A Brain Network Inspired Algorithm: Pre-trained Extreme Learning Machine

Yongshan Zhang[1], Jia Wu[2], Zhihua Cai[1(✉)], and Siwei Jiang[1]

[1] Department of Computer Science, China University of Geosciences, Wuhan 430074, China
{yszhang,zhcai}@cug.edu.cn
[2] Department of Computing, Faculty of Science and Engineering, Macquarie University, Sydney, NSW 2109, Australia
jia.wu@mq.edu.au

Abstract. Extreme learning machine (ELM) is a promising learning method for training "generalized" single hidden layer feedforward neural networks (SLFNs), which has attracted significant interest recently for its fast learning speed, good generalization ability and ease of implementation. However, due to its manually selected network parameters (e.g., the input weights and hidden biases), the performance of ELM may be easily deteriorated. In this paper, we propose a novel pre-trained extreme learning machine (P-ELM for short) for classification problems. In P-ELM, the superior network parameters are pre-trained by an ELM-based autoencoder (ELM-AE) and embedded with the underlying data information, which can improve the performance of the proposed method. Experiments and comparisons on face image recognition and handwritten image annotation applications demonstrate that P-ELM is promising and achieves superior results compared to the original ELM algorithm and other ELM-based algorithms.

Keywords: Extreme learning machine · ELM-based autoencoder · Pre-trained parameter · Classification

1 Introduction

Extreme learning machine (ELM) [1] is a useful learning method for training "generalized" single hidden layer feedforward neural networks (SLFNs), which shows its good performance in various research studies [2]. Compared with traditional neural networks which adjust the network parameters iteratively, in ELM, the input weights and hidden layer biases are randomly generated, while the output weights are analytically determined by using Moore-Penrose (MP) generalized inverse. Due to its extremely fast learning speed, good generalization

The original version of this chapter was revised. The title of the paper has been corrected. The erratum to this chapter is available at https://doi.org/10.1007/978-3-319-70139-4_94

D. Liu et al. (Eds.): ICONIP 2017, Part V, LNCS 10638, pp. 14–23, 2017.
https://doi.org/10.1007/978-3-319-70139-4_2

ability and ease of implementation, ELM has drawn great attention in academia [3,4]. However, the manually assigned network parameters often degrade the performance of ELM.

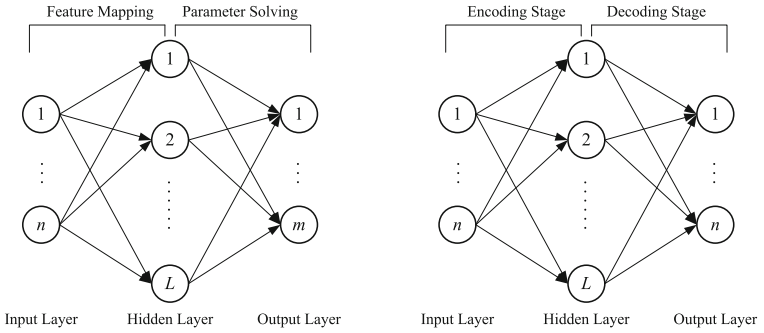
In order to enhance the performance of ELM, researchers have proposed a number of improved methods from different perspectives, such as ensemble learning [5], voting scheme [6], weighting method [7] and instance cloning [4]. Liu and Wang [5] embedded ensemble learning into the training phase of ELM to mitigate the overfitting problem and improve the predictive stability. Cao et al. [6] incorporated multiple independent ELM models into a unified framework to enhance the performance in a voting manner. Zong et al. [7] proposed a weighting scheme method for ELM by assigning different weights for each example. The aforementioned methods for ELM have achieved good performance in some specific problems. However, they do not solve the primary problem in ELM (i.e., the random generation of the network parameters). Therefore, the performance of the above-mentioned methods may be compromised. How to select suitable network parameters for ELM is still an opening problem.

In reality, the original data can provide valuable information according to its different representations. Therefore, it is imperative for ELM to determine the network parameters based on the original data. A straightforward approach to solve the above problem is to use the idea of autoencoder. Autoencoder [8] is a special case of artificial neural network usually used for unsupervised learning, where the output layer are with the same neurons as the input layer. In autoencoder, the learning procedure can be divided into the processes of encoding and decoding [9,10]. The input data is mapped to a high-level representation in the encoding stage, while the high-level representation is mapped back to the original input data in the decoding stage. By doing so, autoencoder can explore the underlying data information and encode these information into the output weights.

Based on the above observations, in this paper, we propose a novel pre-trained extreme learning machine (P-ELM for short), where an ELM-based autoencoder (ELM-AE) is adopted to pre-train the suitable network parameters. The proposed P-ELM encodes the data information into the learned network parameters, which can achieve satisfactory performance for further learning. Experiments on face image recognition and handwritten image annotation applications demonstrate that the proposed P-ELM consistently outperforms other state-of-the-art ELM algorithms. The advantages of P-ELM can be summarized as follows:

- P-ELM falls into the category of data-driven methods, which can successfully find the proper network parameters for further learning.
- P-ELM is simple in both theory and implementation, which inherits the advantages of the original ELM.
- P-ELM is a nonlinear learning model and flexible in modeling different complex real-world relationships.

The remainder of the paper is structured as follows. Section 2 surveys the related work. Section 3 presents the proposed P-ELM method. The experiments are demonstrated in Sect. 4. Finally, we conclude the paper in Sect. 5.



(a) ELM (b) ELM-AE

Fig. 1. Illustration of network structures for (a) ELM and (b) ELM-AE.

2 Related Work

Extreme learning machine (ELM) is an elegant learning method, which was originally proposed for SLFNs and then extended to "generalized" SLFNs [1]. In ELM, the hidden neurons need not be neuron alike and the networks parameters are without iterative tunning. The network structure of ELM is shown in Fig. 1(a). The basic ELM can fundamentally be regarded as a two-stage learning system, which can be spilt into feature mapping and parameter solving [11,12]. In the feature mapping stage, ELM randomly selects the input weights and hidden biases to calculate the hidden layer output matrix via an activation function. In the parameter solving stage, the output weights are analytically determined according to the Moore-Penrose (MP) generalized inverse and the smallest norm least-squares solution of general linear system. To accelerate the learning speed, Huang et al. [13] presented a constrained-optimization-based ELM and provided two effective solution for different size of training data. The learning theories and real-world applications of ELM are well-developed in the literature [2].

Apart from ELM-based SLFNs, the ELM theories can be also applied to built an ELM-based autoencoder (ELM-AE) [14]. Autoencoder is always used to be a feature extractor and usually functions as a basic unit in a multilayer learning model [15]. In recent years, autoencoder has been widely used for tackling numerous real-world applications, e.g., cross-language learning problem and domain adaption problem. Similar to the ELM, an ELM-AE can be also regarded as a two-stage process, where the input data is first mapped to a high-level representation, and then the high-level latent representation is mapped back to the original input data [16]. The network structure of ELM-AE is shown in Fig. 1(b). The main difference between ELM and ELM-AE is the output layer. In ELM, the output layer is to predict the target value for given data. By contrast, in ELM-AE, the output layer is to reconstruct the original input data. Due to the unique learning mechanism, ELM-AE extracts the informative features through the hidden layer and encodes the underlying data information into the output

weights. Motivated by these, we propose to employ an ELM-AE to pre-train the network parameters for P-ELM in this paper.

3 Proposed Method

In this section, we present the proposed pre-trained extreme learning machine (P-ELM). Specifically, P-ELM is achieved through the following steps: (1) Employ an ELM-AE for network parameter learning; (2) Train the P-ELM model with the learned network parameters; and (3) Predict the class labels of the testing instances. Algorithm 1 reports the learning process of the proposed P-ELM.

3.1 Parameter Learning

In P-ELM, the most important aspect is to choose the suitable network parameters based on the original data. To this end, we use an ELM-AE to learn the network parameters. Given N distinct training examples $\mathcal{D}= \{(\boldsymbol{x}_i, \boldsymbol{t}_i)\}_{i=1}^{N}$, where $\boldsymbol{x}_i \in \mathbb{R}^n$ is the input data and $\boldsymbol{t}_i \in \mathbb{R}^m$ is the expectation output, the encoding process in ELM-AE with L hidden neurons can be presented as the following equation:

$$\boldsymbol{h}(\boldsymbol{x}_i) = g(\boldsymbol{\alpha} \cdot \boldsymbol{x}_i + \boldsymbol{b}), \quad i = 1, 2, ..., N; \tag{1}$$

where $\boldsymbol{\alpha} \in \mathbb{R}^{L \times n}$ is the input weight matrix, $\boldsymbol{b} \in \mathbb{R}^{L \times 1}$ is the hidden neuron bias vector, $g(\cdot)$ is an activation function, and $\boldsymbol{h}(\boldsymbol{x}_i)$ is the high-level latent representation for the input data $\boldsymbol{x}_i$. By contrast, the decoding process in ELM-AE can be formulated as follows:

$$\boldsymbol{h}(\boldsymbol{x}_i)\boldsymbol{\varpi} = \boldsymbol{x}_i, \quad i = 1, 2, ..., N; \tag{2}$$

where $\boldsymbol{\varpi} \in \mathbb{R}^{L \times n}$ is the output weight matrix. Equation (2) can be also rewritten as the compacted form based on the whole dataset:

$$\mathbf{H}\boldsymbol{\varpi} = \mathbf{X}. \tag{3}$$

To enhance the performance of ELM-AE, the output weight matrix $\boldsymbol{\varpi}$ can be updated by minimizing the objective fuction: $L(\boldsymbol{\varpi}) = \frac{1}{2}||\boldsymbol{\varpi}||^2 + \frac{C}{2}||\mathbf{X} - \mathbf{H}\boldsymbol{\varpi}||^2$. The calculation of the output weight matrix $\boldsymbol{\varpi}$ can be solved by Eq. (4) according to the relationship between the number of training samples N and the number of hidden neurons L.

$$\boldsymbol{\varpi} = \begin{cases} \left(\dfrac{\mathbf{I}}{C} + \mathbf{H}^T\mathbf{H}\right)^{-1}\mathbf{H}^T\mathbf{X}, & if \;\; N \geq L \\ \mathbf{H}^T\left(\dfrac{\mathbf{I}}{C} + \mathbf{H}\mathbf{H}^T\right)^{-1}\mathbf{X}, & if \;\; N < L \end{cases} \tag{4}$$

The unique parameter learning mechanism enables ELM-AE to encode the underlying information of the original data into the output weights, which can be used as the input weights for the P-ELM model to achieve better performance. This is a data-driven method, which can adaptively search the suitable network parameters based on the specific data.

Algorithm 1. Pre-trained Extreme Learning Machine (P-ELM)

Input:

Training dataset $\mathcal{D}= \{(\boldsymbol{x}_i, \boldsymbol{t}_i)\}_{i=1}^{N}$; Any testing instance $\boldsymbol{x}^{test} \in \mathcal{D}^{test}$;
Activation function $g(\cdot)$; Number of hidden neurons L; Parameter C;

Output:

The predicted class label $c(\boldsymbol{x}^{test})$ of testing instance $\boldsymbol{x}^{test}$;

//P-trained Parameter Learning:

1: Randomly assign the input weights $\boldsymbol{\alpha}$ and hidden biases $\boldsymbol{b}$ for ELM-AE;
2: Calculate the hidden layer output $\mathbf{H}$ in ELM-AE by Eq. (1)
3: Calculate the output weights $\boldsymbol{\varpi}$ in ELM-AE by Eq. (4);

//P-ELM Model Training:

4: Compute the input weights as $\boldsymbol{\varpi}^T$ and the hidden biases as $\boldsymbol{b}'$, where the ith hidden layer bias $b'_i = (\sum_{j=1}^{n} \varpi_{ij})/n, i = 1, 2, ..., L$ in P-ELM;
5: Calculate the hidden layer output matrix $\mathbf{H}'$ in P-ELM by Eq. (5);
6: Calculate the output weights $\boldsymbol{\beta}$ in P-ELM by Eq. (7);

//Instance Label Prediction:

7: Predict the underlying class label $c(\boldsymbol{x}^{test})$ for testing instance $\boldsymbol{x}^{test}$;
8: Return the class label $c(\boldsymbol{x}^{test})$.

3.2 Model Training

In this section, we aim to formulate the learning model of the proposed pre-trained extreme learning machine (P-ELM). As described in the previous section, we use the output weights $\boldsymbol{\varpi}$ learned by ELM-AE as the input weights for P-ELM. In P-ELM, the input weights can be represented as $\boldsymbol{\varpi}^T$, and the hidden layer biases can be expressed as $\boldsymbol{b}'$, where the ith hidden layer bias is $b'_i = (\sum_{j=1}^{n} \varpi_{ij})/n, i = 1, 2, ..., L$. Therefore, the proposed P-ELM with L hidden neurons can be formulated as:

$$\begin{aligned} \boldsymbol{t}_i &= \sum\nolimits_{j=1}^{L} \boldsymbol{\beta}_j g(\boldsymbol{\varpi}_j^T \cdot \boldsymbol{x}_i + b'_j) \\ &= g(\boldsymbol{\varpi}^T \cdot \boldsymbol{x}_i + \boldsymbol{b}')\boldsymbol{\beta} \\ &= \boldsymbol{h}'(\boldsymbol{x}_i)\boldsymbol{\beta} \end{aligned} \quad , \quad i = 1, 2, ..., N; \tag{5}$$

where $\boldsymbol{h}'(\boldsymbol{x}_i) = g(\boldsymbol{\varpi}^T \cdot \boldsymbol{x}_i + \boldsymbol{b}')$ is the hidden layer output for the input data $\boldsymbol{x}_i$ and $\boldsymbol{\beta} \in \mathbb{R}^{L \times m}$ is the output weight matrix of the proposed P-ELM. Mathematically, Eq. (5) can be rewritten as the following compacted form:

$$\mathbf{H}'\boldsymbol{\beta} = \mathbf{T}. \tag{6}$$

To calculate the output weight matrix $\boldsymbol{\beta}$, Eq. (6) can be solved by minimizing the objective function: $L(\boldsymbol{\beta}) = \frac{1}{2}||\boldsymbol{\beta}||^2 + \frac{C}{2}||\mathbf{T} - \mathbf{H}'\boldsymbol{\beta}||^2$. Similar to Eq. (4), the output weight matrix $\boldsymbol{\beta}$ can be calculated as the following equation according to the relationship between the number of training samples N and the number of hidden neurons L.

$$\boldsymbol{\beta} = \begin{cases} \left(\frac{\mathbf{I}}{C} + \mathbf{H'}^T\mathbf{H'}\right)^{-1}\mathbf{H'}^T\mathbf{T}, & if \ \ N \geq L \\ \mathbf{H'}^T\left(\frac{\mathbf{I}}{C} + \mathbf{H'}\mathbf{H'}^T\right)^{-1}\mathbf{T}, & if \ \ N < L \end{cases} \tag{7}$$

The training process of P-ELM is determined by Eq. (5). Different from the traditional ELM with randomly generated network parameters, the proposed P-ELM uses the network parameters pre-trained by ELM-AE for model training. By doing so, the performance of P-ELM can be improved. This is the major difference between P-ELM and the original ELM.

3.3 Label Prediction

In the testing phase, the class labels of each testing instance is predicted by the trained P-ELM model. The testing instances are used to calculate the output of hidden layer based on the pre-trained input weights and hidden layer biases. Then, the class labels of the testing instances can be determined by Eq. (6). Indeed, instance label prediction in the proposed P-ELM is similar to the prediction process in ELM.

4 Experimental Results

To validate the performance of the proposed method, the experiments are conducted on face image recognition [17] and handwritten image annotation [14] respectively. Classification accuracy [18,19] and running time [20] are used as the evaluation metrics. The reported results are based on 10-fold cross validation (CV). In P-ELM, the parameter C is tuned by a grid-search strategy from $\{0.01, 0.1, 1, 10, 100, 1000\}$, the sigmoid function is applied as the activation function for the hidden layer, and the setting of the number of hidden neurons depends on specific applications. For comparison purposes, we use four ELM-based methods compared to P-ELM, including a faster ELM method (ELM) [13], ensemble based ELM (EN-ELM) [5], voting based ELM (V-ELM) [6] and weighting based ELM (W-ELM) [7].

4.1 Face Image Recognition

In this section, we report the performance of P-ELM on face image recognition real-world application. The corresponding datasets used in the experiments are the ORL and Yale face image recognition datasets[1]. The ORL dataset contains 400 face images with the size of 32×32, which belongs to 10 different people. These images were taken at different times, varying the lighting, facial expressions and facial details. The Yale dataset has 165 face images with the size of 32×32 of different facial expressions conducted by 10 different people (Fig. 2).

[1] http://www.cad.zju.edu.cn/home/dengcai/Data/FaceData.html.

(a) ORL (b) Yale

Fig. 2. Example images from different face image databases: (a) ORL and (b) Yale.

In Table 1, we report the experimental results of P-ELM and other baselines with 50 hidden neurons on two different face image datasets. The results indicate that P-ELM are with high testing accuracy and low standard deviation compared to other baselines. P-ELM achieves 74.50% testing accuracy with 4.06% standard deviation on the ORL dataset, and 60.63% testing accuracy with 8.04% standard deviation on the Yale dataset. In terms of both training time and testing time, P-ELM is superior to EN-ELM and V-ELM, and slightly inferior to ELM and W-ELM. Besides, the experimental results for all compared methods with different numbers of hidden neurons are given in Fig. 3. From Fig. 3, we can observe that P-ELM always significantly outperforms other baselines on both the ORL and Yale datasets. P-ELM's remarkable performance on face image recognition owes to the unique of parameter learning mechanism, which guarantees that P-ELM can achieve superior performance.

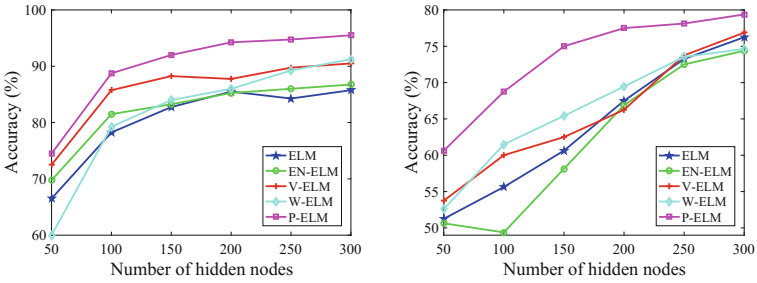
Table 1. Performance comparison on face image recognition.

Dataset	Measure	Algorithm				
		ELM	EN-ELM	V-ELM	W-ELM	P-ELM
ORL	Accuracy (%)	69.00	69.75	72.5	57.25	74.50
	Acc. Std. (%)	5.92	6.92	5.14	6.58	4.06
	Training time (s)	0.0109	0.7472	0.0905	0.0106	0.0328
	Testing time (s)	0.0042	0.4992	0.0094	0.0041	0.0047
Yale	Accuracy (%)	51.25	50.63	53.75	52.58	60.63
	Acc. Std. (%)	13.76	11.58	10.29	11.49	8.04
	Training time (s)	0.0078	0.2590	0.0406	0.0086	0.0312
	Testing time (s)	0.0047	0.1888	0.0187	0.0042	0.0062

4.2 Handwritten Image Annotation

For handwritten image annotation application, we report the performance of P-ELM in this section. In the experiments, we use the USPS and MNIST handwritten image annotation datasets[2]. The USPS dataset contains 9298 different gray-scale handwritten digit images with the size of 16×16. The MNIST dataset

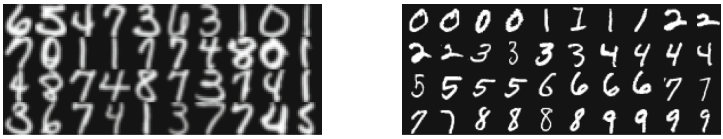
[2] http://www.cad.zju.edu.cn/home/dengcai/Data/MLData.html.



(a) ORL (b) Yale

Fig. 3. Performance comparison with respect to the number of hidden neurons on face image recognition: (a) ORL and (b) Yale.

used in the experiments consists of 10000 images of handwritten numbers with the size of 28×28, where each digital number consists of 1000 images. For the USPS and MNIST datasets, they are both associated with 10 different categories of "0" through "9" (Fig. 4).



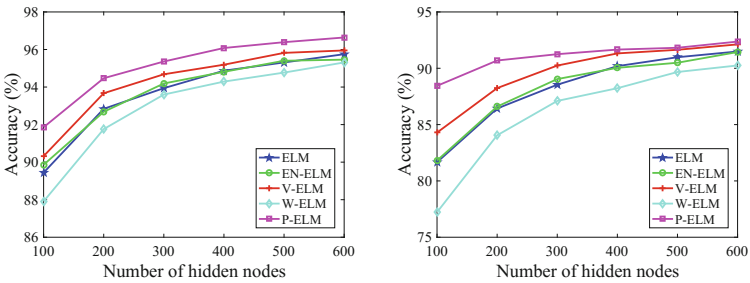
(a) USPS (b) MNIST

Fig. 4. Example images from different handwritten image databases: (a) USPS and (b) MNIST.

In Table 2, the results on handwritten image datasets show the performance of P-ELM and other baselines with 100 hidden neurons. P-ELM achieves 91.86% testing accuracy with 0.79% standard deviation on the USPS dataset, and 88.44% testing accuracy with 0.94% standard deviation on the MNIST dataset, which shows its superiority compared to other baselines. In terms of training time, P-ELM needs a little more running time than ELM, achieves slightly superior performance than W-ELM, and runs much faster than EN-ELM and V-ELM. In terms of testing time, P-ELM is slightly inferior to ELM and W-ELM, and significantly superior to EN-ELM and V-ELM. In addition, the simulation results for P-ELM and other baseline methods with various numbers of hidden neurons are presented in Fig. 5. As can be observed from Fig. 5, P-ELM is always superior to the baselines on the USPS dataset, and achieves better or comparable performance compared to other baselines on the MNIST dataset. The above observation suggests that P-ELM is also effective on handwritten image annotation, mainly because that it uses an ELM-AE to learn the suitable network parameters for P-ELM.

Table 2. Performance comparison on handwritten image annotation.

Dataset	Measure	Algorithm				
		ELM	EN-ELM	V-ELM	W-ELM	P-ELM
USPS	Accuracy (%)	89.44	89.61	90.32	87.91	91.86
	Acc. Std. (%)	1.02	1.07	1.11	0.95	0.79
	Training time (s)	0.2309	6.9748	1.6357	0.6257	0.4212
	Testing time (s)	0.0156	4.7471	0.0796	0.0152	0.0172
MNIST	Accuracy (%)	81.66	81.81	84.29	77.22	88.44
	Acc. Std. (%)	1.24	1.45	1.17	1.74	0.94
	Training time (s)	0.5647	12.9094	1.7023	0.8375	0.7192
	Testing time (s)	0.0172	8.4287	0.0858	0.0203	0.0265



(a) USPS (b) MNIST

Fig. 5. Performance comparison with respect to the number of hidden neurons on handwritten image annotation: (a) USPS and (b) MNIST.

5 Conclusion

In this paper, we proposed a novel method called pre-trained extreme learning machine (P-ELM for short). The proposed P-ELM is a data-driven method, which uses an ELM-AE to intelligently determine the suitable network parameters for diverse learning tasks. The unique parameter learning mechanism, including the processes of encoding and decoding, ensures that P-ELM can encode the underlying information of the original data into the network parameters. Experiments and comparisons on face image recognition and handwritten image annotation (each application contains two datasets) demonstrate the superior performance of the proposed P-ELM compared to baseline methods.

Acknowledgments. This work is supported in part by the National Nature Science Foundation of China (Grant Nos. 61403351 and 61773355), the Key Project of the Natural Science Foundation of Hubei Province, China (Grant No. 2013CFA004), the National Scholarship for Building High Level Universities, China Scholarship Council (No. 201706410005), and the Self-Determined and Innovative Research Founds of CUG (No. 1610491T05).

References

1. Huang, G.B., Zhu, Q.Y., Siew, C.K.: Extreme learning machine: theory and applications. Neurocomputing **70**(1–3), 489–501 (2006)
2. Huang, G., Huang, G.B., Song, S., You, K.: Trends in extreme learning machines: a review. Neural Netw. **61**, 32–48 (2015)
3. Zhang, Y., Wu, J., Cai, Z., Zhang, P., Chen, L.: Memetic extreme learning machine. Pattern Recogn. **58**, 135–148 (2016)
4. Zhang, Y., Wu, J., Zhou, C., Cai, Z.: Instance cloned extreme learning machine. Pattern Recogn. **68**, 52–65 (2017)
5. Liu, N., Wang, H.: Ensemble based extreme learning machine. IEEE Signal Process. Lett. **17**(8), 754–757 (2010)
6. Cao, J., Lin, Z., Huang, G.B., Liu, N.: Voting based extreme learning machine. Inf. Sci. **185**(1), 66–77 (2012)
7. Zong, W., Huang, G.B., Chen, Y.: Weighted extreme learning machine for imbalance learning. Neurocomputing **101**(3), 229–242 (2013)
8. Ap, S.C., Lauly, S., Larochelle, H., Khapra, M., Ravindran, B., Raykar, V.C., Saha, A.: An autoencoder approach to learning bilingual word representations. In: Advances in Neural Information Processing Systems, pp. 1853–1861 (2014)
9. Vincent, P., Larochelle, H., Bengio, Y., Manzagol, P.A.: Extracting and composing robust features with denoising autoencoders. In: 25th International Conference on Machine Learning, pp. 1096–1103 (2008)
10. Wang, H., Shi, X., Yeung, D.Y.: Relational stacked denoising autoencoder for tag recommendation. In: 29th AAAI Conference on Artificial Intelligence, pp. 3052–3058 (2015)
11. Bai, Z., Huang, G.B., Wang, D., Wang, H., Westover, M.B.: Sparse extreme learning machine for classification. IEEE Trans. Cybern. **44**(10), 1858–1870 (2014)
12. Zhang, R., Lan, Y., Huang, G.B., Xu, Z.B.: Universal approximation of extreme learning machine with adaptive growth of hidden nodes. IEEE Trans. Neural Netw. Learn. Syst. **23**(2), 365–371 (2012)
13. Huang, G.B., Zhou, H., Ding, X., Zhang, R.: Extreme learning machine for regression and multiclass classification. IEEE Trans. Syst. Man Cybern. Part B Cybern. **42**(2), 513–529 (2012)
14. Kasun, L.L.C., Zhou, H., Huang, G.B., Chi, M.V.: Representational learning with elms for big data. IEEE Intell. Syst. **28**(6), 31–34 (2013)
15. Hinton, G.E., Salakhutdinov, R.R.: Reducing the dimensionality of data with neural networks. Science **313**(5786), 504–507 (2006)
16. Tang, J., Deng, C., Huang, G.B.: Extreme learning machine for multilayer perceptron. IEEE Trans. Neural Netw. Learn. Syst. **27**(4), 809–821 (2015)
17. Yang, Y., Wu, Q.J.: Multilayer extreme learning machine with subnetwork nodes for representation learning. IEEE Trans. Cybern. **46**(11), 2570–2583 (2016)
18. Wu, J., Cai, Z., Zeng, S., Zhu, X.: Artificial immune system for attribute weighted naive bayes classification. In: IEEE International Joint Conference on Neural Networks, pp. 1–8 (2013)
19. Wu, J., Hong, Z., Pan, S., Zhu, X., Cai, Z., Zhang, C.: Multi-graph-view learning for graph classification. In: 14th IEEE International Conference on Data Mining, pp. 590–599 (2014)
20. Wu, J., Pan, S., Zhu, X., Zhang, C., Wu, X.: Positive and unlabeled multi-graph learning. IEEE Trans. Cybern. **47**(4), 818–829 (2017)