Neural Comput & Applic (2013) 22:559–567
DOI 10.1007/s00521-011-0798-9

On extreme learning machine for ε-insensitive regression in the primal by Newton method

S. Balasundaram · Kapil

Received: 1 August 2011 / Accepted: 26 December 2011 / Published online: 16 February 2012

Abstract In this paper, extreme learning machine (ELM) for ε-insensitive error loss function-based regression problem formulated in 2-norm as an unconstrained optimization problem in primal variables is proposed. Since the objective function of this unconstrained optimization problem is not twice differentiable, the popular generalized Hessian matrix and smoothing approaches are considered which lead to optimization problems whose solutions are determined using fast Newton–Armijo algorithm. The main advantage of the algorithm is that at each iteration, a system of linear equations is solved. By performing numerical experiments on a number of interesting synthetic and real-world datasets, the results of the proposed method are compared with that of ELM using additive and radial basis function hidden nodes and of support vector regression (SVR) using Gaussian kernel. Similar or better generalization performance of the proposed method on the test data in comparable computational time over ELM and SVR clearly illustrates its efficiency and applicability.

Keywords Extreme learning machine · Generalized Hessian matrix · Newton method · Single hidden layer feedforward neural networks · Smoothing technique · Support vector regression

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

Kapil
e-mail: navkapil@gmail.com

1 Introduction

Prediction by regression is an important method of solution for forecasting. The goal of regression problem is in determining the underlying mathematical relationship between the given inputs and their output values. Two very important and widely used models in the literature for regression are feedforward neural networks and support vector regression (SVR). Though feedforward neural networks training algorithms have the ability to learn adaptively from the given data, they suffer from several disadvantages such as presence of local minima, imprecise learning rate, over fitting and slow rate of convergence.

Over the last decade, with the introduction of ε-insensitive error loss function by Vapnik [28], the method of Support Vector Machines (SVMs) has emerged as a paradigm of choice for regression due to its excellent performance on problems of real-world applications. SVR has been widely applied in many fields of study of importance such as image segmentation [5] and time series prediction [22, 23, 27]. Inspired by the study of "equivalent" 2-norm SVM formulations for classification problem proposed in [19–21], "equivalent" 2-norm ε-insensitive SVR (ε-SVR) has been formulated as a constrained minimization problem and solved in its dual form using active set strategy in [24]. For the formulation of ε-SVR as an unconstrained minimization problem in its primal and solving it using Newton method with Armijo stepsize, see [1, 16].

Recently, Huang et al. [15] proposed a new learning algorithm for single hidden layer feedforward neural networks (SLFNs) architecture called extreme learning machine (ELM) method that overcomes the problems of feedforward neural networks training algorithms. For the randomly chosen input weights and hidden layer biases, ELM formulation leads to a least squares solution of a system of linear equations for the unknown output weights having the smallest norm property [15]. Although ELM is a simple and an efficient learning algorithm, the number of hidden nodes of the SLFNs is an input and is to be known at the beginning of the algorithm. However, the choice of the optimal number of hidden nodes, for a given problem, is unknown. For the study of adding hidden nodes to SLFNs in an incremental manner, we refer the reader to [7, 11, 12].

Replacing the SVM kernels by ELM kernels in the SVM formulation, it was shown in [8, 18] that better generalization can be achieved. Formulating the ELM for classification as an optimization problem, ELM for SVM was proposed recently in [14] and it was observed further that the proposed method achieves better generalization performance than SVM and is less sensitive to the input parameters. Motivated by their work, we propose in this paper the study of ELM for ε-insensitive regression in 2-norm formulated as an unconstrained optimization problem in primal variables. Since its objective function is not twice differentiable, we considered generalized Hessian matrix [1, 9, 10] and smoothing approaches [16, 17] that lead to optimization problems whose solutions are obtained using fast Newton–Armijo algorithm. Its main advantage is that the solution of the problem is obtained by solving a system of linear equations at a finite number of times rather than solving a quadratic programming problem as in the case of standard SVR. For either of the problem formulation proposed, the global convergence of Newton–Armijo algorithm to its unique solution of the optimization problem will follow from [1, 16].

The paper is organized as follows. In Sect. 2, SVR formulation is outlined. A brief review of ELM [15] is given in Sect. 3. The study of ELM for ε-insensitive regression in primal by Newton–Armijo algorithm is proposed in Sect. 4. Experimental results obtained by the proposed method with additive and radial basis function (RBF) hidden nodes are compared with the results of ELM and the standard SVR with Gaussian kernel in Sect. 5. Finally, we conclude our paper in Sect. 6.

Throughout in this work all the vectors are assumed as column vectors. For any two vectors x, y in the n-dimensional real-space R^n, the inner product of the vectors will be denoted by $x^t y$ where x^t is the transpose of the vector x. The 2-norm of a vector x will be denoted by $||x||$. For any vector $x \in R^n$, x_+ is a vector in R^n obtained by setting all the negative components of x to zero. Further we define the step function x_* as: $(x_*)_i = 1$ for $x_i > 0$, $(x_*)_i = 0$ if $x_i < 0$ and $(x_*)_i = 0.5$ when $x_i = 0$. For $x \in R^n$, we denote $\mathrm{diag}(x)$ to be the diagonal matrix of order n whose diagonal elements being the components of the vector x. For any real-matrix $H \in R^{m\times \ell}$ of size $m \times \ell$, its transpose is denoted by H^t. The identity matrix of appropriate size is denoted by I and the column vector of ones of dimension m by e. If f is a real valued function of the variable $x = (x_1, \ldots, x_n)^t \in R^n$, then its gradient is denoted by $\nabla f = (\partial f/\partial x_1, \ldots, \partial f/\partial x_n)^t$ and the Hessian matrix of f by $\nabla^2 f = \left(\partial^2 f/\partial x_i \partial x_j\right)_{i,j=1,\ldots,n}$.

2 Support vector regression (SVR) formulation

In this section, we briefly describe the standard SVR formulation. Consider the set of input examples $\{(x_i, y_i)\}_{i=1,2,\ldots,m}$ where for each input example $x_i \in R^n$, let $y_i \in R$ be its corresponding target output. In SVR, the input examples are mapped into a higher dimensional feature space by a nonlinear mapping $\varphi(.)$ and a linear regression problem is solved in the feature space. The generic SVR estimating function $f(\cdot)$ is assumed to be of the form

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

where w is a vector in the feature space and $b \in R$ is a threshold. Our goal is in determining w and b as the solution of the unconstrained minimization problem of the following form [6, 28]

$$\min_{w,b} \frac{1}{2} w^t w + C \sum_{i=1}^{m} |f(x_i) - y_i|_\varepsilon, \tag{1}$$

where $C > 0, \varepsilon > 0$ are input parameters and the ε-insensitive error loss function is defined by

$$|f(x_i) - y_i|_\varepsilon = \max\{|f(x_i) - y_i| - \varepsilon, 0\}. \tag{2}$$

The ε-insensitive error loss function allows in discarding approximation errors whenever the training samples fall within the ε-tube about the fitting function, which enables representing the decision function using fewer data samples.

By considering the square of 2-norm ε-insensitive error loss with weight $(C/2)$ instead of the usual 1-norm loss with weight C and adding the term $(b^2/2)$ in the definition of the objective function of (1), which was initially suggested for regression problems in [24], the minimization problem (1) is reformulated in primal variables as

$$\min_{w,b} \frac{1}{2}(w^t w + b^2) + \frac{C}{2} \sum_{i=1}^{m} |f(x_i) - y_i|_\varepsilon^2. \tag{3}$$

For the solution of the primal problem (3) using the generalized Hessian matrix and the smoothing procedures, see [1, 16].

3 Extreme learning machine (ELM) method

Let $\{(x_i, y_i)\}_{i=1,\ldots,m}$ be a set of training examples given where for the input example $x_i = (x_{i1}, \ldots, x_{in})^t \in R^n$, let

$y_i \in R$ be its corresponding observed value. 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 weight 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

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

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

$$Hw = y, \tag{4}$$

where

$$H = \begin{bmatrix} G(a_1, b_1, x_1) & \cdots & G(a_\ell, b_\ell, x_1) \\ . & \cdots & . \\ G(a_1, b_1, x_m) & \cdots & G(a_\ell, b_\ell, x_m) \end{bmatrix}_{m\times\ell} \tag{5}$$

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

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

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

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

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

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

For a given SLFN having m number of distinct training examples with the number of hidden nodes ℓ equals to m for which the activation function $g(.)$ in any interval is infinitely differentiable and 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 [15] that with probability one the hidden layer output matrix H of the SLFN defined by (5) is invertible and $||Hw - y|| = 0$.

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 (4). In fact, w is determined to be the minimum norm least squares solution of (4) which can be explicitly obtained by [15]

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

where $H^\dagger$ is the Moore–Penrose generalized inverse [25] of the matrix H. Finally by obtaining the solution $w \in R^\ell$, the regression estimation function $f(.)$ for any input example $x \in R^n$ is determined using

$$f(x) = \sum_{s=1}^{\ell} w_s G(a_s, b_s, x). \tag{6}$$

Note that, 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. Further, 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. It is important to note that ELM works for generalized feedforward networks which may not be neuron alike [11–13]. Finally, as an interesting application of ELM for time series prediction and the problem of simultaneous learning of a function and its derivatives, the interested reader is referred to [2, 26].

4 Proposed ELM for ε-insensitive regression

In this section, we propose the extension of the study of ELM learning approach for SVM applied for classification problems in [14] to regression problems by formulating the problem of regression as an unconstrained optimization problem in the primal variables and solving it by Newton–Armijo algorithm.

Let $\{(x_i, y_i)\}_{i=1,\ldots,m}$ be a set of input examples given where for the input vector $x_i \in R^n$ let its observed value be $y_i \in R$. Then for a given SLFN having ℓ number of hidden nodes, the randomly assigned values for 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 of the sth hidden node $b_s \in R$ with the activation function $g: R \to R$, the ELM learning method determines the unknown output weight vector $w = (w_1, \ldots, w_\ell)^t \in R^\ell$ connecting the hidden nodes to the output node under the minimum norm least squares constraint [15]:

$$\min ||w|| \quad \text{and} \quad \min ||Hw - y||,$$

where the matrix H is given by (5) and $y = (y_1, \ldots, y_m)^t \in R^m$ is the vector of observed values.

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 an unconstrained optimization problem of the following form:

$$\min_{w\in R^\ell} L(w) = \frac{1}{2}w^t w + \frac{C}{2}||Hw-y||_\varepsilon^2, \tag{7}$$

where

$$||Hw-y||_\varepsilon^2 = \sum_{i=1}^m |f(x_i)-y_i|_\varepsilon^2 \tag{8}$$

is such that the error loss $|f(x_i)-y_i|_\varepsilon$ and the regression estimation $f(.)$ are given by (2) and (6), respectively.

Since for any $x \in R$ and $\varepsilon > 0$,

$$|x|_\varepsilon = \max\{0, |x|-\varepsilon\} = (x-\varepsilon)_+ + (-x-\varepsilon)_+$$

is true and further $(x-\varepsilon)_+ \cdot (-x-\varepsilon)_+ = 0$ holds, the following important identity [16] can be easily verified

$$|x|_\varepsilon^2 = (x-\varepsilon)_+^2 + (-x-\varepsilon)_+^2. \tag{9}$$

Now, using the identity (9), we obtain

$$\begin{aligned}|f(x_i)-y_i|_\varepsilon^2 &= |H_i w - y_i|_\varepsilon^2 \\ &= (H_i w - y_i - \varepsilon)_+^2 + (y_i - H_i w - \varepsilon)_+^2,\end{aligned} \tag{10}$$

where H_i is the ith row of the hidden layer output matrix H of the network.

Replacing the square of the ε-insensitive error loss function (8) by (10), the unconstrained optimization problem (7) can be rewritten as

$$\begin{aligned}\min_{w\in R^\ell} L(w) =& \frac{1}{2}w^t w + \frac{C}{2}\sum_{i=1}^m \Big[(H_i w - y_i - \varepsilon)_+^2 \\ &+ (y_i - H_i w - \varepsilon)_+^2\Big].\end{aligned} \tag{11}$$

Now we state the Newton iterative Algorithm with Armijo stepsize [1, 16, 17] starting from any initial point $w^0 \in R^\ell$.

4.1 Newton algorithm with Armijo stepsize

Choose any initial guess $w^0 \in R^\ell$

1. Stop the iteration if $\nabla L(w^i) = 0$
   Else
   Determine the direction vector $d^i \in R^\ell$ as the solution of the following system of linear equations in ℓ variables
   $\nabla^2 L(w^i) d^i = -\nabla L(w^i)$
2. Armijo stepsize: define
   $w^{i+1} = w^i + \lambda_i d^i$

where $\lambda_i = \max\{1, \frac{1}{2}, \frac{1}{4}, \ldots\}$ is the stepsize so that

$$L(w^i) - L(w^i + \lambda_i d^i) \geq -\delta\lambda_i \nabla L(w^i)^t d^i$$

and $\delta \in (0, \frac{1}{2})$.

Note that to apply Newton–Armijo algorithm, both the gradient vector and the Hessian matrix of L need to be known. It can be verified that for any $w \in R^\ell$, the gradient of L is obtained to be of the form [1]

$$\nabla L(w) = w + CH^t[(Hw-y-\varepsilon e)_+ - (y-Hw-\varepsilon e)_+] \tag{12}$$

Since the gradient $\nabla L(.)$ given by (12) is not differentiable, the Hessian matrix of second order partial derivatives of L is not defined in the usual sense. To solve this problem, we employ in this work generalized Hessian matrix approach and smoothing technique that have been extensively used in the literature [1, 9, 16, 17].

By using the results of [1], it can be verified that the generalized Hessian matrix of L in the sense of [10] exists everywhere and is defined to be the following $\ell \times \ell$ matrix

$$\begin{aligned}\nabla^2 L(w) = I + CH^t \mathrm{diag}((Hw-y-\varepsilon e)_* \\ + (y-Hw-\varepsilon e)_*)H,\end{aligned} \tag{13}$$

where diag(.) is a diagonal matrix of order m.

Now, using (12), (13) and applying Newton–Armijo iterative algorithm, we can solve the ELM for regression (ELM-R) problem (11). It can be shown that [1] for any starting vector w^0 in R^ℓ, the sequence $\{w^i\}$ obtained using the Newton–Armijo algorithm described above converges globally and terminates at the global minimum in a finite number of iterations.

For our smooth approach, the plus function appearing in the definition of the objective function of (11) is replaced by an approximate function of infinite order of differentiability and the reformulated problem is then solved by Newton–Armijo algorithm.

For this, consider the accurate smooth approximation $p(x, \alpha)$ of the plus function x_+ defined by [16, 17]

$$p(x,\alpha) = x + \frac{1}{\alpha}\log(1+\exp(-\alpha x)), \tag{14}$$

where $\alpha > 0$ is a parameter.

By defining the vector $p(Hw - y - \varepsilon e, \alpha)$ in R^m whose ith component is given by

$$p(Hw-y-\varepsilon e, \alpha)_i = p(H_i w - y_i - \varepsilon, \alpha) \tag{15}$$

and replacing the plus function in (11) by its smooth approximation (15) with parameter $\alpha > 0$, we can obtain the smooth reformulation of (11) as an unconstrained minimization problem of the form:

$$\begin{aligned}\min_{w\in R^\ell} L(w) = \frac{1}{2}w^t w + \frac{C}{2}(||p(Hw-y-\varepsilon e,\alpha)||^2 \\ + ||p(y-Hw-\varepsilon e,\alpha)||^2).\end{aligned} \tag{16}$$

Since $p(., \alpha)$ is infinitely differentiable, the gradient vector and the Hessian matrix of the modified objective function $L(.)$ exist and they can be easily derived.

Now, we solve the smooth ELM for regression (SELM-R) problem (16) using Newton–Armijo algorithm. Following the proof of [16], it can be shown that for any starting vector w^0 in R^ℓ, the sequence $\{w^i\}$ obtained using the Newton–Armijo algorithm described above converges to the unique solution of (16) globally and quadratically. Further, when the smoothing parameter $\alpha > 0$ converges to infinity, the unique solution of (16) converges to the unique solution of the original minimization problem (11) [16].

Note that Armijo stepsize is used to guarantee, theoretically, the global convergence of the Newton–Armijo algorithm. However, in this work both ELM-R and SELM-R defined by (11) and (16), respectively, are solved numerically using Newton method [9, 19], that is, the unknown w^{i+1} at the $(i+1)$th iteration is determined using the current ith iterate w^i, given by

$$\nabla^2 L(w^i)(w^{i+1} - w^i) = -\nabla L(w^i) \quad \text{for} \quad i = 0, 1, 2, \ldots$$

5 Numerical experiments and comparison of results

We demonstrate the effectiveness of the proposed method by performing numerical experiments on few synthetic datasets and also on a number of interesting real-world datasets namely Citigroup, Microsoft, Google, IBM, Redhat, S&P 500, Intel, Lorenz, Mackey–Glass time series datasets and bodyfat dataset, and comparing their results with ELM and SVR. Numerical experiments were performed without Armijo stepsize. They were carried out in MATLAB R2008b running on Windows 7 with 3 GHz Intel Core 2 Duo, 64 bit having 8 GB RAM.

In all our experiments, the value of ε and the smoothing parameter value α in (14) are taken to be 0.01 and 5, respectively. For an observed vector y and its corresponding prediction vector $\tilde{y}$, the relative error RE is calculated using the following formula

$$RE = \frac{||y - \tilde{y}||}{||y||}.$$

For the proposed method and ELM, we have chosen

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

and

$$G(a, b, x) = \exp\left(-b||x - a||^2\right)$$

of additive and RBF hidden nodes, respectively. The input weights and hidden biases are chosen randomly with uniform distribution from the interval [0, 1]. The values of the parameters C and ℓ were taken, respectively, from the sets: $\{10^1, \ldots, 10^6\}$ and {5, 10, 20, 50, 100, 200, 500, 1,000, 2,000} for both the synthetic and real-world datasets. After extensive simulations, it was found that, in general, for large values of C and ℓ good generalization performance was achieved. This behavior was shown in Fig. 1 for the synthetic dataset "Function 1" by plotting the test accuracy for each (C, ℓ). Further it was observed that, in general, from $\ell = 200$ onwards, "improved" performance was obtained and keeping in mind the computational complexity associated with large value of ℓ, we set $\ell = 200$ in all simulations. The optimal C was obtained by performing tenfold cross-validation. Similarly, the optimal value of ℓ is

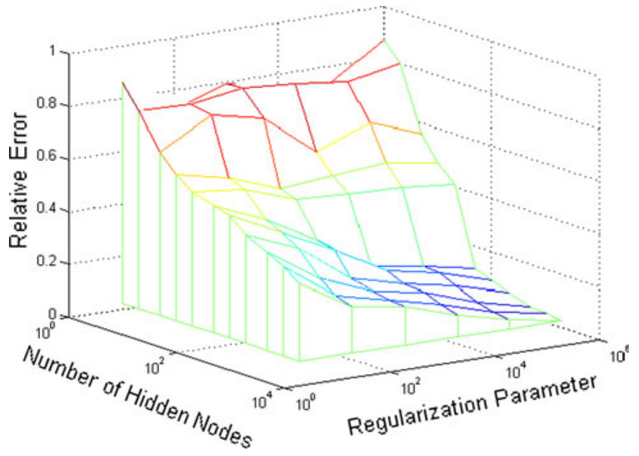

Fig. 1 Test accuracy plot, showing. (i) Insensitivity performance to the parameters C and ℓ. (ii) Good generalization performance for large values of C and ℓ, when ELM-R with RBF was applied on "Function 1" dataset. Additive noise $N(0, 0.02)$ was assumed

Table 1 Functions used for generating synthetic datasets

Name	Function	Domain of definition
Function 1	$y = \frac{4}{\lvert x\rvert+2} + \cos(2x) + \sin(3x)$	$x \in [-6, 6]$
Function 2	$y = 0.79 + 1.27x_1x_2 + 1.56x_1x_4 + 3.42x_2x_5 + 2.06x_3x_4x_5$	$x_i \in [0, 1],\ 1 \leq i \leq 5$
Function 3	$y = \frac{40\exp\left(8\left\{(x_1-0.5)^2+(x_2-0.5)^2\right\}\right)}{\exp\left(8\left\{(x_1-0.2)^2+(x_2-0.7)^2\right\}\right)+\exp\left(8\left\{(x_1-0.7)^2+(x_2-0.2)^2\right\}\right)}$	$x_1, x_2 \in [0, 1]$
Function 4	$y = \frac{1+\sin(2x_1+3x_2)}{3.5+\sin(x_1-x_2)}$	$x_1, x_2 \in [-2, 2]$
Function 5	$y = 1.3356\left(\exp(3(x_2 - 0.5))\sin\left(4\pi(x_2 - 0.9)^2\right) + 1.5(1 - x_1) + \exp(2x_1 - 1)\sin\left(3\pi(x_1 - 0.6)^2\right)\right)$	$x_1, x_2 \in [-0.5, 0.5]$

determined for ELM. With this optimal value, the RE on the test data is calculated.

The standard SVR was solved by LIBSVM [4]. In this case, the Gaussian kernel defined by: for $x_1, x_2 \in R^n$,

$$k(x_1, x_2) = \exp\left(-\mu\|x_1 - x_2\|^2\right)$$

is used where $\mu > 0$ is a parameter. The best prediction performance on the training sets is obtained by varying the regularization parameter C and the kernel parameter μ from the sets $\{10^{-5}, \ldots, 10^{5}\}$ and $\{2^{-10}, \ldots, 2^{10}\}$, respectively, using tenfold cross-validation.

By considering the same training and test samples, the performance of the proposed method is compared with ELM and SVR. The average results for ELM-R, SELM-R and ELM are obtained over 10 trials.

5.1 Synthetic datasets

To verify the performance of ELM-R and SELM-R, we considered 5 synthetic datasets that were obtained according to the list of functions given in Table 1. In these experiments, the observed values y_i for the randomly generated training samples x_i with uniform distribution were taken as:

$$y_i = f(x_i) + \zeta_i$$

where ζ_i is a uniform additive noise from [−0.2, 0.2] and $i = 1, 2, \ldots, m$. Using the procedure explained above, the optimal parameter values were determined. Assuming these values for training, the REs on the test data were calculated. The results obtained by the proposed method, ELM and SVR are summarized in Table 2.

To analyze the sensitivity of SVR, ELM, ELM-R and SELM-R for different level of disturbances, we considered again the synthetic dataset "Function 1" as an example by polluting the observed values with Gaussian additive noises: N(0, 0.02), N(0, 0.05), N(0, 0.1), N(0, 0.2) and N(0, 0.5) where $N(\mu, \sigma)$ denotes the Gaussian distribution with mean μ and SD σ. The performance of ELM-R with RBF hidden nodes for different values of ℓ and C is shown in Fig. 2a and b, respectively. Also, the results for ELM and SVR are illustrated in Fig. 2c and d, respectively. From the figures, it can be observed that the generalization performance is not sensitive to different level of noises. This behavior was observed, in general, for SVR, ELM, ELM-R and SELM-R on all the datasets considered.

5.2 Real-world benchmark datasets

In all the experiments considered in this sub-section, each attribute of the original data is normalized as follows:

Table 2 Performance comparison of ELM-R and SELM-R with ELM having additive and RBF hidden nodes, and with SVR using Gaussian kernel on synthetic datasets

Dataset (train size, test size)	LIBSVM	ELM		ELM-R		SELM-R	
	RE ± std (train time)	Additive RE ± std (train time)	RBF RE ± std (train time)	Additive RE ± std (train time)	RBF RE ± std (train time)	Additive RE ± std (train time)	RBF RE ± std (train time)
Function 1	0.0289 ± 0.0199	**0.0211** ± 0.0293	0.0266 ± 0.0371	0.4746 ± 0.6642	0.0757 ± 0.1069	0.4760 ± 0.6694	0.0803 ± 0.1124
(500 × 1, 1,500 × 1)	(56.0342)	(0.2995)	(0.2699)	(0.4493)	(0.4930)	(0.5366)	(0.6770)
Function 2	0.0174 ± 0.0327	0.0176 ± 0.0499	0.0189 ± 0.0535	**0.0138** ± 0.0393	0.0169 ± 0.048	0.0146 ± 0.0417	0.0157 ± 0.0445
(500 × 5, 1,500 × 5)	(2.429)	(0.06)	(0.108)	(0.5273)	(0.532)	(0.9656)	(0.9251)
Function 3	**0.0121** ± 0.039	0.0265 ± 0.1385	0.0171 ± 0.0902	0.195 ± 1.0277	0.0887 ± 0.4699	0.1939 ± 1.022	0.087 ± 0.4588
(500 × 2, 1,500 × 2)	(2.688)	(4.62)	(2.3)	(0.4914)	(0.4992)	(0.6084)	(0.638)
Function 4	0.149 ± 0.0335	**0.1325** ± 0.0487	0.1899 ± 0.0696	0.178 ± 0.0657	0.2204 ± 0.0814	0.174 ± 0.0632	0.2177 ± 0.0795
(500 × 2, 1,500 × 2)	(0.0716)	(0.075)	(0.093)	(0.4774)	(0.5117)	(0.858)	(0.9001)
Function 5	**0.0352** ± 0.0494	0.0641 ± 0.1447	0.0718 ± 0.1622	0.1746 ± 0.3926	0.1199 ± 0.27	0.1753 ± 0.3964	0.1203 ± 0.2717
(500 × 2, 1,500 × 2)	(0.1125)	(4.628)	(0.201)	(0.4836)	(0.5023)	(0.6349)	(0.6848)

For the optimal values of μ, C and $\ell = 200$, the training time and test accuracy are shown. The best result is indicated in boldface

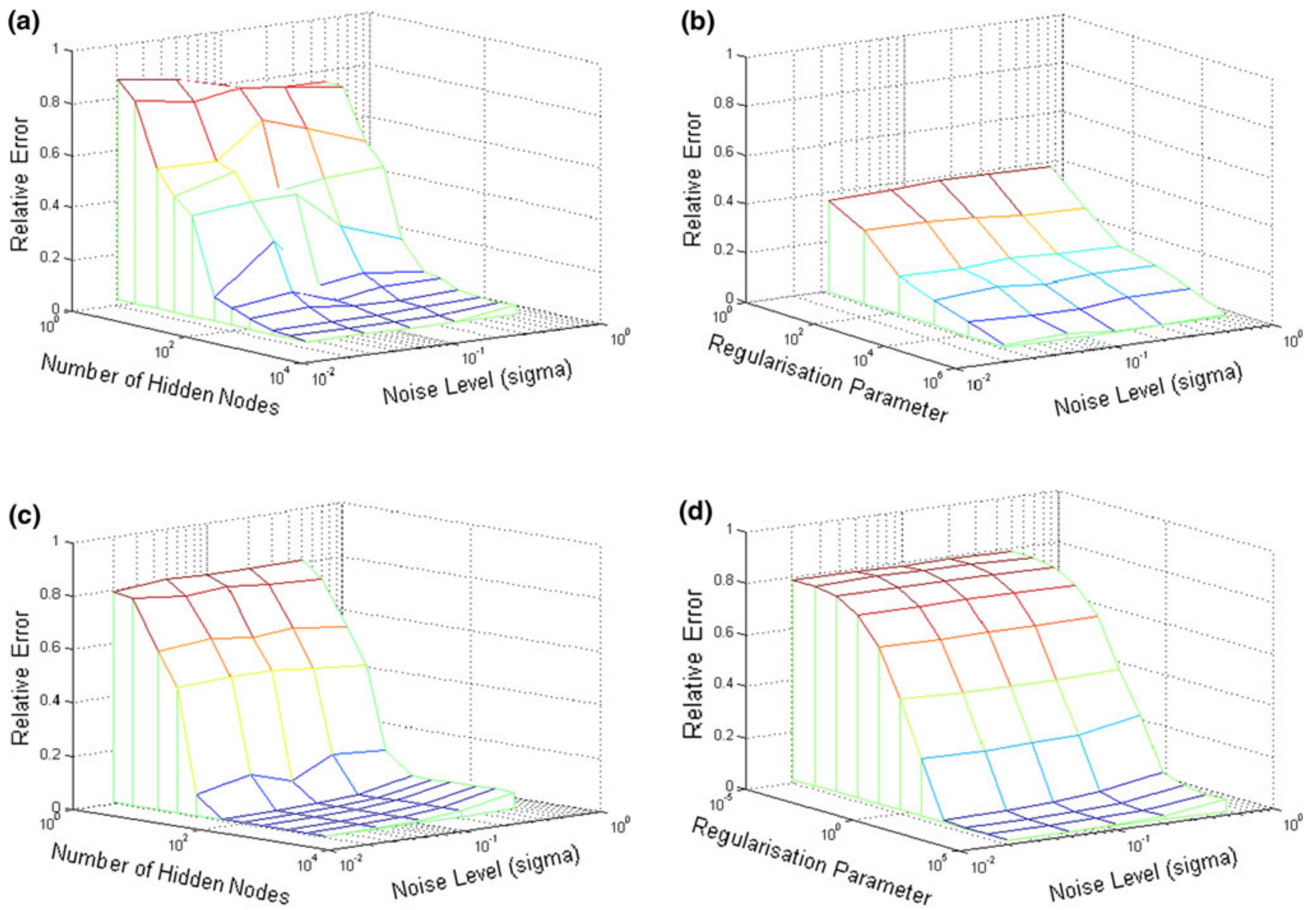

Fig. 2 Insensitivity performances of ELM-R, ELM with RBF hidden nodes and SVR for different level of additive Gaussian noise disturbances. **a** ELM-R with $C = 1{,}000{,}000$. **b** ELM-R with $\ell = 200$. **c** ELM. **d** SVR with $\mu = 0.25$

$$\bar{x}_{ij} = \frac{x_{ij} - x_j^{\min}}{x_j^{\max} - x_j^{\min}}$$

where x_{ij} is the (i, j)th element of the input matrix A, $\bar{x}_{ij}$ is its corresponding normalized value and $x_j^{\min} = \min_{i=1}^{m}(x_{ij})$ and $x_j^{\max} = \max_{i=1}^{m}(x_{ij})$ denote the minimum and maximum values, respectively, of the jth column of A.

The Citigroup, Microsoft, Google, IBM, Red Hat, S&P 500 and Intel are financial datasets of stock index taken from the website: http://www.dailyfinance.com. In all the financial time series examples, we considered 755 closing prices starting from 01-01-2006 to 31-12-2008. Five previous closing prices are used to predict the current closing price. For Citigroup, Microsoft and Intel datasets, the first 200, 380 and 600 samples are taken, respectively, for training and the remaining samples for testing. However, for Google, IBM, Red Hat and S&P 500 datasets, the first 500 samples are used for training and the remaining 250 samples for testing. By choosing the optimal parameter values using tenfold cross-validation, the RE on the test data is determined.

By 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$_{0.20}$ are generated as time series values associated with the variable x of the Lorenz differential equation [22] given by

$$\dot{x} = \rho(y - x), \quad \dot{y} = rx - y - xz \quad \text{and} \quad \dot{z} = xy - bz,$$

using fourth-order Runge–Kutta method. To avoid the initial transients, the first 1,000 values were discarded. We considered the next 3,000 values for our experiment. Five previous values are used to predict the current value. As the first 500 samples are taken for training, we get the remaining 2,495 samples for testing.

Consider the Mackey–Glass time delay differential equation [3, 22] given by

$$\frac{\partial x(t)}{\partial t} = -bx(t) + a\frac{x(t-\tau)}{1 + x(t-\tau)^{10}},$$

Table 3 Comparison of ELM-R and SELM-R with ELM having additive and RBF hidden nodes, and with SVR using Gaussian kernel

Dataset (train size, test size)	LIBSVM RE ± std (train time)	ELM		ELM-R		SELM-R	
		Additive RE ± std (train time)	RBF RE ± std (train time)	Additive RE ± std (train time)	RBF RE ± std (train time)	Additive RE ± std (train time)	RBF RE ± std (train time)
Citigroup	**0.0631** ± 0.0403	0.1836 ± 0.1761	0.1961 ± 0.1895	0.0856 ± 0.0844	0.1031 ± 0.0989	0.1191 ± 0.1137	0.1191 ± 0.1137
(200 × 5,550 × 5)	(0.028)	(0.261)	(0.38)	(0.1607)	(0.17)	(0.1513)	(0.1513)
Microsoft	**0.1701** ± 0.1172	0.2231 ± 0.2225	0.3388 ± 0.3346	0.1827 ± 0.1825	0.281 ± 0.2757	0.1958 ± 0.1955	0.1958 ± 0.1955
(380 × 5,370 × 5)	(0.457)	(0.203)	(0.283)	(0.3697)	(0.4025)	(0.4493)	(0.4493)
Google	**0.1587** ± 0.1078	0.3797 ± 0.1992	0.2419 ± 0.1295	0.1911 ± 0.1878	0.2289 ± 0.2278	0.1954 ± 0.1916	0.1954 ± 0.1916
(500 × 5,250 × 5)	(1.192)	(0.045)	(0.043)	(0.5897)	(0.6256)	(0.9734)	(0.9734)
IBM	0.1381 ± 0.0867	**0.1136** ± 0.1076	0.1809 ± 0.1741	0.1987 ± 0.188	0.1608 ± 0.1578	0.1939 ± 0.1846	0.1939 ± 0.1846
(500 × 5,250 × 5)	(0.067)	(0.044)	(0.067)	(0.5866)	(0.5928)	(0.7847)	(0.7847)
Red hat	0.1339 ± 0.0856	**0.0441** ± 0.0441	0.0985 ± 0.0979	0.1686 ± 0.1683	0.1544 ± 0.1537	0.2405 ± 0.2328	0.2405 ± 0.2328
(500 × 5,250 × 5)	(0.043)	(0.174)	(0.257)	(0.5834)	(0.5803)	(0.5476)	(0.5476)
S&P 500	0.1465 ± 0.1005	**0.0166** ± 0.0165	0.0169 ± 0.0168	0.2038 ± 0.1911 (0.5569)	0.1495 ± 0.1457 (0.61)	0.2149 ± 0.2007 (0.6942)	0.2149 ± 0.2007
(500 × 5,250 × 5)	(7.266)	(4.303)	(5.666)				(0.6942)
Intel (600 × 5,150 × 5)	**0.1617** ± 0.1058 (0.054)	0.2099 ± 0.2079 (0.26)	0.3611 ± 0.3545 (0.371)	0.1679 ± 0.1679 (0.7379)	0.1684 ± 0.1683 (0.7753)	0.1679 ± 0.1678 (0.989)	0.1679 ± 0.1678 (0.989)
Lorenz$_{0.05}$	0.1583 ± 0.1507	0.2041 ± 0.2	0.1969 ± 0.1922	0.0701 ± 0.0699	**0.0368** ± 0.0364	0.0537 ± 0.0537	0.0537 ± 0.0537
(500 × 5, 2,495 × 5)	(0.108)	(0.269)	(0.372)	(0.5944)	(0.6443)	(1.0187)	(1.0187)
Lorenz$_{0.2}$	0.1677 ± 0.1562	0.1838 ± 0.1832	0.1772 ± 0.1769	0.0231 ± 0.0223	0.0189 ± 0.0185	**0.0154** ± 0.0153	**0.0154** ± 0.0153
(500 × 5, 2,495 × 5)	(0.019)	(0.312)	(0.47)	(0.4399)	(0.4446)	(1.0327)	(1.0327)
MG$_{17}$ (500 × 5,995 × 5)	0.0135 ± 0.008	0.0131 ± 0.0129	0.0126 ± 0.0125	0.0164 ± 0.0163	0.027 ± 0.0268	**0.0127** ± 0.0126	**0.0127** ± 0.0126
	(0.101)	(0.603)	(0.709)	(0.4399)	(0.3619)	(0.7706)	(0.7706)
MG$_{30}$ (500 × 5,995 × 5)	**0.086** ± 0.0555 (2406.829)	0.1649 ± 0.1648 (0.17)	0.1141 ± 0.114 (0.736)	0.1398 ± 0.1398 (0.5554)	0.1275 ± 0.1274 (0.6068)	0.1297 ± 0.1297 (0.9516)	0.1297 ± 0.1297 (0.9516)
Bodyfat	**0.0957** ± 0.0252	0.1838 ± 0.1834	0.1858 ± 0.185	0.2322 ± 0.1341	0.1423 ± 0.0502	0.2159 ± 0.1261	0.2159 ± 0.1261
(200 × 14, 52 × 14)	(0.008)	(0.278)	(0.377)	(0.2356)	(0.259)	(0.351)	(0.351)

The training time and test accuracy are shown for the optimal values of μ and C for SVR, the optimal value of ℓ for ELM, and the optimal value of C with $\ell = 200$ for ELM-R and SELM-R. The best result is indicated in boldface where a, b are parameters and τ is the time delay. We performed our experiments on two time series generated by the above differential equation using the parameter values $a = 0.2,\ b = 0.1$ and $\tau = 17,\ 30$. We denote the time series corresponding to $\tau = 17$ and $\tau = 30$ by MG_{17} and MG_{30}, respectively. They are taken from http://www.cse.ogi.edu/~ericwan. Five previous values are used to predict the current value. The first 500 number of samples are considered for training and the remaining 995 for testing.

Bodyfat is a well-known dataset and is taken from the Statlib collection: http://lib.stat.cmu.edu/datasets. It consists of 252 samples with 14 number of attributes. The first 200 samples are taken for training and the remaining for testing.

For the real-world datasets considered, the optimal parameter values were determined using tenfold cross-validation on the training set, and by choosing them, the RE error was calculated on the test data. The number of training and test samples, attributes, the training time and the REs obtained using ELM-R, SELM-R, ELM and SVR are summarized in Table 3. From Table 3, we can conclude that both ELM-R and SELM-R show competitive generalization performance in comparable computational time over ELM and SVR.

6 Conclusion

In this work, we proposed the study of ELM for ε-insensitive regression formulated in 2-norm as an unconstrained optimization problem in primal variables and obtained its solution using fast, Newton–Armijo algorithm. Our method does not need any optimization packages. Similar or better generalization performance of the proposed method in comparable computational time on synthetic and real-world datasets in comparison with ELM and SVR clearly shows its efficiency and practical use. Future work will be on the study of ELM for ε-insensitive regression formulation in dual variables and its applications.

Acknowledgments The authors would like to thank the referees for their valuable comments that greatly improved the earlier version of the paper. Mr. Kapil acknowledges the financial support given as scholarship by Council of Scientific and Industrial Research, India.

References

1. Balasundaram S, Singh R (2010) On finite Newton method for support vector regression. Neural Comput Appl 19:967–977
2. Balasundaram S, Kapil (2011) Application of error minimized extreme learning machine for simultaneous learning of a function and its derivatives. Neurocomputing 74:2511–2519
3. Casdagli M (1989) Nonlinear prediction of chaotic time series. Physica D 35:335–356
4. Chang C-C, Lin C-J LIBSVM: a library for support vector machines. http://www.csie.ntu.edu.tw/~cjlin/libsvm
5. Chen S, Wang M (2005) Seeking multi-threshold directly from support vectors for image segmentation. Neurocomputing 67:335–344
6. Cristianini N, Shawe-Taylor J (2000) An introduction to support vector machines and other kernel based learning methods. Cambridge University Press, Cambridge
7. 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
8. 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
9. Fung G, Mangasarian OL (2003) Finite Newton method for Lagrangian support vector machine. Neurocomputing 55:39–55
10. Hiriart-Urruty J-B, Strodiot JJ, Nguyen H (1984) Generalized Hessian matrix and second order optimality conditions for problems with C^{L1} data. Appl Math Optim 11:43–56
11. Huang G-B, Chen L (2007) Convex incremental extreme learning machine. Neurocomputing 70:3056–3062
12. Huang G-B, Chen L (2008) Enhanced random search based incremental extreme learning machine. Neurocomputing 71:3460–3468
13. 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
14. Huang G-B, Ding X, Zhou H (2010) Optimization method based extreme learning machine for classification. Neurocomputing 74:155–163
15. Huang G-B, Zhu Q-Y, Siew C-K (2006) Extreme learning machine: theory and applications. Neurocomputing 70:489–501
16. Lee YJ, Hsieh W-F, Huang C-M (2005) ε-SSVR: a smooth support vector machine for ε-insensitive regression. IEEE Trans Knowl Data Eng 17(5):678–685
17. Lee YJ, Mangasarian OL (2001) SSVM: a smooth support vector machine for classification. Comput Optim Appl 20(1):5–22
18. Liu Q, He Q, Shi Z (2008) Extreme support vector machine classifier. Lect Notes Comput Sci 5012:222–233
19. Mangasarian OL (2002) A finite Newton method for classification. Optim Methods Softw 17:913–929
20. Mangasarian OL, Musicant DR (2001) Lagrangian support vector machines. J Mach Learn Res 1:161–177
21. Mangasarian OL, Musicant DR (2000) Active set support vector machine classification. In: Lee TK, Dietterich TG, Tesp V (eds) Advances in neural information processing systems, vol 13, pp 577–586
22. 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. Proceedings of IEEE signal processing society workshop, Amelia Island, pp 511–520
23. 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
24. Musicant DR, Feinberg A (2004) Active set support vector regression. IEEE Trans Neural Netw 15(2):268–275
25. Rao CR, Mitra SK (1971) Generalized inverse of matrices and its applications. Wiley, New York
26. Singh R, Balasundaram S (2007) Application of extreme learning machine for time series analysis. Int J Intell Technol 2:256–262
27. Tay FEH, Cao LJ (2001) Application of support vector machines in financial time series with forecasting. Omega 29(4):309–317
28. Vapnik VN (2000) The nature of statistical learning theory, 2nd edn. Springer, New York