ± 0.0075 (50) 0.0022	0.0564 ± 0.0045 (50) 0.0045	0.0478 ± 0.0030 (50) 0.0043
Hydraulic actuator (511×5,510×5)	0.0151 (2^{8},2^{-5}) 10.1713	0.0148 ± 0.0001 (10) 0.0007	0.0147 ± 0.0005 (20) 0.0006	0.0149 ± 0.0066 (20) 0.0006	0.0154 ± 0.0032 (50) 0.0027
Bank-32fh (800×32,7392×32)	0.1538 (2^{1},2^{-5}) 25.3073	0.1472 ± 0.0014 (50) 0.0031	0.1823 ± 0.0015 (100) 0.0115	0.1457 ± 0.0103 (50) 0.0032	0.1477 ± 0.0007 (100) 0.0045
Kin-32fh (800×32,7392×32)	0.0978 (2^{-1},2^{-5}) 25.1334	0.1003 ± 0.0011 (100) 0.0111	0.1379 ± 0.0010 (100) 0.0120	0.0976 ± 0.0179 (50) 0.0046	0.0967 ± 0.0006 (100) 0.0144
Wine-quality red (800×11,3898×11)	0.1505 (2^{-3},2^{0}) 25.4309	0.1463 ± 0.0031 (10) 0.0011	0.1528 ± 0.0058 (20) 0.0014	0.1543 ± 0.0063 (20) 0.0011	**0.1362** ± 0.0024 (50) 0.0032
Wine-quality white (1000×11,3898×11)	0.1718 (2^{-1},2^{0}) 43.7916	0.1880 ± 0.0203 (50) 0.0033	0.2059 ± 0.0251 (50) 0.0040	0.1872 ± 0.0253 (20) 0.0020	0.1746 ± 0.0135 (100) 0.0046
Demo (1000×4,1048×4)	0.0910 (2^{-3},2^{4}) 44.7248	0.0945 ± 0.0009 (20) 0.0012	0.0933 ± 0.0021 (20) 0.0009	0.0970 ± 0.0007 (50) 0.0040	0.0997 ± 0.01 (100) 0.005
Space_ga (1000×6,2107×6)	0.1793 (2^{9},2^{-3}) 43.1917	**0.1684** ± 0.0153 (20) 0.0015	0.1710 ± 0.0129 (20) 0.0010	0.1727 ± 0.0187 (20) 0.0013	0.1731 ± 0.0139 (100) 0.0051
Abalone (1000×8,3177×8)	0.1564 (2^{6},2^{-5}) 43.9882	0.1279 ± 0.0155 (10) 0.0004	0.1957 ± 0.0382 (20) 0.0020	0.1450 ± 0.0302 (10) 0.0057	**0.0939** ± 0.0045 (100) 0.0043
Parkinson (1000×16,4875×16)	0.3540 (2^{1},2^{4}) 44.0213	0.3973 ± 0.0590 (50) 0.0036	0.4174 ± 0.0485 (100) 0.0133	0.3923 ± 0.0310 (100) 0.0135	**0.3430** ± 0.0386 (100) 0.0147

Table 1 continued

Datasets (train size, test size)	OB-ELM			FELMR		
	Sigmoid (C, ℓ) Time	Multiquadric (C, ℓ) Time	Gaussian (C, ℓ) Time	Sigmoid (C, ℓ) Time	Multiquadric (C, ℓ) Time	Gaussian (C, ℓ) Time
Flexible robot arm (500×9,519×9)	**0.0164** ± 0.0003 (2^{18},50) 1.2989	0.0179 ± 0.0004 (2^{14},100) 2.2421	0.0182 ± 0.0005 (2^{15},100) 1.2479	**0.0164** ± 0.0002 (2^{19},50) 0.0071	0.0176 ± 0.0003 (2^{15},200) 0.0333	0.0182 ± 0.0004 (2^{17},200) 0.0282
Gas furnace (100×6,193×6)	0.0319 ± 0.0014 (2^{6},10) 0.0217	0.033 ± 0.0009 (2^{17},200) 0.0794	0.0295 ± 0.0006 (2^{6},200) 0.0785	0.0291 ± 0.0005 (2^{13},20) 0.0047	0.0332 ± 0.0006 (2^{20},200) 0.0103	0.0317 ± 0.0026 (2^{19},20) 0.0013
Servo (100×4,67×4)	0.1432 ± 0.0015 (2^{19},200) 0.0577	0.1458 ± 0.0079 (2^{7},200) 0.0704	0.1306 ± 0.0016 (2^{14},200) 0.0784	0.1438 ± 0.0022 (2^{20},200) 0.0122	0.1625 ± 0.019 (2^{17},50) 0.0030	0.1330 ± 0.0020 (2^{13},200) 0.0100
Auto-MPG (100×7,292×7)	0.1695 ± 0.0026 (2^{11},200) 0.0646	0.1677 ± 0.0124 (2^{9},20) 0.0389	0.1719 ± 0.0062 (2^{3},10) 0.0308	0.1630 ± 0.0029 (2^{18},100) 0.0062	0.1647 ± 0.0046 (2^{11},20) 0.0011	**0.1609** ± 0.0037 (2^{9},100) 0.0041
Machine CPU (120×7,89×7)	0.038 ± 0.0102 (2^{20},50) 0.0510	0.0519 ± 0.0074 (2^{5},50) 0.0865	0.0365 ± 0.0051 (2^{15},50) 0.0718	**0.0363** ± 0.0044 (2^{19},200) 0.0207	0.0456 ± 0.005 (2^{16},200) 0.0108	0.0445 ± 0.0117 (2^{18},20) 0.0019
Pollen grains (150×5,3698×5)	0.0791 ± 0.0007 (2^{7},20) 0.0521	0.0823 ± 0.0045 (2^{7},20) 0.0880	0.0837 ± 0.0030 (2^{10},20) 0.0806	0.0826 ± 0.0009 (2^{12},10) 0.0005	**0.0788** ± 0.0017 (2^{7},50) 0.0026	0.0843 ± 0.0029 (2^{20},10) 0.0005
Bodyfat (150×14,102×14)	**0.0156** ± 0.0018 (2^{6},50) 0.0591	0.0581 ± 0.0063 (2^{7},200) 0.1402	0.0698 ± 0.0144 (2^{6},200) 0.1418	0.0528 ± 0.0113 (2^{5},50) 0.0080	0.0595 ± 0.0077 (2^{9},100) 0.0051	0.0732 ± 0.0099 (2^{7},200) 0.0136
NO2 (200×7,300×7)	0.1558 ± 0.0016 (2^{9},20) 0.0998	0.1553 ± 0.0043 (2^{17},20) 0.1831	0.1608 ± 0.0040 (2^{12},20) 0.1687	0.1519 ± 0.0024 (2^{12},10) 0.0004	0.1490 ± 0.0018 (2^{10},50) 0.0023	0.1514 ± 0.0047 (2^{17},20) 0.0026
Concrete CS (200×8,830×8)	0.1459 ± 0.0125 (2^{8},100) 0.1264	0.1619 ± 0.0233 (2^{4},50) 0.1783	**0.1348** ± 0.0053 (2^{1},200) 0.2512	0.1395 ± 0.0092 (2^{10},200) 0.0141	0.1528 ± 0.0139 (2^{8},100) 0.0073	0.1428 ± 0.0091 (2^{5},100) 0.0026
Sunspots (150×5,140×5)	0.0912 ± 0.0010 (2^{15},200) 0.1116	0.0832 ± 0.0010 (2^{6},200) 0.1448	0.0913 ± 0.0035 (2^{17},20) 0.0777	0.0909 ± 0.0016 (2^{19},50) 0.0055	**0.0825** ± 0.0011 (2^{9},200) 0.0117	0.0869 ± 0.0007 (2^{6},200) 0.0093
Citigroup (200×5,550×5)	0.0236 ± 0.0025 (2^{5},20) 0.1004	0.0231 ± 0.0019 (2^{4},20) 0.1747	0.0289 ± 0.0007 (2^{8},200) 0.2582	0.0242 ± 0.0014 (2^{10},20) 0.0075	**0.0227** ± 0.0005 (2^{6},200) 0.0115	0.0237 ± 0.0025 (2^{16},20) 0.0016
Intel (200×5,550×5)	**0.0332** ± 0.0003 (2^{2},200) 0.1615	0.0359 ± 0.0009 (2^{8},10) 0.1661	0.0383 ± 0.0019 (2^{5},200) 0.2465	0.0342 ± 0.0001 (2^{7},200) 0.0126	0.0381 ± 0.0016 (2^{11},50) 0.0028	0.0648 ± 0.0203 (2^{16},20) 0.0017
Microsoft (200×5,550×5)	0.0317 ± 0.0004 (2^{8},10) 0.0991	0.0343 ± 0.0010 (2^{18},10) 0.1845	0.0323 ± 0.0005 (2^{9},20) 0.1564	0.0317 ± 0.0005 (2^{12},10) 0.0006	0.0331 ± 0.0007 (2^{10},20) 0.0030	0.0371 ± 0.0052 (2^{19},10) 0.0006
RedHat (200×5,550×5)	0.0346 ± 0.0002 (2^{16},10) 0.0912	0.0351 ± 0.0002 (2^{2},100) 0.1972	0.0376 ± 0.0010 (2^{15},20) 0.1689	0.0350 ± 0.0002 (2^{12},20) 0.0068	0.0364 ± 0.0002 (2^{8},20) 0.0017	0.0386 ± 0.0032 (2^{12},20) 0.0024
SNP500 (200×5,550×5)	0.0268 ± 0.0003 (2^{13},20) 0.0955	0.0282 ± 0.0007 (2^{7},50) 0.1831	0.0314 ± 0.002 (2^{14},20) 0.1614	**0.0263** ± 0.001 (2^{18},10) 0.0005	0.0273 ± 0.0003 (2^{9},50) 0.0013	0.0313 ± 0.0026 (2^{18},20) 0.0022
Lorenz$_{0.05}$ (400×5,2595×5)	0.0051 ± 0.0000 (2^{5},200) 0.7006	0.0054 ± 0.0001 (2^{10},200) 1.2962	0.0052 ± 0.0002 (2^{9},200) 1.0477	0.0053 ± 0.0001 (2^{18},20) 0.0102	0.0054 ± 0.0002 (2^{16},200) 0.0238	0.0056 ± 0.0001 (2^{18},200) 0.0239
Lorenz$_{0.2}$ (400×5,2595×5)	0.0070 ± 0.0008 (2^{7},10) 0.5880	0.0074 ± 0.0005 (2^{4},200) 1.2903	0.0069 ± 0.0006 (2^{19},100) 1.1923	0.0087 ± 0.0009 (2^{12},10) 0.0029	0.0169 ± 0.0077 (2^{20},10) 0.0034	0.0080 ± 0.0006 (2^{12},50) 0.0029
MG$_{17}$ (500×5,995×5)	0.0158 ± 0.0007 (2^{20},200) 1.5249	0.0146 ± 0.0006 (2^{16},200) 2.4546	0.0111 ± 0.0003 (2^{16},200) 2.08	0.0156 ± 0.0006 (2^{20},200) 0.0491	0.0141 ± 0.0006 (2^{20},200) 0.0279	0.0104 ± 0.0005 (2^{20},200) 0.0282
MG$_{30}$ (500×5,995×5)	0.0355 ± 0.0006 (2^{19},200) 1.4169	0.0389 ± 0.0007 (2^{16},200) 2.471	0.0316 ± 0.0003 (2^{17},200) 2.1021	0.0354 ± 0.0004 (2^{20},200) 0.0538	0.0388 ± 0.0007 (2^{20},200) 0.0290	0.0317 ± 0.0005 (2^{20},200) 0.0281

Table 1 continued

Datasets (train size, test size)	OB-ELM			FELMR		
	Sigmoid (C, ℓ) Time	Multiquadric (C, ℓ) Time	Gaussian (C, ℓ) Time	Sigmoid (C, ℓ) Time	Multiquadric (C, ℓ) Time	Gaussian (C, ℓ) Time
SantaFeA (500×5,495×5)	0.0503 ± 0.0015 (2^{17},200) 1.4158	**0.0404** ± 0.0026 (2^{9},100) 2.1477	0.0433 ± 0.0014 (2^{12},200) 1.9585	0.0502 ± 0.0014 (2^{20},200) 0.0390	0.0475 ± 0.0031 (2^{14},100) 0.0081	0.0560 ± 0.0022 (2^{16},200) 0.0305
Hydraulic actuator (511×5,510×5)	0.0152 ± 0.0000 (2^{8},200) 1.3288	0.0149 ± 0.0003 (2^{12},200) 2.4034	0.0155 ± 0.0001 (2^{4},200) 1.4067	0.0147 ± 0.0000 (2^{10},200) 0.0351	**0.0143** ± 0.0002 (2^{16},100) 0.0108	0.0146 ± 0.0001 (2^{11},200) 0.0315
Bank-32fh (800×32,7392×32)	0.1523 ± 0.0009 (2^{1},100) 3.9700	0.1498 ± 0.0007 (2^{0},200) 7.3904	0.1588 ± 0.0031 (2^{5},200) 4.2096	0.1448 ± 0.0006 (2^{5},100) 0.0105	**0.1438** ± 0.0002 (2^{4},200) 0.0254	0.1516 ± 0.0024 (2^{8},200) 0.0302
Kin-32fh (800×32,7392×32)	0.0994 ± 0.0006 (2^{-3},200) 3.9443	0.0978 ± 0.0006 (2^{20},50) 9.0923	0.1063 ± 0.0024 (2^{5},200) 4.1063	0.0966 ± 0.0004 (2^{1},200) 0.0294	**0.0963** ± 0.0002 (2^{3},200) 0.0228	0.1013 ± 0.0020 (2^{6},200) 0.0287
Wine-quality red (800×11,3898×11)	0.1472 ± 0.0048 (2^{17},20) 4.1300	0.1528 ± 0.0029 (2^{2},200) 7.2830	0.1574 ± 0.0042 (2^{19},20) 6.1397	0.1503 ± 0.0024 (2^{12},50) 0.0123	0.1499 ± 0.0014 (2^{5},100) 0.0073	0.1510 ± 0.0021 (2^{5},50) 0.0045
Wine-quality white (1000×11,3898×11)	0.2001 ± 0.0097 (2^{10},100) 7.3657	0.2267 ± 0.0149 (2^{15},50) 15.3858	0.206 ± 0.0368 (2^{18},50) 8.9857	**0.1715** ± 0.0037 (2^{11},200) 0.0558	0.1873 ± 0.0168 (2^{10},100) 0.0121	0.2012 ± 0.0117 (2^{7},200) 0.0386
Demo (1000×4,1048×4)	0.0928 ± 0.0002 (2^{18},50) 8.1151	0.0944 ± 0.0009 (2^{18},100) 15.0106	0.0928 ± 0.0006 (2^{18},50) 11.5100	0.0935 ± 0.0001 (2^{14},200) 0.0650	0.0946 ± 0.0005 (2^{14},200) 0.0561	**0.0909** ± 0.0001 (2^{8},200) 0.0335
Space_ga (1000×6,2107×6)	0.1835 ± 0.0044 (2^{17},100) 8.5159	0.1774 ± 0.0109 (2^{14},20) 13.8141	0.1702 ± 0.0020 (2^{5},200) 10.8053	0.1785 ± 0.0031 (2^{17},200) 0.0743	0.1786 ± 0.0034 (2^{13},100) 0.0154	0.1708 ± 0.0025 (2^{11},200) 0.0531
Abalone (1000×8,3177×8)	0.1815 ± 0.0209 (2^{20},20) 8.1671	0.1607 ± 0.0117 (2^{3},20) 12.5530	0.1704 ± 0.0335 (2^{20},20) 11.8790	0.1506 ± 0.0244 (2^{20},20) 0.0036	0.1738 ± 0.0271 (2^{12},50) 0.0059	0.1479 ± 0.0092 (2^{5},100) 0.0103
Parkinson (1000×16,4875×16)	0.4039 ± 0.0292 (2^{15},200) 8.3357	0.4054 ± 0.0299 (2^{19},200) 15.4552	0.3610 ± 0.0244 (2^{11},200) 11.2508	0.4043 ± 0.033 (2^{18},200) 0.0793	0.3595 ± 0.0208 (2^{14},200) 0.0484	0.3617 ± 0.0214 (2^{14},200) 0.0502

RMSE is used for comparison. The test accuracy is shown for the optimal parameter values

Time is for training in seconds

The best result is shown in boldface

Table 2 Average ranks of SVR, ELM, OP-ELM, OB-ELM and FELMR on RMSE values for regression

Datasets	SVR	ELM			OP-ELM
		Sigmoid	Multiquadric	Gaussian	Sigmoid
Flexible robot arm	5	4	11	10	3
Gas furnace	4	2	6	10.5	1
Servo	2	5	6.5	11	1
Auto-MPG	4	2	9	7	11
Machine CPU	5	8	10	11	3
Pollen grains	1.5	5	6.5	10	11
Bodyfat	1.5	7	11	9	3
NO2	4	5	7	2	1
Concrete CS	10	11	9	6	7
Sunspots	4	10	8.5	3	11
Citigroup	6	1.5	8	9	11
Intel	7	3	9	8	11
Microsoft	1	2	9	7	11
RedHat	1	3	10	7	11
SNP500	2	6	7	8	11
$\text{Lorenz}_{0.05}$	1	3.5	8	5.5	11
$\text{Lorenz}_{0.2}$	6	1	2	3	11
MG_{17}	4	2	3	6	1
MG_{30}	1	3	4	11	2
SantaFeA	5	11	8	10	4
Hydraulic actuator	8	5	3.5	6.5	10
Bank-32fh	9	4	11	3	5
Kin-32fh	5.5	8	11	4	3
Wine-quality red	6	2	8.5	10	1
Wine-quality white	2	6	9	4	3
Demo	2	8	5	10	11
Space_ga	10	1	4	5	6
Abalone	6	2	11	3	1
Parkinson	2	7	11	6	1
Average Rank	4.3276	4.7586	7.8103	7.0862	6.1036

Datasets	OB-ELM			FELMR		
	Sigmoid	Multiquadric	Gaussian	Sigmoid	Multiquadric	Gaussian
Flexible robot arm	1.5	7	8.5	1.5	6	8.5
Gas furnace	8	9	5	3	10.5	7
Servo	6.5	9	3	8	10	4
Auto-MPG	8	6	10	3	5	1
Machine CPU	4	9	2	1	7	6
Pollen grains	3	4	8	6.5	1.5	9
Bodyfat	1.5	5	8	4	6	10
NO2	10	9	11	8	3	6
Concrete CS	4	8	1	2	5	3
Sunspots	7	2	8.5	6	1	5
Citigroup	4	3	10	7	1.5	5
Intel	1	4	6	2	5	10
Microsoft	3.5	8	5	3.5	6	10
RedHat	2	5	8	4	6	9

Table 2 continued

Datasets	OB-ELM			FELMR		
	Sigmoid	Multiquadric	Gaussian	Sigmoid	Multiquadric	Gaussian
SNP500	3	5	10	1	4	9
$\text{Lorenz}_{0.05}$	2	8	3.5	5.5	8	10
$\text{Lorenz}_{0.2}$	5	7	4	9	10	8
MG_{17}	11	9	7	10	8	5
MG_{30}	8	10	5	7	9	6
SantaFeA	7	1	2	6	3	9
Hydraulic actuator	9	6.5	11	3.5	1	2
Bank-32fh	8	6	10	2	1	7
Kin-32fh	7	5.5	10	2	1	9
Wine-quality red	3	8.5	11	5	4	7
Wine-quality white	7	11	10	1	5	8
Demo	3.5	7	3.5	6	9	1
Space_ga	11	7	2	8	9	3
Abalone	10	7	8	5	9	4
Parkinson	8	10	4	9	3	5
Average Rank	5.7414	6.7759	6.7241	4.8103	5.4310	6.4310

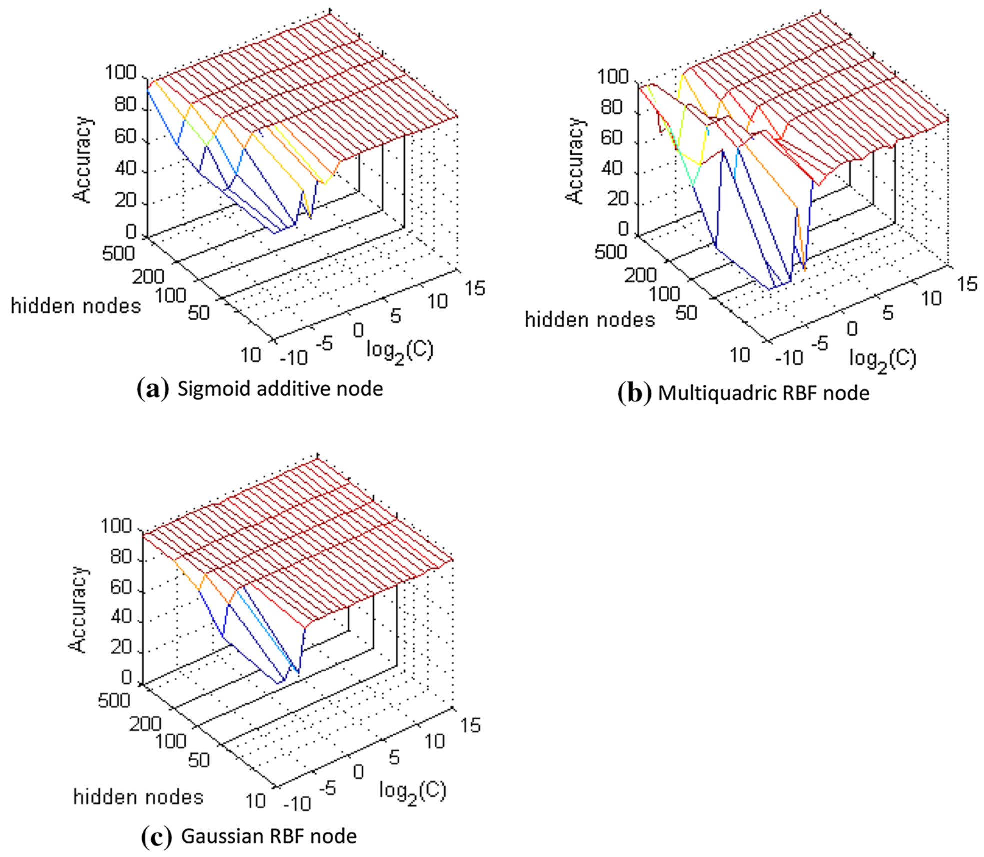
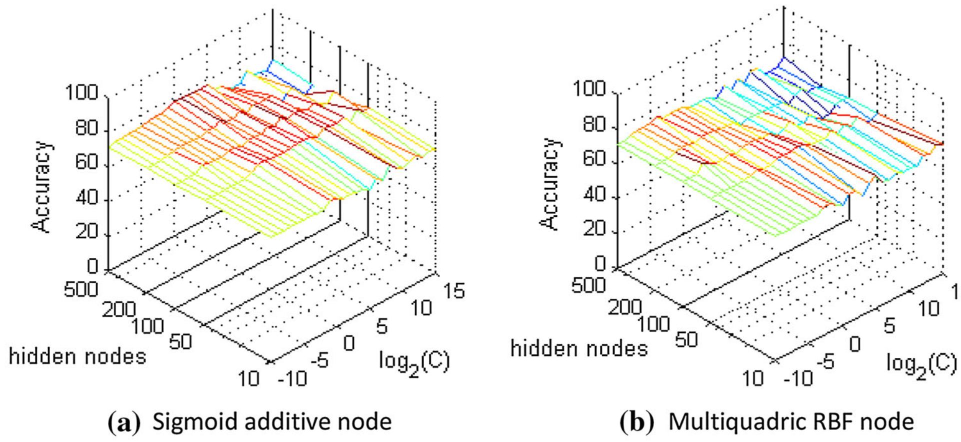


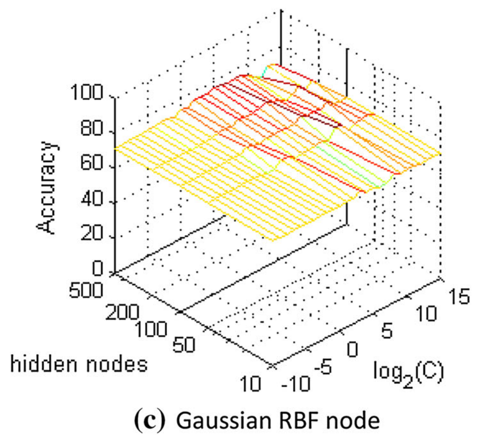
Fig. 4 Insensitivity performance of FELMC for classification to the user specified parameters (C, ℓ) on breast-cancer dataset

that the performance in terms of average rank for either the case of ELM, OB-ELM or FELMR using additive hidden nodes is better than RBF hidden nodes.

For the statistical comparison on the performance of the 11 algorithms where experiments were conducted on 29 datasets, we use the Friedman test with the corresponding

Fig. 5 Insensitivity performance of FELMC for classification to the user specified parameters (C, ℓ) on German dataset





post hoc test recommended in [9] since it is a simple, yet safe and robust non-parametric test. Under the null hypothesis that all the algorithms are equivalent, the Friedman statistic is computed from Table 2 as

$$\chi_F^2 = \frac{12\times 29}{11\times 12}[(4.3276^2 + 4.7586^2 + 7.8103^2 + 7.0862^2 + 6.1036^2 + 5.7414^2 + 6.7759^2 + 6.7241^2 + 4.8103^2 + 5.4310^2 + 6.4310^2) - \frac{11\times 12^2}{4}] = 31.4356$$

$$F_F = \frac{28\times 31.4356}{29\times 10 - 31.4356} = 3.4042.$$

With 11 algorithms and 29 datasets, F_F is distributed according to the F distribution with $(11-1, (11-1)\times(29-1)) = (10, 280)$ degrees of freedom. The critical value of $F(10, 280)$ is 1.8646 for $\alpha = 0.05$. Since $F_F > 1.8646$, we reject the null hypothesis. So, we proceed with Nemenyi test, a post hoc test, for pair wise comparison of algorithms. According to [9], the critical difference (CD) at $p = 0.10$ is $2.9778\sqrt{\frac{11\times 12}{6\times 29}} = 2.5936$.

i) Since the difference between the worst of FELMR and SVR $(6.4310 - 4.3276 = 2.1034)$ is smaller than 2.5936, the post hoc test could unable to detect any significant difference between the algorithms.

ii) For the comparison of FELMR with ELM we proceed as follows

(a) Compare FELMR with ELM using sigmoid function: The difference between the worst of FELMR and ELM using sigmoid $(6.4310 - 4.7586 = 1.6724)$ is smaller than 2.5936, the post hoc test could unable to detect any significant difference between the algorithms.

(b) Compare FELMR using sigmoid function with ELM by multiquadric function: Since the difference between FELMR using sigmoid and ELM by multiquadric $(7.8103 - 4.8103 = 3.0)$ is greater than 2.5936, we see that the performance of FELMR using sigmoid algorithm is better than ELM using multiquadric algorithm.

(c) Compare FELMR using multiquadric and Gaussian functions with ELM by multiquadric function: Since the difference between the best of FELMR using multiquadric and Gaussian and ELM by multiquadric $(7.8103 - 5.4310 = 2.3793)$ is smaller than 2.5936, the post hoc test

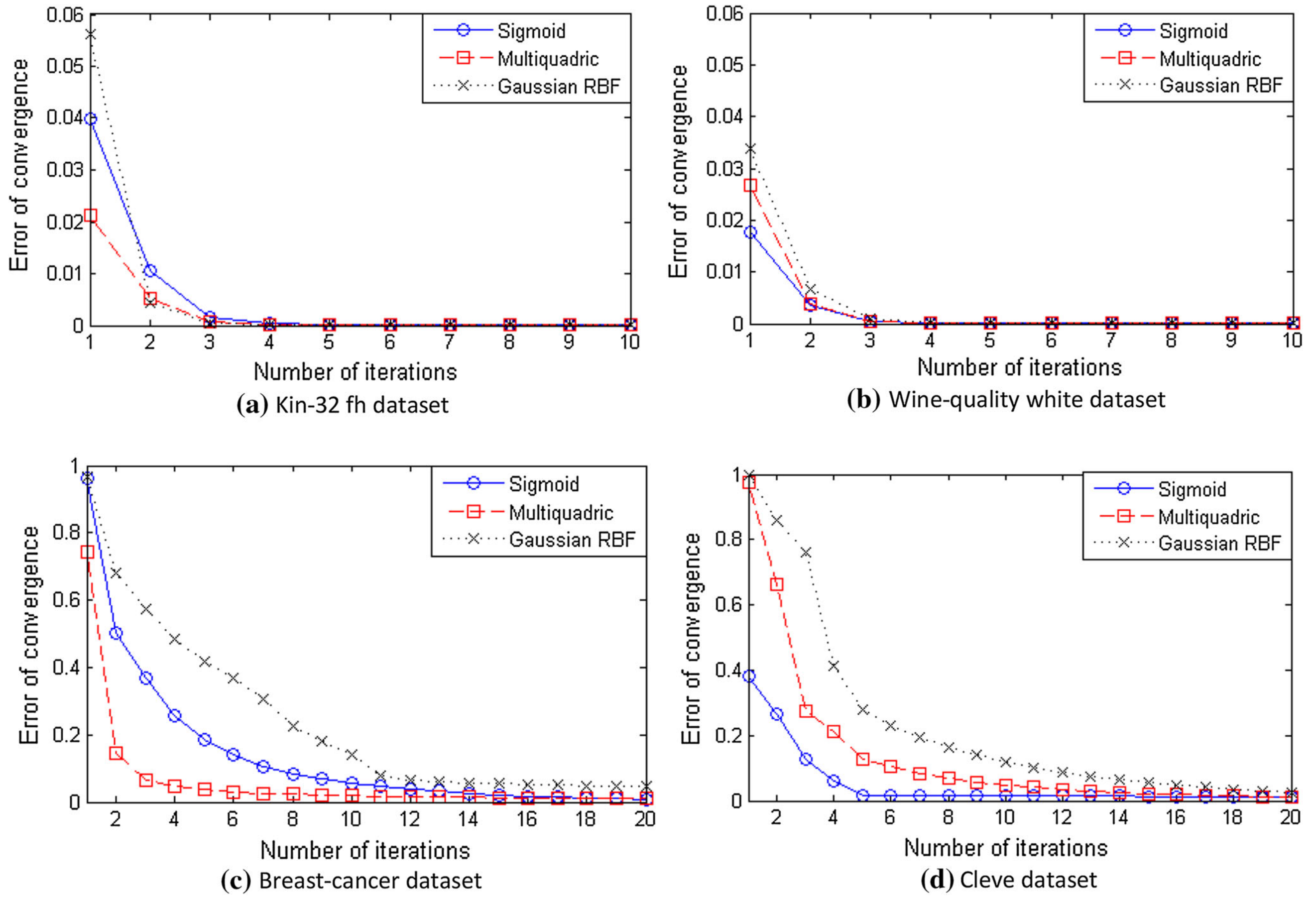


Fig. 6 Error of convergence versus number of iterations of FELM

could unable to detect any significant difference between the algorithms.

(d) Compare FELMR with ELM by Gaussian function: Since the difference between the best of FELMR and ELM using Gaussian function $(7.0862 - 4.8103 = 2.2759)$ is smaller than 2.5936, the post hoc test could unable to detect any significant difference between the algorithms.

iii) For the comparison of FELMR with OP-ELM and OB-ELM, the difference between the best and worst algorithms $(6.7759 - 4.8103 = 1.9656)$ is smaller than 2.5936, the post hoc test could unable to detect any significant difference between the algorithms.

6.2 Classification

In order to verify the effectiveness of the proposed FELMC method for classification, its performance was compared with SVM, ELM, OP-ELM and OB-ELM on 14 binary classification datasets. All the datasets were taken from UCI repository [31].

For the implementation of SVM, the Gaussian nonlinear kernel function with parameter $\sigma > 0$ of the form: $k(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\|\mathbf{x}_i - \mathbf{x}_j\|^2/(2\sigma^2))$ for $i, j = 1,\ldots,m$ has been assumed. The optimal values of C and σ were chosen from predefined sets of values by applying tenfold cross validation methodology. In fact, we assumed $C \in \{10^{-5}, \ldots, 10^5\}$ and $\sigma \in \{2^{-5}, \ldots, 2^5\}$. In case of ELM, the optimal value of the single parameter ℓ was chosen from the set $\{10, 50, 100, 200, 500\}$ and for OP-ELM, however, it was selected from the set $\{10, 20, 50, 80, 100\}$. By choosing the optimal parameter values using tenfold cross-validation, the prediction accuracy was calculated on the test set.

It was observed for FELMC that good generalization performance might be achieved, in general, for medium/large value of ℓ. However, since increase in the number of hidden nodes will result in increase in computational time, we assumed $\ell \in \{10, 50, 100, 200, 500\}$ in both the cases of FELMC and OB-ELM. Also by varying the parameter C from $\{2^{-10}, \ldots, 2^{15}\}$, the optimal values of C and ℓ were obtained using tenfold cross-validation. The average test accuracy was computed by performing 30

Table 3 Performance comparison of FELMC with ELM and OB-ELM having sigmoid, multiquadric and Gaussian functions, OP-ELM having sigmoid function and SVM using Gaussian kernel for binary classification

Datasets (Train size, Test size)	SVM (C, σ)	ELM			OP-ELM
		Sigmoid	Multiquadric	Gaussian	Sigmoid
		(ℓ)	(ℓ)	(ℓ)	(ℓ)
	Time	Time	Time	Time	Time
Breast-cancer (149×10,534×10)	97.38 ($10^0,2^0$) 0.6761	97.28 ± 0.2275 (10) 0.0003	96.18 ± 1.1468 (10) 0.0003	96.80 ± 0.4743 (10) 0.0003	97.36 ± 0.0011 (100) 0.0175
Wpbc (137×32,57×32)	61.40 ($10^5,2^5$) 0.5829	70.88 ± 6.8605 (10) 0.0003	**80.35** ± 5.5990 (10) 0.0003	72.98 ± 3.7848 (50) 0.0027	74.56 ± 0.0447 (10) 0.0019
Cleve (177×13,120×13)	83.33 ($10^1,2^3$) 0.9525	80.58 ± 2.6437 (10) 0.0003	80.58 ± 3.9062 (10) 0.0003	77.17 ± 2.9502 (10) 0.0003	82.58 ± 0.0154 (50) 0.0106
Heart-c (177×13,120×13)	68.33 ($10^0,2^{-1}$) 0.9563	65.42 ± 3.8416 (10) 0.0003	66.08 ± 2.5024 (10) 0.0003	68.17 ± 2.4599 (50) 0.0032	68.50 ± 0.0309 (100) 0.0196
Haberman (200×3,106×3)	76.42 ($10^3,2^1$) 1.2087	76.23 ± 0.9248 (10) 0.0003	76.04 ± 5.5005 (10) 0.0003	76.51 ± 2.6462 (10) 0.0003	76.17 ± 0.0116 (10) 0.0024
Heart-statlog (200×13,70×13)	**84.28** ($10^1,2^2$) 1.2232	80.57 ± 3.6553 (10) 0.0003	82.43 ± 4.1008 (10) 0.0003	80.57 ± 3.0788 (10) 0.0003	82.00 ± 0.0215 (10) 0.0025
Liver-disorders (241×6,104×6)	65.38 ($10^2,2^{-1}$) 1.7576	65.77 ± 3.1295 (50) 0.0022	66.73 ± 2.6971 (50) 0.0024	67.31 ± 2.5344 (50) 0.0025	68.75 ± 0.0380 (100) 0.0114
Ionosphere (246×34,105×34)	88.57 ($10^1,2^0$) 1.8669	83.90 ± 2.6745 (50) 0.0054	85.24 ± 2.5954 (50) 0.0036	80.29 ± 2.5984 (50) 0.0034	89.10 ± 0.0298 (10) 0.0028
Tic tac toe (249×10,709×10)	89.28 ($10^5,2^2$) 1.9075	97.74 ± 0.0000 (500) 0.0754	97.73 ± 0.1107 (50) 0.0027	97.74 ± 0.0000 (200) 0.0338	97.74 ± 0.0000 (50) 0.0149
Votes (306×16,129×16)	95.35 ($10^0,2^1$) 2.8637	96.20 ± 0.7264 (50) 0.0042	96.59 ± 0.6565 (50) 0.0021	94.81 ± 0.9795 (50) 0.0026	96.28 ± 0.0033 (100) 0.0558
Diabetes (537×8,231×8)	78.36 ($10^3,2^2$) 8.9787	79.05 ± 0.7738 (10) 0.0019	**81.30** ± 1.1422 (50) 0.0029	79.44 ± 1.1523 (10) 0.0011	78.48 ± 0.0061 (10) 0.0071
Australian credit (540×14,150×14)	82.67 ($10^1,2^0$) 8.9762	87.53 ± 1.7392 (50) 0.0057	89.73 ± 0.6766 (50) 0.0036	89.87 ± 0.8039 (50) 0.0045	88.00 ± 0.0083 (10) 0.0071
German (800×24, 200×24)	76.50 ($10^4,2^4$) 20.3385	75.00 ± 1.3562 (50) 0.0062	75.40 ± 1.7255 (50) 0.0056	73.10 ± 2.0828 (50) 0.0040	75.10 ± 0.0084 (50) 0.0740
CMC (1000×9, 473×9)	74.63 ($10^{-5},2^{-1}$) 31.1982	74.67 ± 0.0951 (10) 0.0012	**74.69** ± 0.1710 (10) 0.0081	74.63 ± 0.2624 (10) 0.0014	74.57 ± 0.0014 (100) 0.3506

Datasets (Train size, Test size)	OB-ELM			FELMC		
	Sigmoid	Multiquadric	Gaussian	Sigmoid	Multiquadric	Gaussian
	(C, ℓ)	(C, ℓ)	(C, ℓ)	(C, ℓ)	(C, ℓ)	(C, ℓ)
	Time	Time	Time	Time	Time	Time
Breast-cancer (149×10,534×10)	97.40 ± 0.1364 (2^{-3},500) 0.4840	96.85 ± 0.5355 (2^{-4},50) 0.0725	96.74 ± 0.6559 (2^7,10) 0.0294	**97.45** ± 0.1363 (2^2,50) 0.0022	96.84 ± 0.2865 (2^{-4},50) 0.0025	97.19 ± 0.3704 (2^6,10) 0.0007
Wpbc (137×32,57×32)	70.09 ± 5.5032 (2^6,100) 0.0190	62.28 ± 7.0782 (2^5,50) 0.0485	60.7 ± 6.8797 (2^{10},100) 0.0744	73.74 ± 3.3731 (2^9,100) 0.0047	79.30 ± 5.17333 (2^0,100) 0.0050	70.58 ± 3.7472 (2^3,500) 0.1013
Cleve (177×13,120×13)	83.21 ± 0.3053 (2^{-3},500) 0.0396	78.5 ± 3.3054 (2^9,10) 0.0381	74.25 ± 1.5441 (2^7,10) 0.0384	**83.86** ± 2.0646 (2^{-7},500) 0.1227	82.83 ± 1.36668 (2^0,50) 0.0027	79.17 ± 1.6628 (2^3,100) 0.0058
Heart-c (177×13,120×13)	71.17 ± 1.3079 (2^5,100) 0.0355	71.58 ± 0.473 (2^1,100) 0.0925	71.33 ± 1.7033 (2^5,100) 0.0743	**71.78** ± 1.6005 (2^7,200) 0.0187	68.39 ± 2.7310 (2^{13},10) 0.0009	71.64 ± 1.048 (2^0,500) 0.1187
Haberman (200×3,106×3)	76.37 ± 0.211 (2^{14},500) 0.0555	75.38 ± 0.9381 (2^{12},50) 0.064	75.47 ± 0.4447 (2^7,100) 0.0783	75.57 ± 0.3576 (2^6,50) 0.0025	**76.70** ± 0.9510 (2^7,10) 0.0010	75.66 ± 0.3904 (2^7,200) 0.0204
Heart-Statlog (200×13,70×13)	84.07 ± 0.6991 (2^3,50) 0.0407	81.43 ± 0.0000 (2^{-2},200) 0.118	80.00 ± 3.5635 (2^{10},50) 0.064	82.86 ± 0.0000 (2^{-4},500) 0.1361	82.86 ± 1.0399 (2^{-3},50) 0.0026	84.05 ± 1.6216 (2^3,100) 0.0063
Liver-disorders (241×6,104×6)	**73.80** ± 1.3190 (2^{14},500) 0.0742	68.94 ± 2.796 (2^{10},100) 0.1043	71.44 ± 1.5735 (2^8,200) 0.1517	65.58 ± 0.0000 (2^{-4},200) 0.0242	66.44 ± 3.7843 (2^{11},100) 0.0072	70.80 **±** 2.5494 (2^{11},50) 0.0039

Table 3 continued

Datasets (Train size, Test size)	OB-ELM			FELMC		
	Sigmoid (C, ℓ) Time	Multiquadric (C, ℓ) Time	Gaussian (C, ℓ) Time	Sigmoid (C, ℓ) Time	Multiquadric (C, ℓ) Time	Gaussian (C, ℓ) Time
Ionosphere (246×34,105×34)	87.48 ± 0.8334 (2^4,200) 0.0682	87.14 ± 0.9255 (2^4,500) 0.3531	89.14 ± 2.1153 (2^{11},500) 0.3529	87.86 ± 1.9168 (2^3,100) 0.0072	85.30 ± 0.8549 (2^9,100) 0.0072	**89.17** ± 2.5185 (2^{11},500) 0.1578
Tic tac toe (249×10,709×10)	87.01 ± 1.2934 (2^{13},500) 0.0847	97.55 ± 0.3271 (2^8,50) 0.0925	97.19 ± 0.4815 (2^8,200) 0.1653	88.55 ± 1.3523 (2^9,500) 0.1652	**97.78** ± 0.0488 (2^{13},100) 0.0073	97.74 ± 0.0000 (2^{15},500) 0.1782
Votes (306×16,129×16)	95.70 ± 0.4688 (2^{-1},100) 0.1041	96.20 ± 0.4400 (2^{-1},50) 0.1371	95.89 ± 0.6382 (2^3,200) 0.2257	96.49 ± 0.6335 (2^{10},50) 0.0037	**96.90** ± 0.0000 (2^{10},100) 0.0085	96.74 ± 0.2938 (2^{10},500) 0.1936
Diabetes (537×8,231×8)	78.94 ± 0.6004 (2^{11},100) 0.4441	79.65 ± 0.6122 (2^4,500) 0.9784	79.31 ± 0.6388 (2^7,50) 0.4809	76.45 ± 1.3128 (2^{15},100) 0.0150	78.12 ± 1.9737 (2^{11},50) 0.0056	79.47 ± 1.0110 (2^5,50) 0.0055
Australian credit (540×14,150×14)	89.47 ± 0.5118 (2^7,200) 0.5198	85.93 ± 1.6466 (2^{11},200) 0.7491	82.47 ± 1.1353 (2^7,200) 0.6842	88.02 ± 1.9581 (2^{15},10) 0.0015	**89.96** ± 0.8592 (2^3,100) 0.0149	84.62 ± 2.4343 (2^7,50) 0.0056
German (800×24,200×24)	75.18 ± 1.3502 (2^6,100) 1.2242	75.95 ± 1.0395 (2^5,100) 1.4846	73.90 ± 0.7746 (2^9,100) 1.4382	75.83 ± 0.9072 (2^3,100) 0.0279	**76.68** ± 1.1137 (2^6,100) 0.0261	73.75 ± 1.0965 (2^5,100) 0.0246
CMC (1000×9,473×9)	74.63 ± 0.0000 (2^{-10},10) 2.6249	74.63 ± 0.0000 (2^{-10},10) 2.3966	74.63 ± 0.0000 (2^{-10},10) 2.1363	74.63 ± 0.9139 (2^{10},10) 0.0048	**74.69** ± 0.1750 (2^{14},10) 0.0030	74.28 ± 1.0874 (2^{12},10) 0.0029

The test accuracy is shown for the optimal parameter values

Time is for training in seconds

The best result is shown in boldface

Table 4 Average ranks of SVR, ELM, OP-ELM, OB-ELM and FELMC on accuracy values for binary classification

Datasets	SVM	ELM			OP-ELM
		Sigmoid	Multiquadric	Gaussian	Sigmoid
Breast-cancer	3	5	11	9	4
Wpbc	10	6	1	5	3
Cleve	2	6.5	6.5	10	5
Heart-c	8	11	10	9	6
Haberman	3	5	7	2	6
Heart-statlog	1	9.5	6	9.5	7
Liver-disorders	11	9	7	6	5
Ionosphere	4	10	9	11	3
Tic tac toe	9	3.5	6	3.5	3.5
Votes	10	6.5	3	11	5
Diabetes	9	6	1	4	8
Australian Credit	10	7	3	2	6
German	2	8	5	11	7
CMC	6.5	3	1.5	6.5	10
Average rank	6.3214	6.8571	5.5000	7.1071	5.6071

Datasets	OB-ELM			FELMC		
	Sigmoid	Multiquadric	Gaussian	Sigmoid	Multiquadric	Gaussian
Breast-cancer	2	7	10	1	8	6
Wpbc	8	9	11	4	2	7
Cleve	3	9	11	1	4	8
Heart-c	5	3	4	1	7	2
Haberman	4	11	10	9	1	8
Heart-statlog	2	8	11	4.5	4.5	3
Liver-disorders	1	4	2	10	8	3
Ionosphere	6	7	2	5	8	1
Tic tac toe	11	7	8	10	1	3.5
Votes	9	6.5	8	4	1	2
Diabetes	7	2	5	11	10	3
Australian credit	4	8	11	5	1	9
German	6	3	9	4	1	10
CMC	6.5	6.5	6.5	6.5	1.5	11
Average rank	5.3214	6.5000	7.7500	5.4286	4.1429	5.4643

independent trials. Since FELMC is an iterative method, the termination condition was taken as: $||\mathbf{w}^{i+1} - \mathbf{w}^{i}|| < 10^{-3}$. The maximum number of iterations was assumed as 20.

As in the case of regression, it was observed that the performance of FELMC is not sensitive to the values of its parameters C and ℓ. To verify this result, its performance with additive and RBF hidden nodes on Breast-cancer and German credit datasets was illustrated in Figs. 4 and 5. From the figures, one can observe that better accuracy is resulted for medium/large values of C.

Like in the case of regression, the convergence of the proposed FELMC algorithm in terms of the number of iterations were computed as the error of convergence and their results were shown for Breast cancer and Cleve datasets in Fig. 6c and d, respectively. For these classification datasets, the error of convergence becomes stable by the 20th iteration. Also observe that, unlike the case of regression, the rate of convergence of FELMC is slower.

For all the classification datasets considered, the number of training and test samples chosen, the number of attributes, the optimal parameter values determined using ten-fold cross-validation and the accuracies obtained by SVM, ELM, OP-ELM, OB-ELM and FELMC on test sets were summarized in Table 3. One can conclude from the table that the proposed ELM model trained with functional iterative method shows comparable generalization performance in accordance with the rest of the methods considered. Again from Table 3, note that the number of times the best accuracy obtained by SVM, ELM, OP-ELM, OB-ELM and FELMC becomes 1, 3, 0, 1 and 10 respectively. This indicates the supremacy of FELMC. In terms of training time, as expected, ELM and OP-ELM outperform the other methods considered. Like in the case of regression, the training time of FELMC is very close to ELM and OP-ELM and much faster than SVM and OB-ELM. Note that, in case of FELMC, multiquadrtic shows the best performance among the activation functions.

For the statistical comparison on the performance of the 11 algorithms over 14 datasets we use, as in the case of regression, the Friedman test with post hoc test recommended in [9]. For this purpose, the average ranks of all the algorithms on the prediction accuracy values were computed and listed in Table 4. Under the null hypothesis that all the algorithms are equivalent, the Friedman statistic is computed as

$$\chi_F^2 = \frac{12\times 14}{11\times 12}[(6.3214^2 + 6.8571^2 + 5.5^2 + 7.1071^2 + 5.6071^2 + 5.3214^2 + 6.5^2 + 7.75^2 + 5.4286^2 + 4.1429^2 + 5.4643^2) - \frac{11\times 12^2}{4}] = 13.1117$$

$$F_F = \frac{13\times 13.1117}{14\times 10 - 13.1117} = 1.3433.$$

In this case, F_F is distributed according to the F distribution with $(10, 130)$ degrees of freedom. The critical value of $F(10, 130)$ is 1.9042 for $\alpha = 0.05$. Since the value of F_F is smaller than its corresponding critical value, there is no significant error between the algorithms.

7 Conclusion

In this work, a simple novel functional iterative method for the solution of the optimization based ELM in its primal for regression and classification has been proposed. The linear convergence of the proposed method is proved. Numerical experiments have been performed with sigmoid and RBF hidden nodes on a number of real-world, benchmark datasets and their results have been compared with SVM, ELM, OP-ELM and OB-ELM for regression and classification. Comparable generalization performance of the proposed approach with the rest of the methods considered at a faster learning speed than SVM and OB-ELM indicates its usefulness and applicability. Future work will be on the study ELM in dual variables and its applications.

Acknowledgments The authors are extremely thankful to the learned referees for their constructive comments that greatly improved the earlier version of the paper.

References

1. Balasundaram S, Gupta D, Kapil (2014) 1-norm extreme learning machine for regression and multiclass classification using Newton method. Neuocomputing 128:4–14
2. Balasundaram S, Kapil (2013) On extreme learning machine for ε–insensitive regression in the primal by Newton method. Neural Comput Appl 22:559–567
3. Balasundaram S, Kapil (2011) Application of error minimized extreme learning machine for simultaneous learning of a function and its derivatives. Neurocomputing 74:2511–2519
4. Binu PC, Vimal Krishnan VR, Raju G, Babu Anto P (2012) Handwritten character recognition using wavelet energy and extreme learning machine. Int J Mach Learn Cybern 3(2):149–161
5. Box GEP, Jenkins GM (1976) Time series analysis: forecasting and Control. Holden-Day, San Francisco
6. Chen Z, Zhu H, Wang Y (2013) A modified extreme learning machine with sigmoidal activation functions. Neural Comput Appl 22:541–550
7. Cristianini N, Shawe-Taylor J (2000) An introduction to support vector machines and other kernel based learning methods. Cambridge University Press, Cambridge
8. DELVE (2005) Data for evaluating learning in valid experiments, http://www.cs.toronto.edu/~delve/data
9. Demsar J (2006) Statistical comparisons of classifiers over multiple data sets. J Mach Learn Res 7: 1-30
10. Deng W, Chen L (2010) Color image watermarking using regularized extreme learning machine. Neural Netw World 20(3):317–330
11. Gretton A, Doucet A, Herbrich R, Rayner PJW, Scholkopf B (2001) Support vector regression for black-box system identification, In: Proceedings of the 11th IEEE Workshop on Statistical Signal Processing
12. Feng G, Huang G-B, Lin Q, Gay R (2009) Error minimized extreme learning machine with growth of hidden nodes and incremental learning. IEEE Trans Neural Netw 20(8):1352–1357
13. Frenay B, Verleysen M (2010) Using SVMs with randomized feature spaces: an extreme learning approach. In: Proceedings of the 18th European Symposium on Artificial Neural Networks (ESANN), Bruges, Belgium, pp. 315–320
14. Fu A, Dong C, Wang L (2014) An experimental study on stability and generalization of extreme learning machines. Int J Mach Learn Cybern. doi:10.1007/s13042-014-0238-0
15. Huang G-B, Chen L (2007) Convex incremental extreme learning machine. Neurocomputing 70:3056–3062
16. Huang G-B, Chen L (2008) Enhanced random search based incremental extreme learning machine. Neurocomputing 71:3460–3468
17. Huang G-B, Chen L, Siew C-K (2006) Universal approximation using incremental constructive feedforward networks with random hidden nodes. IEEE Trans Neural Netw 17(4):879–892
18. Huang G-B, Ding X, Zhou H (2010) Optimization method based extreme learning machine for classification. Neurocomputing 74:155–163

19. Huang G-B, Wang DH, Lan Y (2011) Extreme learning machines: a survey. Int J Mach Learn Cybern 2:107–122
20. Huang G-B, Zhou H, Ding X, Zhang R (2012) Extreme learning machine for regression and multiclass classification. IEEE Trans Syst, Man Cybern Part B Cybern 42(2):513–528
21. Huang G-B, Zhu Q-Y, Siew C-K (2006) Extreme learning machine: theory and applications. Neurocomputing 70:489–501
22. Jun W, Shitong W, Chung F (2011) Positive and negative fuzzy rule system, extreme learning machine and image classification. Int J Mach Learn Cybern 2(4):261–271
23. Lan Y, Soh C, Huang G-B (2010) Two-stage extreme learning machine regression. Neurocomputing 73:3028–3038
24. Liu Q, He Q, Shi Z (2008) Extreme support vector machine classifier. LNCS 5012:222–233
25. Mangasarian OL (2002) A finite newton method for classification. Optim Methods Softw 17:913–929
26. Miche Y, Sorjamaa A, Bas P, Simula O, Jutten C, Lendasse A (2010) OP-ELM: optimally pruned extreme learning machine. IEEE Trans Neural Netw 21(1):158–162
27. Minhas R, Baradarani A, Seifzadeh S, Wu QMJ (2010) Human action recognition using extreme learning machine based on visual vocabularies. Neurocomputing 73:1906–1917
28. Mohammed AA, Wu QMJ, Sid-Ahmed MA (2010) Application of wave atoms decomposition and extreme learning machine for finger print classification. LNCS 6112:246–256
29. Mukherjee S, Osuna E, Girosi F (1997) Nonlinear prediction of chaotic time series using support vector machines, in: NNSP'97: Neural Networks for Signal Processing VII: in Proceedigs of IEEE Signal Processing Society Workshop, Amelia Island, FL, USA, pp. 511–520
30. Muller KR, Smola AJ, Ratsch G, Schölkopf B, Kohlmorgen J (1999) Using support vector machines for time series prediction. In: Schölkopf B, Burges CJC, Smola AJ (eds) Advances in kernel methods—support vector learning. MIT Press, Cambridge, pp 243–254
31. Murphy PM, Aha DW (1992) UCI repository of machine learning databases. University of California, Irvine. http://www.ics.uci.edu/~mlearn
32. Rao CR, Mitra SK (1971) Generalized inverse of matrices and its applications. Wiley, New York
33. Samet S, Miri A (2012) Privacy-preserving back-propagation and extreme learning machine algorithms. Data Knowl Eng 79–80:40–61
34. Singh R, Balasundaram S (2007) Application of extreme learning machine for time series analysis. Int J Intell Technol 2:256–262
35. Sjoberg J, Zhang Q, Ljung L, Berveniste A, Delyon B, Glorennec P, Hjalmarsson H, Juditsky (1995) Nonlinear black-box modeling in system identification: a unified overview. Automatica 31:1691–1724
36. Souza LGM, Barreto GA (2006) Nonlinear system identification using local ARX models based on the self-organizing map. Learning and Nonlinear Models-Revista da Sociedade Brasileira de Redes Neurals (SBRN) 4(2):112–123
37. Vapnik VN (2000) The nature of statistical learning theory, 2nd edn. Springer, New York
38. Wang X, Shao Q, Qing M, Zhai J (2013) Architecture selection for networks trained with extreme learning machine using localized generalization error model. Neurocomputing 102:3–9
39. Wang X, Chen A, Feng H (2011) Upper integral network with extreme learning mechanism. Neurocomputing 74(16): 2520–2525
40. Wang XX, Chen S, Lowe D, Harris CJ (2006) Sparse support vector regression based on orthogonal forward selection for generalized kernel model. Neurocomputing 70:462–474
41. Yuan Y, Wang Y, Cao F (2011) Optimization approximation solution for regression problem based on extreme learning machine. Neurocomputing 74:2475–2482
42. Zhai J, Xu H, Wang X (2012) Dynamic ensemble extreme learning machine based on sample entropy. Soft Comput 16(9):1493–1502
43. Zhou GL, Toh KC (2005) Super linear convergence of a newton-type algorithm for monotone equations. J Optim Theory Appl 125:205–221
44. Zhu Q-Y, Qin AK, Suganthan PN, Huang G-B (2005) Evolutionary extreme learning machine. Pattern Recogn 38:1759–1763