International Journal of Computational Intelligence Systems
Vol. **13(1)**, 2020, *pp.* 1101–1108
DOI: https://doi.org/10.2991/ijcis.d.200731.001; ISSN: 1875-6891; eISSN: 1875-6883
https://www.atlantis-press.com/journals/ijcis/

Research Article

Scalable Real-Time Attributes Responsive Extreme Learning Machine

Hongbo Wang*, Yuejuan Yao, Xi Liu, Xuyan Tu

School of Computer and Communication Engineering, Beijing Key Lab of Knowledge Engineering for Materials Science, University of Science and Technology Beijing, Xueyuan Road 30, Haidian Zone, Beijing, 100083, China

ARTICLE INFO

Article History

Received 20 Dec 2019
Accepted 27 Jul 2020

Keywords

Extreme learning machine
Attributes scalable
Cropping strategy

ABSTRACT

Extreme learning machine (ELM) has recently attracted many researchers' interest due to its very fast learning speed, and ease of implementation. Its many applications, such as regression, binary and multiclass classification, acquired better results. However, when some attributes of the dataset have been lost, this fixed network structure will be less than satisfactory. This article suggests a Scalable Real-Time Attributes Responsive Extreme Learning Machine (Star-ELM), which can grow its appropriate structure with nodes autonomous coevolution based on the different dataset. Its hidden nodes can be merged to more effectively adjust structure and weight. In the experiments of classical datasets we compare with other relevant variants of ELM, Star-ELM makes better performance on classification learning with loss of dataset attributes in some situations.

1. INTRODUCTION

Extreme learning machine (ELM) [1] is derived from the single-hidden-layer feed-forward neural network (SLFN) design, proposed by Huang *et al*. In this algorithm, the input weights and biases of the hidden layer are randomly generated and need not be adjusted. Unlike the other training algorithms, such as neural networks (NNs), support vector machine (SVM), the ELM's main advantages is low complexity, fast learning speed and better generalization. ELM is an effective solution for excellent learning accuracy and speed in many applications, such as language recognition [2], image classification [3], hardware acceleration [4], disease prediction [5] and classification of time series [6].

Recently, ELM has been extensively studied, and its structure has also been improved. In general, there are two common methods for constructing a SLFN. One is the pruning-oriented, and the other is based on the thinking of constructing. To the generalization ability of NN, pruned-extreme learning machine(P-ELM) [7] and optimally pruned-extreme learning machine(OP-ELM) [8] use prune means in the model and get a satisfactory result, while incremental extreme learning machine (I-ELM) [9], error minimized extreme learning machine (EM-ELM) [10] and constructive hidden nodes selection for ELM (CS-ELM) [11] explore incremental constructive feed-forward networks with random hidden nodes to minimize their error. These two improvements are based on the single-layer NN.

The ELM is still a shallow architecture, its focus on classification and prediction. However, feature learning and testing accuracy on big data are still problems in some practical applications. Inspired by the deep learning (DL) [12], multi-layer extreme learning machine (ML-ELM) [13], hierarchical extreme learning machine(H-ELM) [14] and stacked extreme learning machines(S-ELMs) [15] were developed to solve this issue. For example, S-ELMs divides a single large ELM network into multiple stacked small ELMs. It uses the principal component analysis (PCA) [16] dimensionality reduction method to check the number of hidden nodes in each small ELMs, retaining important nodes from the previous layer and merging with new randomly generated nodes as the input nodes of the next layer.

However, the aforementioned works still face some issues. In the existing immutable structure of the ELM, it will not be as good as solving some problems in reality. For example, in some situations, the dataset may be missing some attributes, or the number of samples in each category in the dataset may be unbalanced. If a fixed network structure is still used to solve such problems, its classification accuracy is difficult to achieve practicality.

In order to improve the effective learning in the condition of the lack of dataset attributes, this paper proposed a network, named Scalable Real-Time Attributes Responsive Extreme Learning Machine (Star-ELM). It can select the appropriate structure with nodes autonomous coevolution according to the different datasets. The nodes of the hidden layer can be merged to more effectively adjust structure and weight.

This paper is organized as follows: Section 2 describes the basics of ELM and its concept of principal components. In Sections 3, the details of proposed Star-ELM are presented. Section 4 checks the performance of Star-ELM on different data sets, and compares it

* *Corresponding author. Email: foreverwhb@126.com*

with ELM and S-ELMs when the attributes of the dataset are missing. Section 5 concludes this article and suggests a future work.

2. RELATED WORK

2.1. Extreme Learning Machine

ELM is a simple and effective algorithm for training. The SLFN [17] proposed by Huang *et al.* in 2004. ELM randomly generated weights and bias of hidden layer node, calculating the weights of the output layer. Huang proved that ELM has uniform approximation ability as SLFN. There are Q samples (x_i, t_i), where $x_i \in \mathbb{R}^n$ means input and $t_i \in \mathbb{R}^m$ means target, respectively.

The number of nodes in the input layer and output layer is determined by n and m, respectively. The number of its hidden layer node L is given by manmade. In an ELM, its input layer to the hidden layer is a random mapping, which maps the training set of samples from the original space to a feature space. The dimension of feature space is determined by the number L of hidden layer nodes. In general, the dimension of the feature space is higher than that of the original space. Compared with other SLFN training modes, the advantage of ELM does not need to adjust the weight parameters and has a very fast learning speed and a very good generalization ability.

The training process can be expressed as

$$t_i = \beta g(Wx_i + b). \tag{1}$$

It can be converted to $H\beta = T$, where

$$H = \begin{bmatrix} g(w_1x_1 + b_1) & \cdots & g(w_Lx_1 + b_L) \\ \vdots & \ddots & \vdots \\ g(w_1x_Q + b_1) & \cdots & g(w_Lx_Q + b_L) \end{bmatrix} \tag{2}$$

represents the hidden layer output matrix of ELM. Huang proves that if the activation function is infinite differentiable, the W and b do not need to be adjusted. We only need to solve the output weight matrix β, which can meet the target output with a minimum error approximation. The solution of β generally expressed as

$$\beta = H^{\dagger}T, \tag{3}$$

where $H^{\dagger}$ is the pseudo inverse (Moore–Penrose) of the hidden layer output matrix H. The structure of the ELM is illustrated in Figure 1.

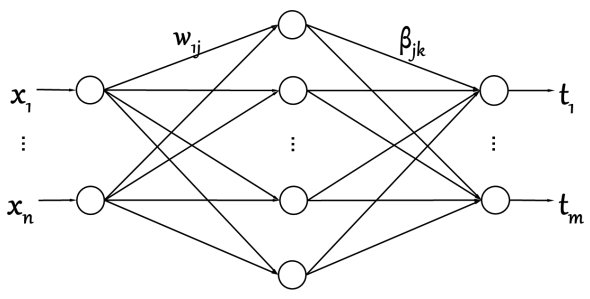

Figure 1 Extreme learning machine (ELM) structure.

2.2. S-ELMs

The S-ELMs divides a single large ELM network into multiple stacked small ELMs. S-ELMs can use small memory to approximate very large ELM network requirements. The main idea is to keep the size of the entire network fixed. They choose the top few significant nodes using their output weights' eigenvalues, and reduce the nodes by multiplying the corresponding eigenvectors. This can be done by performing PCA dimension reduction on the output weights calculated in each small ELM. In PCA process, the top L' eigenvectors generated based on the corresponding eigenvalues can be recorded as $\tilde{V} \in \mathbb{R}^{L \times L'}$. Compared to ELM, S-ELMs can achieve better results, with higher test accuracy, smaller network size, and faster speed, but there is still room for improvement in time complexity and robustness. Through the above description, the steps of the S-ELMs algorithm are summarized as follows:

Step 1: Randomly generate hidden layer parameters input weight w and threshold b, and then calculate hidden layer output matrix H as Equation (2). β is calculated by

$$\beta = \left(\frac{I}{\lambda} + H^TH\right)^{-1} H^TT, \tag{4}$$

where λ is the regularization coefficient.

Step 2: Use the PCA to reduce the dimension of β from L, the number of hidden layer, to L' by $\tilde{V} \in \mathbb{R}^{L \times L'}$.

Step 3: Repeat the following steps until the S layer:

i. randomly generate $(L - L')$ hidden layer nodes and calculate the correlation matrix H_{new};

ii. combine important nodes H'

$$H' = H\tilde{V}; \tag{5}$$

iii. hidden layer output matrix: $H = [H', H_{new}]$;

iv. repeat calculate β and step 2.

Step 4: The output of last layer of the S-ELMs network is $F_{out} = H\beta$.

3. PROPOSED STAR-ELM

3.1. Selection of Hidden Layer Parameters

One of the main features of the ELM is to randomly generate the values of hidden layer parameters, but the disadvantage is that the accuracy of the network model in the application process cannot be guaranteed, and it needs to be adjusted manually, which is a very cumbersome and time-consuming process.

For this reason, Star-ELM has improved in the parameter selection process of the hidden layer. The parameter selection does not need to be adjusted manually, which greatly improves the accuracy of the prediction model.

The main idea is to add a loop program into the traditional ELM. By customizing the number of loops n, the model randomly generates model parameters W and b, and the execution process is looped n times. The network will automatically pick out the set of network

parameters with the highest prediction accuracy from the n cycles. The weights and bias in hidden layer are the most suitable parameters for the network in n cycles. The main method of implementation is to first define four zero matrices $w[]$, $b[]$, $train[]$ and $test[]$ to store the matrix of the input weights, bias, training accuracy and test accuracy, respectively. Finally, W and b are obtained which makes the test accuracy the highest.

3.2. Node Sensitivity

This section describes the definition of node sensitivity in the Star-ELM, which is mainly referred to as the sensitivity to the weight value. In other words, we use quantization to calculate the effect of the output due to changes in the model parameters. That is to say, if the change of a parameter has a large impact on the output, the parameter is sensitive. On the contrary, it indicates that the parameter has little influence on the network structure. You can consider deleting it to make the network structure more streamlined. The calculation method of sensitivity is specifically described below.

There are Q different samples (x_i, t_i), where $x_i \in \mathbb{R}^n$ is input and $t_i \in \mathbb{R}^m$ is target. The number of hidden layer nodes is given by L, and the activation function is $g(x)$. Then it can be expressed as

$$f(x_i) = \sum_{j=1}^{L} \beta_j g(w_j x_i + b_j), \tag{6}$$

where $w_j = (w_{j1}, w_{j2}, \dots, w_{jn})^T$ represents the weight of the input layer to the j-th hidden layer node, b_j is the corresponding bias value, β_j is the output weight vector connected to the j-th hidden layer node and output node and $f(x_i) \in \mathbb{R}^m$ is the output corresponding to the i-th training sample.

We can define

$$k_{ji} = g_j(x) = g(w_j x_i + b_j), \tag{7}$$

where $i = 1, 2, \dots, Q$ and $j = 1, 2, \dots, L$. Based on Equation (7), we can get

$$f(x_i) = \sum_{j=1}^{L} k_{ji} \beta_j. \tag{8}$$

Without loss of generality, suppose we delete the hidden layer node $j = \lambda$, then we can get $f'(x_i)$, that is to say

$$\| f(x_i) - f'(x_i) \| = \| k_{\lambda i} \beta_\lambda \| = |k_{\lambda i}| \cdot \| \beta_\lambda \| . \tag{9}$$

A larger $\| f(x_i) - f'(x_i) \|$ indicates that the λ-th node has a greater influence on $f(x_i)$, that is, it's more important, otherwise this node is less important. Based on this, we define the sensitivity of the λ-th node in hidden layer as

$$S_\lambda = \frac{1}{Q} \sum_{i=1}^{Q} |k_{\lambda i}| \cdot \| \beta_\lambda \| , \tag{10}$$

where Q is the number of samples in the training set, $k_{\lambda i}$ is the output of the λ-th hidden node for the i-th sample and β_λ is the relevant output weight. S_λ represents the influence of λ-th hidden node on the samples in all training sets, so it is obvious that the larger S_λ indicates that the λ-th hidden node is more important to the network model. We can sort S based on the sensitivity of the hidden layer nodes as

$$S_1' \geqslant S_2' \geqslant S_3' \geqslant \cdots \geqslant S_L' \tag{11}$$

where S_1' represents the most sensitive node. After we sort the nodes in the order of their sensitivity, the hidden nodes will be selected and processed.

3.3. Hidden Layer Node Merge Operation

The calculation method of node sensitivity is described in detail in the previous section. In this section, the combination of nodes and layer generation between the layers is introduced for a Star-ELM. First, starting from a simple ELM network with L hidden nodes, the output matrix of the first hidden layer is recorded as H_1, and the corresponding output weight is recorded as β_1. Then, the sensitivity value S of each node is calculated by sensitivity definition as Equation (10).

The nodes in the hidden layer of the network structure are merged in pairs. We chose two adjacent nodes for comparison, because the algorithm is easier to understand and implement in this way. The merge process is based on the sensitivity of the nodes, and the nodes with high sensitivity will be retained, as

$$S_p = \max\{S_{2p-1}, S_{2p}\}, \tag{12}$$

where $p = 1, 2, \dots, \frac{L}{2}$. For example, we have hidden layer nodes $n_1, n_2 \cdots, n_8$, and we calculate and sort the sensitivities of them: $S_2 > S_5 > S_6 > S_3 > S_1 > S_4 > S_8 > S_7$. Then begin to merge nodes. When $p = 1$ we choose node n_2 for $S_2 > S_1$, $p = 2$ we choose node n_3 for $S_3 > S_4$, and so on. Finally we choose n_2, n_3, n_5, n_8 for the second hidden layer, but not the most sensitive four nodes. By this way we can weigh the impact of these attributes in a more balanced way. When some attributes of the sample are lost, or the samples are not balanced. The remaining nodes in the hidden layer, which can participate in the next layer of network operations, is half of the nodes in the initial hidden layer. The new hidden layer output matrix H_1' becomes

$$\begin{aligned} H_1' &= \begin{bmatrix} h(x_1) \\ \vdots \\ h(x_N) \end{bmatrix} \\ &= \begin{bmatrix} h_1(x_1) & \cdots & h_p(x_1) \\ \vdots & \ddots & \vdots \\ h_1(x_N) & \cdots & h_p(x_N) \end{bmatrix}, \end{aligned} \tag{13}$$

where N is the number of training set.

In order to ensure the consistency of the number of hidden layer nodes in the second layer and the first layer, the remaining $L - p$ hidden nodes in the second hidden layer will be randomly generated. The newly generated hidden nodes is

$$\begin{aligned} H_{2_{new}} &= \begin{bmatrix} h_{2_{new}}(x_1) \\ \vdots \\ h_{2_{new}}(x_N) \end{bmatrix} \\ &= \begin{bmatrix} h_1(x_1) & \cdots & h_{L-p}(x_1) \\ \vdots & \ddots & \vdots \\ h_1(x_N) & \cdots & h_{L-p}(x_N) \end{bmatrix}. \end{aligned} \tag{14}$$

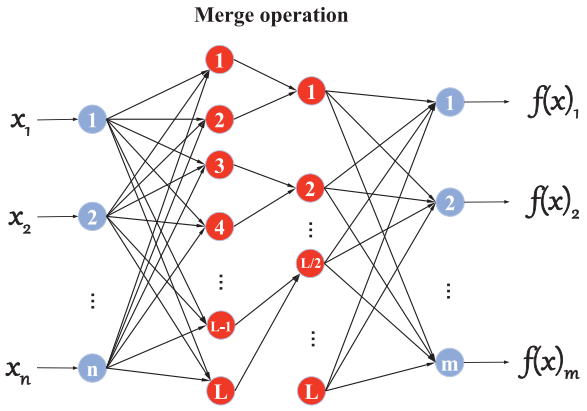

Figure 2 | Scalable Real-Time Attributes Responsive Extreme Learning Machine (Star-ELM) structure.

The second hidden layer in Star-ELM formed after the combination is

$$H_2 = [H_1', H_{2_{new}}], \tag{15}$$

and the combination process of nodes in hidden layer is illustrated in Figure 2. It is worth noting that the index of x and $f(x)$ in the figure refers to the number of their dimensions, which is different from the number of the samples used above.

After the nodes in the first hidden layer have been subjected to the calculation of the sensitivity level, the adjacent nodes are merged in pairs, and relatively important nodes were retained during the merge process, thereby discarding a part of the redundant nodes.

After obtaining nodes that can represent important information of the layer, some new nodes are randomly generated, and the two types of nodes will merge into nodes of the second hidden layer. Through the integration of important nodes in the first layer, the network structure can better adapt to the changes in different datasets. Especially when some attributes in the dataset are lost or face unbalanced datasets, the network structure has better adaptability than traditional network structures.

It is worth noting that, unlike the method of globally selecting nodes based on importance, such as S-ELMs based on PCA operation, the method proposed in this article is a local selection based on importance. When some attributes of the sample are lost, or the samples are not balanced, this method can weigh the impact of these attributes in a more balanced way, rather than discarding all unimportant nodes.

3.4. Star-ELM

According to the above improvements on the selection of hidden layer parameters, node sensitivity definition and description of hidden layer node merging, this section gives detailed steps of Star-ELM.

Given a training dataset with Q different samples (x_i, t_i), where $x_i \in \mathbb{R}^n$ is the input and $t_i \in \mathbb{R}^m$ is the target. We use the activation function $g(x)$ so that Star-ELM can approximate any continuous function [18]. Number of nodes in hidden layer is L, regularization coefficient is λ, the hidden layer output matrix is H and the output layer weight matrix is β.

1. The initial weights w_j and bias b_j are randomly generated, and then an initial ELM operation is performed to calculate the prediction accuracy. Repeat prediction n times according to the set number of loops. According to the prediction accuracy γ, select best the input weight w_j and the bias node b_j;
2. The parameters w_j and b_j with the highest prediction precision selected in the previous step are placed in

$$k_{ji} = g(w_j x_i + b_j); \tag{16}$$

3. Calculate all k_{ji} and combine them into a hidden layer output matrix H;
4. Solving the weight β_1 between the hidden layer and the output layer by H as

$$\beta_1 = \left(\frac{I}{\lambda} + H^T H\right)^{-1} H^T T; \tag{17}$$

5. Calculating the sensitivity value S of each node in the first hidden layer node by

$$S_\lambda = \frac{1}{Q}\sum_{i=1}^{Q} |k_{\lambda i}| \cdot \| \beta_\lambda \|; \tag{18}$$

6. Obtain the nodes' sensitivities and combine the nodes in pairs as

$$S_p = \max\{S_{2p-1}, S_{2p}\}, \tag{19}$$

where $p = 1, 2, \ldots, \frac{L}{2}$;

7. After the nodes are merged, a new output matrix H_1' is obtained, and new remaining nodes matrix $H_{2_{new}}$ is randomly generated. Merge two matrices as

$$H_2 = [H_1', H_{2new}]; \tag{20}$$

8. Solving the weight β_2 based on Equation (17), and then calculate the output value of the NN by the second layer output H_2 and β_2 as

$$F_{out} = H_2 \beta_2. \tag{21}$$

It can be seen from the above description of the principle and steps of the Star-ELM algorithm that the hidden layer nodes in the improved algorithm can be adjusted, and the parameters involved in the operation in the first hidden layer are most suitable for the network structure. In order to prove that the algorithm has better performance than the traditional ELM, in the next section, the classic dataset is selected to verify the validity of the algorithm classification.

The algorithm for Star-ELM is summarized in Table 1.

Table 1 Star-ELM algorithm.

Algorithm 1: Pseudo code for Star-ELM	
Input:	$Q, g(x), L, x$
Output:	F_{out}
1:	Initial weights w_i and bias b_i;
2:	Multiple runs to select best w_i and b_i;
3:	Calculate k_{ji} $k_{ji} = g(w_j x_i + b_j)$;
4:	Combine hidden layer output matrix H;
5:	Solving output weight matrix $\beta_1 = \left(\frac{I}{\lambda} + H^T H\right)^{-1} H^T T$;
6:	Calculate node sensitivity $S_\lambda = \frac{1}{Q}\sum_{i=1}^{Q} \lvert k_{\lambda i}\rvert \parallel \beta_\lambda \parallel$;
7:	Merge sensitivity coefficients in pairs $S_p = \max\{S_{2p-1}, S_{2p}\}$;
8:	Obtain a new output matrix $H_1{}'$;
9:	Randomly generate new nodes H_{2new};
10:	Merge into $H_2 = [H_1{}', H_{2new}]$;
11:	Calculate output value $F_{out} = H_2\beta_2$
Return	F_{out}

Star-ELM, Scalable Real-Time Attributes Responsive Extreme Learning Machine.

Table 2 Performance of different algorithms.

Dataset	Algorithm	Train Accuracy (%)	Test Accuracy (%)	Time (s)
MNIST	ELM	85.23	80.12	3.2313
	S-ELMs	86.71	87.77	4.7390
	Star-ELM	80.07	81.29	11.324
NORB	ELM	80.39	78.13	8.1256
	S-ELMs	80.01	79.86	3.0075
	Star-ELM	86.90	**84.05**	10.321
USPS	ELM	89.08	73.14	2.3906
	S-ELMs	94.02	89.88	5.2851
	Star-ELM	95.20	**92.99**	12.234

ELM, extreme learning machine; Star-ELM, scalable real-time attributes responsive extreme learning machine; S-ELM, stacked extreme learning machine.

4. EXPERIMENTS

4.1. Datasets

This section will check the performance of proposed Star-ELM and ELM, S-ELMs on three datasets: Modified National Institute of Standards and Technology database(MNIST), NYU Object Recognition Benchmark(NORB) and US Postal(USPS) data.

The MNIST [19] database of handwritten digits is a popular database for many researchers to do different machine learning algorithms. It consists of 60000 training samples and 10000 testing samples, each sample have 784 features. Many methods have been tested on this dataset including many other classification algorithms.

The NORB [20] dataset is for experiments in 3-D object recognition from shape. It contains 23000 training samples and 23000 testing samples with 2048 features.

The USPS [21] dataset refers to numeric data obtained from the scanning of handwritten digits from envelopes by the U.S. Postal Service. It contains 7291 training instances and 2007 testing instances with 256 features.

4.2. Experimental Environment

The simulations are carried out in the MATLAB 2016a environment running on an Intel(R) i5–3210M computer with 2.50G Hz and 4GB RAM.

4.3. Mode of Evaluation

To evaluate the performance of a classification algorithm, it is usually judged from two aspects: the time spent in the work and the accuracy of the classification accuracy. In dealing with classification problems, the most important issue of concern is the accuracy of classification. The classification ability of the model is obtained by calculating and recording the test accuracy. In this experiment, the performance of the algorithm was evaluated by the test accuracy and training time of the classification results.

In this experiment, the traditional ELM method uses 500 hidden layer nodes, S-ELMs uses two hidden layers, each hidden layer has 200 nodes. Proposed Star-ELM takes $n = 50$ and two hidden layers, each has 200 nodes. All the above methods use Sigmoid activation functions. Each algorithm is trained 50 times, aaverage the results and two decimal places are retained.

4.4. Results

The training, test accuracy(%) and training time (s) of different algorithms on each dataset are shown in Table 2. The results of the classification of the three datasets using ELM, S-ELMs and Star-ELM are listed in the table, respectively.

As shown in the Table 2, from the perspective of test accuracy, the Star-ELM algorithm has significantly higher test accuracy on the NORB and USPS datasets than the S-ELMs and ELM algorithms. Based on the MNIST dataset, the experimental results of Star-ELM have no advantage over S-ELMs, but significantly exceed the traditional ELM.

From the perspective of training time, the traditional ELM algorithm has the shortest training time and the Star-ELM algorithm has the longest time, mainly because Star-ELM cycles for 50 times to find the best weight w and bias b. However, the training time difference of the algorithms is at the second level, which is acceptable. And Star-ELM's test accuracy on the NORB and USPS datasets is significantly better than the other two algorithms. It can be seen that Star-ELM can automatically adjust to obtain higher accuracy during the training process without manual intervention.

4.5. Scalable Attributes Experiment

Under the complete conditions of the dataset, the advantages of Star-ELM are not well reflected, and now proceed to the next experiment. The main advantage of the Star-ELM algorithm is that the network structure can be reconstructed according to the size of

the dataset, and it has better adaptability to different datasets. In this experiment, by continuously reducing the attribute values of the dataset, observe the changes in the test accuracy of different algorithms.

The first is the MNIST dataset. As shown in Table 3, the initial dataset contains 784 features, and its attribute value is gradually decremented from 784 to 12. The trend of the test accuracy(%) under different algorithms is observed by the change of the number of eigenvalues, as shown in Figure 3.

When the number of features in the MNIST dataset is reduced from 784 to 49, the test accuracy of the ELM is reduced from the highest precision of 80.12% to 53.72%. The accuracy of S-ELMs was reduced from 87.77% to 72.60%, while the accuracy of Star-ELM was only reduced from 81.29% to 77.00%. The decline rates of the three algorithms on the MNIST dataset are 32.95%, 17.28% and 5.28%. It can be seen that when the feature attribute is lost on the MNIST dataset, the Star-ELM can make a better adjustment of the network structure to slow down the rate of test accuracy.

Next, similar experiments were performed on the USPS dataset and the NORB dataset. As shown in Table 4 and Figure 4, when the number of features in the USPS dataset is reduced from 256 to 8, the test accuracy degradation rates of the three algorithms ELM, S-ELMs and Star-ELM are 18.56%, 51.97%, 17.75%, respectively. Among them, Star-ELM still has the lowest rate of decline on the USPS dataset, and the test accuracy is the highest when the dataset is complete. Besides, Star-ELM has also improved the classification accuracy of the dataset.

Finally, the experiment was carried out on the NORB dataset, and the number of attribute features was gradually reduced from 2048 to 32. The performance of three different algorithms on the NORB dataset is shown in Table 5 and Figure 5.

When the number of features in the NORB data set is reduced from 2048 to 32, the test accuracy degradation rates of the three algorithms ELM, S-ELMs and Star-ELM are 70.98%, 65.69%, 15.03%. The rate of decline of ELM on the NORB dataset is still the lowest,

Table 3 Scalable attributes on MNIST dataset test.

Attributes	ELM-Acc (%)	S-ELMs-Acc (%)	Star-ELM-Acc (%)
784	80.12	87.77	81.29
392	80.00	80.14	80.00
196	72.10	80.09	79.02
98	63.04	80.05	78.11
49	53.72	72.60	77.00
25	43.42	60.24	74.74
12	40.10	54.05	71.33

ELM, extreme learning machine; Star-ELM, scalable real-time attributes responsive extreme learning machine; S-ELM, stacked extreme learning machine.

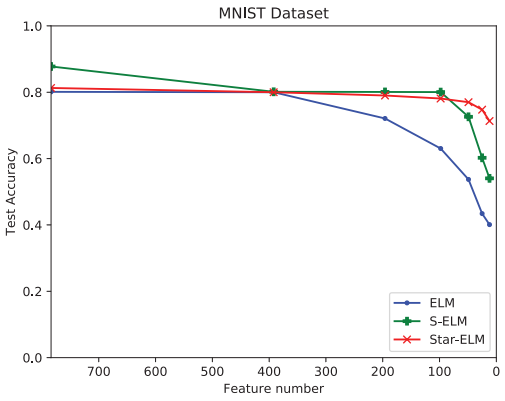

Figure 3 Accuracy changes for MNIST dataset test.

Table 4 Scalable attributes on USPS dataset test.

Attributes	ELM-Acc (%)	S-ELMs-Acc (%)	Star-ELM-Acc (%)
256	73.94	89.98	89.99
128	72.10	80.09	79.83
64	66.75	80.05	78.42
32	63.33	55.77	77.23
8	60.22	43.21	74.01

ELM, extreme learning machine; Star-ELM, scalable real-time attributes responsive extreme learning machine; S-ELM, stacked extreme learning machine.

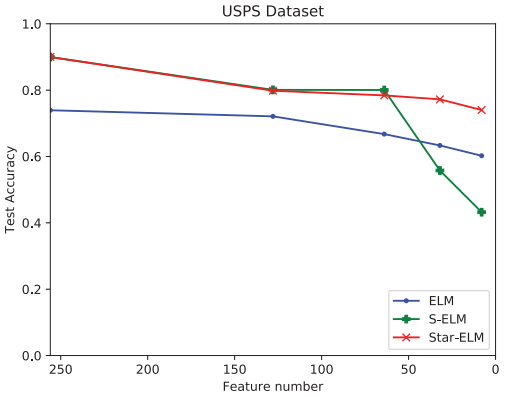

Figure 4 Accuracy changes for USPS dataset test.

Table 5 Scalable attributes on NORB dataset test.

Attributes	ELM	S-ELMs	Star-ELM
2048	73.14	89.98	85.89
1024	70.12	80.14	80.00
512	60.21	77.21	79.00
256	59.22	67.22	76.88
64	30.22	49.21	77.23
32	21.22	30.87	72.98

ELM, extreme learning machine; Star-ELM, scalable real-time attributes responsive extreme learning machine; S-ELM, stacked extreme learning machine.

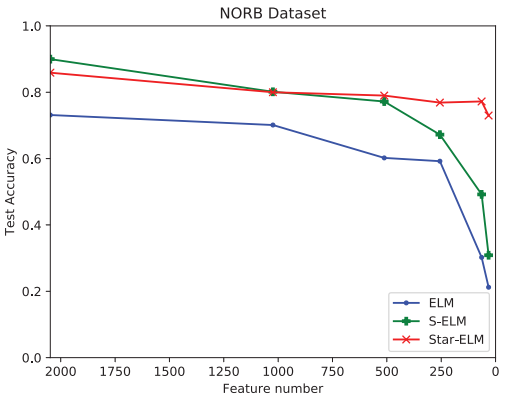

Figure 5 Accuracy changes for NORB dataset test.

and the accuracy of the Star-ELM algorithm is higher than that of the traditional ELM algorithm when the dataset is complete.

The main reason to get better results than ELM is that the network structure of the ELM is fixed, and it cannot be reasonably adjusted as the dataset changes. For S-ELMs, unlike the method of globally selecting nodes based on importance, used by S-ELMs, the method proposed in this article is a local selection based on importance. When some attributes of the dataset are lost, or the samples are not balanced, this method can weigh the impact of these attributes in a more balanced way, rather than discarding all unimportant nodes. Therefore, Star-ELM works better when some attributes of the dataset are missing.

5. CONCLUSIONS

This paper proposes an improved ELM algorithm Star-ELM. In order to verify that the Star-ELM can still be effectively classified during the change of the dataset. The test accuracy of different algorithms will decrease as the feature attributes are lost, but the rate of decrease is different. With the reduction of the number of features in the MNIST, USPS and NORB datasets, Star-ELM's test accuracy declines the slowest. The future work will focus on incremental network, by transferring the learning ability of Star-ELM and process the new incoming big data more effectively.

ACKNOWLEDGMENTS

This work is supported by National Natural Science Foundation of China (NSFC) under Grant NO.61572074.

CONFLICT OF INTEREST

We declare that we have no financial and personal relationships with other people or organizations that can inappropriately influence our work.

AUTHOR CONTRIBUTIONS

Hongbo Wang, Yuejuan Yao, Xi Liu contributed to the conception of the study; Yuejuan Yao, Xi Liu performed the experiment; Hongbo Wang, Yuejuan Yao contributed significantly to analysis and manuscript preparation; Hongbo Wang, Xi Liu performed the data analyses and wrote the manuscript; Xi Liu, Xuyan Tu helped perform the analysis with constructive discussions.

REFERENCES

[1] G.B. Huang, H. Zhou, X. Ding, R. Zhang, Extreme learning machine for regression and multiclass classification, IEEE Trans. Syst. Man Cybern. Part B Cybern. 42 (2011), 513–529.
[2] X.J. Ming, Z.W. Qiang, Y.D. Zhou, L. Jia, X.S. Hong, Manifold regularized extreme learning machine for language recognition, Acta Automat. Sinica. 41 (2015), 1680–1685.
[3] U. Ergul, G. Bilgin, Classification of hyperspectral images with multiple kernel extreme learning machine, in 2018 26th Signal Processing and Communications Applications Conference (SIU), Izmir, Turkey, 2018, pp. 1–4.
[4] A. Safaei, Q.M.J. Wu, T. Akilan, Y. Yang, System-on-a-Chip (SoC)-based hardware acceleration for an Online Sequential Extreme Learning Machine (OS-ELM), IEEE Trans. Comput. Aided Design Integr. Circuits Syst. 38 (2019), 2127–2138.
[5] H.A. Wibawa, I. Malik, N. Bahtiar, Evaluation of kernel-based extreme learning machine performance for prediction of chronic kidney disease, in 2018 2nd International Conference on Informatics and Computational Sciences (ICICoS), Semarang, Indonesia, 2018, pp. 1–4.
[6] K. Yan, Z. Ji, H. Lu, J. Huang, W. Shen, Y. Xue, Fast and accurate classification of time series data using extended ELM: application in fault diagnosis of air handling units, IEEE Trans. Syst. Man Cybern. Syst. 49 (2019), 1349–1356.
[7] H.J. Rong, Y.S. Ong, A.H. Tan, Z. Zhu, A fast pruned-extreme learning machine for classification problem, Neurocomputing. 72 (2008), 359–366.
[8] Y. Miche, A. Sorjamaa, P. Bas, O. Simula, C. Jutten, A. Lendasse, OP-ELM: optimally pruned extreme learning machine, IEEE Trans. Neural Netw. 21 (2009), 158–162.
[9] G.B. Huang, C.K. Chen, L. Siew, Universal approximation using incremental constructive feedforward networks with random hidden nodes, IEEE Trans. Neural Netw. 17 (2006), 879–892.
[10] G. Feng, G.B. Huang, Q. Lin, R. Gay, Error minimized extreme learning machine with growth of hidden nodes and incremental learning, IEEE Trans. Neural Netw. 20 (2009), 1352–1357.
[11] Y. Lan, Y.C. Soh, G.B. Huang, Constructive hidden nodes selection of extreme learning machine for regression, Neurocomputing. 73 (2010), 3191–3199.
[12] G.E. Hinton, S. Osindero, Y.W. Teh, A fast learning algorithm for deep belief nets, Neural Comput. 18 (2006), 1527–1554.
[13] J. Tang, C. Deng, G.B. Huang, Extreme learning machine for multilayer perceptron, IEEE Trans. Neural Netw. Learn. Syst. 27 (2015), 809–821.
[14] K. Phurattanaprapin, P. Horata, Extended hierarchical extreme learning machine with multilayer perceptron, in 2016 13th International Joint Conference on Computer Science and Software Engineering (JCSSE), Khon Kaen, Thailand, 2016, pp. 1–5.
[15] H. Zhou, G.B. Huang, Z. Lin, H. Wang, Y.C. Soh, Stacked extreme learning machines, IEEE Trans. Cybern. 45 (2014), 2013–2025.
[16] T. Jolliffe, Principal Component Analysis, Principal Component Analysis, Springer, Berlin, Heidelberg, Germany, 2011, pp. 1094–1096.
[17] G.B. Huang, Q.Y. Zhu, C.K. Siew, Extreme learning machine: a new learning scheme of feedforward neural networks, in 2004 IEEE international joint conference on neural networks, Budapest, Hungary (IEEE Cat. No. 04CH37541) Neural Netw. Vol. 2 (2004), pp. 985–990. IEEE
[18] C. Huang, L. Chen, Convex incremental extreme learning machine, Neurocomputing. 70 (2007), 3056–3062.
[19] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, Proc. IEEE. 86 (1998), 2278–2324.
[20] Y. LeCun, F.J. Huang, L. Bottou, Learning methods for generic object recognition with invariance to pose and lighting, in Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, Washington, DC, USA, Vol. 2 (2004), pp. 11–104.
[21] J.J. Hull, A database for handwritten text recognition research, IEEE Trans. Pattern Anal. Mach. Intell. 16 (1994), 550–554.