Neural Comput & Applic
DOI 10.1007/s00521-014-1549-5

Applying a new localized generalization error model to design neural networks trained with extreme learning machine

Qiang Liu · Jianping Yin · Victor C. M. Leung · Jun-Hai Zhai · Zhiping Cai · Jiarun Lin

Received: 10 September 2013 / Accepted: 15 January 2014

Abstract High accuracy and low overhead are two key features of a well-designed classifier for different classification scenarios. In this paper, we propose an improved classifier using a single-hidden layer feedforward neural network (SLFN) trained with extreme learning machine. The novel classifier first utilizes principal component analysis to reduce the feature dimension and then selects the optimal architecture of the SLFN based on a new localized generalization error model in the principal component space. Experimental and statistical results on the NSL-KDD data set demonstrate that the proposed classifier can achieve a significant performance improvement compared with previous classifiers.

Keywords Extreme learning machine · Principal component analysis · Architecture selection algorithm · Localized generalization error model

1 Introduction

Classifier design [5, 6, 10] has recently attracted tremendous interest since various applications require highly efficient classifiers for processing huge volumes of data in diverse fields, such as intrusion detection systems (IDS) [3, 12], speaker recognition [7], bioinformatics [2, 8]. For example, the IDS, which is known as a promising technique to counter external intrusions or internal attacks on computer security attributes, i.e., confidentiality, integrity and availability, uses some classifier as the core module of anomaly detection. However, most of the existing classifiers are designed to maximize the classification accuracy while paying less attention to their computation overheads. It is challenging to design highly accurate classifiers with low computation overheads. We propose an improved classifier to meet this challenge, using a single-hidden layer feedforward neural networks (SLFNs) trained with an extreme learning machine (ELM) [6]. The notable merits of SLFNs trained with ELM stem from two aspects: (1) The input weights and hidden layer biases of SLFNs are randomly assigned rather than tuned iteratively; (2) The output weights of SLFNs can be analytically determined through finding the minimum norm least-squares solution thereafter. However, the performance of ELM decreases as the size of training data sets or the size of an input feature vector increases.

This work was supported by the National Natural Science Foundation of China (Grant No. 61170287, No. 60970034, No. 61070198, and No. 61379145).

Q. Liu (✉) · Z. Cai · J. Lin
School of Computer, National University of Defense Technology, Changsha, Hunan, China
e-mail: libra6032009@gmail.com

Z. Cai
e-mail: zpcai@nudt.edu.cn

J. Lin
e-mail: jrlin28@nudt.edu.cn

J. Yin
State Key Laboratory of High Performance Computing, National University of Defense Technology, Changsha, Hunan, China
e-mail: jpyin@nudt.edu.cn

V. C. M. Leung
Department of Electrical and Computer Engineering, University of British Columbia, Vancouver, BC, Canada
e-mail: vleung@ece.ubc.ca

J.-H. Zhai
Key Laboratory of Machine Learning and Computational Intelligence College of Mathematics and Computer Science, Hebei University, Baoding, Hebei, China
e-mail: mczjh@hbu.edu.cn

Published online: 01 February 2014

Specifically, the principal component analysis (PCA) is adopted to extract the significant features from training samples in order to improve the efficiency of the proposed classifier. As mentioned above, the sizes of training data sets and input vectors are limited for the sake of maintaining the performance of ELM. Therefore, PCA is effective for data sets preprocessing with a low computation overhead, resulting in an improved overall performance of classifiers. Since the objective of SLFNs is to approximate all training samples with zero mean square error (MSE), a new localized generalization error model in the principal component space (LGEM-PCS) is proposed. Previous LGEMs [14, 15] assume that all input perturbations have the same distributions, which is not rational in the PCS, since the more significant a feature is, the more sensitive it will be to original input perturbations in the new feature space due to the properties of PCA. We further present an architecture selection algorithm based on LGEM-PCS to select the optimal architecture of the SLFN. Finally, the proposed classifier is applied to an IDS in order to validate its advantages. Our contributions in this work include as follows: (1) We adopt PCA to reduce the feature dimension, which accelerates classification without loss of classification accuracy; (2) We introduce LGEM-PCS and then propose a new architecture selection algorithm for SLFNs trained with ELM. The proposed algorithm works with LGEM-PCS to determine the optimal number of hidden nodes of the SLFN. Experimental and statistical results on the NSL-KDD data set [13] show that the proposed classifier outperforms previous ones in terms of the classification accuracy and the classification time.

2 Background

2.1 Extreme learning machine

Basically, generalization performance and learning speed are of great concern in designing a learning algorithm, which motivates the proposal of ELM. For an infinitely differentiable activation function SLFN trained with ELM, it has been proven that the classifier is able to achieve good generalization performance at an extremely fast learning speed even if the input weights and hidden biases are randomly chosen [6]. After that, the universal approximation capability of ELM has been further investigated. The theoretical results from [4] have demonstrated that SLFNs trained with ELM are universal approximators for any continuous activation functions on any compact input sets. Therefore, rigorous results of the performance of ELM on both the universal approximation capability and the classification capability solidly support artificial and real large applications of SLFNs.

Given a training data set with N samples, $D = \{(\mathbf{x}_i, \mathbf{t}_i)|\mathbf{x}_i \in \mathbf{R}^n, \mathbf{t}_i \in \mathbf{R}^m, i = 1, 2, \ldots, N\}$, where $\mathbf{x}_i = [x_{i1}, x_{i2}, \ldots, x_{in}]^{\mathrm{T}}$ is an input vector and $\mathbf{t}_i = [t_{i1}, t_{i2}, \ldots, t_{im}]^{\mathrm{T}}$ is an output vector, a SLFN with M hidden nodes and an activation function $g(x)$ is modeled as

$$f(\mathbf{x}_j) = \sum_{i=1}^{M} \boldsymbol{\beta}_i g_i(\mathbf{x}_j) = \sum_{i=1}^{M} \boldsymbol{\beta}_i (\boldsymbol{\omega}_i^{\mathrm{T}} \mathbf{x}_j + b_i) = \mathbf{o}_j, \tag{1}$$

where $j = 1, 2, \ldots, N$. $\boldsymbol{\omega}_i = [\omega_{i1}, \omega_{i2}, \ldots, \omega_{in}]^{\mathrm{T}}$ is the weight vector connecting the ith hidden node with the input nodes, $\boldsymbol{\beta}_i = [\beta_{i1}, \beta_{i2}, \ldots, \beta_{im}]^{\mathrm{T}}$ is the weight vector connecting the ith hidden node with the output nodes, and b_i is the threshold of the ith hidden node. Since the objective of SLFNs with M hidden nodes and activation function $g(\mathbf{x})$ is to approximate all N samples with zero MSE, we can select the parameters $\boldsymbol{\beta}_i$, $\boldsymbol{\omega}_i$, and b_i such that

$$\mathbf{H}\boldsymbol{\beta} = \mathbf{T}, \tag{2}$$

where $\mathbf{H}$ is called the hidden layer output matrix, i.e.,

$$\mathbf{H} = \begin{bmatrix} g(\boldsymbol{\omega}_1^{\mathrm{T}} \mathbf{x}_1 + b_1) & \cdots & g(\boldsymbol{\omega}_M^{\mathrm{T}} \mathbf{x}_1 + b_M) \\ \cdots & \cdots & \cdots \\ g(\boldsymbol{\omega}_1^{\mathrm{T}} \mathbf{x}_N + b_1) & \cdots & g(\boldsymbol{\omega}_M^{\mathrm{T}} \mathbf{x}_N + b_M) \end{bmatrix}_{N \times M}, \tag{3}$$

$$\boldsymbol{\beta} = [\boldsymbol{\beta}_1^{\mathrm{T}} \cdots \boldsymbol{\beta}_M^{\mathrm{T}}]_{M \times m}^{\mathrm{T}}, \tag{4}$$

$$\mathbf{T} = [\mathbf{t}_1^{\mathrm{T}} \cdots \mathbf{t}_M^{\mathrm{T}}]_{N \times m}^{\mathrm{T}}. \tag{5}$$

It has been proven [6] that if the activation function g is an infinitely differentiable function, e.g., a sigmoid function, then the number of hidden nodes is no larger than the number of training samples, i.e., $M \leq N$. Furthermore, training an SLFN is equivalent to finding a least-square solution $\boldsymbol{\beta}^*$ once $\boldsymbol{\omega}_i$ and b_i $(i = 1, 2, \ldots, M)$ are randomly assigned, which is formulated by

$$\mathbf{H}\boldsymbol{\beta}^* - \mathbf{T} = \min_{\boldsymbol{\beta}} \mathbf{H}\boldsymbol{\beta} - \mathbf{T}. \tag{6}$$

According to the definition of the Moore-Penrose generalized inverse and the proof in [6], the solution is

$$\boldsymbol{\beta}^* = \mathbf{H}^{\dagger}\mathbf{T}, \tag{7}$$

where $\mathbf{H}^{\dagger}$ is the Moore-Penrose generalized inverse of $\mathbf{H}$.

Finally, the ELM algorithm [6] is introduced as follows:

Given a training data set D, an infinitely differentiable activation function g, and a selection of the parameter M,

1. Randomly assign input weights $\boldsymbol{\omega}_i$ and bias b_i, $i = 1, \ldots, M$.
2. Calculate the hidden layer output matrix $\mathbf{H}$.
3. Calculate the output weight $\boldsymbol{\beta}$ according to (7).

2.2 Localized generalization error model

Since the distribution of the true input space is usually unknown [15], every input perturbation is considered as a random variable with a uniform distribution having zero mean. Based on the assumption, our previous work [14] proposed a LGEM for SLFNs trained with ELM.

Definition 1 (**Q-neighborhood**) The **Q**-neighborhood ($S_{\mathbf{Q}}(\mathbf{x}_b)$) of a training sample $\mathbf{x}_b \in D$ is defined as

$$S_{\mathbf{Q}}(\mathbf{x}_b) = \{\mathbf{x} | \mathbf{x} = \mathbf{x}_b + \Delta\mathbf{x}, 0 < \Delta x_i < Q_i, \forall i = 1, \ldots, n\}, \tag{8}$$

where n denotes the number of features in the PCS, and $\mathbf{Q} \in \mathbf{R}^n$ is a given real vector. $\Delta x_i \in \Delta\mathbf{x}$ is considered as the ith input perturbation which is a random variable having a uniform distribution with zero mean.

Let $S_{\mathbf{Q}}$ be the union of all $S_{\mathbf{Q}}(\mathbf{x}_b)$ and call it the **Q**-union. Then, we can define the localized generalization error for unseen samples within $S_{\mathbf{Q}}$ as follows.

Definition 2 (**Localized generalization error**) Let F be an input-output mapping function to be learned, which is formally defined as follows: Given a testing data set with L samples, $T = \{(\mathbf{x}_i', \mathbf{t}_i') | \mathbf{x}_i' \in \mathbf{R}^n, \mathbf{t}_i' \in \mathbf{R}^m, i = 1, 2, \ldots, L\}$, we have $\forall k\left(F(\mathbf{x}_k') = \mathbf{t}_k'\right)$. Then, the localized generalization error of a SLFN f for unseen samples within $S_{\mathbf{Q}}$ ($R_{SM}(\mathbf{Q})$) is defined as the expected MSE regarding those unseen samples, formally

$$R_{SM}(\mathbf{Q}) = \int_{S_{\mathbf{Q}}} (f(\mathbf{x}) - F(\mathbf{x}))^2 p(\mathbf{x}) d\mathbf{x}, \tag{9}$$

where $f(\mathbf{x})$ and $F(\mathbf{x})$ denote the real outputs of the classifier f and the target outputs of F with respect to the unseen sample $\mathbf{x}$, respectively. $p(\mathbf{x})$ is the probability density function of the sample $\mathbf{x}$ within $S_{\mathbf{Q}}$.

Then, a bound on R_{SM} is given by

$$R_{SM}(\mathbf{Q}) \le \left(\sqrt{R_{\text{emp}}} + \sqrt{E_{S_{\mathbf{Q}}}\left((\Delta y)^2\right) + A}\right)^2 + \varepsilon = R^*_{SM}(\mathbf{Q}) \tag{10}$$

with a probability of $(1 - \eta)$, where $\Delta y = f(\mathbf{x}) - f(\mathbf{x}_b)$, $R_{\text{emp}} = (1/N)\sum_{b=1}^{N}(f(\mathbf{x}_b) - F(\mathbf{x}_b))^2$, $\varepsilon = B\sqrt{\ln \eta/(-2N)}$. A, B, N, and η are the difference between the maximum and minimum value of the target outputs, the possible maximum value of the MSE, the number of training samples, and the confidence of the bound, respectively. Note that A can be fixed when the training data set is given, and B is also computable after the classifier is trained. On the other hand, the term in (10) $E_{S_{\mathbf{Q}}}\left((\Delta y)^2\right) = (1/N)\sum_{b=1}^{N}\int_{S_{\mathbf{Q}}(\mathbf{x}_b)}(\Delta y)^2 p(\mathbf{x}) d\mathbf{x}$ is called the stochastic sensitivity measure (SSM) for the SLFN trained with ELM. The SSM quantifies the change of network outputs with respect to the change of network inputs.

3 Improved architecture selection of SLFNs

In this part, we first extend the LGEM proposed in [14] to a new LGEM model in the principal component space, called LGEM-PCS. Then, an architecture selection algorithm based on LGEM-PCS is proposed to determine the optimal number of hidden nodes of SLFNs in the new feature space.

3.1 Localized generalization error model in the principal component space

Since the derivation of R^*_{SM} for classifiers is of the same form according to [14, 15], we adopt (10) as the definition of R^*_{SM} in LGEM-PCS as well. However, the calculation of SSM should be modified due to the differences in learning algorithms and feature spaces. Let the variance of the ith input perturbation be $\sigma^2_{\Delta x_i}$. Similar to [14], we change the form of SSM to

$$\begin{aligned} E_{S_{\mathbf{Q}}}\left((\Delta y)^2\right) &= E_{S_{\mathbf{Q}}}\left(\left(\sum_{i=1}^{M} \boldsymbol{\beta}_i \left(\exp(-P_i) - \exp(-P_i^*)\right)\right)^2\right) \\ &= \sum_{i=1}^{M}\sum_{j=1}^{M} \boldsymbol{\beta}_i \cdot \boldsymbol{\beta}_j E_{S_{\mathbf{Q}}}(V_i V_j), \end{aligned} \tag{11}$$

where $V_i = \exp(-P_i) - \exp(-P_i^*)$, $P_i = \sum_{k=1}^{n}(\omega_{ik}x_k + b_{ik})$ and $P_i^* = \sum_{k=1}^{n}(\omega_{ik}(x_k + \Delta x_k) + b_{ik})$. $\boldsymbol{\beta}_i \cdot \boldsymbol{\beta}_j$ denotes the inner product of $\boldsymbol{\beta}_i$ and $\boldsymbol{\beta}_j$, and

$$\begin{aligned} E_{S_{\mathbf{Q}}}(V_i V_j) = &\left(\text{var}(P_i + P_j) + \text{var}(P_i^* + P_j^*)\right)/2 \\ &- \left(\text{var}(P_i + P_j^*) + \text{var}(P_i^* + P_j)\right)/2. \end{aligned} \tag{12}$$

Due to the fact that the more significant a feature is, the more sensitive it will be to original input perturbations in the new feature space, the calculation of $\text{var}(P_i^* + P_j^*)$ is given as follows

$$\text{var}(P_i^* + P_j^*) = \sum_{k=1}^{n}\left(\left(\omega_{ki} + \omega_{kj}\right)^2\left(\text{var}(x_k) + \sigma^2_{\Delta x_k}\right)\right), \tag{13}$$

Then,

$$E_{S_{\mathbf{Q}}}\left((\Delta y)^2\right) = \sum_{i=1}^{M} \boldsymbol{\beta}_i^2 \sum_{k=1}^{n}\left(\omega_{ki}^2 \sigma^2_{\Delta x_k}\right). \tag{14}$$

The SSM quantifies the change of SLFN outputs with respect to the change of inputs. A higher output fluctuation

yields a larger result of (14), which measures the sensitivity of the network output to the changes in the input.

Finally, we have

$$R^*_{SM} \approx \left(\sqrt{R_{\text{emp}}} + \sqrt{\sum_{i=1}^{M} \boldsymbol{\beta}_i^2 \sum_{k=1}^{n} \left(\omega_{ki}^2 \sigma_{\Delta x_k}^2 \right)} + A \right) + \varepsilon. \tag{15}$$

3.2 Architecture selection based on LGEM-PCS

As stated in [14, 15], there are two methods to compare two classifiers if every feature has the same Q value. One method is to fix the R^*_{SM} value and then compare the difference between their Q values. The other method is to fix the Q value and then compare the difference between their R^*_{SM} values. A classifier, in contradistinction to other classifiers, is considered to have a better generalization capacity if it yields a larger Q value in the former case or a smaller R^*_{SM} value in the latter one. Here, we employ the second method to select the optimal SLFN trained with ELM because the Q values of features in the PCS are different from each other. Therefore, we fix $\mathbf{Q}$, and then determine the number of hidden nodes to minimize the R^*_{SM} value. Accordingly, we formulate the architecture selection problem as

$$\underset{M}{\text{argmin}} R^*_{SM}(\mathbf{Q}) \quad \text{subject to a given } \mathbf{Q}. \tag{16}$$

For every selected value of M, we can determine $R^*_{SM}(\mathbf{Q})$. Then, we choose the specific M that yields the minimum value of $R^*_{SM}(\mathbf{Q})$.

Finally, the architecture selection algorithm for SLFNs trained with ELM based on LGEM-PCS is shown in Table 1.

The stopping criterion could be selected as "M is equal to the number of training samples". However, it is computationally hard for large data sets, and it is also unnecessary to do so according to the results from the experiments over the NSL-KDD data set [13]. Specifically, we employed KDDTrain+ as the training set, and we applied the learned classifier on two testing sets, namely KDDTest+ and KDDTest-21. Figures 1 and 2 illustrate the experimental results of the normalized average value and standard derivation of R^*_{SM} with respect to the number of hidden nodes. These results validate that the change of R^*_{SM} tends to be very small as the value of M increases. Therefore in our experiments, we empirically set the stopping criterion as "M is equal to 250" when the number of training samples is larger than 250.

Table 1 Architecture selection algorithm for SLFNs trained with ELM based on LGEM-PCS

Architecture selection algorithm
1. Initialize the number of hidden nodes $M = 1$
2. Perform PCA to reduce the dimension of input features
3. Randomly assign input weights $\boldsymbol{\omega}_i$ and biasb_i, $i = 1, \ldots, M$
4. Compute the hidden layer output matrix $\mathbf{H}$ using (3)
5. Compute the output weights $\boldsymbol{\beta}$ using (7)
6. Calculate the value of R^*_{SM} for current network using (15)
7. If the algorithm meets the stopping criterion, go to step 8, otherwise, $M = M + 1$ and go to step 2
8. Calculate $\text{argmin}_M R^*_{SM}$

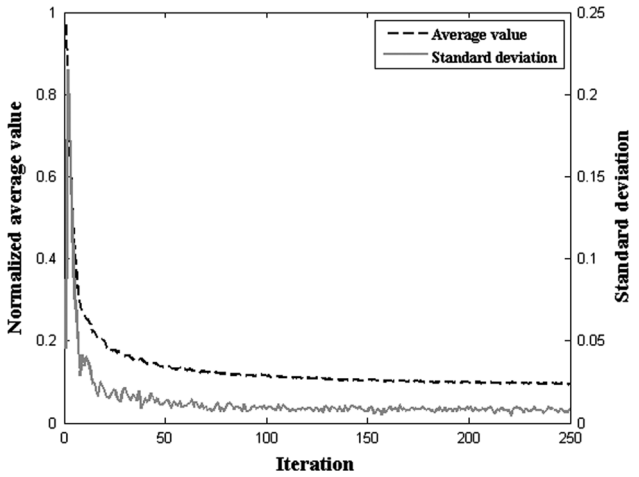

Fig. 1 The normalized average value and standard derivation of R^*_{SM} with respect to the number of hidden nodes, where the testing set is KDDTest+

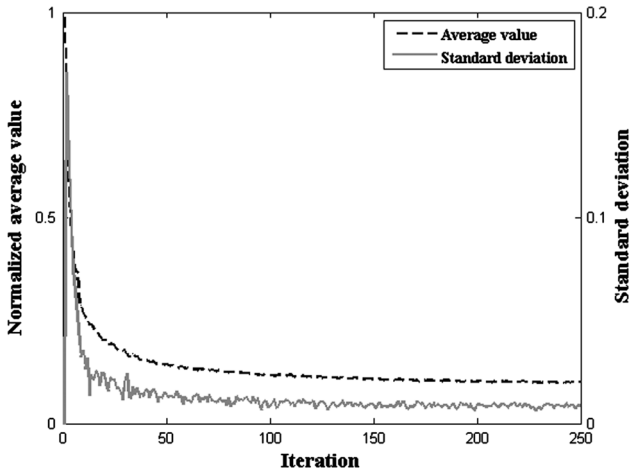

Fig. 2 The normalized average value and standard derivation of R^*_{SM} with respect to the number of hidden nodes, where the testing set is KDDTest-21

4 Discussion

In this part, we briefly discuss the cost and other issues of the classifier in practical usage.

4.1 Computation and memory overheads reduction regarding the ELM algorithm

Highly efficient classifiers are vital for real-time classification and with respect to high volume of data. Therefore, computation and memory overheads are of great concern. Due to various constraints, it is hard in this paper to give a thorough evaluation regarding computation and memory overheads of the proposed classifier. Here, we briefly discuss some overhead reduction methods. According to the ELM algorithm described before, the sizes of input and output weight matrices are $n \times M$ and $M \times m$, respectively, where n is the size of an input vector, and m is that of an output vector. Moreover, the size of the hidden layer output matrix $\mathbf{H}$ is $N \times M$, where N is the total number of records in a training data set. Although n decreases by using some feature reduction techniques, the storage overhead is still very high because the size of $\mathbf{H}$ can not be reduced. Generally speaking, when N is large, the value of M also increases in order to achieve good generalization capacity toward unknown samples. Therefore, the storage cost approximately increases in an exponential manner along with an increase of the size of training data sets, resulting in a heavy burden to system memory. There are two possible ways to reduce the memory overhead: (1) Reducing the size of training data sets. This solution suggests that the proposed classifier could work with other existing classifiers and serve as the last step of classification. The solution is valuable for use in multi-layer classification problems. (2) Delegating the generation of $\mathbf{H}$, the calculation of $\mathbf{H}^{\dagger}$ and the computation of $\boldsymbol{\beta}^{*}$ to a server with sufficient storage capacity. However, the solution may incur additional delays of computing the output weights $\boldsymbol{\beta}$. Note that both solutions are also valuable to reduce the computation overhead because it will significantly reduces the overall CPU times when the size of $\mathbf{H}$ decreases or the heaviest part of the ELM algorithm is delegated to other places.

In order to theoretically evaluate the effectiveness of overhead reduction methods, we analyze a lower bound of the reduction ratio of memory overheads. Here, we do not present analytical results of computation overhead reduction. There are two reasons: Firstly, it is obvious that computation overheads will decrease after applying these overhead reduction methods due to the decrease of data size. Secondly, the data processing at each step of ELM is diverse from each other. Since the advantages of overheads reduction are also obvious by outsourcing the most time and memory consuming parts of ELM to a server, the analysis on the effectiveness of memory overhead reduction is mainly carried out based on the first overhead reduction method. Specifically, we define the reduction ratio of memory overheads (R_{mem}) as the ratio of the total memory assumption (bytes) in two different cases, where the ELM performs over two different data sets, i.e., the reduced data set and the original one. Then, we have the following theorem:

Theorem 1 *Let N be the original size of a training data set. $M^{\dagger}$ ($M^{\dagger} < N$) refers to the number of hidden nodes of a SLFN trained with ELM that performs over the reduced data set. If $M^{\dagger}$ is sufficiently large, then a lower bound of R_{mem} is approximately given by*

$$\underset{M^{\dagger}}{\mathrm{LB}}(R_{mem}) \approx \left(\frac{M^{\dagger}}{N}\right)^{2}. \tag{17}$$

Proof Suppose that the reduced size of the training data set is $N^{\dagger}$ ($N^{\dagger} < N$). Then, the sizes of the input matrix, the input weight matrix, the output matrix, the output weight matrix,the hidden layer output matrix and the bias vector are $n \times N$, $n \times M$, $m \times N$, $M \times m$, $N \times M$ and M, respectively. Thus, the total memory overhead of the SLFN trained with ELM over the training data set is

$$TotO_{mem} = C((m+n)(M+N) + M(N+1)), \tag{18}$$

where C refers to the number of bytes of a storage cell. Similarly, the total memory overhead over the reduced data set is calculated by

$$TotO^{\dagger}_{mem} = C\Big((m+n)\Big(M^{\dagger}+N^{\dagger}\Big) + M^{\dagger}\Big(N^{\dagger}+1\Big)\Big). \tag{19}$$

Then,

$$R_{mem} = \frac{TotO^{\dagger}_{mem}}{TotO_{mem}} \geq \left(\frac{M^{\dagger}}{N}\right)\left(\frac{M^{\dagger}+2m+2n+1}{N+2m+2n+1}\right). \tag{20}$$

When $M^{\dagger}$ is sufficiently large, i.e., $M^{\dagger} \gg m$ and $M^{\dagger} \gg n$, we finally have

$$R_{mem} \geq \left(\frac{M^{\dagger}}{N}\right)^{2}. \tag{21}$$

□

4.2 Performance improvement using cloud computing

As mentioned before, the proposed classifier utilizes the architecture selection algorithm to determine the optimal number of hidden nodes of SLFNs trained with ELM. From the procedure of the algorithm, there are some bottlenecks that severely affect the overall performance of the proposed classifier, i.e., the steps 4, 5, and 6 of the algorithm. On the other hand, cloud computing has been emerging as a

practical technology to handle computation-intensive tasks due to its powerful computation and storage capacities. Therefore, it is intuitive to accelerate the proposed architecture selection algorithm by using the cloud computing [9]. It is worth noticing that such acceleration leads to other practical issues, such as high bandwidth consumption, additional delays, security, and privacy, which are interesting areas to be studied. Moreover, as discussed before, the solution can effectively release conventional users from heavy burden of computation and storage overheads.

5 Performance evaluations

We test the performance of the proposed classifier using the NSL-KDD data set [13], which is an improved version of the well-known KDD CUP 99 data set. NSL-KDD has 41 features, and the simulated attacks in the data set fall in one of the following four categories: (1) Denial-of Service (DoS) (e.g., smurf); (2) Remote to Local (R2L) (e.g., guessing password); (3) User to Root (U2R) (e.g., rootkit); (4) Probing (e.g., port scanning). The type of samples and their sizes for both training and testing data are given in Table 2. Moreover, we run independent trials for 10 times to get average results in each group of experiments, and all experiments are run using Matlab on a computer with Debian 5.0.9, Intel Xeon^{TM} quad core processors at 3.06 GHz, and 1 GB of RAM. The criteria of classification accuracy and time are used for comparing the performance of different classifiers. Regarding a specific data set, the classification accuracy of a classifier is defined as the ratio of the number of correctly classified samples by the classifier to the total number of samples. For the convenience of description, we further define S_{PCA} as the significance level of PCA, which means that the minimum number of selected features (K) in PCS satisfies

$$\sum_{i=1}^{K} \lambda_i \Big/ \sum_{i} \lambda_i \geq S_{PCA}, \tag{22}$$

where λ_i is the eigenvalue of the ith feature, and those features with higher eigenvalues are preferred.

Table 2 Various samples' sizes in NSL-KDD data set

Sample type	Training data	Testing data
Normal	67343	9711
DoS	45927	7458
R2L	942	1656
U2R	105	1298
Probing	11656	2421
Total	125973	22544

5.1 Performance improvements of dimension reduction

To show the effectiveness of PCA in the proposed classifier, we carried out several comparative experiments to show the performance improvement in terms of the classification accuracy and the classification time. For experimental purposes, we employed KDDTrain+ as the training set, and we applied the learned classifier on two testing sets, namely KDDTest+ and KDDTest-21 [13]. Then, we compared the performance of the proposed classifier in three cases: $S_{PCA} = 0.9$, $S_{PCA} = 0.95$, and $S_{PCA} = 1$. Table 3 shows the mean values of the comparative results with respect to different values of S_{PCA}, where R and T represent the classification accuracy and the classification time, respectively. From our experimental results, the number of features decreases from 40 ($S_{PCA} = 1$) to 20 ($S_{PCA} = 0.9$), showing that PCA is valuable to extract more significant features and then to reduce the number of inputs neurons of SLFNs. Furthermore, we see in Table 3 that the classification time is reduced after applying PCA to the data sets while the corresponding accuracy is maintained. The reason is that the number of input neurons of SLFNs and the time for calculating $\mathbf{H}$ decrease as a decrease of the number of features. Therefore, applying PCA can effectively accelerate architecture selection, training and testing procedures.

5.2 Performance comparisons of different classifiers

We further compare the proposed classifier with classifiers trained with ELM using the previous LGEM [14] and the support vector machine (SVM) [1]. For the sake of comparison, we employed KDDTrain+_20%, which contains the first 20% of the records in KDDTrain+, as the training set, and we applied different classifiers on the same testing sets as before, i.e., KDDTest+ and KDDTest-21. Moreover, S_{PCA} was set to 0.9. Figure 3 illustrates the comparative results of different classifiers on these data sets in terms of the classification accuracy. We can see in Fig. 3

Table 3 Comparative results of the accuracy and the time using different testing sets

Testing set	Metric	$S_{PCA} = 0.9$	$S_{PCA} = 0.95$	$S_{PCA} = 1$
KDDTest+	R_{train}	0.9866	0.9872	0.9883
KDDTest+	R_{test}	0.734	0.747	0.7394
KDDTest+	T_{train} (s)	52.844	58.812	60.849
KDDTest+	T_{test} (s)	1.099	1.338	1.353
KDDTest-21	R_{train}	0.9868	0.9873	0.9884
KDDTest-21	R_{test}	0.4629	0.4905	0.4898
KDDTest-21	T_{train} (s)	54.834	56.668	59.741
KDDTest-21	T_{test} (s)	0.657	0.696	0.746

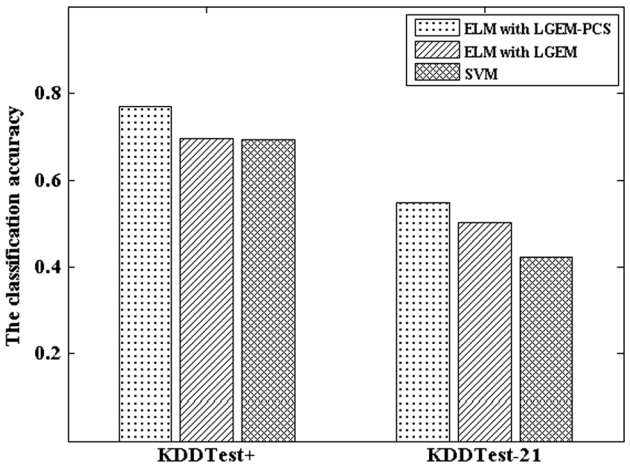

Fig. 3 The comparative results of classifiers with different learning machines on KDDTest+ and KDDTest-21

Table 4 Comparative results of t test on the classification accuracy using different learning machines

Used learning machines	Mean values of the differences in classification accuracy (KDDTest+, KDDTest-21)	T test statistics (KDDTest+, KDDTest-21) ($\alpha = 0.05$)
The proposed vs. ELM/LGEM	(0.071, 0.045)	(**2.36**, 0.721)
The proposed vs. SVM	(0.073, 0.125)	(**7.962**, **13.848**)

that the proposed classifier outperforms the SVM and the classifier trained with ELM using the initial LGEM. Besides, new features of the NSL-KDD data set [13] guarantee that the results are consistent and comparable.

Furthermore, we utilized the *t test* method [11] to test the significance of performance improvement, and the confidence level α was 0.05. Table 4 shows the mean values of the differences in classification accuracy, and the results of the two-sided t tests with a confidence bound of 2.262 ($\alpha = 0.05$). Note that a positive sign with a result indicates that the corresponding result for ELM/LGEM-PCS is larger than that of ELM/LGEM or SVM. Similarly, Table 5 shows the mean values of the classification time (including training time and testing time), and the results of t test statistics of the differences in classification time with the same confidence level. Note that we do not compare ELM/LGEM-PCS with SVM in terms of the classification time because previous work [6] has demonstrated that compared withSVM, the learning time of ELM is dramatically decreased. From Table 4, we see that results of t test statistics exceed the confidence bound, as shown in bold, which shows that the proposed classifier achieves a significant performance improvement at a confidence level

Table 5 Comparative results of t test on the classification time using ELM/LGEM-PCS and ELM/LGEM

Time (s)	Mean values using ELM/LGEM-PCS (KDDTest+, KDDTest-21)	Mean values using ELM/LGEM (KDDTest+, KDDTest-21)	T test statistics of the differences in classification time (KDDTest+, KDDTest-21) ($\alpha = 0.05$)
Training	(1.269, 1.527)	(5.155, 5.944)	(**4.671**, **5.941**)
Testing	(0.207, 0.118)	(0.526, 0.326)	(**4.289**, **5.55**)

of 0.05. We also see in Table 5 that results of t test statistics regarding both training time and testing time exceed the confidence bound, which indicates that the proposed classifier significantly reduces the classification time at the same confidence level as before.

The above results highlight the fact that compared with previous classifiers, the proposed classifier incorporating PCA, ELM and LGEM-PCS can improve the accuracy of anomaly detection while decreasing the detecting time. The results also imply that the proposed classifier is suitable for applications requiring real-time detection.

6 Conclusion

In this paper, we have proposed an improved classifier using a SLFN trained with ELM. The proposed classifier adopts PCA to reduce the feature dimension, and it uses a novel LGEM to select the optimal architecture of the SLFN. Our results have demonstrated a significant performance increase of the proposed classifier compared with previous ones.

References

1. Abe S (2010) Support vector machines for pattern classification, 2nd edn. Springer, New York
2. Chaovalitwongse WA, Jeong YS, Jeong MK, Danish SF, Wong S (2011) Pattern recognition approaches for identifying subcortical targets during deep brain stimulation surgery. IEEE Intell Syst 26(5):54–63
3. Cheng C, Tay WP, Huang GB (2012) Extreme learning machines for intrusion detection. In: Proceedings of 2012 IJCNN, pp 1–8
4. Huang GB, Chen L, Siew CK (2006) Universal approximation using incremental constructive feedforward networks with random hidden nodes. IEEE Trans Neural Netw 17(4):879–892
5. Huang GB, Zhou H, Ding X, Zhang R (2012) Extreme learning machine for regression and multiclass classification. IEEE Trans Syst Man Cybern B Cybern 42(2):513–529
6. Huang GB, Zhu QY, Siew CK (2006) Extreme learning machine: theory and applications. Neurocomputing 70:489–501
7. Lan Y, Hu Z, Soh YC, Huang GB (2013) An extreme learning machine approach for speaker recognition. Neural Comput Appl 22(3-4):417–425

8. Li K, Lu Z, Liu W, Yin J (2012) Cytoplasm and nucleus segmentation in cervical smear images using radiating GVF snake. Pattern Recognit 45(4):1255–1264
9. Lin J, Yin J, Cai Z, Liu Q, Li K, Leung VCM (2013) A secure and practical mechanism for outsourcing elms in cloud computing. To be published in IEEE Intell Syst
10. Liu X, Wang L, Yin J, Zhu E, Zhang J (2013) An efficient approach to integrating radius information into multiple kernel learning. IEEE Trans Cybern 43(2):557–569
11. Moore DS, McCabe GP, Craig BA (2007) Introduction to the practice of statistics, 6th edn. W. H. Freeman and Company, New York
12. Sheikhan M, Jadidi Z, Farrokhi A (2012) Intrusion detection using reduced-size rnn based on feature grouping. Neural Comput Appl 21(6):1185–1190
13. Tavallaee M, Bagheri E, Lu W, Ghorbani AA (2009) A detailed analysis of the kdd cup 99 data set. In: Proceedings of 2009 IEEE CISDA, pp 1–6
14. Wang XZ, Shao QY, Miao Q, Zhai JH (2013) Architecture selection for networks trained with extreme learning machine using localized generalization error model. Neurocomputing 102:3–9
15. Yeung DS, Ng WWY, Wang D, Tsang ECC, Wang XZ (2007) Localized generalization error model and its application to architecture selection for radial basis function neural network. IEEE Trans Neural Netw 18(5):1294–1305