Hindawi
Mathematical Problems in Engineering
Volume 2020, Article ID 9175106, 10 pages
https://doi.org/10.1155/2020/9175106

*Research Article*

# Smoothing $L_0$ Regularization for Extreme Learning Machine

**Qinwei Fan and Ting Liu**

*School of Science, Xi'an Polytechnic University, Xi'an 710048, China*

Correspondence should be addressed to Qinwei Fan; qinweifan@126.com

Received 8 May 2020; Revised 12 June 2020; Accepted 15 June 2020; Published 6 July 2020

Academic Editor: Kauko Leiviskä



Extreme learning machine (ELM) has been put forward for single hidden layer feedforward networks. Because of its powerful modeling ability and it needs less human intervention, the ELM algorithm has been used widely in both regression and classification experiments. However, in order to achieve required accuracy, it needs many more hidden nodes than is typically needed by the conventional neural networks. This paper considers a new efficient learning algorithm for ELM with smoothing $L_0$ regularization. A novel algorithm updates weights in the direction along which the overall square error is reduced the most and then this new algorithm can sparse network structure very efficiently. The numerical experiments show that the ELM algorithm with smoothing $L_0$ regularization has less hidden nodes but better generalization performance than original ELM and ELM with $L_1$ regularization algorithms.

## 1. Introduction

Recently, the studies and applications of artificial intelligence have raised a new high tide. Artificial neural networks, also known as neural networks, are an usual artificial intelligence learning algorithm, which can autonomously learn the characteristics of data, hence avoiding the course of artificial selection.

Among them, the most common network type is feedforward neural networks (SLFNs), such as perceptron, back-propagation (BP), and radial basis function (RBF) networks. Generally, it includes an input layer, several hidden layers, and an output layer. Each neuron is arranged in layers, only connected to the antecedent layer. It takes the output of the previous layer and delivers it to the next layer. There is no feedback between the layers. As a result of its good characteristics, it has been widely used in various fields [1–4]. However, the drawback of the feedforward neural network is inefficiency, especially when it deals with complex data.

Lately, ELM was proposed by Huang et al. [5–7]. Other than the conventional neural network, ELM is a new type of SLFNs, which input weights and the thresholds of the hidden layer can be discretionarily assigned if the activation function of the hidden layer is immortally differentiable. The output weights can be decided by Moore–Penrose generalized inverse. In some simulations, the learning speed of the ELM algorithm can be completed in seconds [8]. It is extremely fast and has better generalization performance than other primitive gradient-based learning algorithms.

However, in order to guarantee the veracity of regression and classification experiments, the ELM method needs a huge number of hidden nodes. This leads to a particularly complex network structure. It inevitably increases the model size and testing time. Therefore, in order to ameliorate the predictive power and generalization performance of ELM, choosing the appropriate quantity of hidden layer nodes is a hot topic in the research of ELM.

Many scholars have done a lot of research on the hidden layer structure optimization of ELM and achieved many remarkable results. Rong et al. [9] proposed a fast pruned ELM (P-ELM) and successfully applied it to the classification problem. This learning mechanism mainly achieves the effect of "pruning" the network structure by deleting hidden layer nodes irrelevant to class labels. Then, Miche et al. [10] proposed an optimal extreme learning machine (OP-ELM) for optimal pruning and extended the pruning method. Incremental extreme learning machine (I-ELM) was put

forward by Huang et al. [11], and it can increase the quantity of hidden layer nodes adaptively. The efficiency of learning is improved, and the network performance is optimized by increasing nodes in the training process. In [12], Yu and Deng proposed a series of new efficient algorithms for training ELM, which exploit both the structure of SLFNs and the gradient information over all training epochs.

In conclusion, the number of neurons is an important factor that determines the structure of the network. There are two methods to adjust the size of the network, one is the growing method, and the other is the pruning method. For the growing method, a common strategy is to begin with a smaller network and then add new hidden nodes during the process of training the network [13, 14]. The pruning method starts with a large network structure and then prunes unimportant weights or nodes [15–19]. There are some network structure optimization methods, such as regularization method, cross validation, sensitivity analysis, mutual information-based, and magnitude-based methods [20–23].

In this paper, based on the regularization method, we propose a new efficient algorithm to train ELM. This novel algorithm updates the weights in the direction along which the overall square error is reduced the most and then to the direction which enforces sparsity of the weights effectively. Our strategy is to combine the $L_0$ regularization term with the standard error function. On the basis of the regularization theory, when $p \longrightarrow 0$, the $L_p$ regularization method tends to produce more sparse results. However, it is not differentiable at the origin of coordinates and leads to NP-hard problem [24]. So, $L_0$ regularization cannot make use of the optimization algorithm directly [25]. Also, there are some other sparse strategies for optimized structure of neural networks [26, 27]. Inspired by the work on smoothing $L_{1/2}$ regularization for feedforward neural networks [28, 29] and other optimizing strategies, this paper draws on some smoothing techniques to overcome the shortcomings of the origin is not differentiable and the phenomenon of oscillation, and we will show the related details in the next section. The main contributions are as follows:

(1) It is shown how the smoothing $L_0$ regularization is proposed to train ELM, which can discriminate important weights from unnecessary weights and drives the unnecessary weights to zeros, thus effectively simplified the structure of the network.

(2) The ideal approximate solution is obtained by using the smoothing approximation techniques. The drawbacks of nonsmooth and nondifferentiable of the normal $L_0$ regularization term can be addressed. And this effectively prevents the oscillatory phenomenon of the training.

(3) Numerical simulations shown that the sparsity effect and the generalization performance of the ELMSL0 are better than those of the original ELM and ELML1. For example, from Tables 1–3, the results clearly show that the novel algorithm ELMSL0 uses fewer neurons but has higher testing accuracy for most of the data sets.

Table 1: Comparison of RMSE in ELM, ELML1, and ELMSL0 algorithms on the SinC data set.

| Learning algorithms | Training (RMSE) | Testing (RMSE) | Average nodes |
|---|---|---|---|
| ELM | 0.1157 | 0.0096 | 50 |
| ELML1 | 0.1158 | 0.0083 | 33.90 |
| ELMSL0 | 0.1158 | 0.0080 | 31.30 |

Table 2: Comparison of RMSE in ELM, ELML1, and ELMSL0 algorithms on Gabor function.

| Learning algorithms | Training (RMSE) | Testing (RMSE) | Average nodes |
|---|---|---|---|
| ELM | 0.2262 | 0.0408 | 100 |
| ELML1 | 0.2279 | 0.0369 | 64.60 |
| ELMSL0 | 0.2272 | 0.0346 | 60.45 |

The remainder of this paper is organized as follows. Section 2 describes the network structure of the traditional ELM algorithm. Section 3 shows the ELM algorithm with smoothing $L_0$ regularization (ELMSL0). Section 4 shows that how the smoothing $L_0$ regularization helps the gradient-based method to bring forth sparse results. The performance of the ELMSL0 algorithm is compared with ELM and ELM with $L_1$ regularization (ELML1) algorithms by regression and classification applications in Section 5. Finally, conclusion is given in Section 6.

## 2. Extreme Learning Machine (ELM)

Next, we will do a description of the traditional ELM. The neuron quantities of the input, hidden, and output layers are $n$, $L$, and $m$, respectively (see Figure 1). The ELM algorithm randomly generates the connection weights of the input layer and hidden layer, and the thresholds of the hidden layer neurons need not be adjusted in the process of training the network. As long as the number of hidden layer neurons is set, the unique optimal solution can be acquired. The input matrix $X$ and output matrix Y of the training set with $Q$ samples are, respectively,

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

On the basis of the structure of ELM, the actual output $T$ of the network is as follows:

Table 3: Comparison of success rate in ELM, ELML1, and ELMSL0 algorithms on classification data sets.

| Data set | Training accuracy | | | Testing accuracy | | | Average nodes | | |
|---|---|---|---|---|---|---|---|---|---|
| | ELM (%) | ELML1 (%) | ELMSL0 (%) | ELM (%) | ELML1 (%) | ELMSL0 (%) | ELM | ELML1 | ELMSL0 |
| Diabetes | 83.40 | 81.62 | 81.13 | 73.97 | 75.50 | 76.68 | 100 | 63.90 | 60.20 |
| Australiancredit | 90.79 | 89.91 | 89.17 | 84.88 | 85.57 | 86.50 | 100 | 63.40 | 59.50 |
| Heart | 89.72 | 87.92 | 88.11 | 78.17 | 80.28 | 82.89 | 50 | 31.80 | 29.05 |
| Sonar | 100.00 | 100.00 | 100.00 | 77.94 | 78.53 | 82.50 | 500 | 315.90 | 310.10 |
| Wiscousin | 98.52 | 97.56 | 97.83 | 94.83 | 96.45 | 96.62 | 100 | 63.50 | 59.70 |
| SPECT | 96.56 | 96.00 | 96.56 | 72.08 | 70.67 | 69.36 | 300 | 189.60 | 186.30 |
| SPECTF | 100.00 | 100.00 | 100.00 | 60.35 | 60.40 | 59.01 | 300 | 192.15 | 187.95 |
| Iris | 98.90 | 99.25 | 98.80 | 91.80 | 92.40 | 96.00 | 50 | 30.70 | 27.60 |
| Wine | 100.00 | 100.00 | 99.92 | 92.76 | 93.28 | 96.55 | 70 | 41.80 | 39.70 |
| Glass | 98.47 | 96.53 | 96.67 | 84.92 | 80.17 | 88.20 | 60 | 35.80 | 33.75 |
| Olitos | 99.35 | 96.79 | 96.00 | 71.86 | 76.94 | 80.75 | 50 | 29.80 | 27.20 |
| *E.coli* | 92.78 | 91.27 | 90.89 | 81.35 | 84.60 | 86.10 | 60 | 35.55 | 32.60 |
| Seeds | 99.35 | 98.57 | 98.20 | 94.05 | 94.29 | 95.00 | 50 | 30.35 | 27.65 |
| Wholesale | 90.21 | 83.36 | 82.37 | 49.62 | 58.75 | 60.57 | 200 | 125.85 | 121.20 |

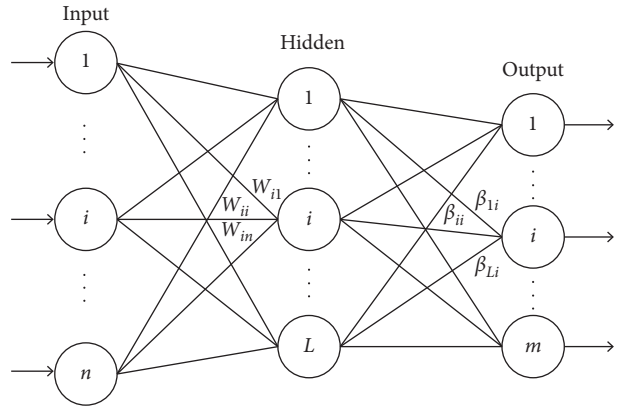


Figure 1: The network structure of extreme learning machine.

$$T = \begin{bmatrix} t_{11} & t_{12} & \cdots & t_{1Q} \\ t_{21} & t_{22} & \cdots & t_{2Q} \\ \vdots & \vdots & \cdots & \vdots \\ t_{m1} & t_{m2} & \cdots & t_{mQ} \end{bmatrix}_{m\times Q}. \tag{2}$$

The normative ELM with $L$ hidden nodes and activation function $g(x)$ can be expressed as

$$\sum_{i=1}^{L} \beta_i g_i(x_j) = \sum_{i=1}^{L} \beta_i g(w_i \cdot x_j + b_i) = t_j, \tag{3}$$

where $t_j = [t_{1j}, t_{2j}, \ldots, t_{mj}]^T$, and the specific expression of $t_j$ is as follows:

$$t_j = \begin{bmatrix} \sum_{i=1}^{L} \beta_{i1} g(w_i \cdot x_j + b_i) \\ \sum_{i=1}^{L} \beta_{i2} g(w_i \cdot x_j + b_i) \\ \vdots \\ \sum_{i=1}^{L} \beta_{im} g(w_i \cdot x_j + b_i) \end{bmatrix}_{m\times 1}, \tag{4}$$

where $j = 1, 2, \ldots, Q$, $x_j = [x_{1j}, x_{2j}, \ldots, x_{nj}]^T$, $w_i = [w_{i1}, w_{i2}, \ldots, w_{\text{in}}]$ is the weight vector connecting the $i$th hidden node and input nodes, $\beta_i = [\beta_{i1}, \beta_{i2}, \ldots, \beta_{im}]^T$ is the weight vector connecting the $i$th hidden node and output nodes, $b_i$ is the threshold of the $i$th hidden node, and $w_i \cdot x_j$ denotes the inner product of $w_i$ and $x_j$.

Equation (4) can be written as $H\beta = T^T$, and H is called the hidden layer output matrix of the ELM, where

$$H(w_1, \ldots, w_L, b_1, \ldots, b_L, x_1, \ldots, x_Q) = \begin{bmatrix} g(w_1 \cdot x_1 + b_1) & \cdots & g(w_L \cdot x_1 + b_L) \\ g(w_1 \cdot x_2 + b_1) & \cdots & g(w_L \cdot x_2 + b_L) \\ \vdots & \cdots & \vdots \\ g(w_1 \cdot x_Q + b_1) & \cdots & g(w_L \cdot x_Q + b_L) \end{bmatrix}_{Q\times L}. \tag{5}$$

By using the Moore–Penrose generalized inverse of $H$, the least-squares solutions of $H\beta = T^T$ can be written as

$$\widehat{\beta} = H^{\dagger} T^T. \tag{6}$$

## 3. Extreme Learning Machine with $L_0$ Regularization (ELMSL0)

In the general case, the weights with large absolute value play a more important role in training. In order to prune the network availably, we need to distinguish the unimportant weights in the first place and then remove them. Therefore, the aim of the network training is to find the appropriate weights $\widehat{\beta}$ that can minimize $\|H\beta - Y^T\|$ as well as $\|\beta\|_p$:

$$\widehat{\beta} = \arg\min \left\|H\beta - Y^T\right\|_2^2 + \lambda\|\beta\|_p^p, \tag{7}$$

where $\lambda$ is the regularization coefficient to balance the training accuracy and the complexity of the network. $\|\beta\|_p^p$ is called $L_p$ regularization, and it shows different properties for different values of $p$. Moreover, $L_2$ regularization can prevent the weight from increasing too large effectively, but it is not sparse. $L_1$ regularization can generate sparse solutions. $L_0$ regularization produces more sparse solutions than $L_1$ regularization. So, we focus on the $L_0$ regularization in this paper. Here, $\|\beta\|_0^0$ is the $L_0$ regularization with the $L_0$ norm defined by

$$\begin{aligned}\|\beta\|_0^0 &= \lim_{p\longrightarrow 0}\left(\left(|\beta_1|^p + |\beta_2|^p + \cdots + |\beta_n|^p\right)^{1/p}\right)^p \\ &= \lim_{p\longrightarrow 0}\left(|\beta_1|^p + |\beta_2|^p + \cdots + |\beta_n|^p\right),\end{aligned} \tag{8}$$

for $\beta = (\beta_1, \beta_2, \ldots, \beta_n)^T$. We know that the quantity of nonzero elements of the vector $\beta$ equals to $\|\beta\|_0^0$. However, according to combinatorial optimization theory, we know that minimizing $\|H\beta - Y^T\|_2^2 + \lambda\|\beta\|_0^0$ is a NP-hard problem. In order to overcome this drawback, the following continuous function $H_\rho(\beta)$ is employed to approximate the $L_0$ regularization at the origin:

$$H_\rho(\beta) = \sum_{i=1}^{n} h_\rho(\beta_i). \tag{9}$$

Here, $h_\rho(\cdot)$ is continuously differentiable on $\mathbb{R}$ and

$$\lim_{p\longrightarrow 0} h_\rho(\beta) = \begin{cases} 1, & \text{if } \beta \neq 0, \\ 0, & \text{if } \beta = 0, \end{cases} \tag{10}$$

where $\rho$ is a positive number, and it is used to control the degree of how $H_\rho(\beta)$ approximates the $L_0$ regularization. A representative choice for $h_\rho(\beta)$ is

$$h_\rho(\beta) = 1 - \exp\left(\frac{-\beta^2}{2\rho^2}\right). \tag{11}$$

To sum up, the error function with $L_0$ regularization has the following form:

$$E(\beta) = \sum_{k=1}^{Q}\left(H\beta - Y^T\right)_{k*}^T\left(H\beta - Y^T\right)_{k*} + \lambda H_\rho(\beta), \tag{12}$$

where $(H\beta - Y^T)_{k*}$ represents the $k$th row of the matrix $(H\beta - Y^T)$. We use the gradient descent method to minimize the error function, and the gradient of the error function is given by

$$E_{\beta_i}(\beta) = \sum_{k=1}^{Q} 2\left(H_{ki}\left(H\beta - Y^T\right)_{ki}\right) + \lambda\frac{\beta_i}{\rho^2}\exp\left(\frac{-\beta_i^2}{2\rho^2}\right), \tag{13}$$

where $i = 1, 2, \ldots, L$. For any initial value $\beta^0$, the batch gradient method with the smoothing $L_0$ regularization term updates the weights by

$$\beta^{n+1} = \beta^n - \eta E_{\beta_i}(\beta), \quad n = 0, 1, 2, \ldots, \tag{14}$$

where $\eta > 0$ is the learning rate.

## 4. Description of Sparsity

Regularized sparse model plays an increasingly important role in machine learning and image processing. It removes a large number of redundant variables and only retains the explanatory variables that are most relevant to the response variables, which simplifies the model while retaining the most important information in the data sets and effectively solves many practical problems.

Next, we show that the ELMSL0 algorithm differentiates between important weights and unimportant weights. Among them, the weights with large absolute value are more important. The curves of $h_\rho(\beta)$ and $h'_\rho(\beta)$ for different $\rho$ are shown in Figures 2 and 3. Notice that the only minimum point of $h_\rho(\beta)$ is $\beta = 0$. On the grounds of (10), for adequately small $\rho$, there exists a positive number $\beta_0$, such that $h'_\rho(\beta) \approx 0$ when $\beta > \beta_0$ and $h'_\rho(\beta)$ will be quite large when $\beta < \beta_0$. Therefore, when using the ELMSL0 algorithm to train the network, the absolute value is greater than the weight of $\beta_0$, which is not easily affected by the regularization term. The unimportant weights are absolutely less than $\beta_0$, they will be driven to zero during the training. This makes clear why the smoothing $L_0$ regularization term can help the gradient-based method achieve sparse results.

In the light of the above discussion, we look forward to more weights fall into the interval $(-\beta_0, \beta_0)$ to achieve the more sparse results. To make this true, one choice is to set the very small initial weights, but it leads to slow convergence at the beginning of the training procedure on account of the gradient of the square error function will be very small [30]. The weight decay method is another choice, and the size of the network weight is reduced compulsively during the training process. As shown in Figure 3, if the parameter $\rho$ is set too small, it will be unstable and affect the performance of the algorithm. Therefore, the parameter $\rho$ should be set to a decreasing sequence with lower bounds.

## 5. Simulation Results

In order to substantiate the reliability of the introduced ELMSL0 algorithm, we conduct some experiments which are both regression and classification applications. In Section 5.1, ELM, ELML1, and ELMSL0 algorithms are used to approximate the SinC and multidimensional Gabor function. There are a few real-word classification data to test the performance of three algorithms in Section 5.2.

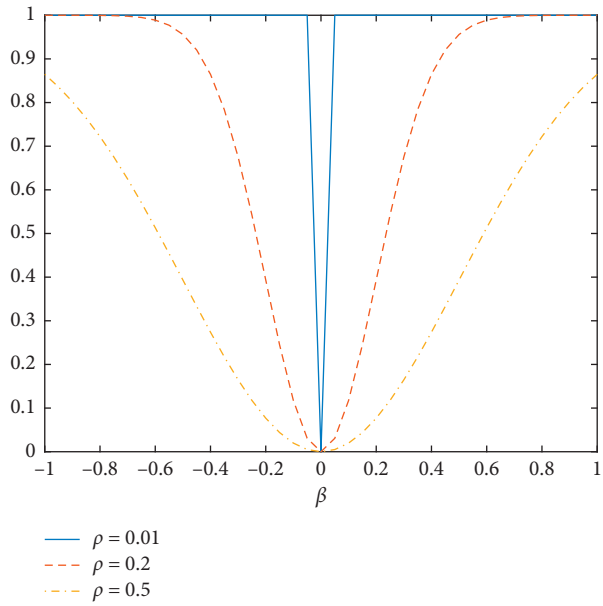


Figure 2: The curve of $h_\rho(\beta)$ for different $\rho$.

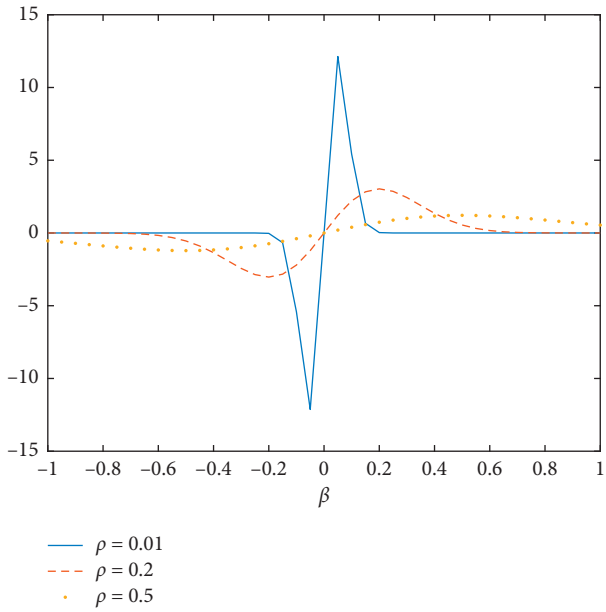


Figure 3: The curve of $h'_\rho(\beta)$ for different $\rho$.

*5.1. Function Regression Problem.* The SinC function is defined by

$$y(x) = \begin{cases} \sin(x)/x, & x \neq 0, \\ 1, & x = 0, \end{cases} \tag{15}$$

and it has a training set $(x_i, y_i)$ and testing set $(x_i, y_i)$ with 5000 data, where $x_i$ is uniformly distributed on the interval $(-10, 10)$. To make the regression problem more real, adding the uniform noise distributed in $[-0.2, 0.2]$ to all the training samples, the testing samples remain noise-free. There are 50 hidden nodes, and the activation function is RBF for ELM, ELML1, and ELMSL0 algorithms. We choose the learning rate $\eta = 0.2$, the regularization coefficient $\lambda = 0.05$, and $h_\rho(\beta) = 1 - \exp(-\beta^2/2\rho^2)$, where $\rho$ is set to be a decreasing sequence. As demonstrated by Figure 3, $h'_\rho(\beta)$ may be absolutely too large when parameter $\rho$ is too little, which may cause instability during the training procedure. Thus, we set a lower bound for $\rho = 0.06$. Figures 4–6 exhibit the prediction results. Figure 4 shows the prediction values of the ELM algorithm for SinC function. Figure 5 shows the prediction values of the ELML1 algorithm. Figure 6 shows the prediction values of the ELMSL0 algorithm. It is obvious that the ELMSL0 algorithm has better prediction performance than other two algorithms.

The root mean square error (RMSE) is usually used as error function for evaluating the performance of the algorithm. The smaller value of the RMSE shows that the algorithm is more accurate to the described experimental data. By the following equation, we can calculate the RMSE:

$$\text{RMSE} = \sqrt{\frac{\sum_{i=1}^{n} [f(x_i) - y_i]^2}{n}}, \tag{16}$$

where $f(x_i)$ denotes the predicted data and $y_i$ denotes the actual data. We run 50 experiments on each of the three algorithms and then give the average training and testing RMSE values in Table 1. By comparing the number of hidden nodes of three algorithms, the ELMSL0 algorithm needs less hidden nodes but the testing sets have the highest accuracy.

Next, we consider using three algorithms to approximate a multidimensional Gabor function individually:

$$Z(x, y) = \frac{1}{2\pi(0.5)^2} \exp\left(-\frac{x^2 + y^2}{2(0.5)^2}\right) \cos(2\pi(x + y)). \tag{17}$$

We select 2601 samples from an equably spaced $51 \times 51$ grid on $-0.5 \leq x \leq 0.5$ and $-0.5 \leq y \leq 0.5$ for training samples, and 2601 testing samples are selected similarly. In order to prevent overfitting in the training process, the noise which is evenly distributed in $[-0.4, 0.4]$ has been added to training samples, and there is no noise in the testing samples. The original ELM algorithm has 100 hidden nodes, and RBF is selected for the activation function. Then, we use ELML1 and ELMSL0 algorithms to prune the network, respectively, choosing the learning rate $\eta = 0.2$, the regularization coefficient $\lambda = 0.05$, and $\rho = 0.06$ to approximate Gabor function (Figure 7). We perform 50 experiments on each of the three algorithms. As shown in Figures 8–10, it is so clear that the ELMSL0 algorithm has better prediction performance than conventional ELM and ELML1 algorithms. Table 2 gives the average training and testing RMSE values and the number of hidden nodes required by three algorithms. The accuracy of the ELMSL0 algorithm in testing sets is higher than other algorithms.

*5.2. Real-Word Classification Problems.* In this section, we compare the generalization performance of ELM, ELML1, and ELMSL0 algorithms in some real-word classification problems, which include seven binary classification problems and seven multiclass classification problems. Tables 4

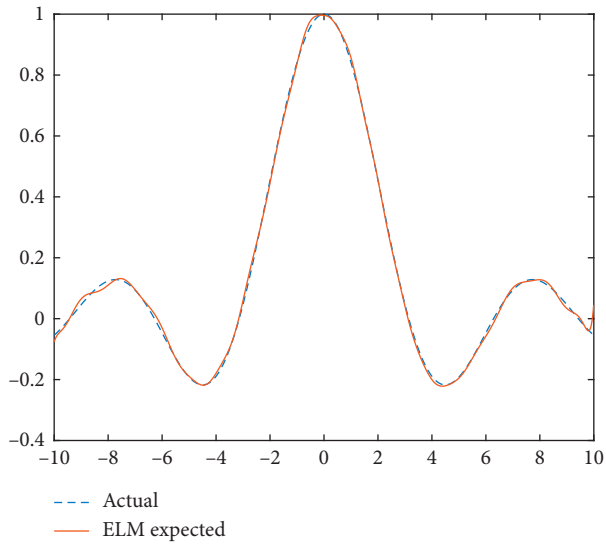


Figure 4: Prediction of the ELM algorithm for SinC function.

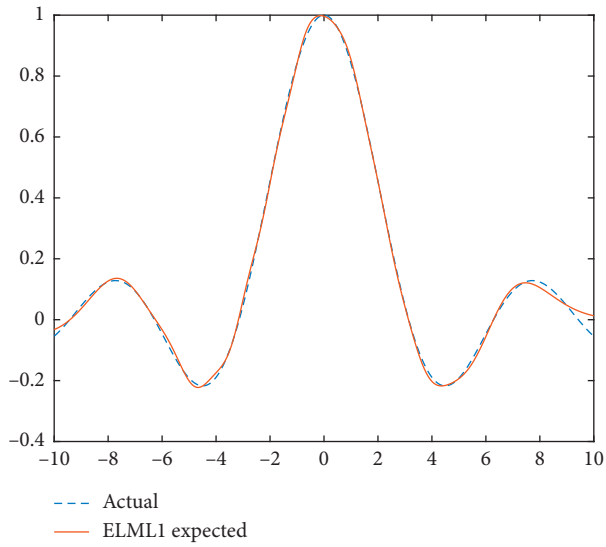


Figure 5: Prediction of the ELML1 algorithm for SinC function.

and 5 describe these classified data clearly, including the number of training and testing data and the attributes of each classification data set.

To explain Tables 4 and 5 clearly, we take the Diabetes and Iris data sets for example. Diabetes data belongs to either positive or negative class. The data from "Pima Indians Diabetes Database" are created by Applied Physics Laboratory. It consists of 768 women over the age of 21 who come from Phoenix, Arizona. Iris data include four features: calyx length, calyx width, petal length, and petal width. It contains three classes: Setosa, Versicolor, and Virginica.

There are different hidden nodes of each data, and the activation function is sigmoid function for three algorithms. We choose the learning rate $\eta = 0.02$, the regularization coefficient $\lambda = 0.03$, and $h_\rho(\beta) = 1 - \exp(-\beta^2/2\rho^2)$, where $\rho$ is a decreasing sequence with a lower bound of

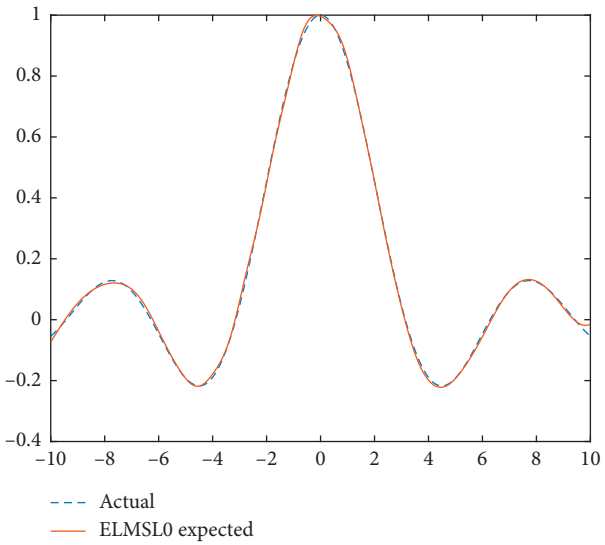


Figure 6: Prediction of the ELMSL0 algorithm for SinC function.

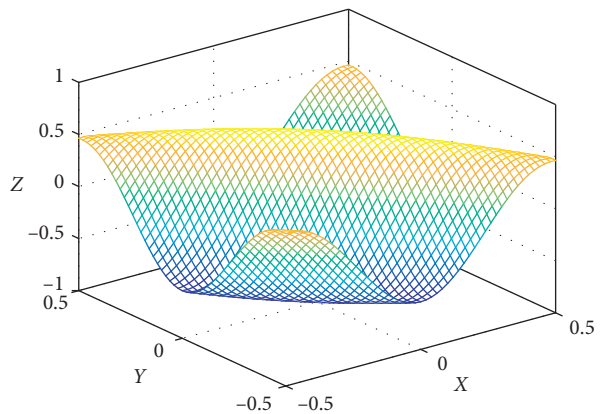


Figure 7: Gabor function.

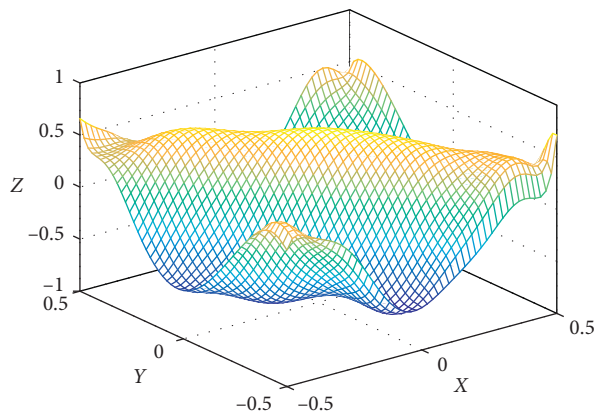


Figure 8: Prediction of the ELM algorithm for Gabor function.

0.08. We run 50 experiments with each data set, and Table 3 shows the average training and testing accuracy. It can be found that the ELMSL0 algorithm requires fewer hidden nodes without affecting the testing accuracy of the

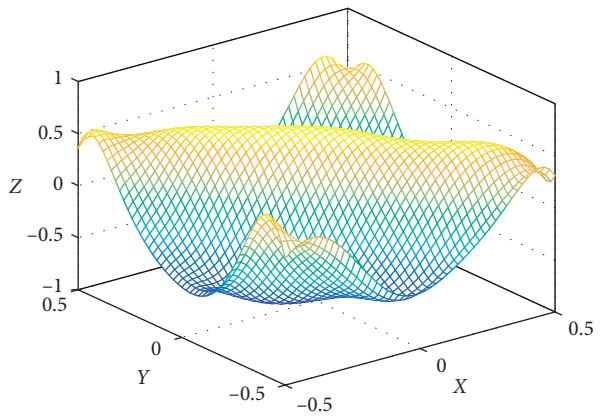


Figure 9: Prediction of the ELML1 algorithm for Gabor function.

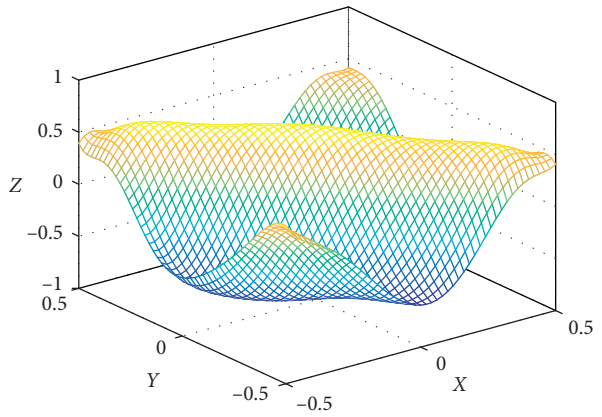


Figure 10: Prediction of the ELMSL0 algorithm for Gabor function.

Table 4: Information of the binary classification data sets.

| Data set | Training | Testing | Attributes |
|---|---|---|---|
| Diabetes | 530 | 238 | 8 |
| Australiancredit | 480 | 210 | 14 |
| Heart | 180 | 90 | 13 |
| Sonar | 140 | 68 | 60 |
| Wiscousin | 480 | 203 | 9 |
| SPECT | 80 | 187 | 22 |
| SPECTF | 80 | 187 | 44 |

Table 5: Information of the multiclass classification data sets.

| Data set | Training | Testing | Attributes | Classes |
|---|---|---|---|---|
| Iris | 100 | 50 | 4 | 3 |
| Wine | 120 | 58 | 13 | 3 |
| Glass | 150 | 64 | 9 | 6 |
| Olitos | 84 | 36 | 25 | 4 |
| *E.coli* | 236 | 100 | 8 | 8 |
| Seeds | 147 | 63 | 7 | 3 |
| Wholesale | 308 | 132 | 7 | 3 |

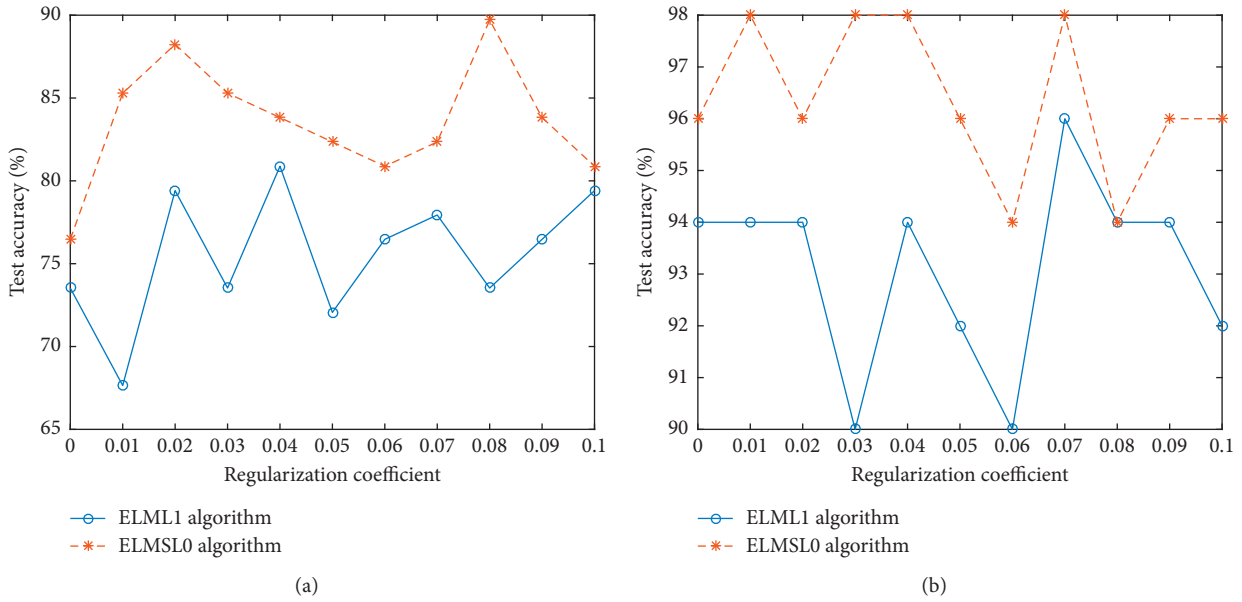


Figure 11: Testing classification performance of (a) Sonar and (b) Iris with different regularization coefficients.

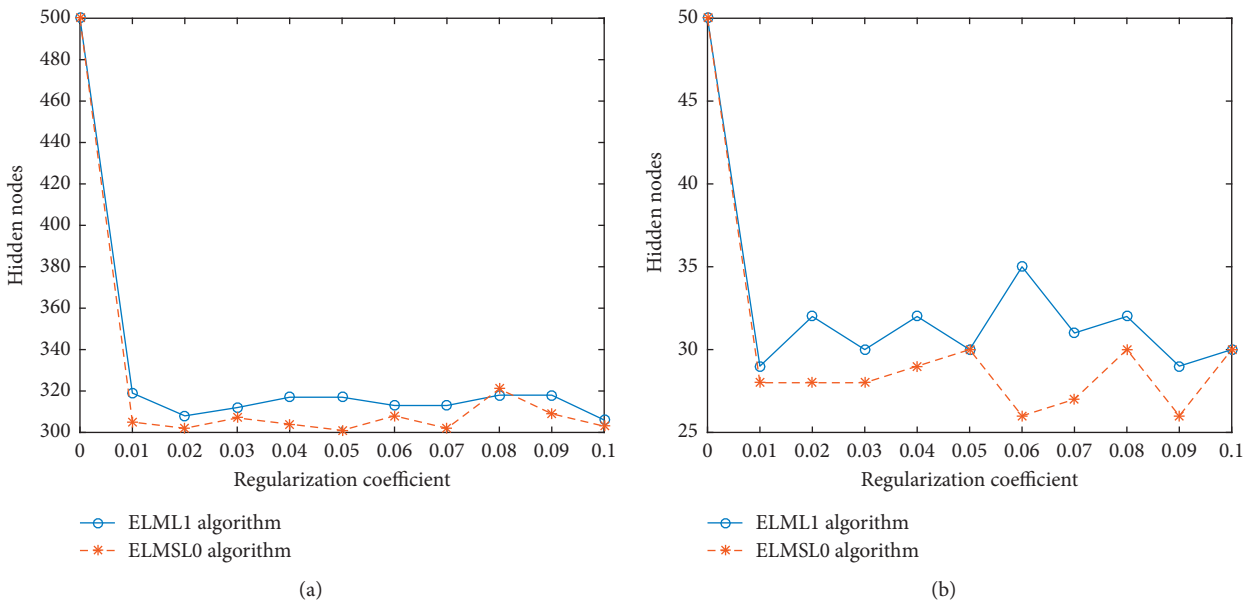


Figure 12: The number of hidden nodes required by ELML1 and ELMSL0 algorithms with different regularization coefficients.

algorithm. So, the ELMSL0 algorithm not only produces sparse results to prune the network but also has better generalization performance than other two algorithms in most classified data sets.

It is well known that regularization coefficient and the number of hidden nodes affect the accuracy of algorithms. Therefore, we need several experiments to select the appropriate regularization coefficients. A binary Sonar data set and a multiclassification Iris data set are selected here. We take the Sonar signal classification data set from the UCI machine learning repository for example. It is a typical benchmark problem in the field of neural networks. All samples are divided into two categories, one is sonar signals bounced off a metal cylinder, the other is those bounced off a roughly cylindrical rock. Here, we consider the number of hidden nodes is 500 in ELM, pruned by ELML1 and ELMSL0 algorithms. Figure 11 shows the testing accuracy of the two algorithms with different regularization coefficients. Figure 12 shows the number of hidden nodes required by the two algorithms as the regularization coefficient increases. From Figures 11 and 12, the ELMSL0 algorithm requires fewer hidden nodes but better generalization performance than the ELML1 algorithm with different regularization coefficients.

## 6. Conclusions

In this paper, we use a smoothing function to approximate $L_0$ regularization, proposing a pruning method with smoothing $L_0$ regularization (ELMSL0) for training and pruning extreme learning machine. And also it is shown that the ELMSL0 algorithm can produce sparse results to prune networks availably. Both regression and classification problems show that the ELMSL0 algorithm has better generalization performance and simpler network structure. In future, we consider applying the intelligent algorithm to the ELM algorithm to find the most appropriate weights and thresholds.

## Data Availability

No data were used to support this study.

## Conflicts of Interest

The authors declare that they have no conflicts of interest.

## Acknowledgments

This work was supported by Special Science Research Plan of the Education Bureau of Shaanxi Province of China (No. 18JK0344), Doctoral Scientific Research Foundation of Xi'an Polytechnic University (No. BS1432) and The 65th China Postdoctoral Science Foundation (No. 2019M652837).

## References

[1] S. K. Sharma and P. Chandra, "Constructive neural networks: a review," *International Journal of Engineering, Science and Technology*, vol. 2, pp. 7847–7855, 2010.

[2] B. Hassibi and D. G. Stork, "Second order derivatives for network pruning: optimal brain surgeon," in *Proceedings of the Advances in Neural Information Processing Systems*, pp. 164–171, Denver, CO, USA, 1993.

[3] M. Attik, L. Bougrain, and F. Alexandra, "Neural network topology optimization," in *Proceedings of the 5th International Conference ICANN'05, Lecture Notes in Computer Science*, vol. 3697, pp. 53–58, Warsaw, Poland, 2005.

[4] J. Wang, W. Wu, and J. M. Zurada, "Computational properties and convergence analysis of BPNN for cyclic and almost cyclic learning with penalty," *Neural Networks*, vol. 33, pp. 127–135, 2012.

[5] 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.

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

[7] G. B. Huang, L. Chen, and C. K. Siew, "Universal approximation using incremental constructive feedforward networks with random hidden nodes," *IEEE Transactions on Neural Networks*, vol. 17, no. 4, 2006.

[8] B. Jia, D. Li, Z. Pan, and G. Hu, "Two-dimensional extreme learning machine," *Mathematical Problems in Engineering*, vol. 2015, Article ID 491587, 8 pages, 2015.

[9] H.-J. Rong, Y.-S. Ong, A.-H. Tan, and Z. Zhu, "A fast pruned-extreme learning machine for classification problem," *Neurocomputing*, vol. 72, no. 1–3, pp. 359–366, 2008.

[10] Y. Miche and A. Sorjamaa, "OP-ELM: optimally pruned extreme learning machine," *IEEE Transactions on Neural Networks*, vol. 21, no. 1, pp. 158–162, 2010.

[11] G.-B. Huang and L. Chen, "Convex incremental extreme learning machine," *Neurocomputing*, vol. 70, no. 16–18, pp. 3056–3062, 2007.

[12] D. Yu and L. Deng, "Efficient and effective algorithms for training single-hidden-layer neural networks," *Pattern Recognition Letters*, vol. 33, no. 5, pp. 554–558, 2012.

[13] R. Parekh, J. Yang, and V. Honavar, "Constructive neural-network learning algorithms for pattern classification," *IEEE Transactions on Neural Networks*, vol. 11, no. 2, pp. 436–451, 2000.

[14] N. B. Karayiannis and G. W. Mi, "Growing radial basis neural networks: merging supervised and unsupervised learning with network growth techniques," *IEEE Transactions on Neural Networks*, vol. 8, no. 6, pp. 1492–1506, 1997.

[15] J. M. Zurada, A. Malinowski, and S. Usui, "Perturbation method for deleting redundant inputs of perceptron networks," *Neurocomputing*, vol. 14, no. 2, pp. 177–193, 1997.

[16] H. M. Shao, L. J. Liu, and G. F. Zheng, "Convergence of a gradient algorithm with penalty for training two-layer neural networks," in *Proceedings of the 2009 2nd IEEE International Conference on Computer Science and Information Technology*, pp. 76–79, Beijing, China, August 2009.

[17] G. E. Hinton, "Connectionist learning procedures," *Artificial Intelligence*, vol. 40, no. 1–3, pp. 185–234, 1989.

[18] H. Zhang, Y. Tang, and X. Liu, "Batch gradient training method with smoothing $$\boldsymbol{\ell}_{\bf 0}$$ ℓ 0 regularization for feedforward neural networks," *Neural Computing and Applications*, vol. 26, no. 2, pp. 383–390, 2015.

[19] W. Wu, Q. Fan, J. M. Zurada, J. Wang, D. Yang, and Y. Liu, "Batch gradient method with smoothing regularization for training of feedforward neural networks," *Neural Networks*, vol. 50, pp. 72–78, 2014.

[20] A. B. Nielsen and L. K. Hansen, "Structure learning by pruning in independent component analysis," *Neurocomputing*, vol. 71, no. 10–12, pp. 2281–2290, 2008.

[21] N. Fnaiech, S. Abid, F. Fnaiech, and M. Cheriet, "A modified version of a formal pruning algorithm based on local relative variance analysis," in *Proceedings of the First International Symposium on Control, Communications and Signal Processing*, pp. 849–852, Hammamet, Tunisia, March 2004.

[22] J.-f. Qiao, Y. Zhang, and H.-g. Han, "Fast unit pruning algorithm for feedforward neural network design," *Applied Mathematics and Computation*, vol. 205, no. 2, pp. 622–627, 2008.

[23] T. Pan, J. Zhao, W. Wu, and J. Yang, "Learning imbalanced datasets based on SMOTE and Gaussian distribution," *Information Sciences*, vol. 512, pp. 1214–1233, 2020.

[24] E. J. Candes and T. Tao, "Decoding by linear programming," *IEEE Transactions on Information Theory*, vol. 51, no. 12, pp. 4203–4215, 2005.

[25] Y. Wang, P. Liu, Z. Li, T. Sun, C. Yang, and Q. Zheng, "Data regularization using Gaussian beams decomposition and sparse norms," *Journal of Inverse and Ill-Posed Problems*, vol. 21, no. 1, pp. 1–23, 2013.

[26] J. Wang, B. Zhang, Z. Sang, Y. Liu, S. Wu, and Q. Miao, "Convergence of a modified gradient-based learning algorithm with penalty for single-hidden-layer feed-forward

networks," *Neural Computing and Applications*, vol. 32, no. 7, pp. 2445–2456, 2018.

[27] X. Xie, H. Zhang, J. Wang, Q. Chang, J. Wang, and N. R. Pal, "Learning optimized structure of neural networks by hidden node pruning with $L_{1}$ regularization," *IEEE Transactions on Cybernetics*, vol. 50, no. 3, pp. 1333–1346, 2020.

[28] Q. Fan, J. M. Zurada, and W. Wu, "Convergence of online gradient method for feedforward neural networks with smoothing L1/2 regularization penalty," *Neurocomputing*, vol. 131, pp. 208–216, 2014.

[29] Q. W. Fan, W. Wu, and J. M. Zurada, "Convergence of batch gradient learning with smoothing regularization and adaptive momentum for neural networks," *SpringerPlus*, vol. 5, no. 1, 2016.

[30] Y. Liu, J. Yang, L. Li, and W. Wu, "Negative effects of sufficiently small initialweights on back-propagation neural networks," *Journal of Zhejiang University Science C*, vol. 13, no. 8, pp. 585–592, 2012.