Memetic Comp.
DOI 10.1007/s12293-017-0238-1

REGULAR RESEARCH PAPER

Improved bidirectional extreme learning machine based on enhanced random search

Weipeng Cao[1] · **Zhong Ming**[1] · **Xizhao Wang**[1] · **Shubin Cai**[1]

Received: 14 September 2016 / Accepted: 30 June 2017

Abstract The incremental extreme learning machine (I-ELM) was proposed in 2006 as a method to improve the network architecture of extreme learning machines (ELMs). To improve on the I-ELM, bidirectional extreme learning machines (B-ELMs) were developed in 2012. The B-ELM uses the same method as the I-ELM but separates the odd and even learning steps. At the odd learning step, a hidden node is added like I-ELM. At the even learning step, a new hidden node is added via a formula based on the former added node result. However, some of the hidden nodes generated by the I-ELM may play a minor role; thus, the increase in network complexity due to the B-ELM may be unnecessary. To avoid this issue, this paper proposes an enhanced B-ELM method (referred to as EB-ELM). Several hidden nodes are randomly generated at each odd learning step, however, only the nodes with the largest residual error reduction will be added to the existing network. Simulation results show that the EB-ELM can obtain higher accuracy and achieve better performance than the B-ELM under the same network architecture. In addition, the EB-ELM can achieve a faster convergence rate than the B-ELM, which means that the EB-ELM has smaller network complexity and faster learning speed than the B-ELM.

Keywords Incremental extreme learning machine · Bidirectional extreme learning machine · Convergence rate · Network architecture

1 Introduction

Feed-forward neural networks with random weights have been studied for over 20 years [1–4]. With the introduction of extreme learning machines (ELMs) [5], this field has attracted the attention of many researchers. Extreme learning machine (ELM) is a special single-hidden layer feed-forward neural network with random weights (NNRW), where the hidden weights are randomly selected and kept fixed throughout the training process while the output weights are obtained analytically. Originally proposed by Huang et al. [5] in 2004, ELMs extreme learning speed and overall performance, has been widely used in scientific and engineering research [6]. Some notable applications include, traffic sign recognition [7], 3D shapes recognition [8], recommendation system [9], unsupervised and semi-supervised learning [10,11], electricity price forecast [12], deep architecture and its applications [13,14], etc. Although the ELM seems simple, Huang et al. [15] have proved that ELMs have universal approximation capabilities. ELMs also have a much faster training speed than traditional tuning-based learning methods, such as the back-propagation algorithm [16], with an acceptable accuracy.

One of the most intractable challenges in ELM is choosing an appropriate network architecture (i.e., the number of hidden neurons). Network architecture has a huge impact on model performance. In the original ELM algorithm, the num-

Electronic supplementary material The online version of this article (doi:10.1007/s12293-017-0238-1) contains supplementary material, which is available to authorized users.

✉ Zhong Ming
mingz@szu.edu.cn

Weipeng Cao
caoweipeng123@gmail.com

Xizhao Wang
xzwang@szu.edu.cn

Shubin Cai
shubin@szu.edu.cn

[1] College of Computer Science and Software Engineering, Shenzhen University, Shenzhen 518060, China

Published online: 27 July 2017

ber of hidden neurons is predefined by users. However, this method can't always obtain the best network architecture and further causes the ELM model to have sub-optimal performance, which likely causes under-fitting or over-fitting.

To avoid the issues, researchers have developed a series of solutions, including incremental ELM (I-ELM) [15], enhanced incremental ELM (EI-ELM) [17], bidirectional ELM (B-ELM) [18], pruned ELM (P-ELM) [19], optimally pruned (OP-ELM) [20], adaptive growth ELM (AG-ELM) [21], error minimized ELM (EM-ELM) [22], and subnetwork ELM [23], in addition to other methods. An I-ELM [15] randomly adds hidden nodes one by one and freezes the output weights of the existing hidden nodes when a new hidden node is added. An EI-ELM [17] is often seen as an enhanced I-ELM. The main difference between an EI-ELM and an I-ELM is that at each learning step in an EI-ELM, several hidden nodes are randomly generated, but only the one node that leads to a minimized residual error will be added to the existing network. A B-ELM [18] divides the learning step into two parts. When the number of hidden nodes in the existing network is even, the new node will be added based on the I-ELM method. While when the number of hidden nodes is odd, the parameters of the new node will be calculated via a set of formulas. A P-ELM [19] uses statistical methods to measure the relevance of neurons and then prunes the irrelevant nodes based on their relevance to class labels. An OP-ELM [20] uses a leave-one-out (LOO) criterion to optimize the P-ELM and extend the application to regression problems. In an AG-ELM [21], the networks are generated within a group at each step. One or more existing hidden nodes may be replaced by newly generated networks, which have fewer hidden nodes and better performance. Thus, the number of hidden nodes in the AG-ELM network changes dynamically. The hidden nodes in an EM-ELM [22] can be added one by one or group by group. Unlike I-ELMs and EI-ELMs, the output weights in an EM-ELM are updated as new hidden nodes are added. Subnetwork-ELMs [23] and B-ELMs both try to pull back the network residual error to the hidden layer. The B-ELM increases hidden nodes one by one, while in the Subnetwork-ELM, the hidden nodes can be a subnetwork (a subnetwork formed by several nodes). Thus, unlike a B-ELM, the input weight in a Subnetwork-ELM is a matrix and the subnetwork hidden nodes are calculated and not generated randomly. The common goal of these methods is to get a better network architecture for the ELM, to get better performance and faster convergence rate.

This paper proposes an improved bidirectional extreme learning machine based on enhanced random search (EB-ELM), where the enhanced random search method is incorporated into a B-ELM. In an EB-ELM, at the odd learning step several hidden nodes are randomly generated and only the one leading to the largest residual error reduction will be added to the existing network. In this way, the reliability of the B-ELM algorithm will be improved. Simulation results on ten benchmark regression problems show that an EB-ELM can obtain better generalization performance and has a faster convergence rate, and we can get a more compact network with an EB-ELM than with a B-ELM or an EI-ELM.

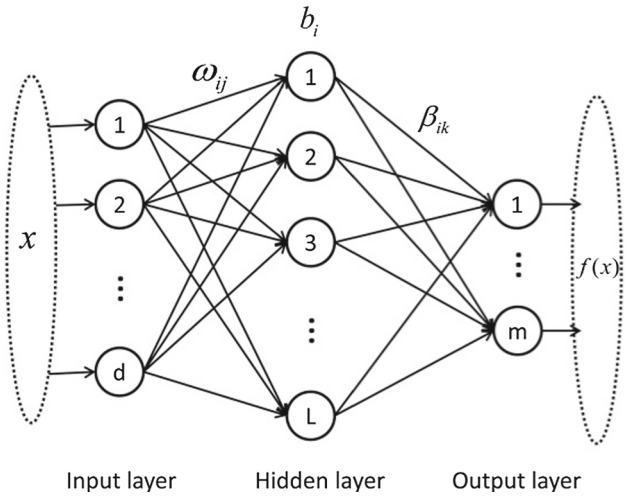

Fig. 1 Basic network structure of an ELM

The organization of this paper is as follows: Sect. 2 briefly introduces the ELM, I-ELM, EI-ELM, and B-ELM methods. The details of the EB-ELM algorithm are proposed in Sect. 3. Section 4 describes the experimental procedure and simulation results. The conclusions are provided in Sect. 5.

2 Review of extreme learning machines

A basic network structure for ELM has three layers, including an input layer, hidden layer, and output layer. Any two adjacent layers are connected via weight parameters as shown in Fig. 1:

In the figure, the term d denotes the number of input neurons, L denotes the number of hidden neurons, and $m$denotes the number of output neurons. The output of ELM is expressed as follows:

$$\sum_{i=1}^{L} \beta_i g(\omega_i \cdot x_j + b_i) = t_j,\ \omega_i \in R^n, b_i \in R,\ j = 1, \ldots, N \tag{1}$$

In Eq. (1), β denotes the weight between hidden and output layers, $g(\omega_i \cdot x_j + b_i)$denotes the output of the ith hidden node with parameters $(\omega_i,\ b_i)$, L denotes the number of hidden nodes, t_j denotes the computational value of ELM, and N is the size of the dataset. Equation (1) can be re-written simply as

$$H\beta = T, \tag{2}$$

where $H = \begin{pmatrix} g(\omega_1 \cdot x_1 + b_1) & \dots & g(\omega_L \cdot x_1 + b_L) \\ \vdots & \ddots & \vdots \\ g(\omega_1 \cdot x_N + b_1) & \cdots & g(\omega_L \cdot x_N + b_L) \end{pmatrix}_{N \times L}$,

$\beta = \begin{bmatrix} \beta_1^T \\ \vdots \\ \beta_L^T \end{bmatrix}_{L \times m}$, $T = \begin{bmatrix} t_1^T \\ \vdots \\ t_N^T \end{bmatrix}_{N \times m}$.

In Eq. (2), H represents the hidden layer output matrix of ELM and T denotes the expected output matrix. The residual error is used to measure the closeness between output function f_n and the target function f, which can be expressed by

$$e_n = f_n - f \tag{3}$$

In the I-ELM algorithm, Huang proposes a method to add hidden nodes one by one and fix the output weights of existing hidden nodes when adding a new hidden node [15]. In this way, the hidden node can be added automatically until the ELM model satisfies the given precision or the number of hidden nodes beyond the given maximum. The output function f_n at the n_{th}step can be expressed by the equation

$$f_n(x) = f_{n-1}(x) + \beta_n G_n(x), \tag{4}$$

where the β_n denotes the weight between the new hidden node and the output nodes, and $G_n(x)$ is the corresponding output of the hidden node at the n-th step.

The network architecture generated by the I-ELM may be very complex because some of the hidden nodes may play a minor role in the network output.

To avoid this problem, Huang et al. [17] and Yang et al. [18] proposed the EI-ELM and B-ELM, respectively. In the EI-ELM algorithm, Huang generates several hidden nodes randomly at each learning step; only the optimal hidden node, which leads to the smallest residual error, is chosen. Obviously, this method can pick a better hidden node than the I-ELM, with a higher probability. As expected, the method can achieve a faster convergence rate and a more compact network architecture [17].

In the B-ELM algorithm, Yang divides the training process into two parts [18]. When the number of hidden nodes $L \in \{2n+1, n \in Z\}$, the hidden node parameter $(\omega_i,\ b_i)$ is generated randomly, as in the I-ELM. When the number of hidden nodes $L \in \{2n, n \in Z\}$, the hidden node parameter $(\omega_i,\ b_i)$ is obtained according to the following formulas.

$$\hat{\omega} 2n = g^{-1}(u(H_{2n})) \cdot x^{-1} \tag{5}$$

$$\hat{b} 2n = \sqrt{mse(g^{-1}(u(H_{2n})) - \omega_{2n} \cdot x)} \tag{6}$$

$$\hat{H} 2n = u^{-1}(g^{-1}(\omega_{2n} \cdot x + b_{2n})) \tag{7}$$

where g^{-1} and u^{-1} represent the inverse functions of g and u, respectively.

We can infer that the ELM model accuracy is sharply affected by parameters from the odd hidden nodes in the B-ELM. In the learning process of the B-ELM, some newly added hidden nodes only play a minor role in the final network, and thus may increase the network complexity without an appropriate advantage.

To avoid this issue, we take advantage of the EI-ELM to improve the B-ELM. At each odd learning step of the B-ELM, several hidden nodes are randomly generated and we only choose to add the node that leads to the largest residual error reduction to the existing network.

3 The proposed algorithm

In this section, we propose an improved bidirectional extreme learning machine that is based on an enhanced random search (EB-ELM). In the EB-ELM, we use an enhanced random search method originated in the EI-ELM and incorporate it into the B-ELM.

In the EB-ELM, when the number of hidden nodes $L \in \{2n+1, n \in Z\}$, we first generate k random hidden nodes and calculate the corresponding hidden layer output, which are denoted by $\{H_1,\ H_2, \dots, H_k\}$, respectively. Then, the corresponding residual errors, denoted by $\{E_1, E_2, \dots, E_k\}$, are determined. Finally, we choose the node that leads to the $\min\{E_1, E_2, \dots, E_k\}$ as the newly added hidden node. It should be noted that the B-ELM is a specific case of the EB-ELM when $k = 1$.

With number of hidden node $L \in \{2n, n \in Z\}$, we use the same method presented in the B-ELM. The proposed EB-ELM algorithm can be summarized as follows:

EB-ELM Algorithm: Given a training set $\{(x_i,t_i)\}_{i=1}^{N} \subset R^n \times R$, the hidden-node output function $G(\omega,x,b)$, with a maximum number $L_{\max}$ of hidden nodes, with expected learning accuracy ε, and a maximum number k of trials of assigning randomly hidden nodes at each step.

Step 1 Initialization: Let the number of hidden nodes $L=0$ and the residual error be $E=T$.

Step 2 Learning step:

while $L<L_{\max}$ and $\|E\|>\varepsilon$ **do**

(a) Increase the number of hidden nodes $L: L=L+1$ by one.

(b) if $L\in\{2n+1, n\in Z\}$ **then**

for $j=1:k$

Assign random parameters $(\omega_{(j)},b_{(j)})$ for the new hidden node L. The random parameters will satisfy any continuous sampling distribution probability.

Calculate the output weight $\beta_{(j)}$ for the new hidden node according to the relation

$$\beta_{2n+1}=\frac{\langle e_{2n},H_{2n+1}\rangle}{\|H_{2n+1}\|^2}.$$

Calculate the residual error E after adding the new hidden node $L: E_{(j)}=E-H_L\bullet\beta_{(j)}$.

endfor

(c) Let $j^*=\{j\,|\,\min_{1\le j\le k}\|E_{(j)}\|\}$. Set $\omega_L=\omega_{(j^*)}$, $b_L=b_{(j^*)}$, $E=E_{(j^*)}$, and $\beta_L=\beta_{(j^*)}$.

endif

(d) if $L\in\{2n, n\in Z\}$ **then**

Calculate the error feedback function sequence H_L according to the equation

$$H_{2n} = e_{2n-1} \bullet (\beta_{2n-1})^{-1}.$$

Calculate the input weight ω_L, bias b_L and update H_L for the new hidden node L based on (5),(6), and (7).

Calculate the output weight β_L for the new hidden node according to the equation

$$\beta_{2n} = \frac{\langle e_{2n-1}, H_{2n} \rangle}{\|H_{2n}\|^2}.$$

Calculate E after adding the new hidden node L: $E = E - H_L \beta_L$.

end if

end while

4 Experimental verification

In this section, we compare the performance of the EB-ELM, B-ELM and EI-ELM on ten benchmark regression problems from the UCI database [24]. The specification of these benchmark problems is given in Table 1. In our experiment, the input data is normalized into $[-1, 1]$. The weight ω, between input layer and hidden layer, is randomly chosen from $[-1, 1]$, while the threshold b of hidden nodes is chosen from $[0, 1]$ based on a uniform sampling distribution probability. The Sigmoid function is chosen as the activation function of all the algorithms, i.e., $G(\omega, x, b) = 1/(1 + \exp(-(\omega \cdot x + b)))$. The simulations are conducted in the MATLAB 2008a environment and the same Windows 7 machine with Intel Core i7 3.60 GHz CPU and 8 GB RAM. For each problem, the average results over 50 trials are obtained for the EB-ELM, B-ELM, and EI-ELM.

Table 1 Specification of ten benchmark regression problems

Name	Training data	Testing data	Attributes
Airfoil Self-noise	750	753	5
Combined cycle power plant	4500	5068	4
Concrete compressive strength	500	530	8
Energy efficiency	400	368	8
Housing	250	256	13
Abalone	2000	2177	8
Wine quality	2000	2898	11
Auto MPG	200	192	8
Machine CPU	100	109	6
Gas Turbines	6000	5934	16

The generalization performance of the EB-ELM, B-ELM, and EI-ELM, with the same number of hidden nodes, are first compared with ten real benchmark problems: Airfoil Self-noise, Combined Cycle Power Plant [25], Concrete Compressive Strength [26], Energy efficiency [27], Housing, Abalone, Wine Quality [28], Auto MPG, Machine CPU, and Gas Turbines [29]. We set 200 as the maximum number of hidden nodes in every algorithm, and we suppose ten hidden nodes are randomly generated at each step of the EB-ELM and EI-ELM, i.e., $k = 10$. The average performance [the training time, training root mean square error (Training RMSE) and the testing root mean square error (Testing RMSE)] of the EB-ELM, B-ELM, and EI-ELM are shown in Table 2. The closed results are underlined and the improved results are in boldface.

As shown in Table 2, the EB-ELM obtains a smaller training RMSE and testing RMSE than the B-ELM under the same node in most cases, compared with the B-ELM using

the Combined Cycle Power Plant Data Set case. In addition, the EB-ELM produces a smaller standard deviation of the testing RMSE as opposed to the B-ELM in most cases, which means the performance of the EB-ELM is more stable than the B-ELM. In other words, the EB-ELM obtains higher accuracy and better generalization performance than the B-ELM under the same network architecture. Compared with the EI-ELM, the EB-ELM achieves better performance and a smaller standard deviation in most cases, except the Energy efficiency and Gas Turbines cases.

Furthermore, Fig. 2 shows the training RMSE with updated curves for the EB-ELM, B-ELM, and EI-ELM on Housing data. From Fig. 2, we can infer that the EB-ELM achieves a faster convergence rate than the two algorithms, which means the EB-ELM can obtain the most compact network architecture compared with the B-ELM and EI-ELM, under the same accuracy. Similar results can be found in other cases.

5 Conclusions

As described above, the performance of B-ELM model is drastically affected by parameters generated at the odd learning step. That is, some of odd hidden nodes generated in B-ELM play a minor role and may cause the unnecessary increase in network complexity. To deal with this problem, in this paper, we proposed an improved bidirectional extreme learning machine based on enhanced random search (EB-ELM) in which an enhanced random search method is incorporated into B-ELM. In EB-ELM, several hidden nodes

Table 2 Performance comparison of the EB-ELM, B-ELM, and EI-ELM with 200 hidden nodes

Datasets	Algorithm	Testing time (s)	SD	Mean	
				Training RMSE	Testing RMSE
Airfoil Self-noise	**EB-ELM**	0.0178	**0.0023**	0.0709	**0.0726**
	B-ELM	0.0197	0.0027	0.0723	0.0745
	EI-ELM	0.0056	0.3161	0.0472	0.2574
Combined cycle power plant	**EB-ELM**	0.0602	**0.0002**	0.0233	**0.0233**
	B-ELM	0.0599	0.0002	0.0233	0.0233
	EI-ELM	0.0625	0.0934	0.0213	0.0370
Concrete compressive strength	**EB-ELM**	0.0175	**0.0021**	0.0217	**0.0233**
	B-ELM	0.0159	0.021	0.0229	0.0316
	EI-ELM	0.0075	0.0209	0.0076	0.0433
Energy efficiency	**EB-ELM**	0.0172	0.0004	0.0092	0.0093
	B-ELM	0.0159	0.0016	0.0094	0.0097
	EI-ELM	0.0044	**0.0001**	0.0011	**0.0019**
Housing	**EB-ELM**	0.0178	**0.0011**	0.0178	**0.0195**
	B-ELM	0.014	0.0061	0.0217	0.0233
	EI-ELM	0.0044	0.1363	0.0043	0.2089
Abalone	**EB-ELM**	0.9928	**0.0070**	0.0092	**0.0161**
	B-ELM	1.0366	0.0155	0.0132	0.0168
	EI-ELM	0.0294	0.0071	0.0004	0.0168
Wine quality_ white	**EB-ELM**	0.0421	**0.0014**	**0.0149**	**0.0163**
	B-ELM	0.0443	0.002	0.0158	0.0173
	EI-ELM	0.0469	45.5285	0.0108	24.2822
Auto MPG	**EB-ELM**	0.9969	**0.0002**	0.0026	**0.0027**
	B-ELM	1.0394	0.0005	0.0026	0.0029
	EI-ELM	0.0003	0.0691	0.0012	0.0488
Machine CPU	**EB-ELM**	0.0641	**0.0074**	0.0145	**0.0220**
	B-ELM	0.0744	0.0570	0.0161	0.0421
	EI-ELM	0.0006	2.8013	0.0069	1.9659
GT_Turbine	**EB-ELM**	1.0781	0.0006	0.0003	0.0004
	B-ELM	1.0994	0.0084	0.0054	0.0056
	EI-ELM	0.0869	<**0.0001**	<0.0001	<**0.0001**

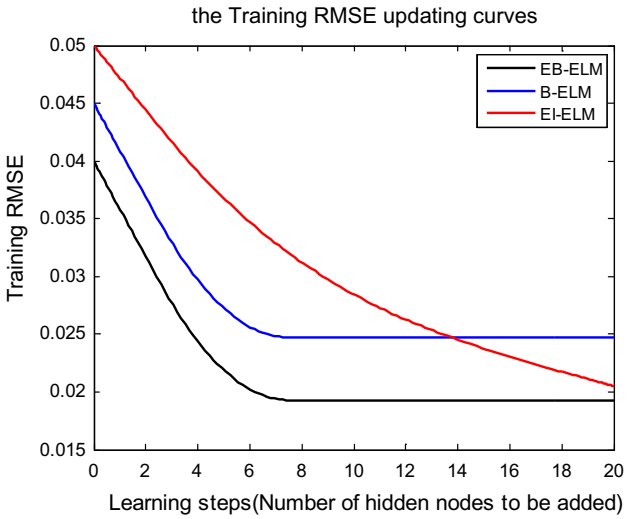

Fig. 2 The Training RMSE Updating Curves of the EB-ELM, B-ELM, and EI-ELM

are randomly generated at each odd learning step, however, only the nodes with the largest residual error reduction are added to the existing network. The parameters of even hidden nodes are calculated in a same way as in the original B-ELM. It should be noted that B-ELM is a specific case of EB-ELM when k= 1. According to the simulation results of ten benchmark regression problems, we can find that the EB-ELM achieves superior performance and smaller standard deviation when compared with the B-ELM and EI-ELM. In addition, the EB-ELM has the fastest convergence rate among the three algorithms, which means that the EB-ELM can achieve a much more compact network architecture. In the future, we will construct a multi-layer EB-ELM and apply it to more complex problems. In addition, the performance of the EB-ELM using additional hidden nodes will be reported in future work.

Acknowledgements The authors would like to thank the editor and reviewers for their invaluable suggestions to improve the quality of this paper. This research is supported by the National Natural Science Foundation of China under Grant No. 61672358.

References

1. Schmidt WF, Kraaijveld MA, Duin RPW (1992) Feedforward neural networks with random weights. In: Proceedings of the 11th IAPR international conference on pattern recognition. doi:10.1109/ICPR.1992.201708
2. Pao YH, Takefuji Y (1992) Functional-link net computing: theory, system architecture, and functionalities. Computer 25(5):76–79
3. Igelnik B, Pao YH (1995) Stochastic choice of basis functions in adaptive function approximation and the functional-link net. IEEE Trans Neural Netw 6(6):1320–1329
4. Zhang L, Suganthan PN (2016) A comprehensive evaluation of random vector functional link networks. Inf Sci 367–368:1094–1105
5. Huang GB, Zhu QY, Siew CK (2004) Extreme learning machine a new learning scheme of feedforward neural networks. In: Proceedings of 2004 IEEE international joint conference on neural networks, vol 2. IEEE, pp 985–990
6. Huang G, Huang GB, Song SJ, You KY (2015) Trends in extreme learning machines: a review. Neural Netw 61:32–48
7. Huang ZY, Yu YL, Gu J, Liu HP (2016) An efficient method for traffic sign recognition based on extreme learning machine. IEEE Trans Cybern 47(4):920–933
8. Xie ZG, Xu K, Shan W, Liu LG, Xiong YS, Huang H (2015) Projective feature learning for 3D shapes with multi-view depth images. Comput Graph Forum 34(7):1–11
9. Zhao XG, Ma ZY, Zhang Z (2017) A novel recommendation system in location-based social networks using distributed ELM. Memet Comput. doi:10.1007/s12293-017-0227-4
10. Zhang N, Ding SF (2017) Unsupervised and semi-supervised extreme learning machine with wavelet kernel for high dimensional data. Memet Comput 9(2):129–139
11. Das SP, Padhy S (2016) Unsupervised extreme learning machine and support vector regression hybrid model for predicting energy commodity futures index. Memet Comput. doi:10.1007/s12293-016-0191-4
12. Xiao CX, Dong ZY, Xu Y, Meng K, Zhou X, Zhang X (2016) Rational and self-adaptive evolutionary extreme learning machine for electricity price forecast. Memet Comput 8(3):223–233
13. Tissera MD, McDonnell MD (2016) Deep extreme learning machines: supervised autoencoding architecture for classification. Neurocomputing 174(22):42–49
14. Liu HP, Li FX, Xu XY, Sun FC (2017) Active object recognition using hierarchical local-receptive-field-based extreme learning machine. Memet Comput. doi:10.1007/s12293-017-0229-2
15. Huang GB, Chen L, Siew CK (2006) Universal approximation using incremental constructive feedforward networks with random hidden nodes. IEEE Trans Neural Netw 17(4):879–892
16. LeCun Y, Boser B, Denker JS, Howard RE, Hubbard W, Jackel LD, Henderson D (1989) Handwritten digit recognition with a back-propagation network. In: Advances in Neural Information Processing Systems, Kaufmann, San Francisco, CA, USA, pp 396–404
17. Huang GB, Chen L (2008) Enhanced random search based incremental extreme learning machine. Neurocomputing 71(s 16–18):3460–3468
18. Yang YM, Wang YN, Yuan XF (2012) Bidirectional extreme learning machine for regression problem and its learning effectiveness. IEEE Trans Neural Netw Learn Syst 23(9):1498–1505
19. Rong HJ, Ong YS, Tan AH, Zhu Z (2008) A fast pruned-extreme learning machine for classification problem. Neurocomputing 72(1):359–366
20. Miche Y, Sorjamaa A, Bas P, Simula O, Jutten C, Lendasse A (2010) OPELM: optimally pruned extreme learning machine. IEEE Trans Neural Netw 21(1):158–162
21. Zhang R, Lan Y, Huang GB, Xu ZB (2012) Universal approximation of extreme learning machine with adaptive growth of hidden nodes. IEEE Trans Neural Netw Learn Syst 23(2):365–371
22. Feng GR, Huang GB, Lin QP, Gay R (2009) Error minimized extreme learning machine with growth of hidden nodes and incremental learning. IEEE Trans Neural Netw 20(8):1352–1357
23. Yang YM, Wu JQM (2016) Extreme learning machine with subnetwork hidden nodes for regression and classification. IEEE Trans Cybern 46(12):2570–2583
24. Lichman M (2013) UCI machine learning repository. http://archive.ics.uci.edu/ml. University of California, School of Information and Computer Science, Irvine
25. Tüfekci P (2014) Prediction of full load electrical power output of a base load operated combined cycle power plant using machine learning methods. Int J Electr Power Energy Syst 60:126–140

26. Yeh IC (1998) Modeling of strength of high performance concrete using artificial neural networks. Cem Concr Res 28(12):1797–1808
27. Tsanas A, Xifara A (2012) Accurate quantitative estimation of energy performance of residential buildings using statistical machine learning tools. Energy Build 49:560–567
28. Cortez P, Cerdeira A, Almeida F, Matos T, Reis J (2009) Modeling wine preferences by data mining from physicochemical properties. Decis Support Syst 47(4):547–553
29. Coraddu A, Oneto L, Ghio A, Savio S, Anguita D, Figari M (2014) Machine learning approaches for improving condition-based maintenance of naval propulsion plants. J Eng Marit Environ 230(8):136–153