Some Tricks in Parameter Selection for Extreme Learning Machine

Weipeng Cao¹, Jinzhu Gao², Zhong Ming¹,* and Shubin Cai¹

¹College of Computer Science and Software Engineering, Shenzhen University, Shenzhen 518060, China
²School of Engineering and Computer Science, University of the Pacific, CA 95211, USA

*Corresponding Author: mingz@szu.edu.cn

Abstract. Extreme learning machine (ELM) is a widely used neural network with random weights (NNRW), which has made great contributions to many fields. However, the relationship between the parameters and the performance of ELM has not been fully investigated yet, i.e. the impact of the number of hidden layer nodes, the randomization range of the weights between the input layer and hidden layer, the randomization range of the threshold of hidden nodes, and the type of activation functions. In this paper, eight benchmark functions are used to study this relationship. We have some interesting findings, such as more hidden layer nodes cannot guarantee the best performance of ELM, the empirical randomization range of the hidden weights (i.e., [-1, 1]) and the empirical randomization range of the threshold of hidden nodes (i.e., [0, 1]) may not lead to the optimal performance of ELM models, and ELM with sigmoid as the activation function always achieves better performance on some regression problems than ELM withtribas as the activation function. We hope the findings from our work could provide a useful guidance for researchers to select right parameters for ELM.

1. Introduction

Traditional artificial neural network (ANN), including deep learning, train models using the BP-based method [1]. That is, iteratively tuning all the weights and biases based on the back-propagated derivatives of loss function. This training mechanism has several notorious shortcomings such as local minima problem, slow convergence rate, time-consuming, and model uncertainty.

By contrast, neural network with random weights (NNRW) is a non-iterative algorithm that can avoid the above issues. NNRW was first proposed in 1992 by Schmidt et al. [2] and Pao’s group [3], respectively. The biggest difference between NNRW and the BP-based method is that the hidden weights are selected randomly while the output weights are obtained analytically. Compared with the BP-based method, NNRW can achieve much faster training speed. The combination of NNRW and traditional deep learning can effectively improve the computational efficiency of deep learning, which has been proven in many applications [4-6].

The name of NNRW proposed by Pao’s group [3] is random vector functional link networks (RVFL). In RVFL, the input layer is directly connected to both the hidden layer and the output layer. The hidden parameters are randomly selected from [-1, 1], while the weights between the input layer and output layer and the weights between the hidden layer and output layer are obtained by Moore-
Penrose pseudo-inverse. Zhang et al. [7] and Li’s group [8] showed that the direct links between the input layer and the output layer have a significant impact on the performance of RVFL and the empirical randomization range (i.e., [-1, 1]) of input weights and biases may not always be the right one. In addition, they studied the impact of the different activation functions on the performance of RVFL.

The NNRW proposed by Schmidt et al. [2] is somewhat different from RVFL. In Schmidt’s model, there is no direct link between the input layer and output layer and they used the Fisher method to determine the output weights. This type of NNRW was further investigated by Huang et al. who proposed a generalized algorithm, that is, extreme learning machine (ELM), in 2004 [9]. They have given a lot of theoretical analysis and rigorous theoretical proof for ELM [10, 11]. Huang et al. demonstrated that ELM has universal approximation capability and can approximate any continuous target function with probability one under certain conditions. In addition, any nonlinear piecewise continuous random hidden nodes can be used in ELM, such as Sigmoid nodes, Radial basis function nodes, Wavelet, etc. ELM and its variants have been widely used in real-world applications, including human face recognition [12, 13], fault diagnosis [14, 15], handwritten digit recognition [16], 3D image classification [17], fuzzy nonlinear regression [18], etc.

The idea behind RVFL, Schmidt’s method, and ELM, is similar, but their implementations are different [19]. Inspired by [7, 8], we did a comprehensive study on the relationship between the parameters and the performance of ELM using eight different benchmark functions. The parameters we studied include empirical randomization range of hidden parameters, the number of hidden layer nodes, and the selection of activation functions. In this paper, we share some interesting findings that could provide a useful guidance to help researchers select parameters wisely.

The rest of this paper is organized as follows. Section 2 gives a brief introduction to ELM. The details of experiment settings are shown in Section 3. In Section 4, we present the simulation results and our discussions. Conclusions and future work are given in Section 5.

2. Extreme learning machine

Extreme learning machine (ELM) is a special single hidden layer feed-forward neural network (SLFN), which was proposed by Huang’s group in 2004 [9]. A typical structure of ELM is shown in Figure 1.

![Figure 1. The structure of ELM.](image)

where $\omega$ denotes the weights between the input layer and hidden layer, $b$ denotes the threshold of hidden nodes, and $\beta$ denotes the output weights. It is noted that there is an activation function in the hidden layer to perform nonlinear transformation.

The biggest difference between ELM and other traditional SLFNs (BP-based methods) is that the hidden parameters (the weights between the input layer and hidden layer and the threshold of hidden nodes) are randomly selected and the output weights are determined by solving a system of linear matrix equations. Compared with the BP-based methods, ELM can achieve much faster learning speed.
with acceptable accuracy.

A typical ELM has \( L \) hidden layer nodes, \( N \) instances, and an activation function \( G(\chi) \), which can be modeled as:

\[
\sum_{i=1}^{L} \beta_i G(\omega_{ij} \cdot x_j + b_j) = o_j, j = 1, 2, ..., N
\]

where \( o_j \) is the predictive value of ELM model. According to the ELM theory [11], the optimization goal of ELM is

\[
\min_{\beta} \left( \min_{\|t\|} \sum_{i=1}^{N} \| t_i - o_i \|^2 \right),
\]

where \( t \) is the actual value of each instance, and \( \| \cdot \| \) is the matrix norm in Euclidean space. Equation (1) and (2) can be rewritten as

\[
H\beta = T,
\]

where

\[
H = \begin{bmatrix}
G(\omega_1 \cdot x_1 + b_1) & \cdots & G(\omega_L \cdot x_1 + b_L) \\
\vdots & \ddots & \vdots \\
G(\omega_1 \cdot x_N + b_1) & \cdots & G(\omega_L \cdot x_N + b_L)
\end{bmatrix}_{N \times L}, \quad \beta = \begin{bmatrix}
\beta_1^T \\
\vdots \\
\beta_L^T
\end{bmatrix}_{L \times m}, \quad T = \begin{bmatrix}
t_1^T \\
\vdots \\
t_N^T
\end{bmatrix}_{N \times m}
\]

\( H \) is the output matrix of the hidden layer, and the output weight \( \beta \) can be solved by

\[
\beta = H^{-1}T
\]

where \( H^{-1} \) is the Moore–Penrose generalized inverse of \( H \).

In general, the weights \( \omega \) are generated randomly within the range of \([-1, 1]\) and the thresholds of hidden layer nodes, \( b \), are generated randomly within the range of \([0, 1]\) under a uniform sampling distribution. The output weight \( \beta \) is determined by the Moore–Penrose generalized inverse as shown in equation (4). The commonly used activation functions \( G(\chi) \) include Sigmoid function, Sine function, Hardlim function, Triangular basis function, and Radial basis function.

Although ELM and its variants have been widely applied to numerous applications, to the best of our knowledge, there is no thorough investigation on the relationship between the parameters and the performance of ELM. In this paper, we use eight benchmark functions to fully study the impact of these parameters on the performance of ELM by answering the following questions.

(1) Is it true: the more hidden layer nodes, the better performance of ELM?
(2) Can the empirical randomization range of hidden weights (i.e., \([-1, 1]\)) lead to the optimal performance of ELM?
(3) Can the empirical randomization range of the hidden threshold (i.e., \([0, 1]\)) lead to the optimal performance of ELM?
(4) What’s impact of different activation functions on the performance of ELM?

3. Experiment settings

All our experiments are conducted in the MATLAB R2014a environment on the same Windows 10 machine with Intel Core i5-5300U 2.3 GHz CPU and 8 GB RAM. For each benchmark function, the average results over 50 trials are obtained for ELM.

3.1. Data preparation

In our experiments, eight commonly used benchmark functions are used to test the performance of ELM, i.e., Sphere function (Function 1), Step function (Function 2), Schwefel function (Function 3), Rastrigin function (Function 4), Ackley function (Function 5), Rosenbrock function (Function 6), Schaferf function (Function 7), and Quartic function (Function 8). The details of these eight functions are shown in Table 1.
Each benchmark function is used to generate a dataset with 2000 records, respectively. In each experiment, we shuffled the records, randomly selected 1000 records for training, and saved the remaining 1000 records for testing. The indicators for ELM performance testing include Testing Time, Standard Deviation of Testing Accuracy, the Root-Mean-Square Error of Training (Training RMSE), and the Root-Mean-Square Error of Testing (Testing RMSE).

Table 1. The details of eight benchmark functions

| Function name | Function model                                                                 | Initial range  | $f_{\text{min}}$ |
|---------------|--------------------------------------------------------------------------------|----------------|-----------------|
| Sphere        | $f = \sum_{i=1}^{D} x_i^2$                                                     | $[-100,100]^D$ | 0               |
| Step          | $f = \sum_{i=1}^{D} (x_i + 0.5)^2$                                             | $[-100,100]^D$ | 0               |
| Schwefel      | $f = \sum_{i=1}^{D} x_i \times \sin \sqrt{|x_i|}$                              | $[-500,500]^D$ | -12569.6        |
| Rastrigin     | $f = \sum_{i=1}^{D} [x_i^2 - 10 \cos(2\pi x_i) + 10]$                         | $[-5.12,5.12]^D$ | 0               |
| Ackley        | $f = -20 \exp(-0.2\sqrt{\frac{1}{D} \sum_{i=1}^{D} x_i^2}) - \exp(\frac{1}{D} \sum_{i=1}^{D} \cos 2\pi x_i) + 20 + e$ | $[-32,32]^D$ | 0               |
| Rosenbrock    | $f = \sum_{i=1}^{D} [100(x_{i+1} - x_i)^2 + (x_i - 1)^2]$                      | $[-30,30]^D$ | 0               |
| Schafferf     | $f = \sum_{i=1}^{D} [(x_i^2 + x_{i+1}^2)^{0.25} (\sin(50(x_i^2 + x_{i+1}^2)^{0.1})^2 + 1)]$ | $[-100,100]^D$ | 0               |
| Quartic       | $f = \sum_{i=1}^{D} x_i^4 + \text{random}[0,1]$                              | $[-1.28,1.28]^D$ | 0               |

3.2. Different ELM configurations
In our experiments, the variables for testing include the number of hidden layer nodes, the randomization range of the weights between the input layer and hidden layer, the randomization range of the threshold of hidden nodes, and the type of activation functions. In this paper, we use the single variable method to carry on the experiments, i.e., when we test the impact of a specific variable on the performance of ELM, the other variables remain unchanged. All the experimental results and analysis are shown in Section 4.

4. Simulation results and discussions
Supported by the experiments described in Section 3, we conducted a comprehensive study on the relationship between the parameters and performance of ELM using eight benchmark functions. The experimental results are shown in Table 2-5. It is noted that the near results are underlined and the best results are in boldface. In the rest of this section, we will give a detailed explanation on these results.

4.1. Is it true: the more hidden layer nodes, the better the performance of ELM?
Although Huang et al. have theoretically proved that the ELM model can approximate any continuous target function with any degree of accuracy provided it has as many hidden layer nodes as possible
[10]. However, in practical applications, would more hidden layer nodes lead to better performance of ELM model?
To answer this question, we conducted a validation experiment on the eight benchmark functions. The experimental results are shown in Table 2. In this experiment, the randomization range of the weights between the input layer and hidden layer is set to [-1, 1], the randomization range of the threshold of hidden nodes is set to [0, 1], and Sigmoid function is used as the activation function of ELM. Here, the variable of testing is the number of the hidden layer nodes $L$, which is set to

$$L = k \times N, K = \{0.2, 0.4, 0.6, 0.8\}$$

where $N$ is the number of training records.

| Dataset | Nodes number | Testing time(s) | Standard deviation | Mean | | Training RMSE | Testing RMSE |
|----------|--------------|-----------------|--------------------|------|------|----------------|----------------|
| Function1 | 0.2 * N | 1.56E-02 | 4.98E-08 | 2.79E-08 | 3.62E-08 |
|          | 0.4 * N | 3.25E-02 | 1.83E-08 | 1.15E-08 | 1.46E-08 |
|          | 0.6 * N | 4.72E-02 | 1.41E-08 | 1.51E-08 | 1.71E-08 |
|          | 0.8 * N | 6.44E-02 | 8.88E-08 | 5.84E-08 | 5.95E-08 |
| Function2 | 0.2 * N | 1.59E-02 | 6.80E-09 | 6.18E-09 | 7.33E-09 |
|          | 0.4 * N | 3.16E-02 | 1.64E-07 | 3.92E-08 | 4.05E-08 |
|          | 0.6 * N | 4.41E-02 | 1.94E-07 | 8.14E-08 | 8.16E-08 |
|          | 0.8 * N | 6.22E-02 | 1.23E-07 | 7.38E-08 | 7.40E-08 |
| Function3 | 0.2 * N | 1.72E-02 | 1.67E-02 | 2.76E-01 | 4.16E-01 |
|          | 0.4 * N | 2.69E-02 | 1.42E-02 | 2.02E-01 | 4.84E-01 |
|          | 0.6 * N | 4.72E-02 | 2.17E-02 | 1.28E-01 | 5.85E-01 |
|          | 0.8 * N | 6.06E-02 | 3.92E-02 | 6.56E-02 | 7.46E-01 |
| Function4 | 0.2 * N | 1.75E-02 | 2.92E-06 | 1.93E-06 | 1.93E-06 |
|          | 0.4 * N | 3.03E-02 | 5.36E-06 | 4.90E-06 | 4.90E-06 |
|          | 0.6 * N | 4.63E-02 | 4.71E-06 | 5.20E-06 | 5.20E-06 |
|          | 0.8 * N | 6.34E-02 | 3.21E-06 | 3.70E-06 | 3.70E-06 |
| Function5 | 0.2 * N | 1.97E-02 | 2.79E-04 | 2.20E-04 | 3.07E-04 |
|          | 0.4 * N | 3.22E-02 | 1.42E-04 | 8.38E-05 | 1.52E-04 |
|          | 0.6 * N | 4.66E-02 | 1.14E-04 | 4.82E-05 | 1.35E-04 |
|          | 0.8 * N | 6.53E-02 | 1.24E-04 | 2.35E-05 | 1.28E-04 |
| Function6 | 0.2 * N | 1.69E-02 | 5.77E-06 | 6.75E-06 | 5.77E-06 |
|          | 0.4 * N | 3.13E-02 | 1.04E-05 | 1.13E-05 | 1.13E-05 |
|          | 0.6 * N | 4.66E-02 | 5.87E-06 | 7.63E-06 | 7.63E-06 |
|          | 0.8 * N | 6.19E-02 | 7.38E-06 | 1.03E-05 | 1.03E-05 |
| Function7 | 0.2 * N | 1.69E-02 | 6.05E-08 | 2.54E-08 | 2.60E-08 |
|          | 0.4 * N | 2.84E-02 | 7.00E-07 | 1.96E-07 | 1.96E-07 |
|          | 0.6 * N | 4.56E-02 | 1.99E-07 | 1.52E-07 | 1.52E-07 |
|          | 0.8 * N | 6.16E-02 | 7.45E-07 | 3.65E-07 | 3.65E-07 |
| Function8 | 0.2 * N | 1.50E-02 | 1.61E-03 | 7.86E-03 | 1.93E-02 |
|          | 0.4 * N | 3.34E-02 | 2.31E-03 | 2.86E-03 | 1.93E-02 |
|          | 0.6 * N | 5.19E-02 | 2.23E-03 | 9.51E-04 | 2.05E-02 |
|          | 0.8 * N | 6.56E-02 | 3.05E-03 | 3.02E-04 | 2.29E-02 |
From the Table 2, we can see that simply adding hidden layer nodes cannot guarantee the better performance for ELM. In this experiment, the ELM model with 400 hidden layer nodes achieves the best performance on Function 1, while the ELM model with 800 hidden layer nodes achieves the best performance on Function 5. In other cases, 200 hidden layer nodes are enough to obtain the best performance for ELM models. One explanation is that, because there are a lot of noises in the datasets, arbitrarily setting the number of hidden layer nodes or simply using a lot of hidden layer nodes in ELM models may cause the models to be under-fitting or over-fitting. Due to the number of hidden layer nodes has a significant impact on the performance of ELM, in practical applications, one recommended is to automatically deduce the initial number of hidden nodes by using incremental constructive methods such as incremental ELM (I-ELM) [10], enhanced incremental ELM (EI-ELM) [20], bidirectional ELM (B-ELM) [21], pruned ELM (P-ELM) [22], error minimized ELM (EM-ELM) [23], etc. And then optimize the initial model to obtain the optimal network model.

4.2. Can the empirical randomization range of hidden weights (i.e., [-1, 1]) lead to the optimal performance of ELM?

In many ELM applications, we found that the authors always used the empirical randomization range [-1, 1] as the default selection range of hidden weights without giving any explanation [20, 24, 25]. To test the rationality of this range (i.e., whether this range can always lead to the optimal performance of ELM), we conducted a thorough experiment on the eight benchmark functions described in Table 1. Here, we set the number of hidden layer nodes to 200, set the randomization range of the threshold of hidden nodes to [0, 1], and use Sigmoid function as the activation function of ELM. The variables of testing (i.e., the randomization range of the weights between the input layer and hidden layer) are set to [-1, 1], [-10, 10], [-20, 20], and [-30, 30], respectively. The experimental results are shown in Table 3.

| Dataset | Weights range | Testing time(s) | Standard deviation | Mean Training RMSE | Testing RMSE |
|---------|---------------|----------------|--------------------|------------------|--------------|
| Function3 | [-1,1] | 1.72E-02 | 1.67E-02 | 2.76E-01 | **4.16E-01** |
| | [-10,10] | 2.28E-02 | 1.45E-02 | 2.84E-01 | **4.20E-01** |
| | [-20,20] | 1.78E-02 | 1.67E-02 | 2.84E-01 | **4.16E-01** |
| | [-30,30] | 1.25E-02 | 1.25E-02 | 2.83E-01 | **4.21E-01** |
| Function4 | [-1,1] | 1.75E-02 | 2.92E-06 | 1.93E-06 | **1.93E-06** |
| | [-10,10] | 1.81E-02 | 5.41E-06 | 6.09E-06 | **6.09E-06** |
| | [-20,20] | 2.09E-02 | 5.65E-06 | 4.72E-06 | **4.72E-06** |
| | [-30,30] | 1.38E-02 | 1.25E-02 | 4.15E-06 | **4.15E-06** |
| Function5 | [-1,1] | 1.97E-02 | 2.79E-04 | 2.20E-04 | **3.07E-04** |
| | [-10,10] | 1.94E-02 | 6.01E-06 | 3.80E-06 | **3.80E-06** |
| | [-20,20] | 2.00E-02 | 1.11E-03 | 2.43E-06 | **5.73E-04** |
| | [-30,30] | 1.75E-02 | 1.23E-03 | 2.17E-06 | **7.86E-04** |
| Function6 | [-1,1] | 1.69E-02 | 5.77E-06 | 6.75E-06 | **5.77E-06** |
| | [-10,10] | 1.28E-02 | 8.48E-06 | 6.35E-06 | **6.35E-06** |
| | [-20,20] | 1.81E-02 | 5.00E-06 | 4.36E-06 | **4.36E-06** |
| | [-30,30] | 1.22E-02 | 5.27E-06 | 4.27E-06 | **4.27E-06** |

As shown in Table 3, the randomization range of the weights between the input layer and hidden layer has significant impact on the performance of ELM. In addition, we found that the empirical randomization range (i.e., [-1, 1]) may not lead to the optimal performance of ELM models, e.g. Function 3/5/6 cases, which implies that many existing ELM models are likely to achieve better performance by tuning the randomization range of hidden weights.
4.3. Can the empirical randomization range of the hidden threshold (i.e., \([0, 1]\)) lead to the optimal performance of ELM?

Similar to Section 4.2, here we test that whether the most common randomization range (i.e., \([0, 1]\)) of the threshold of hidden nodes can lead to the optimal performance of ELM. In this experiment, the number of hidden layer nodes is set to 200, the randomization range of the hidden weights is set to \([-1, 1]\), and **Sigmoid function** is used as the activation function of ELM. The variables of testing (i.e., the randomization range of the threshold of hidden nodes) are set to \([0, 1]\), \([0, 5]\), \([0, 10]\), and \([0, 15]\), respectively. The experimental results are shown in Table 4.

**Table 4.** The performance of ELM based on different randomization ranges of bias.

| Dataset | Bias range | Testing time(s) | Standard deviation | Mean |
|---------|------------|-----------------|--------------------|------|
|         |            |                 | Training RMSE      | Testing RMSE |
| Function1 | [0, 1] | 1.56E-02 | **4.98E-08** | 2.79E-08 | 3.62E-08 |
|          | [0, 5]   | 1.50E-02 | 9.58E-08 | 3.11E-08 | **3.15E-08** |
|          | [0, 10]  | 1.53E-02 | 7.61E-07 | 5.60E-07 | 5.60E-07 |
|          | [0, 15]  | 1.69E-02 | 6.29E-06 | 5.98E-06 | 5.98E-06 |
| Function2 | [0, 1]   | 1.59E-02 | **6.80E-09** | 6.18E-09 | **7.33E-09** |
|          | [0, 5]   | 1.56E-02 | 8.22E-08 | 5.12E-08 | 5.12E-08 |
|          | [0, 10]  | 1.47E-02 | 2.34E-06 | 1.12E-06 | 1.12E-06 |
|          | [0, 15]  | 1.06E-02 | 4.68E-06 | 4.86E-06 | 4.86E-06 |
| Function3 | [0, 1]   | 1.72E-02 | 1.67E-02 | 2.76E-01 | **4.16E-01** |
|          | [0, 5]   | 1.75E-02 | **1.47E-02** | 2.88E-01 | 4.31E-01 |
|          | [0, 10]  | 1.88E-02 | 2.74E-02 | 2.94E-01 | 4.77E-01 |
|          | [0, 15]  | 1.34E-02 | 8.35E-01 | 2.94E-01 | 1.21E+00 |
| Function4 | [0, 1]   | 1.75E-02 | **2.92E-06** | 1.93E-06 | **1.93E-06** |
|          | [0, 5]   | 1.84E-02 | 5.30E-06 | 4.96E-06 | 4.96E-06 |
|          | [0, 10]  | 1.34E-02 | 7.41E-06 | 8.06E-06 | 8.06E-06 |
|          | [0, 15]  | 1.16E-02 | 1.53E-05 | 1.63E-05 | 1.63E-05 |
| Function5 | [0, 1]   | 1.97E-02 | 2.79E-04 | 2.20E-04 | 3.07E-04 |
|          | [0, 5]   | 2.09E-02 | 6.16E-05 | 2.31E-05 | 3.77E-05 |
|          | [0, 10]  | 1.41E-02 | 6.68E-07 | 3.76E-07 | 5.28E-07 |
|          | [0, 15]  | 1.53E-02 | **2.05E-08** | 1.20E-08 | **1.34E-08** |
| Function6 | [0, 1]   | 1.69E-02 | **5.77E-06** | 6.75E-06 | **5.77E-06** |
|          | [0, 5]   | 1.09E-02 | 9.27E-06 | 1.06E-05 | 1.06E-05 |
|          | [0, 10]  | 1.38E-02 | 1.40E-05 | 1.48E-05 | 1.48E-05 |
|          | [0, 15]  | 1.44E-02 | 1.31E-05 | 1.42E-05 | 1.42E-05 |
| Function7 | [0, 1]   | 1.69E-02 | **6.05E-08** | 2.54E-08 | **2.60E-08** |
|          | [0, 5]   | 1.44E-02 | 6.81E-07 | 1.87E-07 | 1.87E-07 |
|          | [0, 10]  | 2.06E-02 | 4.23E-06 | 2.39E-06 | 2.39E-06 |
|          | [0, 15]  | 1.25E-02 | 7.83E-06 | 7.19E-06 | 7.19E-06 |
| Function8 | [0, 1]   | 1.50E-02 | **1.61E-03** | 7.86E-03 | 1.93E-02 |
|          | [0, 5]   | 1.72E-02 | 2.46E-03 | 3.65E-03 | **1.64E-02** |
|          | [0, 10]  | 1.00E-02 | 1.85E-02 | 9.89E-04 | 2.52E-02 |
|          | [0, 15]  | 1.91E-02 | 1.08E-01 | 5.64E-04 | 7.12E-02 |

From Table 4, we can see that the empirical randomization range (i.e., \([0, 1]\)) may not always lead to the best performance of ELM models, e.g. Function 1/3/5/8 cases. Similarly, it implies that some existing ELM models may achieve better performance by tuning the randomization range of the threshold of hidden nodes.
To test the impact of different activation functions on the performance of ELM model, we conducted a set of experiments. Here, we set the number of hidden layer nodes to 200, set the randomization range of the hidden weights to $[-1, 1]$, and set the randomization range of the threshold of hidden nodes to $[0, 1]$. The variables of testing (i.e., the functions that used as activation functions) are set to $sigmoid$ (Sigmoid function), $sine$ (Sine function), $hardlim$ (Hardlim function), $tribas$ (Triangular basis function), and $radbas$ (Radial basis function), respectively. The experimental results are shown in Table 5.

**Table 5.** The performance of ELM based on different activation functions.

| Dataset | Activation functions | Testing time(s) | Standard deviation | Mean Testing RMSE |
|---------|----------------------|-----------------|--------------------|-------------------|
| Function1 | Sigmoid             | 1.56E-02        | 4.98E-08           | 2.79E-08          | 3.62E-08         |
|         | sine                | 3.30E-03        | 1.10E-04           | 2.61E-03          | 3.30E-03         |
|         | hardlim            | 9.69E-03        | 2.57E-04           | 4.31E-15          | 2.44E-04         |
|         | tribas              | 1.19E-02        | 6.40E-03           | 4.53E-02          | 4.82E-02         |
|         | radbas             | 8.75E-03        | 2.48E-03           | 1.43E-02          | 1.73E-02         |
| Function2 | Sigmoid             | 1.59E-02        | 6.80E-09           | 6.18E-09          | 7.33E-09         |
|         | sine                | 2.03E-02        | 1.55E-04           | 1.30E-03          | 1.66E-03         |
|         | hardlim            | 1.06E-02        | 1.55E-04           | 4.82E-15          | 7.99E-05         |
|         | tribas              | 1.22E-02        | 5.59E-03           | 3.28E-02          | 3.46E-02         |
|         | radbas             | 1.34E-02        | 1.38E-03           | 7.60E-03          | 9.08E-03         |
| Function3 | Sigmoid             | 1.72E-02        | 1.67E-02           | 2.76E-01          | 4.16E-01         |
|         | sine                | 2.19E-02        | 1.63E-02           | 5.04E-01          | 6.48E-01         |
|         | hardlim            | 8.44E-03        | 1.41E-02           | 2.89E-01          | 4.18E-01         |
|         | tribas              | 8.13E-03        | 2.45E-02           | 5.29E-01          | 7.05E-01         |
|         | radbas             | 1.53E-02        | 2.14E-02           | 4.70E-01          | 6.36E-01         |
| Function4 | Sigmoid             | 1.75E-02        | 2.92E-06           | 1.93E-06          | 1.93E-06         |
|         | sine                | 1.25E-02        | 1.02E-07           | 2.04E-06          | 2.57E-06         |
|         | hardlim            | 1.75E-02        | 4.30E-05           | 8.81E-15          | 6.14E-06         |
|         | tribas              | 1.50E-02        | 3.07E-04           | 1.61E-03          | 1.65E-03         |
|         | radbas             | 1.53E-02        | 1.04E-05           | 2.41E-05          | 2.76E-05         |
| Function5 | Sigmoid             | 1.97E-02        | 2.79E-04           | 2.20E-04          | 3.07E-04         |
|         | sine                | 2.06E-02        | 5.82E-03           | 1.41E-01          | 1.78E-01         |
|         | hardlim            | 1.13E-02        | 1.06E-03           | 2.85E-15          | 2.53E-03         |
|         | tribas              | 1.09E-02        | 3.08E-01           | 1.33E-01          | 2.41E-01         |
|         | radbas             | 1.38E-02        | 1.31E+02           | 8.64E-02          | 3.63E+01         |
| Function6 | Sigmoid             | 1.69E-02        | 5.77E-06           | 6.75E-06          | 5.77E-06         |
|         | sine                | 1.38E-02        | 5.01E-09           | 8.61E-08          | 1.12E-07         |
|         | hardlim            | 6.25E-03        | 4.43E-05           | 8.54E-15          | 6.32E-06         |
|         | tribas              | 1.13E-02        | 4.56E-05           | 2.93E-04          | 3.02E-04         |
|         | radbas             | 1.78E-02        | 5.55E-07           | 9.26E-07          | 1.05E-06         |
| Function7 | Sigmoid             | 1.69E-02        | 6.05E-08           | 2.54E-08          | 2.60E-08         |
|         | sine                | 2.22E-02        | 2.46E-05           | 5.46E-04          | 7.00E-04         |
|         | hardlim            | 1.19E-02        | 1.78E-04           | 6.34E-15          | 1.18E-04         |
|         | tribas              | 1.19E-02        | 2.25E-02           | 2.04E-02          | 2.53E-02         |
|         | radbas             | 1.19E-02        | 6.44E-04           | 3.59E-03          | 4.19E-03         |
As shown in Table 5, we found that the choice of the activation function has a significant impact on the performance of ELM. In our experiments, the ELM models with sigmoid as activation function can achieve the best performance in most cases, while the models with tribas as activation function always obtain suboptimal results. One explanation is that the datasets generated by different benchmark functions have different data distributions, and the effect of the feature mappings generated by each data distribution under different activation functions is quite different, which directly affect the performance of ELM. In other words, the sigmoid function may perform better nonlinear transformations and obtain more meaningful feature mappings than the tribas function on these benchmark functions.

5. Conclusions and future work
In this paper, we did a comprehensive study on the relationship between the parameters and the performance of ELM based on eight benchmark functions. Based on the experimental results, we have the following interesting findings.

1) More hidden layer nodes may not always lead to the better performance of ELM. Arbitrarily setting the number of hidden nodes may cause the ELM model to be under-fitting or over-fitting. One recommendation is to use the incremental constructive method to obtain an initial network model, and then prune the unimportant hidden nodes to obtain the optimal network model.

2) The empirical randomization range (i.e., [-1, 1]) of the weights between the input layer and hidden layer and the empirical randomization range (i.e., [0, 1]) of the threshold of hidden nodes cannot guarantee better performance for ELM models. One recommendation is to use the grid search or random search to obtain the best parameters for ELM model.

3) The effects of five commonly used activation functions on the performance of ELM are shown in Table 5. It turns out that the sigmoid function always leads to better performance than the tribas function on these regression problems.

In the future, we will give a complete theoretical proof for these phenomena and establish the quantitative relationship between the parameters and the performance of ELM. In addition, ELMs with kernel functions and the multi-layer ELMs will be studied. We hope this study could provide guidance for researchers to select appropriate parameters for ELM.

Acknowledgments
This research is supported by the National Natural Science Foundation of China under Grant nos. 61672358.

References
[1] Chauvin Y and Rumelhart DE 1995 Backpropagation: theory, architectures, and applications (Hillsdale, NJ: Lawrence Erlbaum)
[2] Schmidt WF, Kraaijveld MA and Duin RPW 1992 Feedforward neural networks with random weights, Proceedings of 11th IAPR International Conference on Pattern Recognition, Conference B: Pattern Recognition Methodology and Systems, IEEE, Vol. 2, pp. 1–4.
[3] Pao YH and Takefji Y 1992 Functional-link net computing: theory, system architecture, and functionalities, Computer, Vol. 25, No. 5, pp. 76–79.
[4] Tang J, Deng C and Huang GB 2016 Extreme learning machine for multilayer perceptron, IEEE Trans. Neural Netw. Learn Syst., Vol. 27, No. 4, pp. 809–821.
[5] Hu JY, Zhang JS, Zhang CX and Wang J 2016 A new deep neural network based on a stack of single-hidden-layer feedforward neural networks with randomly fixed hidden neurons, Neurocomputing, Vol. 171, pp. 63-72.
[6] Gu Y, Chen YQ, Liu JF and Jiang XL 2015 Semi-supervised deep extreme learning machine for Wi-Fi based localization, Neurocomputing, Vol. 166, pp. 282-293.
[7] Zhang L and Suganthan PN 2016 A comprehensive evaluation of random vector functional link networks, Inf. Sci., Vol. 367, pp. 1094-1105.
[8] Li M and Wang DH 2016 Insights into randomized algorithms for neural networks Practical issues and common pitfalls, *Inf. Sci.*, Vol. 382-383, pp. 170-178.
[9] Huang GB, Zhu QY and Siew CK 2004 Extreme learning machine a new learning scheme of feedforward neural networks, *Proceedings of 2004 IEEE International Joint Conference on Neural Networks*, IEEE, Vol. 2, pp. 985-990.
[10] Huang GB, Chen L and Siew CK 2006 Universal approximation using incremental constructive feedforward networks with random hidden nodes, *IEEE Trans. Neural Netw.*, Vol. 17, pp. 879–892.
[11] Huang GB, Zhu QY and Siew CK 2006 Extreme learning machine: theory and applications, *Neurocomputing*, Vol. 70, pp. 489–501.
[12] Mohammed AA, Minhas R, Wu QMJ and Sid-Ahmed MA 2011 Human face recognition based on multidimensional PCA and extreme learning machine, *Pattern Recog.*, Vol. 44, No. 10, pp. 2588–2597.
[13] Lu J, Zhao JW and Cao FL 2014 Extended feed forward neural networks with random weights for face recognition, *Neurocomputing*, Vol. 136, pp. 96-102.
[14] Wong PK, Yang ZX, Vong CM and Zhong JH 2014 Real-time fault diagnosis for gas turbine generator systems using extreme learning machine, *Neurocomputing*, Vol. 128, pp. 249-257.
[15] Bakri AE, Koumir M and Boughnidi I 2016 Extreme learning machine for fault detection and isolation in wind turbine, *International Conference on Electrical and Information Technologies (ICEIT)*, IEEE.
[16] Guo LL and Ding SF 2015 A hybrid deep learning CNN-ELM model and its application in handwritten numeral recognition, *Journal of Computational Information Systems*, Vol. 11, No. 7, pp. 2673-2680.
[17] Xie ZG, Xu K, Shan W, Liu LG, Xiong YS and Huang H 2015 Projective Feature Learning for 3D Shapes with Multi-View Depth Images, *Comput. Graph. Forum*, Vol. 34, No. 7, pp. 1-11.
[18] He YL, Wang XZ and Huang JZX 2016 Fuzzy nonlinear regression analysis using a random weight network, *Inf. Sci.*, Vol. 364-365, pp. 222-240.
[19] Huang GB 2015 What are extreme learning machines? Filling the gap between frank Rosenblatt’s dream and John von Neumann’s puzzle, *Cogn. Comput.*, Vol. 7, No. 3, pp. 263-278.
[20] Huang GB and Chen L 2008 Enhanced random search based incremental extreme learning machine, *Neurocomputing*, Vol. 71, pp. 3460-3468.
[21] Yang YM, Wang YN and Yuan XF 2012 Bidirectional extreme learning machine for regression problem and its learning effectiveness, *IEEE Trans. Neural Netw. Learn Syst.*, Vol. 23, No.9, pp.1498-1505.
[22] Rong HJ, Ong YS, Tan AH and Zhu Z 2008 A fast pruned-extreme learning machine for classification problem, *Neurocomputing*, Vol. 72, pp. 359-366.
[23] Feng GR, Huang GB, Lin QP and Gay R 2009 Error minimized extreme learning machine with growth of hidden nodes and incremental learning, *IEEE Trans. Neural Netw.*, Vol. 20, No. 8, pp. 1352-1357.
[24] Zhao XM, Cao WP, Zhu HY, Ming Z and Ashfaq RAR 2016 An initial study on the rank of input matrix for extreme learning machine, *Int. J. Mach. Learn. Cybern.*, pp. 1-13.
[25] Salam MA, Zawbaa HM, Emary E, Ghany KKA and Parv B 2016 A hybrid dragonfly algorithm with extreme learning machine for prediction, *International Symposium on INnovations in Intelligent SysTems and Applications (INISTA)*, IEEE.