Hindawi
Mathematical Problems in Engineering
Volume 2017, Article ID 4670187, 10 pages
https://doi.org/10.1155/2017/4670187

Research Article

A Multiple Hidden Layers Extreme Learning Machine Method and Its Application

Dong Xiao,[1] Beijing Li,[1] and Yachun Mao[2]

[1] *Information Science & Engineering School, Northeastern University, Shenyang 110004, China*
[2] *College of Resources and Civil Engineering, Northeastern University, Shenyang 110004, China*

Correspondence should be addressed to Dong Xiao; xiaodong@ise.neu.edu.cn

Received 30 August 2017; Accepted 14 November 2017; Published 13 December 2017

Academic Editor: Mauro Gaggero

Extreme learning machine (ELM) is a rapid learning algorithm of the single-hidden-layer feedforward neural network, which randomly initializes the weights between the input layer and the hidden layer and the bias of hidden layer neurons and finally uses the least-squares method to calculate the weights between the hidden layer and the output layer. This paper proposes a multiple hidden layers ELM (MELM for short) which inherits the characteristics of parameters of the first hidden layer. The parameters of the remaining hidden layers are obtained by introducing a method (make the actual output zero error approach the expected hidden layer output). Based on the MELM algorithm, many experiments on regression and classification show that the MELM can achieve the satisfactory results based on average precision and good generalization performance compared to the two-hidden-layer ELM (TELM), the ELM, and some other multilayer ELM.

1. Introduction

At present, artificial neural network has been widely applied in many research fields, such as pattern recognition, signal processing, and short-term prediction. Among them, the single-hidden-layer feedforward neural network (SLFN) is the most widely used type of the artificial neural network [1, 2]. Because the parameters of traditional feedforward neural network are usually determined by gradient-based error backpropagation algorithms, the network bears the time-expensive training and testing process and easily falls into the local optimum. Now, many algorithms have been proposed to improve the SLFN operation rate and precision such as the backpropagation algorithm (BP) and its improved algorithms [3, 4]. With limitations of BP algorithms, generalization ability of networks is unsatisfactory and the over learning easily occurs. In 1989, Lowe proposed the RBF neural network [5] which indicated that the parameters of the SLFNs can also be randomly selected in his articles. In 1992, Pao Y. H. et al. proposed the theory of the random vector functional link network (RVFL) [6, 7], and they presented that only one parameter of the output weights should be calculated during the training process.

In 2004, Huang G. B. proposed the extreme learning machine (ELM) reducing the training time of network and improving the generalization performance [8–10]. Traditional neural network learning algorithms (such as BP) need to randomly set all the training parameters and use iterative algorithm to update the parameters. Also it is easy to generate local optimal solution. But ELM only needs to randomly set the weights and bias of the hidden neurons, and the output weights are determined by using the Moore-Penrose pseudoinverse under the criterion of least-squares method. In recent years, various ELM variants have been proposed aiming to achieve better achievements, such as the deep ELM with kernel based on Multilayer Extreme Learning Machine (DELM) algorithm [11]; two-hidden-layer extreme learning machine (TELM) [12]; a Four-Layered Feedforward Neural Network [13]; online sequential extreme learning machine [14, 15]; multiple kernel extreme learning machine (MK-ELM) [16]; two-stage extreme learning machine [17], using noise detection and improving the classifier accuracy [18, 19].

First, consider the DELM with kernel based on ELM-AE algorithm (DELM) presented in [11], which quotes the ELM autoencoder (ELM-AE) [20–22] as the learning algorithm in each layer. The DELM also has multilayer network structure divided into two parts: the first part uses the ELM-AE to deep learn the original data aiming at obtaining the most representative new data; the second part calculates the network parameters by using the Kernel ELM algorithm with a three-layer structure (the output of the first part, hidden layer, and output layer). But for the MELM we do not need the data processing (such as extract the representative data from the original data) but make the actual output of the hidden layers more closer to the expected output of the hidden layers by calculating step by step in the multiple hidden layers.

Next, a two-hidden-layer feedforward network (TLFN) was proposed by Huang in 2003 [23]. This article demonstrates that the TLFNs could learn arbitrary N training samples with a very small training error by employing $2\sqrt{(m+3)N}$ hidden neurons [24]. But the changing process of the TLFNs structure is very complicated. First the TLFN has a three-layer network with L output neurons, then adds $2L$ neurons (two parts) to the hidden layer aiming at making the original output layer transform into the second hidden layer with L hidden neurons, and finally adds a output layer to the structure. Eventually the final network structure has one input layer, two hidden layers, and one output layer. But the MELM has a relatively simple stable network structure and the simple calculation process, and the MELM is a time-saving algorithm compared with TLFNs.

In the paper, we propose a multiple hidden layers extreme learning machine algorithm (MELM) that the MELM adds some hidden layers to the original ELM network structure, randomly initializes the weights between the input layer and the first hidden layer as well as the bias of the first hidden layer, utilizes the method (make the actual each hidden layer output approach the expected each hidden layer output) to calculate the parameters of the hidden layers (except the first hidden layer), and finally uses the least square method to calculate the output weights of the network. In the following chapters, we have carried out many experiments with the ideas proposed. The MELM experimental results on regression problems and some popular classification problems have shown satisfactory advantages in terms of average accuracy compared to other ELM variants. Our experiments also study the effect of different numbers of the hidden layer neurons, the compatible activation function, and the different numbers of the hidden layers on the same problems.

The rest of this paper is organized as follows. Section 2 reviews the original ELM; Section 3 presents the method and framework structure of two-hidden-layer ELM; Section 4 presents the proposed the MELM technique: multihidden-layer ELM; Section 5 reports and analyzes experimental results; Section 6 presents the conclusions.

2. Extreme Learning Machine

The extreme learning machine (ELM) proposed by Huang G.B. aims at avoiding time-costing iterative training process and improving the generalization performance [8–10, 25]. As a single-hidden-layer feedforward neural networks (SLFNs), the ELM structure includes input layer, hidden layer, and output layer. Different from the traditional neural network learning algorithms (such as BP algorithm) randomly setting all the network training parameters and easily generating local optimal solution, the ELM only sets the number of hidden neurons of the network, randomizes the weights between the input layer and the hidden layer as well as the bias of the hidden neurons in the algorithm execution process, calculates the hidden layer output matrix, and finally obtains the weight between the hidden layer and the output layer by using the Moore-Penrose pseudoinverse under the criterion of least-squares method. Because the ELM has the simple network structure and the concise parameters computation processes, so the ELM has the advantages of fast learning speed. The original structure of ELM is expressed in Figure 1.

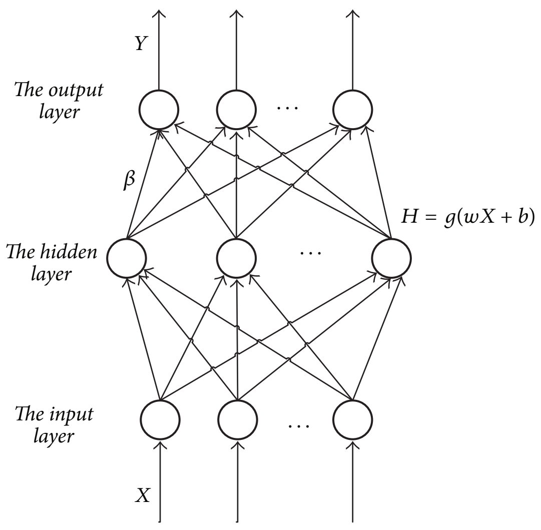


FIGURE 1: The structure of the ELM.

Figure 1 is the extreme learning machine network structure which includes n input layer neurons, l hidden layer neurons, and m output layer neurons. First, consider the training sample $\{X, Y\} = \{x_i, y_i\}$ $(i = 1, 2, \ldots, Q)$, and there is an input feature $X = [x_{i1} \;\; x_{i2} \;\; \cdots \;\; x_{iQ}]$ and a desired matrix $Y = [y_{j1} \;\; y_{j2} \;\; \cdots \;\; y_{jQ}]$ comprised of the training samples, where the matrix X and the matrix Y can be expressed as follows:

$$X = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1Q} \\ x_{21} & x_{22} & \cdots & x_{2Q} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{nQ} \end{bmatrix},$$
$$Y = \begin{bmatrix} y_{11} & y_{12} & \cdots & y_{mQ} \\ y_{21} & y_{22} & \cdots & y_{mQ} \\ \vdots & \vdots & \ddots & \vdots \\ y_{m1} & y_{m2} & \cdots & y_{mQ} \end{bmatrix}, \quad (1)$$

where the parameters n and m are the dimension of input matrix and output matrix.

Then the ELM randomly sets the weights between the input layer and the hidden layer:

$$w = \begin{bmatrix} w_{11} & w_{12} & \cdots & w_{1n} \\ w_{21} & w_{22} & \cdots & w_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ w_{l1} & w_{l2} & \cdots & w_{ln} \end{bmatrix}, \quad (2)$$

where w_{ij} represents the weights between the jth input layer neuron and ith hidden layer neuron.

Third, the ELM assumes the weights between the hidden layer and the output layer that can be expressed as follows:

$$\beta = \begin{bmatrix} \beta_{11} & \beta_{12} & \cdots & \beta_{1m} \\ \beta_{21} & \beta_{22} & \cdots & \beta_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ \beta_{l1} & \beta_{l2} & \cdots & \beta_{lm} \end{bmatrix}, \quad (3)$$

where β_{jk} represents the weights between the jth hidden layer neuron and kth output layer neuron.

Fourth, the ELM randomly sets the bias of the hidden layer neurons:

$$B = \begin{bmatrix} b_1 & b_2 & \cdots & b_n \end{bmatrix}^T. \quad (4)$$

Fifth, the ELM chooses the network activation function $g(x)$. According to Figure 1, the output matrix T can be expressed as follows:

$$T = [t_1, t_2, \ldots, t_Q]_{m\times Q}. \quad (5)$$

Each column vector of the output matrix T is as follows:

$$t_j = \begin{bmatrix} t_{1j} \\ t_{2j} \\ \vdots \\ t_{mj} \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{l} \beta_{i1} g\left(w_i x_j + b_i\right) \\ \sum_{i=1}^{l} \beta_{i2} g\left(w_i x_j + b_i\right) \\ \vdots \\ \sum_{i=1}^{l} \beta_{im} g\left(w_i x_j + b_i\right) \end{bmatrix} \quad (6)$$
$$(j = 1, 2, 3, \ldots, Q).$$

Sixth, consider formulae (5) and (6), and we can get

$$H\beta = T', \quad (7)$$

where T' is the transpose of T and H is the output of the hidden layer. In order to obtain the unique solution with minimum-error, we use least square method to calculate the weight matrix values of β [8, 9].

$$\beta = H^{+}T'. \quad (8)$$

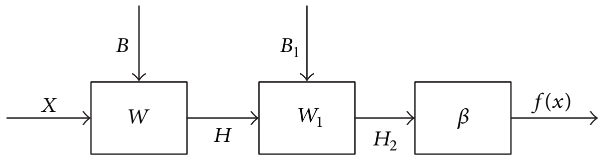


FIGURE 2: The workflow of the TELM.

To improve the generalization ability of network and make the results more stable, we add a regularization term to the β [26]. When the number of hidden layer neurons is less than the number of training samples, β can be expressed as

$$\beta = \left(\frac{I}{\lambda} + H^T H\right)^{-1} H^T T'. \quad (9)$$

When the number of hidden layer nodes is more than the number of training samples, β can be expressed as

$$\beta = H^T \left(\frac{I}{\lambda} + H H^T\right)^{-1} T'. \quad (10)$$

3. Two-Hidden-Layer ELM

In 2016, B. Y. Qu and B. F. Lang proposed the two-hidden-layer extreme learning machine (TELM) in [11]. The TELM tries to make the actual hidden layers output approach expected hidden layer outputs by adding parameter setting step for the second hidden layer. Finally, the TELM finds a better way to map the relationship between input and output signals, which is the two-hidden-layer ELM. The network structure of TELM includes one input layer, two hidden layers, one output layer, and each hidden layer with l hidden neurons. The activation function of the network is selected for $g(x)$.

The focus of the TELM algorithm is the process of calculating and updating the weights between the first hidden layer and the second hidden layer as well as the bias of the hidden layer and the output weights between the second hidden layer and the output layer. The workflow of the TELM architecture is depicted in Figure 2.

Consider the training sample datasets $\{X, \mathrm{T}\} = \{x_i, t_i\}$ $(i = 1, 2, 3, \ldots, Q)$, where the matrix X is input samples and T is the labeled samples.

The TELM first puts the two hidden layers as one hidden layer, so the output of the hidden layer can be expressed as $H = g(WX + B)$ with the parameters of the weight W and bias B of the first hidden layer randomly initialized. Next, the output weight matrix β between the second hidden layer and the output layer can be obtained by using

$$\beta = H^{+}T. \quad (11)$$

Now the TELM separates the two hidden layers merged previously, so the network has two hidden layers. According to the workflow of Figure 2, the output of the second hidden layer can be obtained as follows:

$$g\left(W_1 H + B_1\right) = H_1, \quad (12)$$

where W_1 is the weight matrixes between the first hidden layer and the second hidden layer, H is the output matrix of the first hidden layer, B_1 is the bias of the second hidden layer, and H_1 is the expected output of the second hidden layer.

However the expected output of the second hidden layer can be obtained by calculating

$$H_1 = T\beta^{+}, \tag{13}$$

where β^{+} is the generalized inverse of the matrix β.

Now the TELM defines the matrix $W_{HE} = [B_1 \;\; W_1]$, so the parameters of the second hidden layer can be easily obtained by using formula (12) and the inverse function of the activation function.

$$W_{HE} = g^{-1}(H_1) H_E^{+}, \tag{14}$$

where H_E^{+} is the generalized inverse of $H_E = [1 \;\; H]^T$, **1** denotes a one-column vector of size Q, and its elements are the scalar unit 1. The notation $g^{-1}(x)$ indicates the inverse of the activation function $g(x)$.

With selecting the appropriate activation function $g(x)$, the TELM calculates (14), so the actual output of the second hidden layer is updated as follows:

$$H_2 = g(W_{HE} H_E). \tag{15}$$

So the weights matrix β between the second hidden layer and the output layer is updated as follows:

$$\beta_{\text{new}} = H_2^{+} T, \tag{16}$$

where H_2^{+} is the generalized inverse of H_2, so the actual output of the TELM network can be expressed as

$$f(x) = H_2 \beta_{\text{new}}. \tag{17}$$

To sum up, the TELM algorithm process can be expressed as follows.

Algorithm 1. (1) Assume the training sample dataset is $\{X, T\} = \{x_i, t_i\}$ $(i = 1, 2, 3, \dots, Q)$, where the matrix X is input samples and the matrix T is the labeled samples; each hidden layer with l hidden neurons; the activation function $g(x)$.

(2) Randomly generate the weights W between the input layer and the first hidden layer and bias B of the first hidden neurons $W_{IE} = [B \;\; W]$, $X_E = [1 \;\; X]^T$.

(3) Calculate the equation $H = g(W_{IE} X_E)$.

(4) Obtain the weights between the second hidden layer and output layer $\beta = H^{+} T$.

(5) Calculate the expected output of the second hidden layer $H_1 = T\beta^{+}$.

(6) According to formulae (12)–(14) and the algorithm steps (4, 5), calculate the weights W_1 between the first hidden layer and the second hidden layer and the bias B_1 of the second hidden neurons $W_{HE} = g^{-1}(H_1) H_E^{+}$.

(7) Obtain and update the actual output of the second hidden layer $H_2 = g(W_{HE} H_E)$.

(8) Update the weights matrix β between the second hidden layer and the output layer $\beta_{\text{new}} = H_2^{+} T$.

(9) Calculate the output of the network $f(x) = g\{[W_H g(WX + B) + B_1]\}\beta_{\text{new}}$.

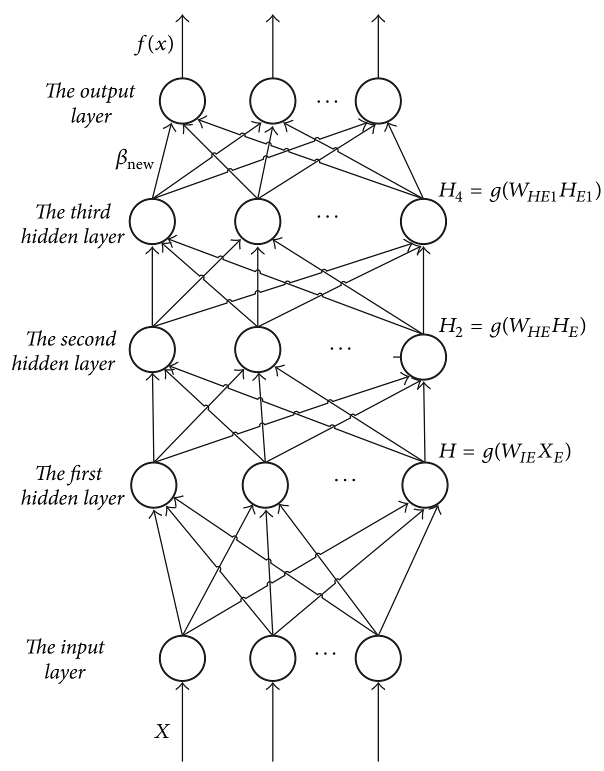


Figure 3: The structure of the three-hidden-layer ELM.

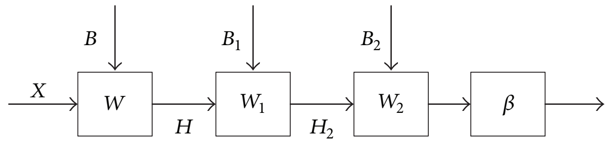


Figure 4: The workflow of the three-hidden-layer ELM.

4. Multihidden-Layer ELM

At present, some scholars have proposed many improvements on ELM algorithm and structure and obtained some achievements and advantages. Such advantages of neural network motivate us to explore the better ideas behind the ELM. Based on the above articles, we adjust the structure of ELM neural network. Thus we propose an algorithm named multiple hidden layers extreme learning machine (MELM). The structure of the MELM (select the three-hidden-layer ELM for example) is illustrated in Figure 3. The workflow of the three-hidden-layer ELM is illustrated in Figure 4.

Here we take a three-hidden-layer ELM, for example, and analyze the MELM algorithm below. First give the training samples $\{X, \mathrm{T}\} = \{x_i, t_i\}$ $(i = 1, 2, 3, \dots, Q)$ and the three-hidden-layer network structure (each of the three-hidden-layer has l hidden neurons) with the activation function $g(x)$. The structure of the three-hidden-layer ELM has input layer, three hidden layers, and output layer. According to the theory of the TELM algorithm, now we put the three hidden layers as two hidden layers (the first hidden layer is still the first hidden layer, but put the second hidden layer and the third hidden layer together as one hidden layer), so that the structure of the network is the same as the TELM network mentioned above. So we can obtain the weights matrix β_{new} between the second hidden layer and the output layer. According to the number of the actual samples, we can use formula (8) or formula (9) to calculate the weights β, which can improve the generalization ability of the network.

Then the MELM separates the three hidden layers merged previously, so the MELM structure has three hidden layers. So the expected output of the third hidden layer can be expressed as follows:

$$H_3 = T\beta_{\text{new}}^{+}. \tag{18}$$

β_{new}^{+} is generalized inverse of the weights matrix β_{new}.

Third the MELM defines the matrix $W_{HE1} = [B_2 \;\; W_2]$, so the parameters of the third hidden layer can be easily obtained by calculating formula (18) and the formula $H_3 = g(H_2W_2 + B_2) = g(W_{HE1}H_{E1})$.

$$W_{HE1} = g^{-1}(H_3) H_{E1}^{+}, \tag{19}$$

where H_2 is the actual output of the second hidden layer, W_2 is the weights between the second hidden layer and the third hidden layer, B_2 is the bias of the third hidden neurons, H_{E1}^{+} is the generational inverse of $H_{E1} = [1 \;\; H_2]^T$, $\mathbf{1}$ denotes a one-column vector of size Q, and its elements are the scalar unit 1. The notation $g^{-1}(x)$ indicates the inverse of the activation function $g(x)$.

In order to test the performance of the proposed MELM algorithm, we adopt different activation functions for regression and classification problems to experiment. Generally, we adopt the logistic sigmoid function $g(x) = 1/(1 + e^{-x})$. The actual output of the third hidden layer is calculated as follows:

$$H_4 = g(\mathrm{W}_{HE1}H_{E1}). \tag{20}$$

Finally, the output weights matrix β_{new} between the third hidden layer and the output layer is calculated as follows: when the number of hidden layer neurons is less than the number of training samples, β can be expressed as follows:

$$\beta_{\text{new}} = \left(\frac{I}{\lambda} + H_4^T H_4\right)^{-1} H_4^T T. \tag{21}$$

When the number of hidden layer neurons is more than the number of training samples, β can be expressed as follows:

$$\beta_{\text{new}} = H_4^T \left(\frac{I}{\lambda} + H_4 H_4^T\right)^{-1} T. \tag{22}$$

The actual output of the three-hidden-layer ELM network can be expressed as follows:

$$f(x) = H_4\beta_{\text{new}}. \tag{23}$$

To ensure that the actual final hidden output more approaches the expected hidden output during the training process, the operation process is the optimization of the network structure parameters starting from the second hidden layer.

The above is the parameter calculation process of three-hidden-layer ELM network, but the purpose of this paper is to calculate the parameter of the multiple hidden layers ELM network and the final output of the MELM network structure. We can use cycle calculation theory to illustrate the calculating process of the MELM. When the four-hidden-layer ELM network occurs, we can recalculate formula (18) to formula (22) in the calculation process of the network, obtain and record the parameters of each hidden layer, and finally calculate the final output of the MELM network. If the number of hidden layers increased, the calculation process can be recycled and executed in the same way. The calculation process of MELM network can be described as follows.

Algorithm 2. (1) Assume the training sample dataset is $\{X, T\} = \{x_i, t_i\}$ $(i = 1, 2, 3, \ldots, Q)$, where the matrix X is the input samples and the matrix T is the labeled samples. Each hidden layer has l hidden neurons with the activation function $g(x)$.

(2) Randomly initialize the weights W between the input layer and the first hidden layer as well as the bias B of the first hidden neurons $W_{IE} = [B \;\; W]$, $X_E = [1 \;\; X]^T$.

(3) Calculate the equation $H = g(W_{IE}X_E)$.

(4) Calculate the weights between the hidden layers and the output layer $\beta = (I/\lambda + H^TH)^{-1}H^TT$ or $\beta = H^T(I/\lambda + HH^T)^{-1}T$.

(5) Calculate the expected output of the second hidden layer $H_1 = T\beta^{+}$.

(6) According to formulae (12)–(14) and the algorithm steps (4, 5), calculate the weights W_1 between the first hidden layer and the second hidden layer and the bias B_1 of the second hidden neurons $W_{HE} = g^{-1}(H_1)H_E^{+}$.

(7) Obtain and update the actual output of the second hidden layer $H_2 = g(W_{HE}H_E)$.

(8) Update the weights matrix β between the hidden layer and the output layer $\beta_{\text{new}} = (I/\lambda + H_2^TH_2)^{-1}H_2^TT$ or $\beta_{\text{new}} = H_2^T(I/\lambda + H_2H_2^T)^{-1}T$.

(9) If the number of the hidden layer is three, we can calculate the parameters by recycle executing the above operation from step (5) to step (9). Now β_{new} is expressed as follows: $\beta_{\text{new}} = \beta$, $H_E = [1 \;\; H_2]^T$.

(10) Calculate the output, $f(x) = H_2\beta_{\text{new}}$.

If the number K of the hidden layer is more than three, recycle is executing step (5) to step (9) for $(K - 1)$ times. All the H matrix (H_1, H_2) must be normalized between the range of -0.9 and 0.9, when the max of the matrix is more than 1 and the min of the matrix is less than -1.

5. Application

In order to verify the actual effect of the algorithm proposed in this paper, we have done the following experiments which are divided into three parts: regression problems, classification problems, and the application of mineral selection industry. All the experiments are conducted in the MATLAB 2010b computational environment running on a computer with a 2.302 GHZ in i3 CPU.

Table 1: The RMSE of the three function examples.

Algorithm	Training RMSE	Testing RMSE
$f_1(x)$		
TELM	$8.6233E-9$	$1.0797E-8$
MELM (three-hidden-layer)	$8.722E-15$	$1.3055E-14$
MLELM	$1.8428E-6$	$1.5204E-5$
$f_2(x)$		
TELM	0.2867	0.2775
MELM (three-hidden-layer)	0.0011	0.0019
MELM	0.1060	0.1098
$f_3(x)$		
TELM	0.3528	0.6204
MELM (three-hidden-layer)	0.2110	0.4177
MLELM	0.5538	0.4591

5.1. Regression Problems. To test the performance of the regression problems, several widely used functions are listed below [27]. We use these functions to generate a dataset which includes random selection of sufficient training samples and the remaining is used as a testing samples, and the activation function is selected as the hyperbolic tangent function $g(x) = (1-e^{-x})/(1+e^{-x})$.

(1) $f_1(x) = \sum_{i=1}^{D} x_i$.

(2) $f_2(x) = \sum_{i=1}^{D-1}(100(x_i^2 - x_{i+1})^2 + (x_i - 1)^2)$.

(3) $f_3(x) = -20e^{-0.2\sqrt{\sum_{i=1}^{n} x_i^2/D}} - e^{\sum_{i=1}^{n}\cos(2\pi x_i)/D} + 20$.

The symbol D that is set as a positive integer represents the dimensions of the function we are using. The function $f_2(x)$ and $f_3(x)$ are the complex multimodal function. Each function-experiment has 900 training samples and 100 testing samples. The evaluation criterion of the experiments is the average accuracy, namely, the root mean square error (RMSE) in the regression problems. We first need to conduct some experiments to verify the average accuracy of the neural network structure in different number of hidden neurons and hidden layers. Then we can observe the optimal neural network structure with specific hidden layers number and hidden neurons number compared with the results of the TELM on regression problems.

In Table 1, we give the average RMSE of the above three functions in the case of the two-hidden-layer structure, the MLELM, and the three-hidden-layer structure (we do a series of experiments, including the four-hidden-layer structure, five-hidden-layer structure, and eight-hidden-layer structure. But the experiments proved that the three-hidden-layer structure for the three functions $f_1(x)$, $f_2(x)$, $f_3(x)$ can achieve the best results, so we only give the value of the RMSE of the three-hidden-layer structure) in regression problems. The table includes the testing RMSE and the training RMSE of the three algorithm structure. We can see that the MELM (three-hidden-layer) has the better results.

Table 2: The three datasets for the classification.

Datasets	Training samples	Testing samples	Attributes	Class
Mice Protein	750	330	82	8
svmguide4	300	312	10	6
vowel	568	422	10	11
AutoUnix	380	120	101	4
Iris	100	50	4	3

5.2. Classification Problems. To test the performance of the MELM algorithm proposed on classification datasets, so we quote some datasets (such as Mice Protein, svmguide4, vowel, AutoUnix, and Iris) that are collected from the University of California [28] and the LIBSVM website [29]. The training data and testing data of each experiment are randomly selected from the original datasets. This information of the datasets introduced is given in Table 2.

The classification performance criteria of the problems are the average classification accuracy of the testing data. In the Figure 5, the average testing classification correct percentage for the ELM, TELM, MLELM, and MELM algorithm is shown clearly by using (a) Mice Protein, (b) svmguide4, (c) Vowel, (d) AutoUnix, and (e) Iris, and different datasets (a, b, c, d, and e) have an important effect on the classification accuracy for the three algorithms (ELM, TELM, MLELM, and MELM). But from the changing trend of the average testing correct percentage for the three algorithms with the same datasets, we can see that the MELM algorithm can select the optimal number of hidden layers for the network structure to adapt to the specific datasets, aiming at obtaining the better results. So the datasets of the Mice Protein and the Forest type mapping use the three-hidden-layer ELM algorithm to experiment and the datasets of the svmguide4 use the five-hidden-layer ELM algorithm to experiment.

5.3. The Application of Ores Selection Industry. In this part, we use the actual datasets to analyze and experiment with the model we have established and test the performance of the MELM algorithm. First, our datasets come from AnQian Mining Group, namely, the mining area of Ya-ba-ling and Xi-da-bei. And the samples we selected are the hematite (50 samples), magnetite (90 samples), granite (57 samples), phyllite (15 samples), and chlorite (32 samples). On the precise classification of the ores and the prediction of total iron content of iron ores, people usually use the chemical analysis method. But the analysis process of total iron content is a complex process, a long process, and the color of the solution sometimes without the significantly change in the end. Due to the extensive using of near infrared spectroscopy for chemical composition analysis, we can infer the structure of the unknown substance according to the position and shape of peaks in the absorption spectra. So we can use the theory to obtain the correct information of the ores with the low influence of environment on data acquisition. HR1024 SVC spectrometer is used to test the spectral data of each sample, and the total iron content of hematite and magnetite is obtained from the chemical analysis center of Northeastern

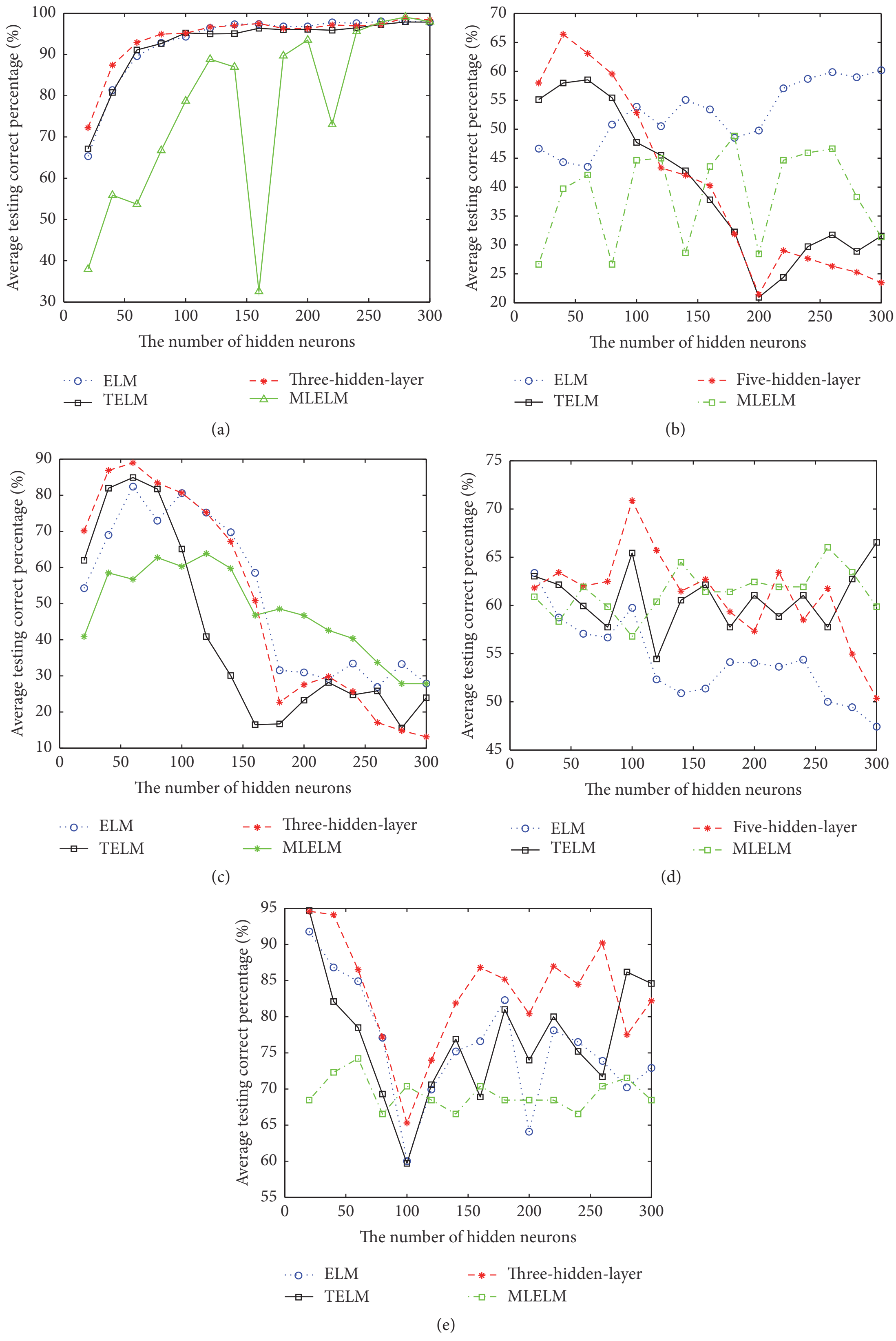


Figure 5: The average testing classification correct percentage for the ELM, TELM, and MELM using (a) Mice Protein, (b) svmguide4, (c) Vowel, (d) AutoUnix, and (e) Iris.

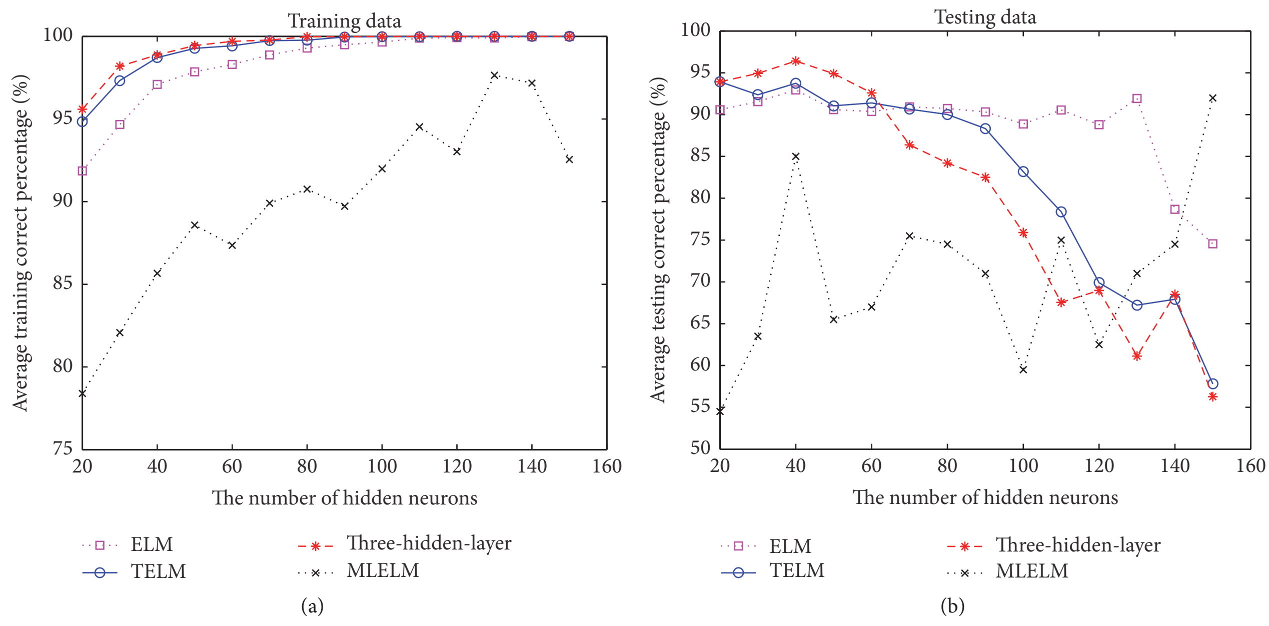


FIGURE 6: Average classification accuracy of different hidden layers.

University of China. Our experiments' datasets are the near infrared spectrum information of iron ores which are the absorption rate of different iron ores under the condition of the near infrared spectrum. The wavelength of near infrared spectrum is 300 nm–2500 nm, so the datasets of the iron ores we obtained have the very high dimensions (973 dimensions). We use the theory of the principal component analysis to reduce the dimensions of the datasets.

(A) Here we use the algorithm proposed in this paper to classify the five kinds of ores and compare with the results of the ELM algorithm and TELM algorithm. The activation function of the algorithm in classification problems is selected as the sigmoid function $g(x) = 1/(1 + e^{-x})$.

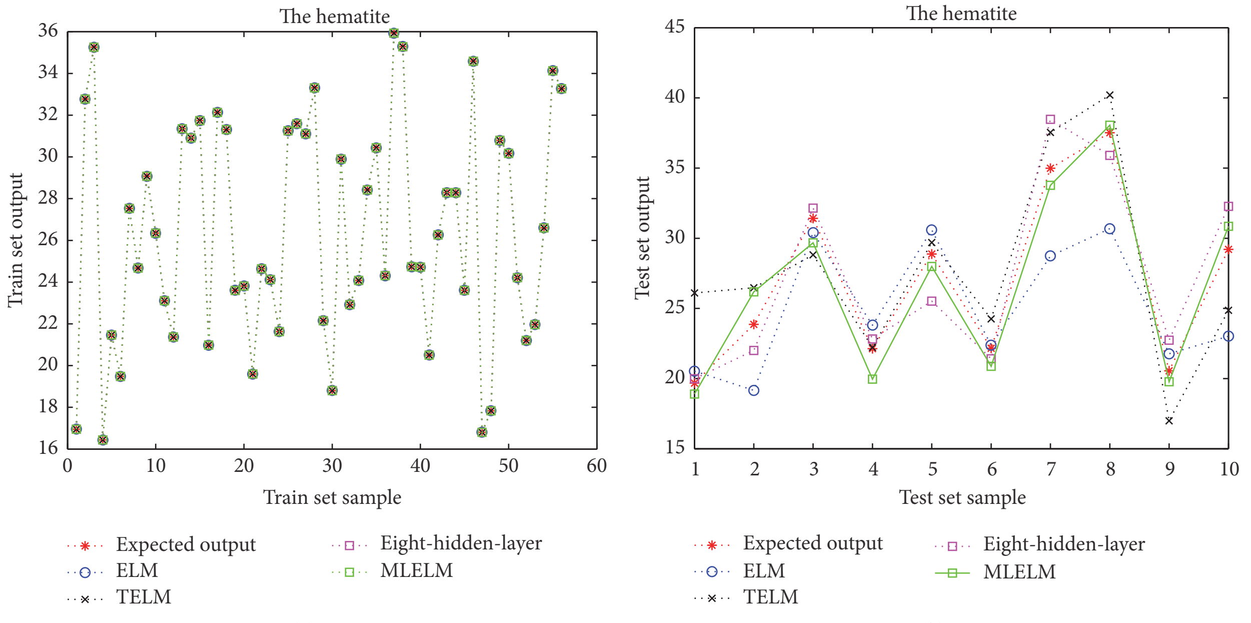
Figure 6 expresses the average classification accuracy of the ELM, TELM, MLELM, and MELM (three-hidden-layer ELM) algorithm in different hidden layers. After we conduct a series of experiments with different hidden layers of the MELM structure, finally we select the three-hidden-layer ELM structure which can obtain the optimal classification results in the MELM algorithm. Figure 6(a) is the average training correct percentage and the MELM has the high result compared with others in the same hidden neurons. Figure 6(b) is the average testing correct percentage and the MELM has the high result in the hidden neurons between 20 and 60. In the actual situation, we can reasonably choose the number of each hidden neurons to model.

(B) Here we use the method proposed in this paper to test the total iron content of the hematite and magnetite and compare with the results of the ELM algorithm and TELM algorithm. Here is the prediction of total iron content of hematite and magnetite with the optimal hidden layers and optimal hidden layer neurons. The activation function of the algorithm in regression problems is selected as the hyperbolic tangent function $g(x) = (1 - e^{-x})/(1 + e^{-x})$.

Figures 7 and 8, respectively, expresses the total iron content of the hematite and magnetite by using the ELM, TELM, MLELM, and MELM algorithm (the MELM includes the eight-hidden-layer structure which is the optimal structure in Figure 7; the six-hidden-layer structure which is the optimal structure in Figure 8). The optimal results of each algorithm are obtained by constantly experimenting with the different number of hidden layer neurons. In the prediction process of the hematite, we can see that the MELM has the better results and each hidden layer of this structure has 100 hidden neurons. In the prediction process of the magnetite, we can see that the MELM has the better results and each hidden layer of this structure has 20 hidden neurons. So the MELM algorithm has strong performance to the ores selection problems, which can achieve the best adaptability to the problems by adjusting the number of hidden layers and the number of hidden layer neurons to improve the ability of system regression analysis.

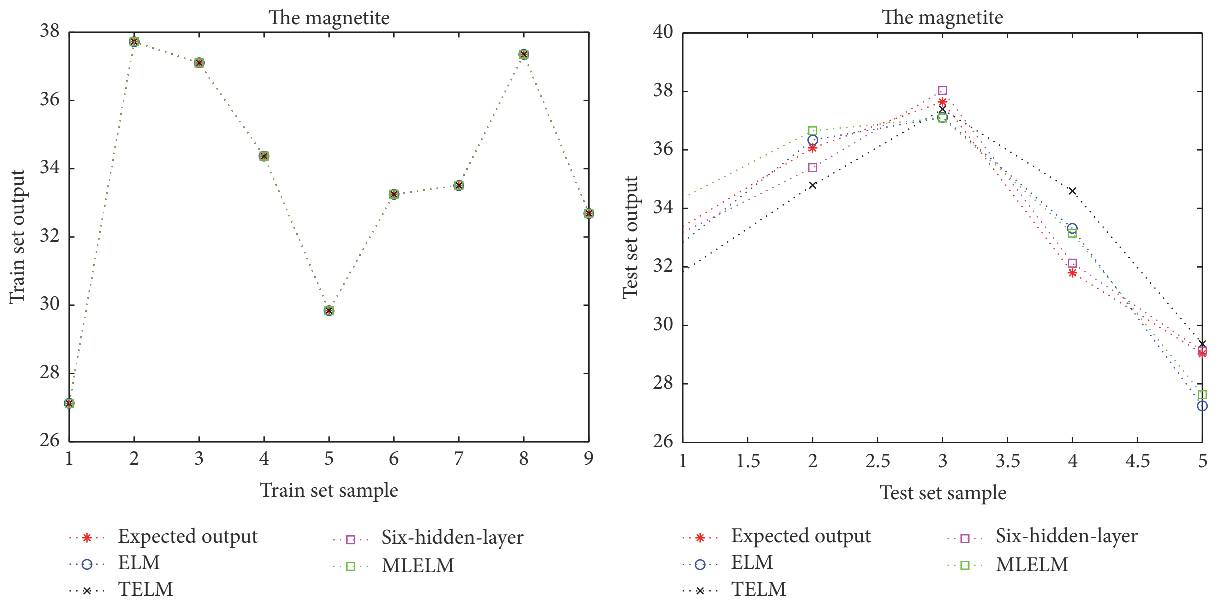
6. Conclusion

The MELM we proposed in this paper solves the two problems which exist in the training process of the original ELM. The first is the stability of single network that the disadvantage will also influence the generalization performance. The second is that the output weights are calculated by the product of the generalized inverse of hidden layer output and the system actual output. And the parameters randomly selected of hidden layer neurons which can lead to the singular matrix or morbid matrix. At the same time, the MELM



(a) (b)

Figure 7: The total iron content of the hematite.



(a) (b)

Figure 8: The total iron content of the magnetite.

network structure also improves the average accuracy of training and testing performance compared to the ELM and TELM network structure. The MELM algorithm inherits the characteristics of traditional ELM that randomly initializes the weights and bias (between the input layer and the first hidden layer), also adopts a part of the TELM algorithm, and uses the inverse activation function to calculate the weights and bias of hidden layers (except the first hidden layer). Then we make the actual hidden layer output approximate to the expected hidden layer output and use the parameters obtained above to calculate the actual output. In the function regression problems, this algorithm reduces the least mean square error. In the datasets classification problems, the average accuracy of the multiple classifications is significantly higher than that of the ELM and TELM network structure. In such cases, the MELM is able to improve the performance of the network structure.

Conflicts of Interest

The authors declare no conflicts of interest.

Acknowledgments

This research is supported by National Natural Science Foundation of China (Grants nos. 41371437, 61473072, and 61203214), Fundamental Research Funds for the Central Universities (N160404008), National Twelfth Five-Year Plan for Science and Technology Support (2015BAB15B01), China, and National Key Research and Development Plan (2016YFC0801602).

References

[1] K. Hornik, "Approximation capabilities of multilayer feedforward networks," *Neural Networks*, vol. 4, no. 2, pp. 251–257, 1991.

[2] M. Leshno, V. Y. Lin, A. Pinkus, and S. Schocken, "Multilayer feedforward networks with a nonpolynomial activation function can approximate any function," *Neural Networks*, vol. 6, no. 6, pp. 861–867, 1993.

[3] R. Hecht-Nielsen, "Theory of the backpropagation neural network," in *Proceedings of the International Joint Conference on Neural Networks (IJCNN '89)*, vol. 1, pp. 593–605, Washington, DC, USA, June 1989.

[4] H.-C. Hsin, C.-C. Li, M. Sun, and R. J. Sclabassi, "An Adaptive Training Algorithm for Back-Propagation Neural Networks," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 25, no. 3, pp. 512–514, 1995.

[5] D. Lowe, "daptive radial basis function nonlinearities, and the problem of generalization," in *First IEE International Conference on Artificial Neural Networks*, pp. 313-171, 1989.

[6] S. Ding, G. Ma, and Z. Shi, "A Rough RBF Neural Network Based on Weighted Regularized Extreme Learning Machine," *Neural Processing Letters*, vol. 40, no. 3, pp. 245–260, 2014.

[7] C. Sun, W. He, W. Ge, and C. Chang, "Adaptive Neural Network Control of Biped Robots," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 47, no. 2, pp. 315–326, 2017.

[8] S.-J. Wang, H.-L. Chen, W.-J. Yan, Y.-H. Chen, and X. Fu, "Face recognition and micro-expression recognition based on discriminant tensor subspace analysis plus extreme learning machine," *Neural Processing Letters*, vol. 39, no. 1, pp. 25–43, 2014.

[9] G. Huang, "An insight into extreme learning machines: random neurons, random features and kernels," *Cognitive Computation*, 2014.

[10] G. B. Huang, Q. Y. Zhu, and C. K. Siew, "Extreme learning machine: theory and applications," *Neurocomputing*, vol. 70, no. 1–3, pp. 489–501, 2006.

[11] S. Ding, N. Zhang, X. Xu, L. Guo, and J. Zhang, "Deep Extreme Learning Machine and Its Application in EEG Classification," *Mathematical Problems in Engineering*, vol. 2015, Article ID 129021, 2015.

[12] B. Qu, B. Lang, J. Liang, A. Qin, and O. Crisalle, "Two-hidden-layer extreme learning machine for regression and classification," *Neurocomputing*, vol. 175, pp. 826–834, 2016.

[13] S. I. Tamura and M. Tateishi, "Capabilities of a four-layered feedforward neural network: four layers versus three," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 8, no. 2, pp. 251–255, 1997.

[14] J. Zhao, Z. Wang, and D. S. Park, "Online sequential extreme learning machine with forgetting mechanism," *Neurocomputing*, vol. 87, no. 15, pp. 79–89, 2012.

[15] B. Mirza, Z. Lin, and K.-A. Toh, "Weighted online sequential extreme learning machine for class imbalance learning," *Neural Processing Letters*, vol. 38, no. 3, pp. 465–486, 2013.

[16] X. Liu, L. Wang, G. B. Huang, J. Zhang, and J. Yin, "Multiple kernel extreme learning machine," *Neurocomputing*, vol. 149, part A, pp. 253–264, 2015.

[17] Y. Lan, Y. C. Soh, and G. B. Huang, "Two-stage extreme learning machine for regression," *Neurocomputing*, vol. 73, no. 16-18, pp. 3028–3038, 2010.

[18] C.-T. Lin, M. Prasad, and A. Saxena, "An Improved Polynomial Neural Network Classifier Using Real-Coded Genetic Algorithm," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 45, no. 11, pp. 1389–1401, 2015.

[19] C. F. Stallmann and A. P. Engelbrecht, "Gramophone Noise Detection and Reconstruction Using Time Delay Artificial Neural Networks," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 47, no. 6, pp. 893–905, 2017.

[20] E. Cambria, G.-B. Huang, and L. L. C. Kasun, "Extreme learning machines," *IEEE Intelligent Systems*, vol. 28, no. 6, pp. 30–59, 2013.

[21] W. Zhu, J. Miao, L. Qing, and G.-B. Huang, "Hierarchical Extreme Learning Machine for unsupervised representation learning," in *Proceedings of the International Joint Conference on Neural Networks, IJCNN 2015*, Ireland, July 2015.

[22] W. Yu, F. Zhuang, Q. He, and Z. Shi, "Learning deep representations via extreme learning machines," *Neurocomputing*, pp. 308–315, 2015.

[23] G.-B. Huang, "Learning capability and storage capacity of two-hidden-layer feedforward networks," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 14, no. 2, pp. 274–281, 2003.

[24] G.-B. Huang and H. A. Babri, "Upper bounds on the number of hidden neurons in feedforward networks with arbitrary bounded nonlinear activation functions," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 9, no. 1, pp. 224–229, 1998.

[25] G.-B. Huang, H. Zhou, X. Ding, and R. Zhang, "Extreme learning machine for regression and multiclass classification," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 42, no. 2, pp. 513–529, 2012.

[26] J. Wei, H. Liu, G. Yan, and F. Sun, "Robotic grasping recognition using multi-modal deep extreme learning machine," *Multidimensional Systems and Signal Processing*, vol. 28, no. 3, pp. 817–833, 2016.

[27] J. J. Liang, A. K. Qin, P. N. Suganthan, and S. Baskar, "Comprehensive learning particle swarm optimizer for global optimization of multimodal functions," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 3, pp. 281–295, 2006.

[28] University of California, Irvine, Machine Learning Repository, http://archive.ics.uci.edu/ml/.

[29] L. Website, https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/.