Neural Comput & Applic (2013) 22:551–558
DOI 10.1007/s00521-012-0859-8

Symmetric extreme learning machine

Xueyi Liu · Ping Li · Chuanhou Gao

Received: 9 August 2011 / Accepted: 21 January 2012 / Published online: 23 February 2012

Abstract Extreme learning machine (ELM) can be considered as a black-box modeling approach that seeks a model representation extracted from the training data. In this paper, a modified ELM algorithm, called symmetric ELM (S-ELM), is proposed by incorporating a priori information of symmetry. S-ELM is realized by transforming the original activation function of hidden neurons into a symmetric one with respect to the input variables of the samples. In theory, S-ELM can approximate N arbitrary distinct samples with zero error. Simulation results show that, in the applications where there exists the prior knowledge of symmetry, S-ELM can obtain better generalization performance, faster learning speed, and more compact network architecture.

Keywords Extreme learning machine · Symmetry · Generalization performance · Black-box model · Chaotic time series

1 Introduction

Extreme learning machine (ELM) [1, 2] is a recent approach for training single hidden layer feedforward networks (SLFNs). In ELM algorithm, the input weights and biases of hidden nodes are randomly generated, and then the weights of the output layer are obtained using a least squares method based on calculating the Moore–Penrose generalized inverse [3]. Therein, no iteration is needed and the computational cost is much lower than when using other traditional methods, such as back-propagation algorithm (BP) [4] or support vector machine (SVM) [5]. Meanwhile, ELM can also obtain similar generalization ability with SVM [1]. Recently, its application to different fields is becoming more and more common [6, 7].

However, ELM constitutes a black-box approach that seeks a model representation extracted from the training samples. A full back-box model is appropriate when there is no prior knowledge underlying the samples [8]. Nevertheless, if there exists some type of prior knowledge about the system to be modeled, it should be incorporated into the ELM modeling process. For example, many real-life phenomena exhibit inherent symmetry properties. But ELM can hardly learn this symmetry exactly especially from noisy data, although it has universal approximation ability [9].

This paper tries to incorporate symmetry into the ELM model. A few works have discussed the possible means to impose symmetry in different black-box models, such as radial basis function (RBF) networks [10–12], least squares SVMs (LS-SVM) [13], and k-nearest neighbor method [14]. Aguirre et al. [10] and Chen et al. [11, 12] modify the model structure to guarantee the symmetry of the network, while Espinoza et al. [13] impose symmetric constraints in the LS-SVM algorithm, and McNames et al. [14] introduce additional virtual samples according to the symmetry of the problem. It is worth noticing that how to incorporate the prior knowledge of symmetry into a black-box model is highly problem-dependent. None of the methods above is

X. Liu · P. Li (✉)
School of Aeronautics and Astronautics, Zhejiang University, Hangzhou 310027, Zhejiang, China
e-mail: pli@iipc.zju.edu.cn

X. Liu
e-mail: liuxy@cjlu.edu.cn

C. Gao
Department of Mathematics, Zhejiang University, Hangzhou 310027, China
e-mail: gaochou@zju.edu.cn

certainly applicable to other problems. Furthermore, when incorporating the prior knowledge we often hope that little additional computation complexity is involved and little modification on the original algorithm is made. From this point of view, this paper is devoted to presenting a simple yet effective method for incorporating symmetry into ELM. The rest of this paper is organized as follows. In Sect. 2, ELM is reviewed briefly. In Sect. 3, the modified ELM incorporating the prior knowledge of symmetry is proposed. Section 4 is devoted to experiments, followed by the conclusion in Sect. 5.

2 Brief of ELM

ELM presents a novel training solution for SLFNs, which is very efficient and effective. In ELM, parameters are randomly assigned, and the output weights can be analytically determined by the generalized inverse operation.

Given N distinct samples $\{(\mathbf{x}_i, y_i)|i = 1, \ldots, N;\ \mathbf{x}_i \in \mathbb{R}^d;\ y_i \in \mathbb{R}\}$, the output of ELM with L hidden neuron nodes is

$$f(\mathbf{x}_i) = \sum_{j=1}^{L} \beta_j G(\mathbf{a}_j, b_j, \mathbf{x}_i), \quad i = 1, \ldots, N \tag{1}$$

where β_j is the output weight from the jth hidden node to the output node; $\mathbf{a}_j \in \mathbb{R}^d$ and $b_j \in \mathbb{R}$ are the input node parameters generated randomly; $G(\mathbf{a}_j, b_j, \mathbf{x}_i) = g(\mathbf{a}_j \cdot \mathbf{x}_i + b_j)$ is the output of jth hidden node with respect to the input $\mathbf{x}_i$, with $\mathbf{a}_j \cdot \mathbf{x}$ is the inner product of $\mathbf{a}_j$ and $\mathbf{x}$ and $g(\cdot)$ being the activation function of hidden node. The activation functions $g(\cdot)$ can be sigmoid function as well as the radial biases, sine, cosine, exponential, and many other non-regular functions as shown in Huang and Babri [15]. For additive hidden node, $G(\mathbf{a}_j,\ b_j,\ \mathbf{x}_i)$ is given by $G(\mathbf{a}_j, b_j, \mathbf{x}) = g(\mathbf{a}_j^T\mathbf{x} + b_j)$, while given by $G(\mathbf{a}_j, b_j, \mathbf{x}) = g\left(b_j\|\mathbf{x} - \mathbf{a}_j\|\right)$ for RBF hidden node.

Then, training an ELM model is equivalent to obtaining the minimum norm least squares solution of the system of equations (1), which can be given by

$$\boldsymbol{\beta} = \mathbf{H}^{\dagger}\mathbf{Y} \tag{2}$$

where $\mathbf{H} = \begin{bmatrix} G(\mathbf{a}_1, b_1, \mathbf{x}_1) & \cdots & G(\mathbf{a}_L, b_L, \mathbf{x}_1) \\ \vdots & \cdots & \vdots \\ G(\mathbf{a}_1, b_1, \mathbf{x}_N) & & G(\mathbf{a}_L, b_L, \mathbf{x}_N) \end{bmatrix}_{N\times L}$ is the hidden layer output matrix; $\mathbf{Y} = [y_1 \quad y_2 \quad \cdots \quad y_N]^T$; and $\mathbf{H}^{\dagger}$ is the Moore–Penrose generalization inverse of $\mathbf{H}$.

Compared with the conventional popular learning algorithms, ELM offers many significant advantages such as good generalization performance, fast learning speed, and ease of implementation. On the other hand, ELM also has inherent drawbacks such as unsteadiness derived from the randomness of input weights, and incompact network architecture. Moreover, ELM belongs to a black-box-type modeling approach, so it is not trivial to impose a priori knowledge such as periodicity, symmetry, or monotonicity in ELM. As we have known, incorporating a priori knowledge into a black-box modeling method can often lead to better performance. As an important kind of a priori knowledge, symmetry will be incorporated into ELM in this paper.

3 Proposed symmetric ELM (S-ELM)

The inclusion of odd or even symmetry in the ELM learning framework can be realized by modification its structure. Assuming that $G(\mathbf{a}_j,\ b_j,\ \mathbf{x}_i)$ is activation function of the original ELM model, we propose to adopt the following symmetric activation function in S-ELM

$$\begin{aligned}\tilde{G}(\mathbf{x}) &\triangleq G(\mathbf{a}_j, b_j, \mathbf{x}) + \delta G(\mathbf{a}_j, b_j, -\mathbf{x}) \\ &= \begin{cases} g(\mathbf{a}_j^T\mathbf{x} + b_j) + \delta g(-\mathbf{a}_j^T\mathbf{x} + b_j), \textbf{\textit{for additive}}\ g(\cdot), \\ g(\|\mathbf{x} - \mathbf{a}_j\|/b_j) + \delta g(\|\mathbf{x} + \mathbf{a}_j\|/b_j), \textbf{\textit{for RBF}}\ g(\cdot). \end{cases}\end{aligned} \tag{3}$$

where $\delta = \pm 1$. Obviously, $\tilde{G}(\mathbf{x})$ is even if $\delta = +1$, and odd if $\delta = -1$.

Then, the output of S-ELM can be expressed as follows:

$$\begin{aligned} f(\mathbf{x}) &= \sum_{j=1}^{L} \beta_j \tilde{G}(\mathbf{a}_j, b_j, \mathbf{x}) \\ &= \sum_{j=1}^{L} \beta_j (G(\mathbf{a}_j, b_j, \mathbf{x}) + \delta G(\mathbf{a}_j, b_j, -\mathbf{x})) \end{aligned} \tag{4}$$

Here, the output $f(\mathbf{x})$ can always be guaranteed to obey the same odd or even symmetry as the underlying function.

Let $\{(\mathbf{x}_i, y_i)|i = 1, \ldots, N; \mathbf{x}_i \in \mathbb{R}^d; y_i \in \mathbb{R}\}$ be a set of training samples. Solving S-ELM is equivalent to solving the following linear system

$$\tilde{\mathbf{H}}\beta = \mathbf{Y} \tag{5}$$

where $\tilde{\mathbf{H}} = \begin{bmatrix} \tilde{G}(\mathbf{a}_1, b_1, \mathbf{x}_1) & \cdots & \tilde{G}(\mathbf{a}_L, b_L, \mathbf{x}_1) \\ \vdots & \cdots & \vdots \\ \tilde{G}(\mathbf{a}_1, b_1, \mathbf{x}_N) & \cdots & \tilde{G}(\mathbf{a}_L, b_L, \mathbf{x}_N) \end{bmatrix}_{N\times L}$; $\mathbf{Y} = [y_1 \quad y_2 \quad \cdots \quad y_N]^T$; and $\boldsymbol{\beta}$ is the output weight vectors.

The universal approximation ability of the standard ELM with additive or RBF activation function has been proved by Huang et al. [9]. Notably, the symmetric activation function of S-ELM, defined by (3), belongs to neither type mentioned above. Therefore, the approximation ability of S-ELM is discussed firstly.

Theorem 1 *Given a SLFN with N hidden nodes and symmetry activation function* $\tilde{G}(\mathbf{a}_j, b_j, \mathbf{x})$ *defined by (3) where* $g : \mathbb{R} \to \mathbb{R}$ *is infinitely differentiable in any interval, for N arbitrary distinct samples* $\{(\mathbf{x}_i, y_i)|i = 1, \ldots, N; \mathbf{x}_i \in \mathbb{R}^d; y_i \in \mathbb{R}\}$, *for any randomly chosen* $\mathbf{a}_j \in \mathbb{R}^d$ *and* $b_j \in \mathbb{R}$ *from any intervals of* $\mathbb{R}^d$ *and* $\mathbb{R}$, *respectively, according to any continuous probability distribution, the hidden layer output matrix* $\tilde{\mathbf{H}}$ *is full rank with probability one.*

Proof Following the same method of proof by Huang et al. [1], we will prove our result by contradiction.

Assume all the biases $\{b_i\}_{i=1}^N$ are randomly generated from (s, t), any interval of $\mathbb{R}$. Then consider the ith column of $\tilde{\mathbf{H}}$ given by $\boldsymbol{c}(b_i) = [\tilde{G}(\mathbf{a}_i, b_i, \mathbf{x}_1), \tilde{G}(\mathbf{a}_i, b_i, \mathbf{x}_2), \ldots, \tilde{G}(\mathbf{a}_i, b_i, \mathbf{x}_N)]^T$.

Note that $\mathbf{a}_i$ are randomly chosen based on continuous probability distribution, hence we can assume that when $k \neq k'$, $\mathbf{a}_i^T \mathbf{x}_k \neq \mathbf{a}_i^T \mathbf{x}_{k'}$. Suppose that the vector $\boldsymbol{c}(b_i)$ belongs to a subspace of dimension $N - 1$. Then there exists a non-zero vector $\alpha \in \mathbb{R}^N$ which is orthogonal to this subspace

$$\alpha^T(\boldsymbol{c}(b_i) - \boldsymbol{c}(s)) = \alpha_1^T \tilde{G}(\mathbf{a}_i, b_i, \mathbf{x}_1) + \cdots + \alpha_N^T \tilde{G}(\mathbf{a}_i, b_i, \mathbf{x}_N) - \alpha^T \boldsymbol{c}(s) = 0. \quad (6)$$

Without loss of generality, suppose $\alpha_N \neq 0$, then (6) can be reformulated into

$$\tilde{G}(\mathbf{a}_i, b_i, \mathbf{x}_N) = \gamma^T \boldsymbol{c}(s) - \sum_{j=1}^{N-1} \gamma_j^T \tilde{G}(\mathbf{a}_i, b_i, \mathbf{x}_1) \quad (7)$$

where $\gamma_j = \alpha_j/\alpha_N$, $j = 1, \ldots, N$.

Noting that $\tilde{G}(\mathbf{a}_i, b_i, \mathbf{x}) = g(\mathbf{a}_j^T \mathbf{x} + b_j) + \delta g(-\mathbf{a}_j^T \mathbf{x} + b_j)$, or $g(\|\mathbf{x} - \mathbf{a}_j\|/b_j) + \delta g(\|\mathbf{x} + \mathbf{a}_j\|/b_j)$, and $g(\cdot)$ is infinitely differentiable in any interval, $\tilde{G}(\mathbf{a}_i, b_i, \mathbf{x})$ is also infinitely differentiable in any interval. Consequently, (7) implies

$$\left(\tilde{G}(a_i, b_i, x_N)\right)^{(l)} = \sum_{j=1}^{N-1} \gamma_j^T \left(\tilde{G}(a_i, b_i, x_1)\right)^{(l)}, \quad l = 1, \ldots, N, N+1, \ldots \quad (8)$$

where $(\cdot)^{(l)}$ denotes the lth derivative with respect to b_i.

This is a contradiction since there are only $N - 1$ free coefficients for the derived more number of linear equations. Thus, vector $\boldsymbol{c}$ does not belong to any subspace whose dimension is less than N. Hence for any N different values of $\mathbf{a}_i$ and b_i chosen from any intervals of $\mathbb{R}^d$ and $\mathbb{R}$, respectively, according to any continuous probability distribution, then with probability one, the vectors $\boldsymbol{c}(b_1)$, $\boldsymbol{c}(b_2)$, $\ldots$, $\boldsymbol{c}(b_N)$ are linearly independent, and hence $\tilde{\mathbf{H}}$ can be made full rank. □

Furthermore, we have

Theorem 2 *For any* $\varepsilon > 0$ *and activation function* $g : \mathbb{R} \to \mathbb{R}$ *being infinitely differentiable in any interval, there exists* $\tilde{N} \leq N$ *such that* $\|\tilde{\mathbf{H}}\beta - \mathbf{Y}\| < \varepsilon$ *holds for N arbitrary distinct samples* $\{(\mathbf{x}_i, y_i)|i = 1, \ldots, N; \mathbf{x}_i \in \mathbb{R}^d; y_i \in \mathbb{R}\}$, *for any randomly chosen* $\mathbf{a}_j \in \mathbb{R}^d$ *and* $b_j \in \mathbb{R}$ *from any intervals of* $\mathbb{R}^d$ *and* $\mathbb{R}$, *respectively, according to any continuous probability distribution.*

Theorems 1 and 2 tell that with at most N hidden neuron nodes, S-ELM can learn N arbitrary distinct training samples with arbitrary negligible error, just as ELM can do. In spite of this, it is very important to emphasize that S-ELM always use symmetry function to estimate the system to be modeled, even when the underlying system is unsymmetric. Thus, although S-ELM has the ability to approximate N arbitrary distinct data points with zero error, it is not a universal approximator. The first example in Sect. 4 later gives a more intuitive illustration to this point.

The output of S-ELM is unstable due to the randomness of the input weights and biases of the hidden nodes, because of which the simulation results of S-ELM, together with ELM, are given by the mean results of fifty trials. It should also be mentioned that since the activation function of the hidden layer of ELM can include sigmoid functions as well as all others which is infinitely differentiable in any interval [15], many different kinds of symmetric ones of S-ELM can be established based on them using (3), which will be further discussed later in the next section.

4 Simulation results

To evaluate the proposed S-ELM algorithm, we compare it with the ELM algorithm in different kinds of cases. The sigmoid function $g(x) = 1/(1 + e^{-x})$ is adopted as activation function in ELM and is also employed to establish the symmetric one in S-ELM by (3). The numbers of hidden neurons of ELM and S-ELM are selected based on cross-validation together with grid search method. ELM source code can be downloaded from [16]. In order to reduce the disturbance from randomness, all the experimental results are the mean values of 50 trails. All the simulations are carried out in MATLAB7.5 environment running in a desktop PC with a 2.01 GHz AMD Athlon (tm) 64 × 2 Dual Core 3600 + Processor and a 3.00 GB memory.

Firstly, the scope of application of S-ELM is discussed on a simple example. Consider the problem of approximating a piecewise function composed of two parts: $y = x^2$, for $-2.5 \leq x \leq 0$; $y = 0.5x$, for $0 < x \leq 2.5$, which is obviously not symmetric. Six data points are taken as the training set in $[-2.5, 2.5]$, as shown in Fig. 1a. We apply ELM, even S-ELM, and odd S-ELM to learning

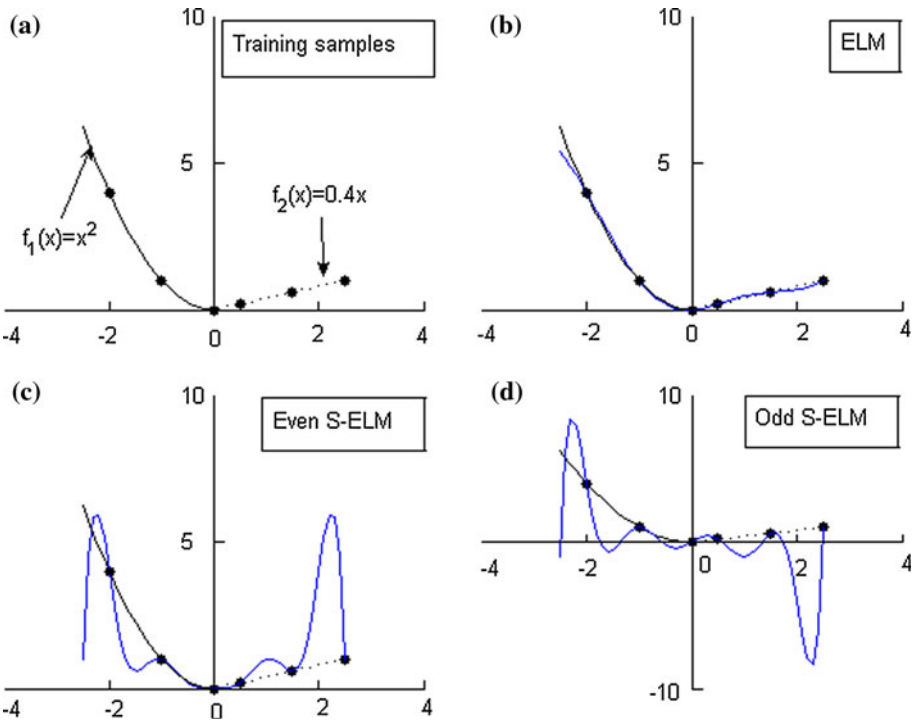

Fig. 1 Six training samples and the sampling function are shown in (**a**). The approximation results of ELM, even S-ELM, and odd S-ELM are presented in (**b**, **c**), and (**d**), respectively. The numbers of hidden nodes of all the three models are six

these training data, respectively. The numbers of hidden nodes of the three models are all 6. Figure 1b, c, and d gives the corresponding results. It is clear that all the three methods can learn the training samples exactly. However, compared with ELM, even and odd S-ELM will generalize worse in such unsymmetric case. Consequently, although S-ELM has the capability to approximate N arbitrary distinct samples exactly, it is only applicable to the cases where the prior knowledge of symmetry exists. The symmetric activation function makes S-ELM not to be a universal approximator any more, which is an important difference from ELM.

Secondly, the performance of S-ELM is evaluated on two toy function approximation problems, which are odd and even, respectively. The first problem is to approximate the sinc function. The training and testing samples $\{(x_i, y_i)\}$ are uniformly randomly generated with x_i drawn from the interval $(-10, 10)$, and $y_i = \sin(x_i)/x_i + e$ where e is the random noise uniformly distributed in $[-0.2, 0.2]$ for training samples, and zero for testing samples. For the purpose of exploring the performance of S-ELM under different level of training data size, 3 groups of training samples with the size being, namely 1,000, 5,000, 10,000, have been tried. The testing samples always have the same size as the training ones in all simulations here. The corresponding results (including training time, training and testing root mean squared errors (RMSE), and the number of hidden neuron nodes (#Nodes)) are shown in Table 1, in which the bold values indicate the best performance (the same as in all tables). To make the comparison more convincing, two other algorithms are also compared with S-ELM: ELM-V, which generates virtual training samples according to the symmetry attribute and trains the ELM model with the new training set $\{(\mathbf{x}_i, y_i)\}_{i=1}^{N} \cup \{(-\mathbf{x}_i, \delta y_i)\}_{i=1}^{N}$, and regularized ELM (R-ELM).

A look at Table 1 might suggest the following information. (a) S-ELM and ELM-V achieve the very similar generalization performance for all three different training set sizes, and both perform better than ELM and R-ELM. (b) S-ELM obtains the most compact network architecture among all the algorithms. ELM and ELM-V tend to need much more hidden nodes than S-ELM for yielding good performance, which will lead to the longer response time to new external unknown stimuli in real applications. (c) Finally, as far as the training time is concerned, S-ELM also outperforms all other algorithms thanks to the particular structure of the activation function, while ELM-V is most time-consuming due to the introduction of additional virtual training samples. It should also be mentioned that, because of the involving of calculating the inverse of a matrix of much lower dimension, the training time of R-ELM is also much shorter than that of ELM, but is still longer than those of S-ELM.

The second problem, taken from Chen [12], is to approximate the odd function defined by

$$f(x,y) = 10\left(\frac{\sin(x-5)\sin(y-5)}{(x-5)(y-5)} - \frac{\sin(x+5)\sin(y+5)}{(x+5)(y+5)}\right) \tag{9}$$

where $x, y \in [-5\pi, 5\pi]$. Figure 2a shows this function using a uniform grid of 100*100 points which are taken as the

Table 1 Performance comparison between S-ELM, ELM, ELM-V, and R-ELM on sinc case

Case	Algorithm	Training time (s)	Training		Testing		#Nodes
			RMSE	Dev	RMSE	Dev	
1,000	ELM	0.0056	0.1152	0.0015	0.0144	0.0026	14
	S-ELM	**0.0037**	0.1149	0.0017	**0.0103**	0.0024	**8**
	ELM-V	0.0121	0.1151	0.0010	0.0105	0.0025	15
	R-ELM	0.0042	0.1146	0.0016	0.0140	0.0027	16
5,000	ELM	0.0401	0.1153	0.0007	0.0069	0.0012	17
	S-ELM	**0.0239**	0.1154	0.0007	**0.0049**	0.0012	**9**
	ELM-V	0.1016	0.1154	0.0005	**0.0049**	0.0009	18
	R-ELM	0.0253	0.1152	0.0007	0.0066	0.0012	20
10,000	ELM	0.0869	0.1153	0.0004	0.0048	0.0008	18
	S-ELM	**0.0469**	0.1154	0.0005	**0.0035**	0.0007	**9**
	ELM-V	0.2083	0.1154	0.0004	**0.0035**	0.0007	19
	R-ELM	0.0638	0.1154	0.0005	0.0046	0.0008	25

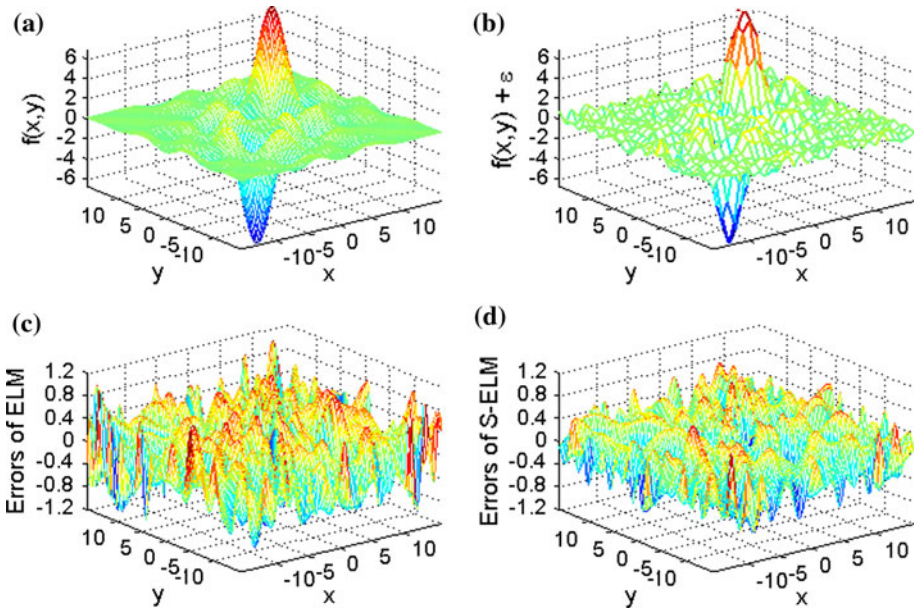

Fig. 2 **a** Odd function to be approximated; **b** noisy training data points; **c** approximation errors of ELM; **d** approximation errors of S-ELM

testing dataset. The training set contains 31*31 data points, where the Gaussian noise of zero mean and standard deviation 0.4 has been added, as shown in Fig. 2b. Figure 2c and d shows the approximation errors of the two algorithms over the grid of 100*100 points. The errors of S-ELM are much smaller than those of ELM. From Table 2, we can see that, besides the testing RMSE, both the training time and the number of hidden neuron nodes are also much smaller, compared with ELM.

Thirdly, the performance of S-ELM with different activation function is evaluated using the above two cases. Another symmetric activation function, $\tilde{G}(\mathbf{x}) = \sin(\mathbf{a}_j \cdot \mathbf{x} + (\delta + 1)\pi/4)$, is considered instead. It is obvious that $\tilde{G}(\mathbf{x})$ is even when $\delta = +1$, and odd when $\delta = -1$. The simulation results are given in Table 3, from which we can find that ELM_{sine} (the ELM with sine activation function, similarly hereinafter) outperforms ELM_{sigm} on both cases. For example, in the odd case, the testing RMSE of the ELM_{sine} model with 190 hidden neurons is 0.1767, while that of ELM_{sigm} with 430 hidden neurons is 0.3279. In spite of this, remarkable improvements can also be made by imposing symmetry in ELM_{sine}, since $\text{S-ELM}_{\text{sine}}$ reduces the testing RMSE and the number of hidden neurons to 0.1353 and 110, respectively.

In order to explore the performance of S-ELM with different size of hidden neurons, Fig. 3 shows how ELM_{sigm}, ELM_{sine}, $\text{S-ELM}_{\text{sigm}}$, and $\text{S-ELM}_{\text{sine}}$ generalize with the number of hidden neurons varying. S-ELM displays different characteristics when the network size changes: (1) the generalization ability of S-ELM increases faster than

Table 2 Performance comparison between ELM and S-ELM on the odd case

Algorithm	Training time (s)	Training		Testing		#Nodes
		RMSE	Dev	RMSE	Dev	
ELM	2.0728	0.3530	0.0166	0.3279	0.0246	430
S-ELM	**0.4788**	0.3916	0.0160	**0.2617**	0.0218	**220**

Table 3 Performance comparison between ELM and S-ELM with sine activation function

Sample size	Algorithm	Training time (s)	Training		Testing		#Nodes
			RMSE	Dev	RMSE	Dev	
Sinc 1,000	S-ELM$_{\mathbf{sine}}$	**0.0009**	0.1151	0.0017	**0.0094**	0.0026	**7**
	ELM$_{\mathbf{sine}}$	0.0050	0.1148	0.0016	0.0135	0.0031	14
Sinc 5,000	S-ELM$_{\mathbf{sine}}$	**0.0111**	0.1153	0.0008	**0.0043**	0.0012	**7**
	ELM$_{\mathbf{sine}}$	0.0286	0.1153	0.0007	0.0060	0.0011	14
Sinc 10,000	S-ELM$_{\mathbf{sine}}$	**0.0273**	0.1155	0.0005	**0.0030**	0.0008	**7**
	ELM$_{\mathbf{sine}}$	0.0617	0.1154	0.0006	0.0042	0.0008	14
Odd case	S-ELM$_{\mathbf{sine}}$	**0.1875**	0.3806	0.0094	**0.1353**	0.0104	**110**
	ELM$_{\mathbf{sine}}$	0.4197	0.3620	0.0115	0.1767	0.0113	190

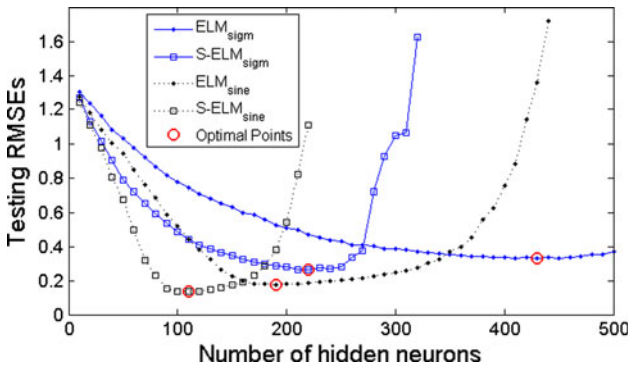

Fig. 3 Relation between the generalization performance and the numbers of hidden neurons in ELM and S-ELM with sigmoid and sine activation functions

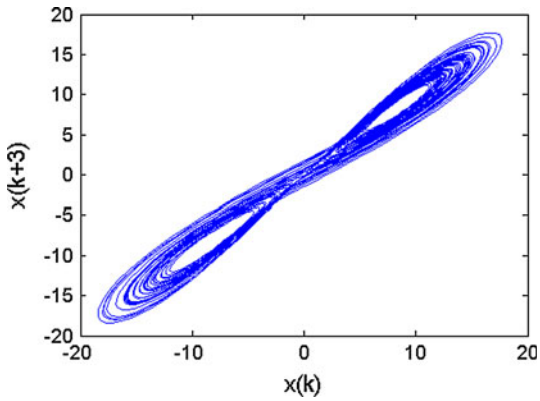

Fig. 4 Bidimensional delay reconstruction of the Lorenz attractor from the x-coordinate. There lies odd symmetry in this reconstructed attractor

that of ELM with the number of hidden nodes increasing, and the minimum testing error can be reached at a smaller network size; (2) S-ELM will generalize worse rapidly after the optimal number of hidden neurons, which is different from ELM, the generalization performance of which is relatively stable on a wide range of number of hidden nodes, as pointed by Huang [1, 17].

Finally, the effectiveness of imposing symmetry in ELM is presented using two chaotic time series prediction tasks. The first task is to model the well-known Lorenz system

$$\begin{cases} \dot{x} = a(y - x) \\ \dot{y} = (c - z)x - y \\ \dot{z} = xy - bz \end{cases} \tag{10}$$

where $a = 10$, $b = 8/3$, $c = 28$. The solution of (10) settles to the well-known Lorenz attractor shown in Fig. 4.

Here the x-coordinate of the Lorenz system is used as an example of a time series to be modeled. Being aware of the symmetry of Lorenz attractor shown in Fig. 4, S-ELM is potential to model this chaotic system. The nonlinear autoregressive (NAR) model, $x(t) = \varphi(x(t-1), \ldots, x(t-\lambda)) + e(t)$, is formulated to be indentified by ELM and S-ELM, where λ is an the order of NAR and is selected using one-step-ahead cross-validation method. Two thousand data points are used as the training samples, which are shown in Fig. 5 (solid line). The dotted line part in Fig. 5 denotes the future evolution of the series (test zone). After the NAR model is estimated using the training set, an iterative multi-step-ahead prediction for the test set is done, where the prediction is computed with past predictions as inputs, that is,

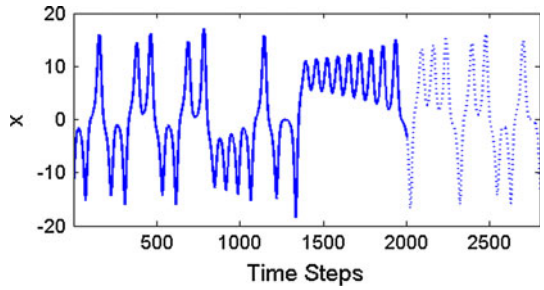

Fig. 5 The series from the x-coordinate of the Lorenz chaotic time series, part of which is used for training (*solid line*)

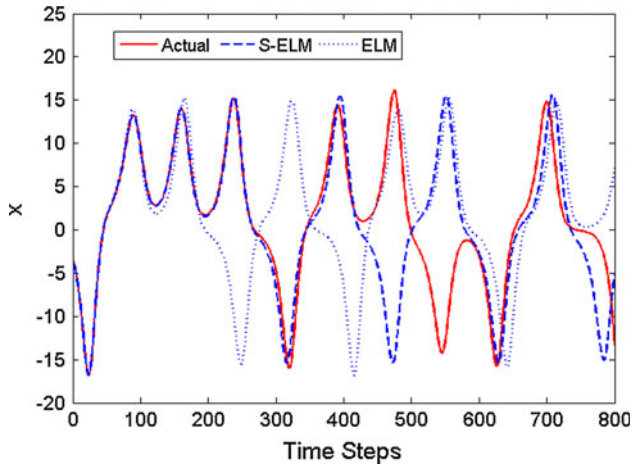

Fig. 6 Simulations with ELM and S-ELM compared to the actual values. The number of hidden neurons of ELM and S-ELM are 270 and 140, respectively

$$\hat{x}(t) = \varphi(\hat{x}(t-1), \hat{x}(t-1), \ldots, \hat{x}(t-\lambda)).$$

Figure 6 presents the simulations obtained with ELM and S-ELM of one trail, and Fig. 7 shows the mean values of the prediction errors of 50 trials. As observed from these two figures, the prediction errors gradually enhance during the iterative prediction process. Thanks to incorporating of symmetry, S-ELM can model the chaotic system more accurately at the very beginning of the test and then is able to simulate the Lorenz system for a longer time, which is more than 300 time steps, far beyond the time, less than 200 time steps, that can be simulated by ELM. Meanwhile, the number of hidden nodes of S-ELM is 140, which is much smaller than 270, that of ELM.

The second task is to model a multi-scroll attractor system. The dataset had been used for the K. U. Leuven Time Series Prediction Competition [18]. There exists a training set of 2,000 points for model estimation, shown in Fig. 8, and the goal is to forecast the next 200 points out of the training sample. As pointed by [14], a look at Fig. 8 indicated that the series is probably chaotic with 3 unstable equilibria at approximately -0.25, 0, and 0.25, and the

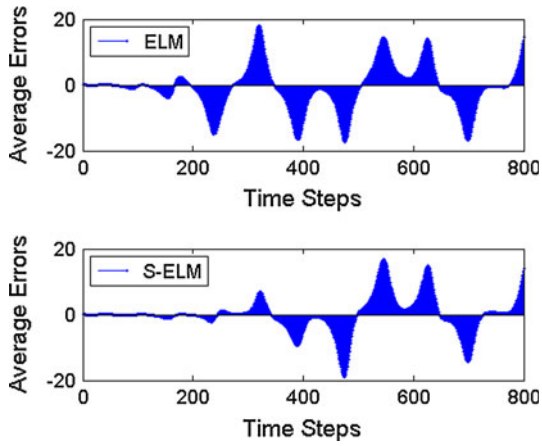

Fig. 7 Average prediction errors of ELM and S-ELM of 50 trials. The number of hidden neurons of ELM and S-ELM are 270 and 140, respectively

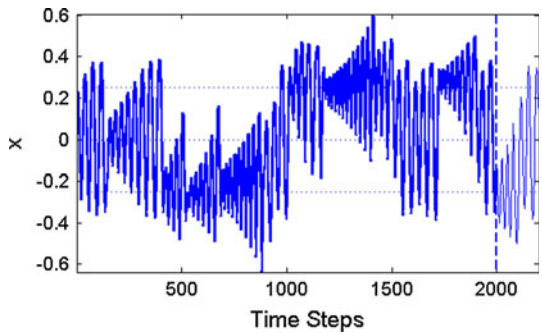

Fig. 8 The training sample (*thick line*) and future evolution (*thin line*) of the series from the K. U. Leuven Time Series Competition

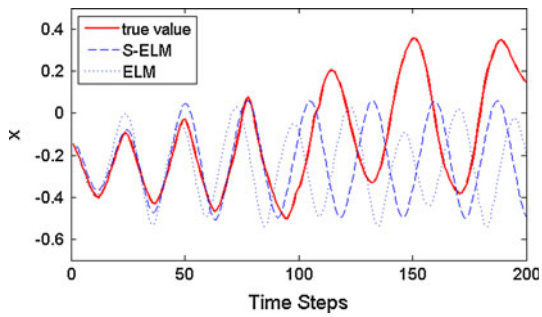

Fig. 9 Simulations with ELM, S-ELM compared with the actual values for the next 200 steps of the K. U. Leuven series. The number of hidden neurons of ELM and S-ELM are 92 and 45, respectively

source of time series seems to be symmetrical. Therefore, it is reasonable to apply S-ELM to this dataset to model the unknown system and then compare S-ELM with ELM. Following the approach used by the winner of the competition, both ELM and S-ELM are trained using 10-step-ahead cross-validation for parameter selection. Figure 9 shows the prediction results obtained with ELM and S-ELM, and Table 4 gives the mean results of 50 trials.

Table 4 Performance comparison between ELM and S-ELM on the K. U. Leuven time series

Algorithm	Training time (s)	Training RMSE	Testing RMSE				#Nodes
			1–20	1–50	1–100	1–200	
ELM	0.1984	0.0051	0.0407	0.0622	0.1331	0.2758	92
S-ELM	**0.0672**	0.0052	**0.0265**	**0.0488**	**0.1110**	**0.2643**	**45**

To illustrate the difference between ELM and S-ELM, the testing RMSEs considering the first n ($n = 20$, 50, 100, 200) test points of the test dataset are presented in Table 4. It can be seen that, though both models are trained using exactly the same methodology for order and parameter selection, S-ELM can achieve higher prediction precision at the beginning of testing and simulate the system longer than ELM, with compacter network structure and shorter training time.

5 Conclusions

In this paper, it is shown that the prior information of symmetry can be directly incorporated into ELM by transforming the activation function of the hidden nodes into a symmetric one. S-ELM preserves the capability of approximating N arbitrary distinct samples exactly. Simulation results show that, in the applications where there exists the prior knowledge of symmetry, S-ELM can obtain much better generalization performance, more compact network architecture, and faster learning speed. S-ELM also shows the potential for chaotic time series prediction. Our further research works will include how to apply the proposed S-ELM algorithm to the real-world industrial cases.

Acknowledgments This work was supported by the National High-Technologies Research and Development Program of China (2006AA04Z184), the National Natural Science Foundation of China under Grant No. 10901139, 60911130510, 60874029, and the Public Benefit Technologies R & D Program of Science and Technology Department of Zhejiang Province under Grant No. 2011C21020.

References

1. Huang GB, Zhu QY, Siew CK (2006) Extreme learning machine: theory and applications. Neurocomputing 70:489–501
2. Huang GB, Zhu QY, Siew CK (2004) Extreme learning machine: a new learning scheme of feedforward neural networks. In: Proceedings of International joint conference on neural networks (IJCNN 2004), vol 2, pp 25–29
3. Golub G, Charles F, Loan V (1983) Matrix computations. Johns Hopkins University Press, Baltimore
4. Haykin S (1999) Neural networks: a comprehensive foundation. Prentice Hall, New Jersey
5. Cortes C, Vapnik V (1995) Support vector networks. Mach Learn 20:273–297
6. Minhas R, Baradarani A, Seifzadeh S, Wu QMJ (2010) Human action recognition using extreme learning machine based on visual vocabularies. Neurocomputing 73:1906–1917
7. Zhang R, Huang GB, Sundararajan N, Saratchandran P (2007) Multi-category classification using an extreme learning machine for microarray gene expression cancer diagnosis. IEEE/ACM Trans Comput Biol Bioinf 4:485–495
8. Sjöberg J, Zhang Q, Ljung L, Benveniste A, Deylon B, Glorennec P, Hjalmarsson H, Juditsky A (1995) Nonlinear black-box modelling in system identification: a unified overview. Automatica 31:1691–1724
9. Huang GB, Chen L, Siew CK (2006) Universal approximation using incremental constructive feedforward networks with random hidden nodes. IEEE Trans Neural Netw 17:879–892
10. Aguirre L, Lopes R, Amaral G, Letellier C (2004) Constraining the topology of neural networks to ensure dynamics with symmetry properties. Phys Rev E 69:1–11
11. Chen S, Wolfgang A, Harris C, Hanzo L (2008) Symmetric RBF classifier for nonlinear detection in multiple-antenna-aided systems. IEEE Trans Neural Netw 19:737–745
12. Chen S, Hong X, Harris C (2011) Grey-box radial basis function modelling. Neurocomputing 74:1564–1571
13. Espinoza M, Suykens J, Moor B (2005) Imposing symmetry in least squares support vector machines regression. In: Proceedings of the 44th IEEE conference on decision and control (CDC 2005), pp 5716–5721
14. McNames J, Suykens JAK, Vandewalle J (1999) Winning entry of the K. U. Leuven time series prediction competition. Int J Bifurcat Chaos 9:1485–1500
15. Huang GB, Babri H (1998) Upper bounds on the number of hidden neurons in feedforward networks with arbitrary bounded nonlinear activation functions. IEEE Trans Neural Netw 9:224–229
16. http://www3.ntu.edu.sg/home/egbhuang/
17. Wang D, Huang GB (2005) Protein sequence classification using extreme learning machine. In: Proceedings of International joint conference on neural networks (IJCNN 2005), pp 1406–1411
18. Suykens JAK, Vandewalle J (2000) The K. U. Leuven competition data: a challenge for advanced neural network techniques. In: Proceedings of the European symposium on artificial neural networks (ESANN 2000), pp 299–304