Research Article

An Incremental Optimal Weight Learning Machine of Single-Layer Neural Networks

Hai-Feng Ke,¹ Cheng-Bo Lu ,² Xiao-Bo Li,² Gao-Yan Zhang,¹
Ying Mei ,² and Xue-Wen Shen³

¹*School of Computer & Computing Science, Zhejiang University City College, Hangzhou 310015, China*

²*College of Engineering, Lishui University, Lishui 323000, China*

³*School of Electronics and Information, Zhejiang University of Media and Communications, Hangzhou 310015, China*

Correspondence should be addressed to Cheng-Bo Lu; lu.chengbo@aliyun.com

Received 12 October 2017; Revised 1 January 2018; Accepted 11 January 2018; Published 1 March 2018

Academic Editor: Wenbing Zhao

Copyright © 2018 Hai-Feng Ke et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

An optimal weight learning machine with growth of hidden nodes and incremental learning (OWLM-GHNIL) is given by adding random hidden nodes to single hidden layer feedforward networks (SLFNs) one by one or group by group. During the growth of the networks, input weights and output weights are updated incrementally, which can implement conventional optimal weight learning machine (OWLM) efficiently. The simulation results and statistical tests also demonstrate that the OWLM-GHNIL has better generalization performance than other incremental type algorithms.

1. Introduction

Feedforward neural networks (FNNs) have been extensively used in classification applications and regressions [1]. As a specific type of FNNs, single hidden layer feedforward networks with additive models can approximate any target continuous function [2]. Owing to excellent learning capabilities and fault tolerant abilities, SLFNs play an important role in practical applications and have been investigated extensively in both theory and application aspects [3–7].

Maybe the most popular training method for SLFNs classifiers in recent years was gradient-based back-propagation (BP) algorithms [8–12]. BP algorithms can be easily recursively implemented in real time; however, the slow convergence is the bottleneck of BP, where the fast training is essential. In [13], a kind of novel learning machine named extreme learning machine (ELM) is proposed for training SLFNs, where the learning parameters of hidden nodes, including input weights and biases, are randomly assigned and need not be tuned, while output weights can be obtained by simple generalized inverse computation. It has been proven that, even without updating the parameters of the hidden layer, SLFNs with randomly generated hidden neurons and tunable output

weights maintain their universal approximation and excellent generalization performance [14]. It has been shown that ELM is faster than most state-of-the-art training algorithms for SLFNs and it has been applied widely to many practical cases such as classification, regression, clustering, recognition, and relevance ranking problems [15, 16].

Since input weights and hidden layer biases of SLFNs trained with ELM are randomly assigned, the minor changes of data in input vectors maybe cause large changes of data in hidden layer output matrix of the SLFNs. This in turn will lead to large changes of data in the output weight matrix. According to statistical learning theory [17–21], the large changes of data in the output weight matrix will greatly increase both structural and empirical risks of the SLFNs, which will in turn decrease robustness property of the SLFNs regarding the input disturbances. In fact, it has been noted from simulations that the SLFNs trained with the ELM sometimes perform poor generalization performance and robustness with regard to the input disturbances. In view of this situation, OWLM [22] was proposed; it is seen that both input weights and output weights of the SLFNs are globally optimized with the batch learning type of least squares. All feature vectors of classifier can then be placed at the

prescribed positions in feature space in the sense that the separability of those nonlinearly separable patterns can be maximized, and better generalization performance can be achieved compared with conventional ELM.

However, there is still one major issue existing in OWLM, which is OWLM needing more computational cost than ELM, since the input weights are not randomly selected in SLFNs trained with OWLM. With the advent of the big data age, data sets become larger and more complex [23, 24], which reduces the learning efficiency of OWLM further. For implementing OWLM efficiently, this paper proposed an incremental learning machine referred to as optimal weight learning machine with growth of hidden nodes and incremental learning (OWLM-GHNIL). Whenever new nodes are added, the input weights and output weights could be incrementally updated which can implement the conventional OWLM algorithm efficiently. At the same time, owing to the advantages of OWLM, OWLM-GHNIL has better generalization performance than other incremental algorithms such as EM-ELM [25] (an approach that could automatically determine the number of hidden nodes in generalized single hidden layer feedforward networks) and I-ELM [14], which added random hidden nodes to SLFNs only one hidden node each time.

The rest of this paper is organized as follows: in Section 2, the OWLM is briefly described. In Section 3, we present OWLM-GHNIL in detail and analyze its computational complexity. Simulation results are then presented in Section 4, showing that our proposed approach performs more efficiently and has better generalization performance than some existing methods. In Section 5, we give conclusion.

2. Brief of the Optimal Weight Learning Machine

In this section, we briefly describe the OWLM.

For N given input pattern vectors $x_1(k), x_2(k), \dots, x_N(k)$, as well as N corresponding desired output data vectors $o_1(k), o_2(k), \dots, o_N(k)$, respectively, N linear output equations of SLFNs in Figure 1 can be obtained as

$$H\beta = O, \quad (1)$$

where

$$H = [x_1(k) \ x_2(k) \ \cdots \ x_N(k)]^T W$$

$$= \begin{bmatrix} x_1^T w_1 & x_1^T w_2 & \cdots & x_1^T w_{\bar{N}} \\ x_2^T w_1 & x_2^T w_2 & \cdots & x_2^T w_{\bar{N}} \\ \vdots & \vdots & \ddots & \vdots \\ x_N^T w_1 & x_N^T w_2 & \cdots & x_N^T w_{\bar{N}} \end{bmatrix} = XW, \quad (2)$$

with

$$X = [x_1(k), x_2(k), \dots, x_N(k)]^T, \quad (3)$$

$$x_i(k) = [x_{i1}(k), x_{i2}(k), \dots, x_{in}(k)]$$

FIGURE 1: A single hidden layer neural network with linear nodes and an input tapped delay line.

and input weight matrix

$$W = [w_1, w_2, \dots, w_{\bar{N}}], \quad (4)$$

$$w_i = [w_{i1}, w_{i2}, \dots, w_{in}]^T$$

and output weight matrix

$$\beta = [\beta_1, \beta_2, \dots, \beta_m], \quad (5)$$

$$\beta_i = [\beta_{i1}, \beta_{i2}, \dots, \beta_{i\bar{N}}].$$

Let $y_1, y_2, \dots, y_N$ be N feature vectors, corresponding to the input data vectors $x_1, x_2, \dots, x_N$. Then we have

$$[y_1 \ y_2 \ y_3 \ \cdots \ y_N] = W^T [x_1 \ x_2 \ x_3 \ \cdots \ x_N], \quad (6)$$

or

$$Y = W^T X. \quad (7)$$

Let the N reference feature vectors be described by

$$Y_d = [y_{d1} \ y_{d2} \ y_{d3} \ \cdots \ y_{dN}]. \quad (8)$$

Generally, as described in [22], the selection of the N desired feature vectors in (8) mainly depends on the characteristics of the input vectors. By optimizing the input weights of the SLFNs in Figure 1, the OWLM can place the feature vectors of the SLFNs at the “desired position” in feature space. The purpose of the assignment is to further maximize the separability of the vectors in the feature space so that the generalization performance and robustness, seen from the output layer of the SLFNs, can be greatly improved, compared with the SLFNs trained with ELM.

The design of the optimal input weight of the SLFNs can be formulated by the following optimization problem:

$$\begin{aligned} & \text{Minimize} \quad \left\{ \frac{1}{2} \|\varepsilon_f\|^2 + \frac{\lambda_1}{2} \|W\|^2 \right\} \\ & \text{subject to} \quad \varepsilon_f = Y_d - Y = Y_d - W^T X, \end{aligned} \quad (9)$$

where λ_1 is a positive real regularization parameter.

The optimal input weight matrix was derived as follows:

$$W = (\lambda_1 I + XX^T)^{-1} XY_d^T. \quad (10)$$

Similarly, to minimize the error between the desired output pattern T and the actual output pattern O , the design of the optimal output weight of the SLFNs can be formulated by the following optimization problem:

$$\begin{aligned} & \text{Minimize} \quad \left\{ \frac{1}{2} \|\varepsilon\|^2 + \frac{\lambda_2}{2} \|\beta\|^2 \right\} \\ & \text{subject to} \quad \varepsilon = O - T = H\beta - T, \end{aligned} \quad (11)$$

where λ_2 is a positive real regularization parameter.

The optimal output weight matrix was derived as follows:

$$\beta = (\lambda_2 I + H^T H)^{-1} H^T T. \quad (12)$$

The optimal weight learning machine [22] can be summarized as follows.

Algorithm OWLM. Given a training set $\{(x_i, t_i)\}_{i=1}^N$, as well as hidden node number $\tilde{N}$, do the following steps.

Step 1. Randomly assign hidden node parameters (w_i, b_i) , $i = 1, \dots, \tilde{N}$.

Step 2. Calculate the hidden layer output matrix H by (2).

Step 3. Calculate the input weight matrix W_{new} by (10).

Step 4. Recalculate the hidden layer output matrix H_{new} by W_{new} .

Step 5. Calculate the output weight matrix β by (12).

Obviously, the OWLM needs more training time compared with ELM, since it needs additional computational cost for computing the input weight matrix.

3. Growing Hidden Nodes and Incrementally Updating Weights

Given SLFNs with initial hidden nodes m and a training set $\{(x_i, t_i)\}_{i=1}^N$ let N be the number of input patterns, let l be the length of the input patterns, and let h be the length of the output patterns.

We have

$$W_0 = (XX^T + \lambda_1 I)^{-1} XY_{d0}^T, \quad (13)$$

$$H_0 = X^T W_0 \quad (14)$$

$$\beta_0 = (H_0^T H_0 + \lambda_2 I)^{-1} H_0^T T, \quad (15)$$

where Y_{d0} is an $m \times N$ matrix consisting of N desired feature vectors.

Let $E(H_i) = \min \|H_i \beta_i - T\|$ be the network output error; if $E(H_0)$ is less than the target error $\varepsilon > 0$, then no new hidden nodes need to be added and the learning procedure completes. Otherwise, we could add n new nodes to the existing SLFNs; then

$$\begin{aligned} W_1 &= (XX^T + \lambda_1 I)^{-1} X \begin{bmatrix} Y_{d0} \\ Y_{d1} \end{bmatrix}^T \\ &= [W_0 \quad (XX^T + \lambda_1 I)^{-1} XY_{d1}^T] = [W_0 \quad \Delta W_0], \end{aligned} \quad (16)$$

where Y_{d1} is an $n \times N$ matrix consisting of N desired feature vectors. Then,

$$\begin{aligned} H_1 &= X^T [W_0 \quad \Delta W_0] = [H_0 \quad X^T \Delta W_0] = [H_0, \Delta H_0], \\ \beta_1 &= ([H_0, \Delta H_0]^T [H_0, \Delta H_0] + \lambda_2 I)^{-1} [H_0, \Delta H_0]^T T \\ &= \left(\begin{pmatrix} H_0^T H_0 & H_0^T \Delta H_0 \\ \Delta H_0^T H_0 & \Delta H_0^T \Delta H_0 \end{pmatrix} + \lambda_2 I \right)^{-1} \\ &\quad \cdot [H_0, \Delta H_0]^T T \\ &= \begin{pmatrix} H_0^T H_0 + \lambda_2 I & H_0^T \Delta H_0 \\ \Delta H_0^T H_0 & \Delta H_0^T \Delta H_0 + \lambda_2 I \end{pmatrix}^{-1} \\ &\quad \cdot \begin{pmatrix} H_0^T T \\ \Delta H_0^T T \end{pmatrix}. \end{aligned} \quad (17)$$

The Schur complement $P(P = (\Delta H_0^T \Delta H_0 + \lambda_2 I) - \Delta H_0^T H_0 (H_0^T H_0 + \lambda_2 I)^{-1} H_0^T \Delta H_0)$ of $H_0^T H_0 + \lambda_2 I$ is invertible by choosing the suitable λ_2 . Then, using the result on the inversion of 2×2 block matrices [26], we can get

$$\begin{aligned} & \begin{pmatrix} H_0^T H_0 + \lambda_2 I & H_0^T \Delta H_0 \\ \Delta H_0^T H_0 & \Delta H_0^T \Delta H_0 + \lambda_2 I \end{pmatrix}^{-1} \\ &= \begin{pmatrix} (H_0^T H_0 + \lambda_2 I)^{-1} + (H_0^T H_0 + \lambda_2 I)^{-1} (H_0^T \Delta H_0) P^{-1} (\Delta H_0^T H_0) (H_0^T H_0 + \lambda_2 I)^{-1} & -(H_0^T H_0 + \lambda_2 I)^{-1} (H_0^T \Delta H_0) P^{-1} \\ -P^{-1} (\Delta H_0^T H_0) (H_0^T H_0 + \lambda_2 I)^{-1} & P^{-1} \end{pmatrix}; \end{aligned} \quad (18)$$

then

$$\begin{aligned}
\beta_1 &= \begin{pmatrix} H_0^T H_0 + \lambda_2 I & H_0^T \Delta H_0 \\ \Delta H_0^T H_0 & \Delta H_0^T \Delta H_0 + \lambda_2 I \end{pmatrix}^{-1} \begin{pmatrix} H_0^T \\ \Delta H_0^T \end{pmatrix}^T \\
&= \begin{pmatrix} (H_0^T H_0 + \lambda_2 I)^{-1} H_0^T T + (H_0^T H_0 + \lambda_2 I)^{-1} (H_0^T \Delta H_0) P^{-1} (\Delta H_0^T H_0) (H_0^T H_0 + \lambda_2 I)^{-1} H_0^T T - (H_0^T H_0 + \lambda_2 I)^{-1} (H_0^T \Delta H_0) P^{-1} \Delta H_0^T T \\ -P^{-1} (\Delta H_0^T H_0) (H_0^T H_0 + \lambda_2 I)^{-1} H_0^T T + P^{-1} \Delta H_0^T T \end{pmatrix} \quad (19) \\
&= \begin{pmatrix} \beta_0 + (H_0^T H_0 + \lambda_2 I)^{-1} (H_0^T \Delta H_0) P^{-1} (\Delta H_0^T H_0) \beta_0 - (H_0^T H_0 + \lambda_2 I)^{-1} (H_0^T \Delta H_0) P^{-1} \Delta H_0^T T \\ -P^{-1} (\Delta H_0^T H_0) \beta_0 + P^{-1} \Delta H_0^T T \end{pmatrix} = \begin{pmatrix} \beta_0 + Q_0 - R_0 \\ U_0 + V_0 \end{pmatrix}.
\end{aligned}$$

To save computational cost, Q_0, R_0, U_0 , and V_0 in (19) should be computed as the following sequence:

$$\begin{aligned}
P^{-1} (\Delta H_0^T H_0) &\longrightarrow P^{-1} (\Delta H_0^T H_0) \beta_0 (= U_0), \\
(H_0^T H_0 + \lambda_2 I)^{-1} (H_0^T \Delta H_0) &\longrightarrow (H_0^T H_0 + \lambda_2 I)^{-1} \\
&\quad \cdot (H_0^T \Delta H_0) U_0 (= Q_0), \\
P^{-1} \Delta H_0^T &\longrightarrow P^{-1} \Delta H_0^T T (= V_0) \\
&\longrightarrow (H_0^T H_0 + \lambda_2 I)^{-1} (H_0^T \Delta H_0) \\
&\quad \cdot P^{-1} \Delta H_0^T T (= R_0). \quad (20)
\end{aligned}$$

Given the number of initial hidden nodes m_0 , the maximum number of hidden nodes $m_{\max}$, and the expected output error ε , the OWLM-GHNIL for the SLFNs with the mechanism of growing hidden nodes can be summarized as the following two steps.

Algorithm OWLM-GHNIL

Step 1 (initialization step).

- (1) Compute W_0, H_0, β_0 by (13)–(15).
- (2) Compute the corresponding output error $E(H_0) = \min \|H_0 \beta_0 - T\|$.

Step 2 (recursively incremental step). Let $k = 0$, and while $m_k < m_{\max}$ and $E(H_k) > \varepsilon$,

- (1) $k = k + 1$;
- (2) randomly add n (n need not be kept constant) hidden nodes to the existing network; then $W_{k+1}, H_{k+1}, \beta_{k+1}$ can be calculated by (16)–(19).

End

Different from conventional OWLM which needs recalculating the input weight matrix and output weight matrix, whenever the network architecture is changed, the OWLM-GHNIL only needs updating the input weight matrix and output weight matrix incrementally each time; that is why it can reduce the computational complexity significantly.

Moreover, the convergence of the OWLM-GHNIL can be guaranteed by the Convergence Theorem in [25].

Now, we begin to analyze computational complexity of the updated work.

The computational complexity, which we consider, expresses the total number of required scalar multiplications. Some matrix computations need not be done repeatedly including the inversion of matrix in (13), since they have been obtained in the process of computing W_k, H_k, β_k . Then it requires lNn, lNn , and $2m_k n^2 + m_k nN + 3m_k nh + 2m_k^2 n + 2n^2 N + nNh + n^3$ multiplications for W_{k+1}, H_{k+1} , and β_{k+1} , respectively. Thus, the total computational complexity for the weights W_{k+1} and β_{k+1} is

$$\begin{aligned}
C_{\text{OWLM-GHNIL}} &= 2lNn + 2m_k n^2 + m_k nN + 3m_k nh \\
&\quad + 2m_k^2 n + 2n^2 N + nNh + n^3. \quad (21)
\end{aligned}$$

If we compute W_{k+1} and β_{k+1} by (10) and (12) directly, it will cost $C_{\text{OWLM}} = l^3 + 2l^2 N + 2lN(m_k + n) + 2(m_k + n)^2 N + (m_k + n)^3 + (m_k + n)Nh$ multiplications.

Since in most applications m_k and n can be much smaller than the number of training samples N : $m_k, n \ll N$ and h and l are often small number in practical applications, then, with the growth of m_k, n ,

$$\frac{C_{\text{OWLM}}}{C_{\text{OWLM-GHNIL}}} \approx \frac{2(m_k + n)^2}{m_k n + 2n^2}; \quad (22)$$

when $n = (1/2)m_k$,

$$\frac{C_{\text{OWLM}}}{C_{\text{OWLM-GHNIL}}} \approx 4.5; \quad (23)$$

when $n = (1/4)m_k$,

$$\frac{C_{\text{OWLM}}}{C_{\text{OWLM-GHNIL}}} \approx 8.3. \quad (24)$$

It can be seen that the OWLM-GHNIL is much more efficient than the conventional OWLM in such cases.

Similarly, we can get the computational complexity of the EM-ELM and ELM, respectively.

$$\begin{aligned}
C_{\text{EM-ELM}} &= lNn + 2m_k n^2 + m_k nN + 3m_k nh + 2m_k^2 n \\
&\quad + 2n^2 N + nNh + n^3,
\end{aligned}$$

$$C_{\text{ELM}} = lN(m_k + n) + 2(m_k + n)^2 N + (m_k + n)^3 + (m_k + n)Nh. \quad (25)$$

Then, we have

$$C_{\text{OWLM-GHNIL}} - C_{\text{EM-ELM}} = lNn, \quad (26)$$

$$C_{\text{OWLM}} - C_{\text{ELM}} = l^3 + 2l^2N + lN(m_k + n).$$

Obviously, the difference on computational complexity between OWLM-GHNIL and EM-ELM is much less than the difference between OWLM and ELM.

4. Simulation Results

In our experiments, all the algorithms are run in such computer environment: (1) operating system: Windows 7 Enterprise; (2) 3.8 GHZ CPU, Intel i5-3570; (3) memory: 8 GB; (4) simulating software: Matlab R2013a.

The performance of the OWLM-GHNIL has been compared with other growing algorithms including the EM-ELM, I-ELM, and the conventional OWLM.

In order to investigate the performance of the proposed OWLM-GHNIL, some benchmark problems are presented in this section.

The OWLM-GHNIL, EM-ELM, and OWLM have first been run to approximate the artificial ‘‘SinC’’ function which is a popular choice to illustrate neural network.

$$y(x) = \begin{cases} \frac{\sin x}{x}, & x \neq 0 \\ 1, & x = 0. \end{cases} \quad (27)$$

A training set and testing set with 5000 samples, respectively, are generated from the interval $(-10, 10)$ with random noise distributed in $[-0.2, 0.2]$, while testing data remain noise-free. The performances of each algorithm are shown in Figures 2 and 3. In this case, initial SLFNs are given five hidden nodes and then one new hidden node will be added each step until 30 hidden nodes arrive.

It can be seen from Figure 2 that the OWLM and the OWLM-GHNIL obtain similar lower testing root mean square error (RMSE) than the EM-ELM in most cases. Figure 3 shows the training time comparison of the three methods in SinC case. We can see that, with the growth of hidden nodes, the OWLM-GHNIL spent similar training time with the EM-ELM but much less than the OWLM in the case of the same number of nodes.

In the following, nine real benchmark problems including five regression applications and four classification applications are used for further comparison; all of them are available on the Web. For each case, the training data set and testing data set are randomly generated from its whole data set before each trial of simulation, and average results are obtained over 30 trials for all cases. The features of the benchmark data sets are summarized in Table 1.

The generalization performance comparison between the OWLM-GHNIL and two other popular incremental

FIGURE 2: Average testing RMSE of different algorithms in SinC case.

FIGURE 3: Computational complexity comparison of different algorithms in Sinc case.

ELM-type algorithms, EM-ELM and I-ELM, on regression and classification cases is given in Tables 2 and 3. In the implementation of the EM-ELM and OWLM-GHNIL, initial SLFNs are given 50 hidden nodes and then 25 new hidden nodes will be added each step until 150 hidden nodes arrive. In the case of the I-ELM, the initial SLFNs are given 1 hidden node and then the hidden nodes are added one by one until 150 hidden nodes. As observed from test results of average RMSE and accuracy in Tables 2 and 3, it looks that the OWLM-GHNIL obtained better generalization performance than the EM-ELM and I-ELM. In order to obtain an objective statistical measure, we apply a Student's t -test to each data to check if the differences between the OWLM-GHNIL and

TABLE 1: Details of the data sets used for regression and classification.

Data sets	Classes	Attributes	Types	# of pieces of training data	# of pieces of testing data
Delta Ailerons	—	6	Regression	3000	4129
Delta Elevators	—	6	Regression	4517	5000
California Housing	—	8	Regression	8000	12,460
Computer activity	—	8	Regression	4000	4192
Bank domains	—	8	Regression	4500	3692
COLL20	20	1024	Classification	1080	360
G50C	2	50	Classification	414	136
USPST(B)	2	256	Classification	1509	498
Satimage	6	36	Classification	3217	3218

TABLE 2: Comparison of average testing RMSE/accuracy and Student's t -test for the data sets between OWLM-GHNIL and EM-ELM.

Data sets	OWLM-GHNIL		EM-ELM		t -Test
	RMSE/accuracy	Std.	RMSE/accuracy	Std.	p value
Delta Ailerons	0.0583	0.0058	0.1023	0.0081	0.003627
Delta Elevators	0.0896	0.0063	0.1423	0.0154	0.025436
California Housing	0.1841	0.0032	0.2321	0.0045	0.000048
Computer activity	0.0467	0.0021	0.0392	0.0018	0.628794
Bank domains	0.0212	0.0043	0.0611	0.0032	0.007694
COLL20	91.25%	0.0332	86.33%	0.0537	0.000000
G50C	85.31%	0.0553	87.23%	0.0463	0.065632
USPST(B)	90.45%	0.0158	85.21%	0.0547	0.000000
Satimage	88.15%	0.0368	85.37%	0.0446	0.007625

TABLE 3: Comparison of average testing RMSE/accuracy and Student's t -test for the data sets between OWLM-GHNIL and I-ELM.

Data sets	OWLM-GHNIL		I-ELM		t -Test
	RMSE/accuracy	Std.	RMSE/accuracy	Std.	p -value
Delta Ailerons	0.0583	0.0058	0.1364	0.0223	0.000493
Delta Elevators	0.0896	0.0063	0.2265	0.0196	0.003743
California Housing	0.1841	0.0032	0.4365	0.0223	0.000000
Computer activity	0.0467	0.0021	0.0733	0.0072	0.065475
Bank domains	0.0212	0.0043	0.1128	0.0341	0.000000
COLL20	91.25%	0.0332	78.04%	0.0663	0.000000
G50C	85.31%	0.0553	76.45%	0.0772	0.000482
USPST(B)	90.45%	0.0158	84.33%	0.0472	0.000000
Satimage	88.15%	0.0368	80.33%	0.0682	0.001356

the other two algorithms are statistically significant (p value = 0.05, i.e., confidence of 95%). It was shown in Table 2 that, in four of the regression data sets (Delta Ailerons, Delta Elevators, California Housing, and Bank domains) and three of the classification data sets (COLL20, USPST(B), and Satimage), the t -test gave a significant difference between OWLM-GHNIL and EM-ELM with superior generalization performance of the OWLM-GHNIL, whereas no significant difference was found in the two other data sets (computer activity and G50C). In Table 3, the t -test results show that there was a significant difference between OWLM-GHNIL and I-ELM with superior generalization performance of the OWLM-GHNIL in all data sets except computer activity data set.

5. Conclusion

In this paper, we have developed an efficient method, OWLM-GHNIL; it can grow hidden nodes one by one or group by group in SLFNs. The analysis of computational complexity and simulation results on an artificial problem shows that OWLM-GHNIL can significantly reduce the computational complexity of OWLM. The simulation results on nine real benchmark problems including five regression applications and four classification applications also show that OWLM-GHNIL has better generalization performance than the two other incremental algorithms EM-ELM and I-ELM. t -test gave a significant difference with superior generalization performance of the OWLM-GHNIL further.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This research was supported by Zhejiang Provincial Natural Science Foundation of China under Grant no. LY18F030003, Foundation of High-Level Talents in Lishui City under Grant no. 2017RC01, Scientific Research Foundation of Zhejiang Provincial Education Department under Grant no. Y201432787, and the National Natural Science Foundation of China under Grant no. 61373057.

References

- [1] C. M. Bishop, *Neural Networks for Pattern Recognition*, Oxford University Press, New York, NY, USA, 1995.
- [2] G.-B. Huang and H. A. Babri, "Upper bounds on the number of hidden neurons in feedforward networks with arbitrary bounded nonlinear activation functions," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 9, no. 1, pp. 224–229, 1998.
- [3] X.-F. Hu, Z. Zhao, S. Wang, F.-L. Wang, D.-K. He, and S.-K. Wu, "Multi-stage extreme learning machine for fault diagnosis on hydraulic tube tester," *Neural Computing and Applications*, vol. 17, no. 4, pp. 399–403, 2008.
- [4] T.-Y. Kwok and D.-Y. Yeung, "Objective functions for training new hidden units in constructive neural networks," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 8, no. 5, pp. 1131–1148, 1997.
- [5] E. J. Teoh, K. C. Tan, and C. Xiang, "Estimating the number of hidden neurons in a feedforward network using the singular value decomposition," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 17, no. 6, pp. 1623–1629, 2006.
- [6] X. Luo, J. Deng, W. Wang, J.-H. Wang, and W. Zhao, "A quantized kernel learning algorithm using a minimum kernel risk-sensitive loss criterion and bilateral gradient technique," *Entropy*, vol. 19, no. 7, article no. 365, 2017.
- [7] Y. Xu, X. Luo, W. Wang, and W. Zhao, "Efficient DV-HOP localization for wireless cyber-physical social sensing system: A correntropy-based neural network learning scheme," *Sensors*, vol. 17, no. 1, article no. 135, 2017.
- [8] S. Haykin, *Neural networks and learning machines*, Pearson, Prentice-Hall, New Jersey, USA, 3rd edition, 2009.
- [9] S. Kumar, *Neural Networks*, McGraw-Hill Companies Inc., Columbus, OH, USA, 2006.
- [10] X. Yao, "Evolving artificial neural networks," *Proceedings of the IEEE*, vol. 87, no. 9, pp. 1423–1447, 1999.
- [11] G. P. Zhang, "Neural networks for classification: a survey," *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, vol. 30, no. 4, pp. 451–462, 2000.
- [12] J. Hertz, A. Krogh, and R. G. Palmer, *Introduction to the Theory of Neural Computation*, Addison-Wesley Publishing Company, Boston, Mass, USA, 1991.
- [13] 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.
- [14] G. Huang, L. Chen, and C. Siew, "Universal approximation using incremental constructive feedforward networks with random hidden nodes," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 17, no. 4, pp. 879–892, 2006.
- [15] S.-F. Ding, X.-Z. Xu, and R. Nie, "Extreme learning machine and its applications," *Neural Computing and Applications*, vol. 25, no. 3, pp. 549–556, 2014.
- [16] X. Luo, Y. Xu, W. Wang et al., "Towards enhancing stacked extreme learning machine with sparse autoencoder by correntropy," *Journal of The Franklin Institute*, 2017.
- [17] M. Anthony and P. L. Bartlett, *Neural Network Learning: Theoretical Foundations*, Cambridge University Press, Cambridge, UK, 1999.
- [18] V. N. Vapnik, *Statistical Learning Theory*, Adaptive and Learning Systems for Signal Processing, Communications, and Control, Wiley- Interscience, New York, NY, USA, 1998.
- [19] L. Devroye, L. Györfi, and G. Lugosi, *A Probabilistic Theory of Pattern Recognition*, vol. 31 of *Stochastic Modelling and Applied Probability*, Springer-Verlag New York, Berlin, Germany, 1996.
- [20] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification*, Wiley-Interscience, New York, NY, USA, 2nd edition, 2001.
- [21] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone, *Classification and Regression Trees*, Wadsworth, Belmont, Mass, USA, 1984.
- [22] Z. Man, K. Lee, D. Wang, Z. Cao, and S. Khoo, "An optimal weight learning machine for handwritten digit image recognition," *Signal Processing*, vol. 93, no. 6, pp. 1624–1638, 2013.
- [23] X. Luo, J. Deng, J. Liu, W. Wang, X. Ban, and J. Wang, "A quantized kernel least mean square scheme with entropy-guided learning for intelligent data analysis," *China Communications*, vol. 14, no. 7, pp. 127–136, 2017.
- [24] W. Zhao, R. Lun, C. Gordon et al., "A human-centered activity tracking system: toward a healthier workplace," *IEEE Transactions on Human-Machine Systems*, vol. 47, no. 3, pp. 343–355, 2017.
- [25] G. Feng, G.-B. Huang, Q. Lin, and R. Gay, "Error minimized extreme learning machine with growth of hidden nodes and incremental learning," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 20, no. 8, pp. 1352–1357, 2009.
- [26] G. H. Golub and C. F. van Loan, *Matrix Computations*, The Johns Hopkins University Press, Baltimore, Md, USA, 3rd edition, 1996.