Neural Comput & Applic (2013) 22:531–539
DOI 10.1007/s00521-012-0858-9

The extreme learning machine learning algorithm with tunable activation function

Bin Li · Yibin Li · Xuewen Rong

Received: 6 August 2011 / Accepted: 21 January 2012 / Published online: 7 February 2012
© Springer-Verlag London Limited 2012

Abstract In this paper, we propose an extreme learning machine (ELM) with tunable activation function (TAF-ELM) learning algorithm, which determines its activation functions dynamically by means of the differential evolution algorithm based on the input data. The main objective is to overcome the problem dependence of fixed slop of the activation function in ELM. We mainly considered the issue of processing of benchmark problems on function approximation and pattern classification. Compared with ELM and E-ELM learning algorithms with the same network size or compact network configuration, the proposed algorithm has improved generalization performance with good accuracy. In addition, the proposed algorithm also has very good performance in the TAF neural networks learning algorithms.

Keywords Extreme learning machine · Single hidden layer feed-forward neural networks · Tunable activation function · Differential evolution algorithm

1 Introduction

Feed-forward neural networks have been attracted much attention and been extensively used in many fields, such as time series prediction, pattern classification and a variety of real applications, due to their simple architecture and good global approximation performance. In general, there are few faster learning algorithms for feed-forward neural networks, and the traditional learning methods are usually much slower than required. It is not surprising to see that it may take several hours and even more time to train neural networks by using of the traditional methods [1].

A novel learning algorithm for single hidden layer feed-forward neural networks (SLFNs) called extreme learning machine (ELM) [1–3] was presented by Huang et al. for improving the training time of SLFNs. Unlike traditional neural networks approaches, the ELM learning algorithm chooses randomly the input weights and biases of hidden neurons, and the output weights of SLFNs are determined analytically. The training speed of ELM learning algorithm is extremely fast and has good generalization ability with least human intervene, which has been applied in many areas, such as nonlinear system identification [4] and decision-making problems [5], etc.

However, the basis functions themselves influence the performance of neural networks with respect to the same training data, the choice of basis function is problem dependent and has a significant effect on model performance of neural networks. Especially, in the ELM learning algorithm, the configuration of neural networks has quite different approximation ability and generalization performance for different basis functions on the same problem. Consequently, the choice of activation (basis) functions of ELM learning algorithm is problem dependent [6]. Generally, there are two approaches for solving the problem dependence of neural networks. The first one is incorporating a priori information into ELM learning algorithm in order to select the activation function of hidden neurons [7, 8]. However, the priori knowledge of problems is

B. Li · Y. Li (✉) · X. Rong
School of Control Science and Engineering,
Shandong University, Jinan 250061, People's Republic of China
e-mail: liyb@sdu.edu.cn

B. Li
e-mail: ribbenlee@126.com

B. Li
School of Science, Shandong Polytechnic University,
Jinan 250353, People's Republic of China

difficult to acquire. The second method is to choice the neural networks with tunable activation functions, that is, the activation functions do not keep unchanged. They can be adjusted adaptively to embed the information of problems in the training process in order to match the specific task [9–11]. The learning algorithm previously proposed in [9] is back propagation (BP) algorithm suffering from a number of shortcomings which has lower convergence speed and may easily converge to local minima. Furthermore, the algorithm presented in [10, 11] based on the RLS algorithm is more sensitive to initial parameters.

In order to improve the performance of SLFNs and solve the problem dependence of ELM learning algorithm, we propose a novel learning algorithm for ELM called tunable activation function extreme learning machine (TAF-ELM) based on the idea of tunable activation function, which was developed in [9, 10]. In terms of the learning algorithm, the training parameters (input weights, hidden biases and TAF parameters of variables) of neural networks can be assigned randomly, and then the weights between hidden layer and output layer can be solved using the least squares method by the Moore–Penrose (MP) generalized inverse. Afterward, the maximum iteration time of the TAF-ELM algorithm is set. Based on the convergence criteria, the parameters of neural networks are adjusted using differential evolutionary algorithm [12]. Consequently, the parameters of variable of tunable activation functions can be tuned dynamically, and the values of TAF parameters represent the importance of different activation functions. Moreover, the activation functions of hidden neurons contain the information of problems that should be solved self-adaptively.

Existing learning algorithms of neural networks can be categorized into two classes, batch and on-line (or sequential). Sequential algorithms with less training time are applicable to real-time systems. Nevertheless, the generalization ability and approximation accuracy are more important than training time for batch algorithms. As the new proposed learning algorithm is implemented with respect to function approximation and pattern classification benchmark problems, the proposed batch learning method can be used to obtain better generalization performance sometimes with more compact networks, although it takes longer time for training process than ELM and E-ELM learning algorithms. Meanwhile, in the Feigenbaum function approximation problem, the TAF-ELM performs much better than the traditional TAF-BP [9] and TAF-MFNN [10] learning algorithms.

The rest part of this paper is organized as follows. Section 2 presents the TAF-ELM learning algorithm based on the original ELM and the idea of tunable activation function. Section 3 includes different simulation results and analysis of the proposed algorithm for benchmark problems. Finally, the conclusion is summarized in Sect. 4.

2 The TAF-ELM learning algorithm

2.1 The TAF neural model

The activation functions simulating soma's function of the classical M-P model are fixed. The capability of neural networks based on this simple model is limited and fails to deal with many difficult problems. Actually, there are many different kinds of biologic neurons. Different types of neurons can be adapted and used effectively to deal with different information. On the basis of this hypothesis, Wu and Zhao have designed the tunable activation function model [9].

Figure 1 gives the TAF neural model. In the model, a neuron is divided into two parts: synapse and soma. The function of the synapse is regarded as a multi-input and multi-output mapping, denoted by $S = g(X, W, b)$, which includes the hidden bias (or impact factor) b in our paper. Where S is called "inner activation", which is the input of the soma. The function of the soma, called activation function of neural networks, is to transform the inner-activation nonlinearly and can be denoted as $O = G(S, \alpha)$. Where $G(\cdot, \cdot)$ is the mapping function of the soma model and is called TAF. In the function of G, α is a tunable parameter, which enables the soma to have the ability of adaptive training to suit specific input signal [9–11].

Thus, the activation function of TAF model can be generally written as

$$G(S, \alpha) = G(g(X, W, b), \alpha) = F(X, W, b, \alpha) \quad (1)$$

where W is a weight vector between input layer and hidden layer, X is an input signal vector, b is the bias of the hidden neuron, α is a parametric vector, which are used to control

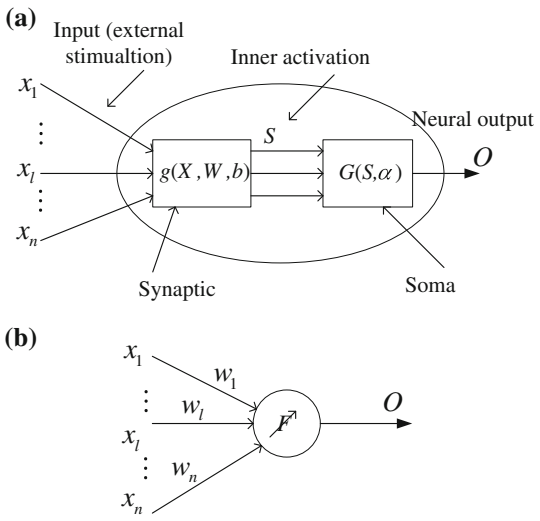

Fig. 1 The general TAF neural model

the adjustability of TAF neurons. S is the output of synapses or can be regraded as input of the soma, so it is referred to as the internal stimulation or called total inner simulation when S is a scalar quantity [9, 10].

In this paper, the TAF activation function represented by a weighted summation of a finite number of basis functions f_j is as follows

$$G(S,\alpha)=\sum_{j=1}^{k}\alpha_{i,j}f_j(S)\quad i=1,2,\ldots,\tilde{N} \tag{2}$$

where $\alpha_{i,j}(i=1,2,\ldots,\tilde{N},\ j=1,2,\ldots,k)$ are all real constants, $\tilde{N}$ is the number of hidden neurons. k is the combinatorial number of basis functions per hidden neuron. $\alpha_i=(\alpha_{i,1},\alpha_{i,2},\ldots,\alpha_{i,k})^T$ is called the TAF parametric vector of the ith hidden neuron and represents the degree of importance of corresponding activation function. f_j (S) is the jth TAF basis function, which always satisfied the condition of infinite differentiability.

2.2 The TAF-ELM neural networks

The ELM with good performance has recently attracted the attention from more and more researchers. But for the same problem, the ELM learning algorithm has different performance with different activation functions [6]. The results indicate that the choice of activation function is data set dependent, and evaluating the proper activation function for different problem is advantageous for improving the generalization performance of neural networks. In order to overcome the data dependence of ELM, the ELM learning algorithm combined with TAF neural model is proposed in this paper.

As is known, the activation function is strictly required for neural networks model in the early research stage of neural networks. The property of activation function ought to be continuous, bounded and nonconstant. Hornik proved that the continuous mappings can be learned uniformly over compact input sets for the feed-forward neural networks on condition that the activation function is continuous, bounded and nonconstant [13]. Afterward, the ELM learning algorithm with infinitely differentiable activation function is available for SLFNs [1, 2]. Such activation functions include the sigmoid function as well as the radial basis, sine, cosine, exponential and many other nonregular functions as shown in [14]. Therefore, the theorems mentioned above extend the choosing space of activation functions for neural networks, and the theoretic basis is supported for constructing the TAF-ELM neural networks learning algorithm.

In order to reduce the computational complexity, a simple form of activation function is preferred exemplifying by activation functions like the sigmoid function, sine, cosine, cubic functions and all-order derivatives of sigmoid functions. However, in order to enlarge the feasible region of tunable activation functions and further to accommodate to more conditions, many other combinational forms of activation functions satisfying the theorems that proposed by Huang et al. in [1, 2] are chosen to construct different activation function types.

The construction of TAF-ELM neural networks with $\tilde{N}$ hidden neurons is shown in Fig. 2, which consists of one input layer, one hidden layer with combination of different activation functions and one output layer. The mathematical model of the TAF-ELM learning algorithm can be written as follows

$$f_{\tilde{N}}(X)=\sum_{i=1}^{\tilde{N}}\beta_i F(W_i,b_i,X,\alpha_i),\qquad X\in R^n,\ W_i\in R^n,\ \beta_i\in R^m,\ \alpha_i\in R^k \tag{3}$$

where $F(W_i, b_i, X, \alpha)$ is the output of ith hidden neuron corresponding to the input X.

$\beta_i=[\beta_{i1},\beta_{i2},\ldots,\beta_{im}]^T$ denotes the weight vector connecting the ith hidden neuron and the output neurons, $W_i=[w_{i1},w_{i2},\ldots,w_{in}]$ is the weight vector connecting the ith hidden neuron and the input neurons, b_i is the bias of the ith hidden neuron. And $\alpha_i=(\alpha_{i,1},\alpha_{i,2},\ldots,\alpha_{i,k})^T$ is the TAF parametric vector of the ith hidden neuron.

For N arbitrary input and output samples (X_i, Y_i), where $X_j=[x_{i1},x_{i2},\ldots,x_{in}]\in R^n$, $Y_j=[y_{i1},y_{i2},\ldots,y_{im}]\in R^m$. Based on the theorems proposed by Huang et al. in [1, 2], the TAF-ELM learning algorithm can approximate these N samples with zero error, if the parameter $\tilde{N}$ of hidden neurons and the activation function $F(W_i, b_i, X_j, \alpha)$ are given. Thus, there exist β_i, W_i, α and b_i such that

$$\sum_{i=1}^{\tilde{N}}\beta_i F(W_i,b_i,X_j,\alpha)=Y_j\quad j=1,\ldots,N \tag{4}$$

The above N equations can be written compactly as

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

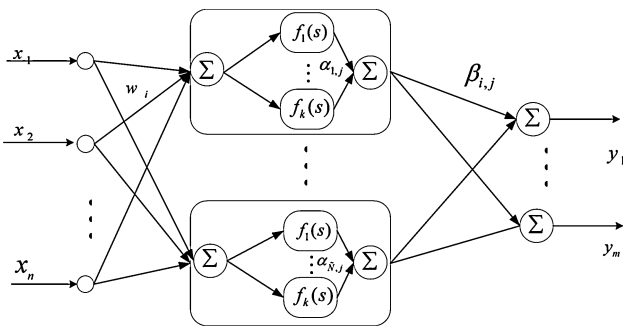

Fig. 2 The construction of TAF-ELM neural networks

where

$$\beta = \begin{bmatrix} \beta_1^T \\ \vdots \\ \beta_{\tilde{N}}^T \end{bmatrix}_{\tilde{N}\times m} \quad \text{and} \quad Y = \begin{bmatrix} Y_1^T \\ \vdots \\ Y_N^T \end{bmatrix}_{N\times m}$$

H is called the hidden layer output matrix of the TAF-ELM learning algorithm [1, 2, 15]. The details presentation form of H is shown in formula (6).

$$\begin{aligned} & H\left(W_1,\ldots,W_{\tilde{N}},b_1,\ldots,b_{\tilde{N}},X_1,\ldots,X_N,\alpha_{1,1},\ldots,\alpha_{1,k},\ldots,\alpha_{\tilde{N},1},\ldots,\alpha_{\tilde{N},k}\right) \\ &= \begin{bmatrix} h(X_1) \\ \vdots \\ h(X_1) \end{bmatrix} \\ &= \begin{bmatrix} F\left(W_1,b_1,X_1,\alpha_{1,1},\ldots,\alpha_{1,k}\right)\ldots F\left(W_{\tilde{N}},b_{\tilde{N}},X_1,\alpha_{\tilde{N},1},\ldots,\alpha_{\tilde{N},k}\right) \\ \vdots \quad \ldots \quad \vdots \\ F\left(W_1,b_1,X_N,\alpha_{1,1},\ldots,\alpha_{1,k}\right)\ldots F\left(W_{\tilde{N}},b_{\tilde{N}},X_N,\alpha_{\tilde{N},1},\ldots,\alpha_{\tilde{N},k}\right) \end{bmatrix}_{N\times\tilde{N}}, \end{aligned} \tag{6}$$

where

$$\begin{aligned} h(X) = & F\left(W_1,b_1,X,\alpha_{1,1},\ldots,\alpha_{1,k}\right)\ldots \\ & F\left(W_{\tilde{N}},b_{\tilde{N}},X,\alpha_{\tilde{N},1},\ldots,\alpha_{\tilde{N},k}\right) \end{aligned} \tag{7}$$

is called the hidden layer feature mapping. The ith column of H is the ith hidden neuron output with respect to inputs $X_1, X_2, \ldots, X_N$. The ith of H is the hidden layer feature mapping with respect to the ith input X_i: $h(X_i)$.

The training process of TAF-ELM learning algorithm is equivalent to finding a least squares solution $\hat{\beta}$ of the linear system $H\beta = Y$. However, in most cases, the number of hidden neurons is much less than the number of training samples of neural networks, $\tilde{N} \ll N$. Therefore, only the smallest norm least squares solution of linear system mentioned above can be solved and the formula can be represented by

$$\hat{\beta} = H^{+}Y \tag{8}$$

where H^{+} is the MP generalized inverse of matrix H [1, 2].

In general, two types of training methods are involved in the TAF-ELM learning algorithm. The first one is that the weight vector that connects the hidden layer neurons with the output layer neurons, and the parameters of TAF are tuned simultaneously in each iterative time. Contrarily, the second one is that the weights between the input layer neurons and the hidden layer neurons, the biases of hidden neurons and the TAF parametric vector are determined firstly, and then the smallest norm least squares solution of linear system mentioned above is solved. After that, the parameters of TAF are adjusted by means of some optimization methods such as differential evolution and particle swarm optimization. In other words, the weight vector in the second method of connecting hidden layer neurons with output layer neurons is invariable, while the biases of hidden neurons, TAF parametric vector and the weights between input layer neurons and hidden layer neurons are adjusted simultaneously. Consequently, the minimum error of the neural networks can be obtained, and the neural networks constructed by the TAF-ELM learning algorithm is adaptive to the problems to be solved.

As the ELM learning algorithm is concerned, the input weights and hidden biases are tuned randomly. Therefore, much of learning time traditionally spent in tuning these parameters is saved. However, in this algorithm, the parameters of input weights and hidden biases are not optimal, and besides, the ELM learning algorithm may require more hidden neurons for improving the performance of neural networks. Unfortunately, the high complexity of neural networks may reduce the generalization performance and also make it respond slowly to unknown testing data. In reference [12], a hybrid approach named E-ELM learning algorithm is proposed using differential evolution (DE) and MP generalized inverse for expecting more compact neural networks in the applications which requires faster response of the trained networks.

In this work, based on the idea of tunable activation function and the ELM learning algorithm, a novel neural networks learning algorithm called TAF-ELM is proposed. In this algorithm, the activation function of hidden neuron is changeable and can be adjusted adaptively to embed the information of the learning task in the training process. The intelligent optimization algorithms including differential evolution and particle swarm optimization can be used to optimize the parameters of ELM neural networks with different activation functions. Generally, the differential evolution algorithm is better than the particle swarm optimization in the parameter optimization process of ELM learning algorithm [12, 16]. Therefore, in this paper, the parameters of the TAF-ELM are optimized through the DE algorithm used in [12]. However, the individual in the population of the proposed algorithm is composed of a set of input weights, hidden biases and TAF parametric vector. The individual of the proposed algorithm is different from that in the E-ELM learning algorithm and is defined as

$$\begin{aligned} \theta = [& w_{11}, w_{12}, \ldots, w_{1k}, w_{21}, w_{22}, \ldots, w_{2k}, \ldots, \\ & w_{\tilde{N}1}, w_{\tilde{N}2}, \ldots, w_{\tilde{N}k}, \ldots, \\ & b_1, b_2, \ldots, b_{\tilde{N}}, \alpha_{1,1}, \ldots, \alpha_{1,k}, \ldots, \alpha_{\tilde{N},1}, \ldots, \alpha_{\tilde{N},k}] \end{aligned} \tag{9}$$

The weight vector of input layer, hidden biases and the TAF parametric vector are adjusted adaptively based on the DE algorithm. Therefore, the optimal parameters of input weights and hidden biases can be obtained, and also the TAF coefficients of activation functions can be tuned dynamically for adapting to the problem. Consequently, an optimal SLFNs can be achieved for real applications.

Based on the ELM and E-ELM learning algorithms and the tunable activation function, the TAF-ELM learning algorithm is derived as the following.

Given a training set $\{(X_j, Y_j)|X_j \in R^n, Y_j \in R^m, j = 1, 2, \ldots, N\}$, the combinational form and the number of basis functions of hidden layer neurons k, the number of hidden neurons $\tilde{N}$ are set experientially.

Step 1 Generate the initial generation composed of parameter vectors $\{\theta_{p,G}|p = 1, 2, \ldots, NP\}$ as the population, where NP is the population size, $G = \tilde{N} * (N+1+k)$.
Step 2 The hidden layer output matrix H and the output weights β are computed analytically by using the MP generalized inverse. Then, the root mean squared error (RMSE) of each individual of population is evaluated as the fitness function $val(p)$.
Step 3 At each generation G, mutation, crossover and selection operations are performed, respectively, and then one new generation $\theta_{p,G}$ is generated.
Step 4 The step 2 and the step 3 are iterated repetitively until the maximal iteration time is satisfied.
Step 5 The TAF parameters, w_{ij} and b_i can be determined, and then the optimal $\theta_{p,G}$ are found. Then, based on the above parameter values, the hidden layer output matrix H is computed.
Step 6 Determine the final output weights β in terms of Eq. 7.

3 Simulations and performance verification

3.1 Performance of the TAF-ELM learning algorithm

In the aspect of function approximation as well as pattern classification benchmark problems, the performance of TAF-ELM learning algorithm is compared with the ELM and E-ELM algorithms. All the simulations for TAF-ELM, ELM and E-ELM learning algorithms are carried out in MATLAB 7.1 environment running in a Pentium 4, 3.2 GHZ CPU. The average results of 10 trials with different training and testing sets are shown for increasing the persuasion of the proposed algorithm. The root mean square error (RMSE) is used to measure the training error and the testing error of the function approximation problems. The performance of pattern classification problems is justified by classification rate. Moreover, the accuracy of the algorithms is also evaluated by the standard deviation (SD) of 10 trials. For convenience in discussion, except for the maximal iteration time, other parameters of DE algorithm are the same as the parameters in [12], the population size NP is set to 200, crossover constant is set to 0.8, etc. In addition, the performance of ELM [1] and E-ELM [12] algorithms is evaluated with additive hidden neurons (which chooses a simple sigmoid function) and radial basis function (RBF) hidden neurons with characteristic of localized receptive fields. The ELM and E-ELM learning algorithms based on the RBF neural networks cases were, respectively, presented by Huang and Siew in [17] and our previous paper [16].

The simulation data sets mainly come from the UCI repository [18]. Two function approximation (regression) and two pattern classification benchmark problems are used for verifying the performance of TAF-ELM learning algorithm. The specification of benchmark problems divided of data sets for the training and testing sets is described in Table 1. All the inputs of data sets have been normalized into [−1, 1].

In order to obtain convincing performance comparison of the complexity of the neural networks, the number of hidden neurons and the added tunable parameters of the TAF-ELM, ELM and E-ELM learning algorithms are used simultaneously. In the TAF-ELM learning algorithm, the added tunable parameters are the number of the TAF parametric vector. In the ELM and E-ELM learning algorithms with additive hidden neurons, the added tunable parameters denoted as the weights of input layer, hidden biases and the weights connecting hidden layer and output layer with increasing of the hidden neurons. While in these algorithms with RBF hidden neurons, the added tunable parameters are the centers and impact factors of RBF neurons and the weights connecting the hidden neurons to the output neurons.

Performance comparison between TAF-ELM learning algorithm and the ELM, E-ELM algorithms on function approximation problems is illustrated in Table 2. The maximal iteration time (abbrevd. max.) of the TAF-ELM learning algorithm in Table 2 is 20. The activation function of the proposed algorithm is the combination of sigmoid function, sine function and linear function, $k = 3$. The experiment results show that the performance of TAF-ELM learning algorithm proposed here has improved significantly under the prerequisite condition of the same neural network size. Based on the maximal iteration time, differential evolution algorithm is used to adjust the TAF parametric vector adaptively in order to match the specific task. Thus, the TAF-ELM learning algorithm embedded the information of the learning task in the training process demonstrates better generalization performance and good convergence accuracy. As shown in Table 2, the performance of TAF-ELM learning algorithm is better than ELM and E-ELM learning algorithms on function approximation problems in most cases. Especially in the Box and Jenkins gas furnace data, the preferable performance of proposed algorithm illustrates that the selection of activation functions in these specific problems is suitable. In conclusion, the simulation results demonstrate that the TAF-ELM learning algorithm has better generalization ability with

Table 1 Specification of benchmark data sets

Types of problems	Data sets	Input attributes	Output attributes	Training data	Testing data
Function	Auto-Mpg	7	1	320	79
Approximation	B. J. gas furnace	10	1	200	90
Pattern	Wine	13	3	150	28
Classification	Iris	4	3	110	27

Table 2 Performance comparison on function approximation problems

Algorithm	Auto-Mpg				Box and Jenkins gas furnace data			
	CPU time (s)	Training err. SD	Testing err. SD	Size of complexity	CPU time (s)	Training err. SD	Testing err. SD	Size of complexity
TAF-ELM (max. = 20)	35.8875	0.0860	0.0891	3 (9)	26.4985	0.0181	0.0185	3 (9)
		0.0054	0.0042			9.1318e−004	7.9924e−004	
E-ELM (sigmoid)	3.7828	0.0855	0.0905	4 (9)	3.3859	0.0299	0.0309	4 (12)
		0.0057	0.0066			0.0041	0.0060	
ELM (sigmoid)	0.0031	0.1014	0.1192	4 (9)	0	0.0567	0.0510	4 (12)
		0.0444	0.0444			0.0234	0.0185	
E-ELM (RBF)	56.2297	0.0841	0.0876	4 (9)	39.7891	0.0203	0.0226	4 (12)
		0.0027	0.0051			8.6210e−004	0.0029	
ELM (RBF)	0.0109	0.3077	0.3043	4 (9)	0.0031	0.4345	0.4770	4 (12)
		0.0897	0.0846			0.0679	0.0654	

fewer parameters and compact network configuration comparing to ELM and E-ELM learning algorithms in regression problems.

The sigmoid function has the nonlinear and continuous differentiable properties, which is good gain control for the input signal. Therefore, the combination of the sigmoid functions and the first, second and third derivative of sigmoidal type functions is chosen in the TAF-ELM learning algorithm in solving the pattern classification problems, that is, $k = 4$. Seen from Table 3, the TAF-ELM learning algorithm in the aspect of testing classification rate greatly improved in most cases when compared to ELM and E-ELM learning algorithms.

As is known, generalization performance which has been widely studied in [19, 20] is the most important property of the feed-forward neural networks. Both the accuracy of training data sets and the complexity of network structure which matters more play significant roles in the generalization performance. As for the Wine classification problem, the TAF-ELM learning algorithm has better generalization performance for the fewer tunable parameters of neural networks. There are 5 times to 100% in the 10 trials of testing classification rate. However, the E-ELM learning algorithm just has 2 times to 100%. It can also be observed that the ELM and E-ELM learning algorithms have different performance with different activation functions based on the simulation results in Tables 2 and 3. And the E-ELM algorithm with RBF hidden neurons runs slower than TAF-ELM learning algorithm.

Figures 3 and 4 show the relationship between the testing performance and the number of hidden neurons on Box and Jenkins gas furnace data and Iris, respectively. As seen in the figures, the TAF-ELM learning algorithm has better generalization ability in most cases. Especially in the function approximation problem, the generalization performance of TAF-ELM learning algorithm almost unchanged with increasing of the number of hidden neurons. Consequently, the TAF-ELM learning algorithm can improve the generalization performance significantly with good accuracy.

3.2 Sensitivity analysis of the TAF-ELM neural networks

Based on the DE algorithm, the TAF parameter α, input weights and hidden biases in the TAF-ELM learning algorithm are adjusted adaptively for improving the performance of neural networks. However, this algorithm is relatively slow in learning process compared to ELM learning algorithm in most cases. Therefore, for increasing

Table 3 Performance comparison on pattern classification problems

Algorithm	Iris				Wine			
	CPU time (s)	Training rate SD	Testing rate SD	Size of complexity	CPU time (s)	Training rate SD	Testing rate SD	Size of complexity
TAF-ELM (max. = 20)	15.0375	0.9473	0.9385	3 (12)	19.3375	0.8507	0.9429	5 (20)
		0.0119	0.0324			0.0616	0.0738	
E-ELM (sigmoid)	4.5328	0.9582	0.9385	5 (12)	5.2094	0.9200	0.9214	7 (30)
		0.0130	0.0324			0.0406	0.0527	
ELM (sigmoid)	0	0.8836	0.8593	5 (12)	0	0.8900	0.9179	7 (30)
		0.0417	0.0600			0.0502	0.0715	
E-ELM (RBF)	21.6250	0.9318	0.8692	5 (12)	35.0969	0.8353	0.8714	7 (30)
		0.0351	0.0633			0.0594	0.0940	
ELM (RBF)	0.0172	0.7436	0.7259	5 (12)	0.0125	0.5940	0.6178	7 (30)
		0.0923	0.1148			0.1015	0.1011	

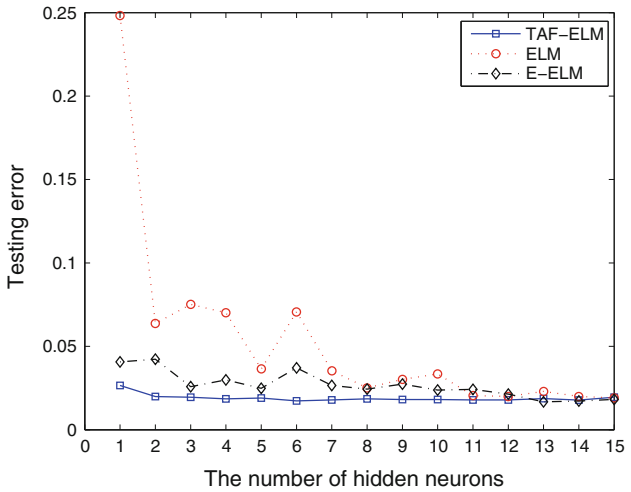

Fig. 3 Relation between testing error and the number of hidden neurons on Box and Jenkins gas furnace data

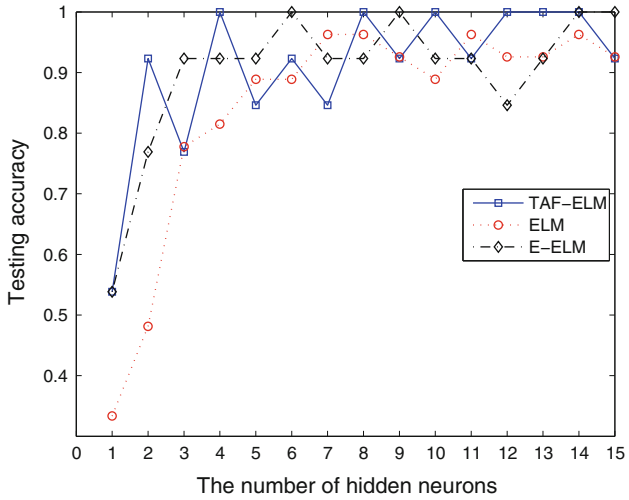

Fig. 4 Relation between testing accuracy and the number of hidden neurons on Iris

the learning speed and evaluating the influence of TAF parameter on the TAF-ELM learning algorithm, perhaps one does not need to adjust the TAF parameters because of the configuration of TAF-ELM neural networks already including different activation functions.

After the input weights, the hidden biases and the TAF parametric vector are chosen randomly, the TAF-ELM can be simply considered as a linear system and the output weights can be analytically determined. In this part, the performance of the TAF-ELM learning algorithm is compared in two different cases: without iteration and the maximum iteration time is set as 10. Tables 4 and 5 show the average performance of 10 trials on benchmark problems in function approximation and pattern classification. As observed from the tables, it is obvious that the testing RMSE is decreasing in the function approximation and the testing rate is increasing in the pattern classification with the increase in iteration time. But, the performance of the proposed algorithm is not obviously improved with the increasing of iteration time, which means that the TAF-ELM neural networks are not much sensitive to input parameters. Moreover, as observed from the standard deviation and generalization performance in tables, the accuracy of the proposed algorithm without iteration is generally better than the ELM learning algorithm.

3.3 Performance comparison of different TAF neural networks

In this section, the performance of TAF-ELM learning algorithm is compared with TAF-BP [9] and TAF-MFNN [10] learning algorithms. The nonlinear Feigenbaum function coming from the references [9, 10] is used to validate the performance of TAF neural networks.

Table 4 Sensitivity analysis of the TAF-ELM neural networks on function approximation

Algorithm	Auto-Mpg				Box and Jenkins gas furnace data			
	CPU time (s)	Training err. SD	Testing err. SD	Size of complexity	CPU time (s)	Training err. SD	Testing err. SD	Size of complexity
TAF-ELM (no iter.)	1.4766	0.0916	0.0970	3 (9)	1.0703	0.0393	0.0398	3 (9)
		0.0056	0.0061			0.0040	0.0056	
TAF-ELM (max. = 10)	16.1969	0.0891	0.0946	3 (9)	11.5688	0.0250	0.0246	3 (9)
		0.0047	0.0060			0.0104	0.0091	

Table 5 Sensitivity analysis of the TAF-ELM neural networks on pattern classification

Algorithm	Iris				Wine			
	CPU time (s)	Training rate SD	Testing rate SD	Size of complexity	CPU time (s)	Training rate SD	Testing rate SD	Size of complexity
TAF-ELM (no iter.)	0.6297	0.9255	0.8846	3 (12)	0.9094	0.8760	0.9000	5 (20)
		0.0417	0.0748			0.0623	0.0603	
TAF-ELM (max. = 10)	6.9797	0.9445	0.8923	3 (12)	10.1656	0.8840	0.9214	5 (20)
		0.0357	0.0649			0.0626	0.0626	

The Feigenbaum function is defined by

$$x(t+1) = \gamma x(t)(1 - x(t)) \tag{10}$$

where parameter γ is equal to 4. The data used to train and test the TAF neural networks are $\{x(t), x(t+1)\}$, and the initial value of $x(t)$ is 1.004×10^{-2}. The number of training data in the TAF neural networks N is 200 [9, 10], and the number of testing data in the TAF-ELM is 100. For the convenience of comparative analysis, the training and testing error are given by

$$10 \lg \left(\sum_{i=1}^{N} (o_i - Y_i)^2 \Big/ \left(\sum_{i=1}^{N} Y_i^2 \right) \right) (dB) \tag{11}$$

where o_i is the output of the neural network, and Y_i is the desired output.

Performance comparison among TAF-ELM, TAF-BP and TAF-MFNN is given in Table 6. In the table, the data of TAF-BP and TAF-MFNN are taken from [9, 10] and the average results of 10 trials of TAF-ELM are shown. The parameter k is the combinatorial number of basis functions per hidden neuron, and the convergence speed of algorithms is indicated by the number of iterations. The combination of TAF is the same as that of the above two function approximation problems. As shown in Table 6, The TAF-ELM learning algorithm achieves much better approximation performance and has simpler network construction than the TAF-BP and TAF-MFNN algorithms with much less number of iterations or without iteration cases. In this special Feigenbaum function approximation problem, the TAF-ELM learning algorithm without iteration can also achieve very good generalization performance. Moreover, the performance cannot be improved significantly by increasing the iteration time, which means the TAF-ELM is an accurate algorithm and not sensitive to the input parameters in this specific problem.

4 Conclusion

In this work, a novel training algorithm, TAF-ELM, has been developed based on the ELM. In this algorithm, the combination of different activation functions is embedded into the ELM, and the parameters of input weights, hidden

Table 6 Performance comparison among different TAF neural networks on Feigenbaum function

Algorithm	CPU time (s)	Training error	Testing error	No. of iterations	No. of hidden neurons	k
TAF-ELM (no iter.)	1.1453	−120.45	−98.41	0	3	3
TAF-ELM (max. = 20)	24.566	−135.43	−99.00	20	3	3
TAF-MFNN	–	−35.03	–	24	6	4
TAF-BP	–	−45	–	20,000	6	4

biases and tunable TAF parametric vector are adjusted by the differential evolution algorithm instead of constant activation function of ELM and E-ELM learning algorithms. The combination of activation functions can be adjusted adaptively, and then the TAF-ELM learning algorithm is adaptive to the information of problem to be solved, provided that the domain of activation function is suitable.

However, how to choose a suitable combination of activation functions is still an unresolved theoretic problem. When the number of basis functions k increases, the algorithm is suggested to be more suitable to the problems, whereas the training time increases and thus the learning speed of algorithm is reduced, which consequently resulting in unsuited for resolution of real-time problems with this algorithm.

Acknowledgments This work was supported by the Independent Innovation Foundation of Shandong University Grant No. 2009JC010 and 2011JC011, the National Nature Science Foundation of China Grant No. 61075091, National Natural Science Foundation for Young Scholars of China (61105100) and the Natural Science Foundation of Shandong Province under grant No. Y2008G21 and Grant No. 2007BS01008.

References

1. Huang GB, Zhu QY, Siew CK (2006) Extreme learning machine: theory and applications. Neurocomputing 70(1–3):489-501
2. Huang GB, Wang DH, Lan Y (2011) Extreme learning machines: a survey. Int J Mach Learn Cybern 2(2):107–122
3. Huang GB, Zhu QY, Siew CK (2004) Extreme learning machine: a new learning scheme of feedforward neural networks. In: Proceedings of the international joint conference on neural networks (IJCNN2004), Budapest, Hungary, pp 25–29
4. Li MB, Er MJ (2006) Nonlinear system identification using extreme learning machine. In: 9th International conference on control, automation, robotics and vision, pp 1–4
5. Li FC, Wang PK, Wang GE (2009) Comparison of the primitive classifiers with extreme learning machine in credit scoring. In: IEEE international conference on industrial engineering and engineering management, Hong Kong, pp 685–688
6. Li B, Li YB (2011) Chaotic time series prediction based on ELM learning algorithm. J Tianjing Unv 44(8):701–704
7. Han F, Huang DS (2006) Improved extreme learning machine for function approximation by encoding a priori information. Neurocomputing 69(16–18):2369–2373
8. Harpham C, Dawson CW (2006) The effect of different basis functions on a radial basis function network for time series prediction: a comparative study. Neurocomputing 69(16–18):2161–2170
9. Wu YS, Zhao MS (2001) A neuron model with trainable activation function (TAF) and its MFNN supervised learning. Sci China (Ser F) 44(5):366–375
10. Shen YJ, Wang BW (2004) A fast learning algorithm of neural network with tunable activation function. Sci China (Ser F) 47(1):126–136
11. Shen YJ, Wang BW, Chen FG, Cheng L (2004) A new multi-output nueral model with tunable activation function and its applications. Neural Process Lett 20:85–104
12. Zhu QY, Qin AK, Suganthan PN, Huang GB (2005) Evolutionary extreme learning machine. Pattern Recogn 38:1759–1763
13. Hornik K (1991) Approximation capabilities of multilayer feedforward networks. Neural Netw 4(2):251–257
14. Huang GB, Babri HA (1998) Upper bounds on the number of hidden neurons in feedforward networks with arbitrary bounded nonlinear activation functions. IEEE Trans Neural Netw 9(1):224–229
15. Huang GB (2003) Learning capability and storage capacity of two hidden layer feedforward networks. IEEE Trans Neural Netw 14(2):274–281
16. Li B, Li YB, Rong XW (2010) Intelligent optimization strategy for ELM-RBF neural networks. J Shandong Univ (Nat Sci) 45(5):48–52
17. Huang GB, Siew CK (2005) Extreme learning machine with randomly assigned RBF kernels. Int J Inf Technol 11(1):16–24
18. Murphy PM, Aha DW (2008) UCI repository of machine learning databases [online]. Availabe: http://archive.ics.uci.edu/ml/datasets.html
19. Redondo MF, Espinosa CH (1999) Generalization capability of one and two hidden layers. In: International joint conference on neural networks, Washington DC, vol 3, pp 1840–1843
20. Wei HK, Xu SX,Song WZ (2001) Generalization theory and generalizaiton methods for neural networks. Acta Automat Sin 27(6):806–815