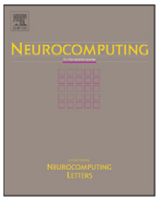
Neurocomputing 102 (2013) 3–9

Contents lists available at SciVerse ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom



Architecture selection for networks trained with extreme learning machine using localized generalization error model

Xi-zhao Wang*, Qing-yan Shao, Qing Miao, Jun-hai Zhai

Key Lab. of Machine Learning and Computational Intelligence, College of Mathematics and Computer Science, Hebei University, Baoding, Hebei 071002, China

ARTICLE INFO

Available online 5 June 2012

Keywords:
Localized generalization error
Extreme learning machine
Network architecture selection
Cross validation (CV)
Sensitivity measure

ABSTRACT

The initial localized generalization error model (LGEM) aims to find an upper bound of error between a target function and a radial basis function neural network (RBFNN) within a neighborhood of the training samples. The contribution of LGEM can be briefly described as that the generalization error is less than or equal to the summation of three terms: training error, stochastic sensitivity measure (SSM), and a constant. This paper extends the initial LGEM to a new LGEM model for single-hidden layer feed-forward neural networks (SLFNs) trained with extreme learning machine (ELM) which is a type of new training algorithms without iterations. The development of this extended LGEM can provide some useful guidelines for improving the generalization ability of SLFNs trained with ELM. An algorithm for architecture selection of the SLFNs is also proposed based on the extended LGEM. Experimental results on a number of benchmark data sets show that an approximately optimal architecture in terms of number of neurons of a SLFN can be found using our method. Furthermore, the experimental results on eleven UCI data sets show that the proposed method is effective and efficient.

1. Introduction

The generalization ability of a trained feed-forward neural network, which refers to the accuracy of the trained network predicting the classes of unseen instances, is the most important index for designing and training a neural network [1–9]. How to improve the generalization ability is the central part of learning feed-forward neural networks. The generalization ability can be equivalently described as generalization error. Given a data set, there are two frequently-used approaches from the literature to measuring and estimating generalization error. One is the cross validation [1] and the other is the error bound model [5–8]. The cross validation is based on a split of the given data set, or generally, based on a K-fold partition of the given data set, where K is an integer bigger than or equal to 2. Specifically, the given data set is first divided to K disjoint subsets, and then the union of any $(K-1)$ subsets is considered as a training set, and the remaining subset is the testing set. Correspondingly, K neural network classifiers are generated and the average of testing errors of the K classifiers on the remaining subset is referred as to an estimation of the generalization error, where the testing error of a classifier is defined as the number of testing instances wrongly predicted by the classifier. The K-fold cross validation is experimentally driven and easy-implementation, but it is time consuming. The error bound model for a trained neural network is to estimate the generalization error by finding an upper bound of the difference between the target function and the trained neural network on the entire input space. The main advantage of error bound models is that the error bound can be definitely suitable for all unseen instances and the main disadvantages are the difficulty of mathematical derivation in finding the bound and the looseness of the derived upper bound. Recently, based on the stochastic sensitivity analysis of the radial basis function neural network (RBFNN), a localized generalization error model (LGEM) is proposed in [10], which gave an upper bound on the generalization error for unseen samples located within neighborhoods of the training samples.

Another important index for training a neural network is the complexity of training algorithm. One of the old frequently-used algorithms for training a feed-forward neural network is the gradient-based back-propagation (BP) algorithm. Even if its performance is often claimed in textbooks or references to be better, it is criticized as time-consuming of iteration and it often reaches local minimum. Recently, a new training algorithm for single-hidden layer feed-forward neural networks (SLFNs) was firstly proposed in [11]. It is referred to as Extreme Learning Machine (ELM) which performs on a SLFN. The ELM's main features include the following points.

(1) ELM does not need iteration to tune the weights and so it is not time-consuming.

* Corresponding author.
E-mail addresses: xizhaowang@ieee.org, wangxz@mail.hbu.edu.cn (W. Xi-zhao), shaoqingyan2008@163.com (S. Qing-yan), miaoqing198607@126.com (M. Qing), mczjh@hbu.edu.cn (Z. Jun-hai).

0925-2312/$ - see front matter © 2012 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.neucom.2011.12.053

(2) The weights between input layer and hidden layer are randomly generated.
(3) The weights between hidden layer and output are determined by solving the system of generalized linear equations.
(4) There is no general conclusion for the generalization ability of SLFNs trained by ELM.

The related references [11–18,25,29–31] can be found from journals and conference proceedings. More recently, a survey paper on ELM, which completely introduces ELM's historical development, newest advances, key issues, and main advantages and disadvantages, has been published in Journal of Machine Learning and Cybernetics [16].

Since there is still no general result for the generalization ability of SLFNs trained by ELM (point (4) mentioned above), based on the error bound model given in [10] for RBFNNs, this paper makes an attempt to establish an upper error bound model for a SLFN trained by ELM. A main concept for deriving the upper error bound formula is called Q-neighbor. It is expected that the establishment of the upper error bound model can provide some useful guidelines for clarifying the generalization ability of SLFNs trained by ELM. Moreover, the upper error bound model here has potential applications to topics such as feature selection and architecture selection of networks.

This paper is organized as follows. Section 1 is the introduction. In Section 2, the ELM for SLFNs and LGEM for RBFNNs are briefly reviewed. In Section 3, the LGEM for SLFNs trained with ELM is derived in detail and its application to architecture selection is presented. In Section 4, a number of numerical experiments using this newly proposed model are conducted and some experimental results and remarks are showed there. Section 5 concludes this paper.

2. Brief review of extreme learning machine and localized generalization error model

In this section, we briefly review the basic concepts and methods of extreme learning machine and localized generalization error model.

2.1. Extreme learning machine (ELM)

The ELM algorithm was proposed by Huang for single-hidden layer feed-forward neural networks (SLFNs). According to Theorem 2.1 in [11], the input weights and biases do not need to be adjusted. It is possible to analytically determine the output weights by finding the least-square solution. The neural network is obtained after a few steps with very low computational cost.

Given a training data set, $L=\{(x_i,t_i)|x_i\in R^n,t_i\in R^m,i=1,2,\cdots,N\}$, where x_i is a $n\times 1$ input vector and t_i is a $m\times 1$ target vector, a SLFN with M hidden nodes is formulated as

$$f_\theta(x_i)=\sum_{j=1}^{M}\beta_j g(w_j\cdot x_i+b_j)=t_i(1\le i\le N) \tag{1}$$

where $w_j=[w_{j1},w_{j2},\cdots w_{jn}]^T$is the weight vector connecting the jth hidden node with the input nodes. $\beta_j=[\beta_{j1},\beta_{j2}\ldots,\beta_{jm}]^T$ is the weight vector connecting the jth hidden node with the output nodes, and b_j is the threshold of the jth hidden node. Eq. (1) can be written in a more compact format as

$$H\beta=T \tag{2}$$

where

$$H=\begin{bmatrix} g(w_1\times x_1+b_1) & \cdots & g(w_M\times x_1+b_M)\\ \vdots & \cdots & \vdots\\ g(w_1\times x_N+b_1) & \cdots & g(w_M\times x_N+b_M)\end{bmatrix}_{N\times M} \tag{3}$$

$$\beta=\begin{bmatrix}\beta_1^T\\ \vdots\\ \beta_M^T\end{bmatrix}_{M\times m}\quad\text{and}\quad T=\begin{bmatrix}t_1^T\\ \vdots\\ t_N^T\end{bmatrix}_{N\times m} \tag{4}$$

H is the hidden layer output matrix of the network [12], where the jth column of H is the jth hidden node's output vector with respect to inputs x_1, x_2, …, x_N, and the ith row of H is the output vector of the hidden layer with respect to input x_i. If the number of hidden nodes is equal to the number of distinct training samples, the matrix H is square and invertible, and SLFNs can approximate these training samples with zero error. But generally, the number of hidden nodes is much less than the number of training samples. Therefore, H is a non-square matrix and we can not expect an exact solution of the system (2). Fortunately, it has been proved in [13,14] that SLFNs with random hidden nodes have the universal approximation capability and the hidden nodes could be randomly generated. According to the definition of the *Moore-Penrose generalized inverse*, the smallest norm least-squares solution of (2) is given in [11]:

$$\hat{\beta}=H^\dagger T \tag{5}$$

where $H^\dagger$ is the Moore-Penrose *generalized inverse* of matrix H [19].

In the following, the *ELM* Algorithm [11] is introduced.

ELM Algorithm: Given a training data set $\{(x_i,t_i)|,x_i\in R^n,t_i\in R^m, i=1,\ldots,N\}$, an activation function g, and the number of hidden nodes M:

(1) Randomly assign input weights w_j and biases b_j, $j=1,\ldots, M$.
(2) Calculate the hidden layer output matrix H;
(3) Calculate output weights matrix $\beta=H^\dagger T$.

2.2. Localized generalization error model (LGEM)

The localized generalization error model [5,6,10] was proposed by Yeung and it is to find an upper bound of MSE for unseen samples which have similar feature values to the training samples (i.e., the unseen samples will only occur with the distance smaller than a given value Q from the training samples in the input space) [5,6].

The Q-neighborhood ($S_Q(x_b)$) of a training sample x_b is defined as $S_Q(x_b)=\{x|x=x_b+\Delta x_i, 0<|\Delta x_i|\le Q,\forall i=1,2,\cdots,n\}$, where n denotes the number of input features, and Q is a given real value. Let S_Q be the union of all $S_Q(x_b)$ and it is called as the Q-union. All samples in $S_Q(x_b)$ except x_b are considered as unseen samples. In classification problem, we usually do not have any knowledge about the distribution of the true input space. Therefore, each unseen sample is expected to have the same chance to appear; correspondingly, Δx_i is considered to be input perturbations which is random variable having uniform distribution with zero mean and variance $\sigma^2_{\Delta x_i}$ .For $0\le Q_1\le\cdots\le Q_k\le\infty$, the following relationship holds:

$$D\subseteq S_{Q_1}\subseteq\cdots\subseteq S_{Q_k}\subseteq T \tag{6}$$

where D is the training data set and T is the entire input space. The localized generalization error is defined as:

$$R_{SM}(Q)=\int_{S_Q}(f_\theta(x)-F(x))^2p(x)dx \tag{7}$$

where f_θ denotes the classifier with parameter set θ, $F(x)$ denotes the true unknown input–output mapping function and $p(x)$

denotes the true unknown probability density function of the input x. By the Hoeffding's inequality [20], with a probability of $(1-\eta)$, we have

$$R_{SM}(Q) = \int_{S_Q} (f_\theta(x) - F(x))^2 p(x)dx \le (\sqrt{E_{S_Q}((\Delta y)^2)} + \sqrt{R_{emp}} + A)^2 + \varepsilon = R^*_{SM}(Q) \quad (8)$$

where $\Delta y = f_\theta(x) - f_\theta(x_b)$, $\varepsilon = B\sqrt{\ln \eta/(-2N)}$, $R_{emp} = \frac{1}{N}\sum_{b=1}^{N}(f_\theta(x_b) - F(x_b))^2$. R_{emp} is the training error, $E_{S_Q}((\Delta y)^2)$ denotes the stochastic sensitivity measure(SSM), A, B and η, the difference between the maximum and minimum value of the target outputs, the possible maximum value of the MSE and the confidence of the bound, respectively. A and B can be fixed when the data set is given.

3. Architecture selection algorithm based on extended LGEM

The number of hidden neurons in feed-forward neural network is usually selected by sequential learning [9,15,26] or by CV. However, the sequential learning technique determines the number of hidden neurons by making use of the training error without considering the generalization capability. The CV technique is time consuming for large data sets. In this paper, we propose an architecture selection algorithm based on extended LGEM which considers the generalization capability and costs less time than CV. We derive the SSM of R^*_{SM} for single-hidden layer feed-forward neural network with sigmoid activation function in Section 3.1 and an algorithm for architecture selection of the SLFNs is proposed in Section 3.2.

3.1. Derivation of the SSM for SLFNs with sigmoid activation function

The stochastic sensitivity measure (SSM) measures the output perturbations (Δy) of the classifier when the input value changes [21,22,27]. In this paper, we assume the training samples are independent and only consider input perturbation. In addition, the input perturbation of the ith input feature is a random variable with a uniform distribution having the mean $\mu_{\Delta x_i} = 0$ and the variance $\sigma^2_{\Delta x_i} = (2Q)^2/12 = Q^2/3$ .Uniform distribution is assumed here because without any prior knowledge on the distribution of unseen samples, we assume that they have an equal chance of occurrence. According to (1), SLFNs with sigmoid activation function could be described as

$$f_\theta(x) = \sum_{j=1}^{M} \beta_j g(w_j \times x + b_j) = \sum_{j=1}^{M} \beta_j \frac{1}{1+\exp(-(w_j \times x + b_j))} \quad (9)$$

According to Taylor's series expansion: $\frac{1}{1+x} = \sum_{t=0}^{\infty}(-1)^t x^t$ $(-1 < x < 1)$,

$$f_\theta(x) = \sum_{j=1}^{M} \beta_j \sum_{t=0}^{\infty} (-1)^t (\exp(-(w_j \times x + b_j)))^t, (0 < \exp(-(w_j \times x + b_j)) < 1) \quad (10)$$

Ignoring the terms with order larger than 1, we have:

$$f_\theta(x) \approx \sum_{j=1}^{M} \beta_j (1 - \exp(-(w_j \times x) + b_j)) \quad (11)$$

Let $S_j = \sum_{i=1}^{n}(w_{ij}x_i + b_{ij})$ and $S^*_j = \sum_{i=1}^{n}(w_{ij}(x_i + \Delta x_i) + b_{ij})$. Then $E_{S_Q}((\Delta y)^2)$

$$= E_{S_Q}\left(\left(\sum_{j=1}^{M}\beta_j(1-\exp(-S^*_j)) - \sum_{j=1}^{M}\beta_j(1-\exp(-S_j))\right)^2\right)$$

$$= E_{S_Q}\left(\left(\sum_{j=1}^{M}\beta_j\left(\exp(-S_j) - \exp(-S^*_j)\right)\right)^2\right) \quad (12)$$

Let $V_j = \exp(-S_j) - \exp(-S^*_j)$, then we have:

$$E_{S_Q}((\Delta y)^2) = \sum_{j=1}^{M}\sum_{i=1}^{M}\beta_i\beta_j E_{S_Q}(V_iV_j) \quad (13)$$

We have:

$$E_{S_Q}(V_iV_j) = E_{S_Q}(\exp(-S_i - S_j)) - E_{S_Q}(\exp(-S_i - S^*_j)) - E_{S_Q}(\exp(-S^*_i - S_j)) + E_{S_Q}(\exp(-S^*_i - S^*_j)) \quad (14)$$

According to the central limit theorem, $\exp(S_j)$and $\exp(S^*_j)$ have a log-normal distribution. Therefore,

$$E_{S_Q}(\exp(-S^*_i - S^*_j)) = \exp\left(\frac{Var(S^*_i + S^*_j)}{2} - E(S^*_i + S^*_j)\right) \approx 1 + \frac{Var(S^*_i + S^*_j)}{2} - E(S^*_i + S^*_j) \quad (15)$$

Then,

$$E_{S_Q}(V_iV_j) = \tfrac{1}{2}(Var(S^*_i + S^*_j) + Var(S_i + S_j) - Var(S^*_i + S_j) - Var(S_i + S^*_j)) \quad (16)$$

Because:

$$Var\left(S^*_i + S^*_j\right) = Var\left(\sum_{k=1}^{n}(w_{ki}(x_k + \Delta x_k) + b_{ki}) + \sum_{k=1}^{n}(w_{kj}(x_k + \Delta x_k) + b_{kj})\right) = \sum_{k=1}^{n}(w_{ki} + w_{kj})^2 Var(x_k) + \frac{Q^2}{3}\sum_{k=1}^{n}(w_{ki} + w_{kj})^2 \quad (17)$$

And, finally we get:

$$E_{S_Q}((\Delta y)^2) = \frac{Q^2}{3}\sum_{j=1}^{M}\beta_j^2\sum_{k=1}^{n}w_{kj}^2 \quad (18)$$

where Q is a given real value, β_j is the output of the jth hidden node and w_{kj} is the weight connecting the jth hidden node and the kth input node.

The sensitivity measure (SM) of neural network [21,22] gives a quantified data on the change of network outputs with respect to change of network inputs. Intuitively, it measures how sensitive the network output is to the input change. Here, the formula (18) measures the output fluctuations of the classifier. A classifier that has high output fluctuations yields high SM because its output varies dramatically when the input value changes. Due to the classifier bias/variance dilemma, a classifier yielding a good generalization capability should minimize both the training error and SM or achieves a good balance between the two [28].

3.2. Architecture selection algorithm based on extended LGEM

There are two ways to compare two classifiers [10]. One way is to fix the value of $R^*_{SM}(Q)$ and compare the magnitude between the Q values. The other is to fix the Q value and compare the $R^*_{SM}(Q)$. Suppose there are two classifiers, f_1 and f_2. Given a value a of R^*_{SM}, if $R^*_{SM}(Q_1) = R^*_{SM}(Q_2) = a$, $Q_1 < Q_2$, then f_2 has a better generalization performance. In order to get the best network structure and find the optimal classifier, we can select the largest Q which satisfies $R^*_{SM}(Q) = a$. On the other hand, we can also compute the R^*_{SM} using the same Q, the lower R^*_{SM} is, the better generalization capability the network has.

According to the first way, we can formulate the architecture selection problem as an optimization problem, that is

$$\max_{Q} R^*_{SM}(Q) \le a \tag{19}$$

According to (8) and (18), the R^*_{SM} for SLFNs with sigmoid activation function is as follows:

$$R^*_{SM} \approx \left(\sqrt{\frac{Q^2}{3} \sum_{j=1}^{M} \beta_j^2 \sum_{k=1}^{n} w_{kj}^2} + \sqrt{R_{emp}} + A \right)^2 + \varepsilon \tag{20}$$

Let $R^*_{SM}(Q) = a$, we have

$$Q^2 \sum_{j=1}^{M} \beta_j^2 \sum_{k=1}^{n} w_{kj}^2 - 3(\sqrt{a-\varepsilon} - \sqrt{R_{emp}} - A)^2 = 0 \tag{21}$$

Eq. (21) is a quadratic equation for Q. There are two solutions for Q and let Q^* be the positive real solution of the quadratic equation.

$$h(M,Q^*) = \begin{cases} 0, & R_{emp} \ge a \\ Q^*, & \text{else} \end{cases} \tag{22}$$

The algorithm for architecture selection of SLFNs trained with ELM based on LGEM (ASLGEM-ELM) is given as follows.

1) Initialize $M=1$;
2) Randomly assign input weight w_j and bias b_j of hidden nodes, $j=1,\ldots,M$;
3) Calculate the hidden layer output matrix H;
4) Calculate output weights matrix $\beta = H^{\dagger}T$;
5) Calculate the Q-value for the current network using (22);
6) If it does not meet the stop criterion, then $M=M+1$ and go to step (2), otherwise, go to (7);
7) Calculate $\underset{M}{\operatorname{argmax}}\, h(M,Q^*)$.

The stop criterion could be "M is equal to the number of training samples". But it is not only time consuming but also unnecessary. Generally speaking, when $M \ll N$, the ELM algorithm also gets high training and testing accuracy [11]. In our experiments, we set the stop criterion as $M=500$ when the number of training samples is larger than 500.

4. Experimental results and analysis

In this section, we investigate the performance of the proposed algorithm ASLGEM-ELM by conducting experiments on eleven benchmark classification data sets and five regression problems chosen from the University of California at Irvine (UCI) Machine Learning Repository [23]. The basic information of the data sets is given in Table 1. All attributes have been scaled to [0,1] to reduce the effect of large values. Note that the proposed method ASLGEM-ELM can be applied to any variation of problems with any number of features, samples and classes.

In the experiments, every data set is randomly divided into two parts, training data set and testing data set, and the simulations are conducted in Matlab 7.1 running on a desktop PC (2.80 G HZ CPU). Three groups of experiments are conducted for different purposes. In experiment 1, we verify our proposed method with ELM by the results of average testing accuracy. In experiment 2, we compare our method with CV and OP-ELM [31] in architecture selection on classification problems. In experiment 3, it is compared with I-ELM [13], CS-ELM [32] and EM-ELM [15] on regression problems. Each experiment is repeated ten times to get ten independent results for each data set. The experimental results show that our proposed algorithm could obtain a network with the best generalization performance.

Table 1
The basic information of the benchmarking data sets.

	No. of features	No. of samples	No. of classes
Classification			
Iris	4	150	3
Breast Cancer	10	699	2
Wine Recognition	13	178	3
Sonar Target	60	208	2
Pima Diabetes	8	768	2
Ionosphere	34	351	2
Car Evaluation	6	1728	4
Waveform	21	5000	3
Optical Digit	64	5620	10
Mushroom	22	8124	2
Wisconsin Breast Cancer	32	569	2
Regression			
Servo	4	167	–
Boston housing	13	506	–
Auto-MPG	8	398	–
Abalone	8	4177	–
Ailerons	40	7154	–

The setting of parameters of in ASLGEM-ELM is as follows. In (8), the difference between the maximum and minimum values of target outputs (A) and the number of training samples (N) are fixed after giving the training data set. The maximum possible value of the MSE (B) and the confidence level of the R^*_{SM} bound (η) could also be selected before any classifier training. From Eq. (22), one may notice that the smaller a is, the larger number of hidden nodes will be selected by ASLGEM-ELM. This is because the Q value will be zero if its training error is larger than a and the training error decreases as the number of hidden nodes increases. Since the experimental results show that the trained SLFNs with training error larger than 0.25 will not yield good generalization capability, the constant a is set as 0.25.

4.1. Experiment 1: Performance verification

In this section, we verify that the proposed algorithm ASLGEM-ELM with ELM. In Table 2, nodes1 denotes the number of the hidden nodes selected by our algorithm and nodes2 denotes the optimal number of hidden nodes by many trials with ELM. nodesi (i=3,4) denotes the number of the hidden nodes two times and half as many as nodes2, respectively. From Table 2, the optimal network architecture can be found by our algorithm, because nodes1 and nodes2 is comparative. However, when the number of hidden nodes in SLFNs trained by ELM is more or less than nodes2, the testing accuracies are lower than our method in all the ten datasets. It could be easily observed that our proposed algorithm ASLGEM-ELM could select the optimal network architecture in terms of number of neurons.

4.2. Experiment 2: Benchmarking with real classification applications

In this section, we compare the ASLGEM-ELM with well known architecture selection method, i.e. the k-fold cross validation (CV) [1] and OP-ELM. The average classification testing accuracies and average running time of ten time experiments of each data set are given in Tables 3 and 4.

Experimental results in Table 3 show that the ASLGEM-ELM performs best among the methods in terms of average classification testing accuracy in most cases and training time on all the data sets.

Table 2
The results of experiment 1: performance verification.

Datasets	ASLGEM-ELM		ELM					
	No. of nodes1	Testing accuracy (%)	No. of nodes2	Testing accuracy (%)	No. of nodes3	Testing accuracy (%)	No. of nodes4	Testing accuracy (%)
Iris	6.7	98.67	7	98.67	14	97.33	4	78.67
Breast Cancer	8.6	96.20	9	96.93	18	95.03	5	93.57
Wine Recognition	13.8	98.88	14	98.88	28	96.63	7	91.01
Sonar Target	45.2	84.48	45	84.48	90	69.23	22	71.15
Pima Diabetes	35.8	83.85	36	84.38	72	75.52	18	76.04
Ionosphere	36.3	90.34	36	90.91	72	85.80	18	84.09
Car Evaluation	349.7	95.37	350	95.37	700	88.08	175	93.75
Waveform	69.5	85.16	70	85.16	140	84.92	35	83.28
Optical Digit	270.4	97.76	270	97.37	540	97.30	135	85.08
Mushroom	207.3	99.98	207	99.93	414	99.89	103	99.68

Table 3
The results of experiment 2: comparison with CV.

Datasets	ASLGEM-ELM			5-CV			10-CV		
	No. of nodes	Testing accuracy (%)	CPU Time1	No. of nodes	Testing accuracy (%)	CPU Time2	No. of nodes	Testing accuracy (%)	CPU Time3
Iris	6.7	98.67	2.2969	13.6	96.53	11.60	10.3	96.07	13.4
Breast Cancer	8.6	96.20	66.9063	14.1	96.99	672	17.9	96.92	1012.7
Wine Recognition	13.8	98.88	1.9063	34.6	90.06	23.8	40.2	91.19	29.1
Sonar Target	45.2	84.48	8.2969	30.4	80.49	44.0	35.2	80.87	54.1
Pima Diabetes	35.8	83.85	81.2188	17.6	75.86	561.2	12.3	75.55	824.1
Ionosphere	36.3	90.34	9.6563	54.1	83.29	243.2	65.1	83.40	279.9
Car Evaluation	349.7	95.37	484.2031	63.4	90.92	2048.0	80.0	91.61	2209.7
Waveform	69.5	85.16	250.8125	32.5	82.60	255.4	54.3	84.20	283.8
Optical Digit	270.4	97.76	290.8750	93.2	94.80	4492.4	112.1	95.25	4663.0
Mushroom	207.3	99.98	267.3906	135.4	99.70	751.2	152.6	99.94	799.7

Table 4
The results of experiment 2: comparison with OP-ELM.

Datasets	OP-ELM		ASLGEM-ELM	
	Time(s)	Testing accuracy (%)	Time(s)	Testing accuracy (%)
Iris	7.4e−2	95.00	2.2969	98.67
Wisconsin B.C.	1.1	95.60	54.7344	96.84
Pima I.D.	0.96	74.90	81.2188	83.85
Wine	0.44	90.70	1.9063	98.88

From Table 3, the learning speed of our method ASLGEM-ELM is the fastest in all cases. ASLGEM-ELM is about 5 to 20 times faster than CV besides waveform and mushroom. The reason that more time is required by CV is that for k-fold CV and L choices of running times, kL classifiers must be trained. As shown in Table 3, for Breast Cancer dataset ASLGEM-ELM gains a little lower testing accuracy than CV, but for all the other data sets, ASLGEM-ELM gains higher testing accuracies than CV. The reason is that CV method estimates the expected generalization error instead of its bound. Thus, they can not guarantee the constructed classifiers have good generalization capability [24]. The presenting experimental results illustrate that our proposed method has better performance than CV on the same training and testing data sets.

According to the test results of Table 4, the OP-ELM is faster than our method. This is comprehensible, because our method trains multiple classifiers in order to find the maximum value of Q (i.e., the classifier with the best generalization performance). It could be obviously concluded that the network architecture selected by our method is superior than that the OP-ELM selected.

4.3. Experiment 3: Benchmarking with regression applications

In this section, we evaluate and compare the performance of the proposed ASLGEM-ELM with I-ELM, EM-ELM and CS-ELM. In our experiments, the I-ELM and EM-ELM algorithms have two parameters that need to be considered: the expected training rate ε and the maximum number of hidden nodes $L_{\max}$. In the evaluations, we aim to get the network when the training rate reaches the specific ε without the constraint of $L_{\max}$. Therefore, in our experiments, $L_{\max}$ is set to be a large value of 2000. For the CS-ELM, only the number of hidden neurons needs to be set by us. According to [32], $L_{\max}$ is set to be 100 in order to obtain a compact network structure.

From Table 5, it is clear that the proposed ASLGEM-ELM achieves better generalization performance than others and comparable compact network structure.

I-ELM achieves comparable performance with the largest network structure for all the regression cases. Comparing CS-ELM, EM-ELM with the I-ELM, the network obtained using CS-ELM and EM-ELM is always more compact than I-ELM. From Table 5, ASLGEM-ELM performs better than others, because our method is proposed based on generalization error.

5. Conclusions and future works

In this paper, we derive the SSM of SLFNs with sigmoid activation function and propose an algorithm for architecture selection of SLFNs trained by ELM based on LGEM. The algorithm is not only simple but also efficient to automatically determine the number of hidden nodes with the best generalization performance. The experimental results demonstrate that the proposed method could select the architecture of SLFNs which have the best generalization performance, and yield higher average classification testing accuracy and less training time than CV.

Since ELM randomly generates input weights and biases which may different each time, the network structure is not steady. In our experiments, we have to compute the average number of hidden nodes. In this paper, we have considered the input

Table 5
The results of experiment 3: benchmarking with regression cases.

Datasets	Algorithms	No. of hidden nodes	Training time(s)	Testing RMSE
Servo	I-ELM	2000	0.1398	0.1440
	EM-ELM	28.6	0.0125	0.1369
	CS-ELM	16.8	0.0078	0.1214
	ASLGEM-ELM	12.8	0.8125	0.1187
Auto-MPG	I-ELM	2000	0.1422	0.0831
	EM-ELM	21.7	0.0086	0.0803
	CS-ELM	13.5	0.0125	0.0806
	ASLGEM-ELM	18.9	1.0625	0.0804
Boston housing	I-ELM	2000	0.1844	01043
	EM-ELM	46.05	0.0219	0.1052
	CS-ELM	22.75	0.0117	0.1086
	ASLGEM-ELM	50.3	0.5469	0.0935
Abalone	I-ELM	1840.6	0.4820	0.0822
	EM-ELM	11.5	0.0117	0.0794
	CS-ELM	20.8	0.0375	0.0771
	ASLGEM-ELM	26.4	2.9219	0.0771
Ailerons	I-ELM	2000	0.8492	0.0658
	EM-ELM	79.8	0.5203	0.0526
	CS-ELM	31.9	0.3414	0.0530
	ASLGEM-ELM	38.2	17.7500	0.0514

perturbation, leaving behind the weights perturbation. A future work is to add the effect of weights perturbation to the SSM.

Acknowledgments

This research is supported by the national natural science foundation of China (61170040, 60903088, 60903089), by the natural science foundation of Hebei Province (F2010000323, F2011201063), by the Key Scientific Research Foundation of Education Department of Hebei Province (ZD2010139), and by the natural science foundation of Hebei University (2011-228).

References

[1] S. Haykin., Neural Networks: A Comprehensive Foundation, 2nd edition, Pearson, Prentice Hall, 2004.
[2] F.M. Ham, I. Kostanic, Principles of Neurocomputing for Science and Engineering, McGraw-Hill, 2003.
[3] X.Z. Wang, C.R. Dong, Improving generalization of fuzzy if-then rules by maximizing fuzzy entropy, IEEE Trans. Fuzzy Syst. 17 (3) (2009) 556–567.
[4] X.Z. Wang, J.H. Zhai, S.X. Lu, Induction of multiple fuzzy decision trees based on rough set technique, Inf. Sci. 178 (16) (2008) 3188–3202.
[5] D.S. Yeung, P.P.K. Chan, W.W.Y. Ng, Radial Basis Function network learning using localized generalization error bound, Inf. Sci. 179 (19) (2009) 3199–3217.
[6] W.W.Y. Ng, D.S. Yeung, M. Firth, E.C.C. Tsang, X.Z. Wang, Feature selection using localized generalization error for supervised classification problems using RBFNN, Pattern Recognit. 41 (12) (2008) 3706–3719.
[7] J.G. Polhill, M.K. Weir, An approach to guaranteeing generalization in neural networks, Neural Networks 14 (8) (2001) 1035–1048.
[8] Y. Liu, Unbiased estimate of generalization error and model selection in neural network, Neural Networks 8 (2) (1995) 215–219.
[9] S.M. Zhang, P. McCullagh, C. Nugent, H.R. Zheng, M. Baumgarten, Optimal model selection for posture recognition in home-based healthcare, Int. J. Mach. Learn. Cybern. 1 (2) (2011) 1–14.
[10] D.S. Yeung, W.W.Y. Ng, D. Wang, E.C.C. Tsang, X.Z. Wang, Localized generalization error and its application to architecture selection for radial basis function neural network, IEEE Trans. Neural Networks 18 (5) (2007) 1294–1305.
[11] G.B. Huang, Q.Y. Zhu, C.K. Siew, Extreme learning machine: Theory and applications, Neurocomputing 70 (1-3) (2006) 489–501.
[12] G.B. Huang, Learning capability and storage capacity of two-hidden-layer feedforward networks, IEEE Trans. Neural Networks 14 (2) (2003) 274–281.
[13] G.B. Huang, L. Chen, C.K. Siew, Universal approximation using incremental constructive feedforward networks with random hidden nodes, IEEE Trans. Neural Networks 17 (4) (2006) 879–892.
[14] G.B. Huang, L. Chen, Convex incremental extreme learning machine, Neurocomputing 70 (16–18) (2007) 3056–3062.
[15] G.R. Feng, G.B. Huang, Q.P. Lin, Robert Gay. Error, Minimized extreme learning machine with growth of hidden nodes and incremental learning, IEEE Trans. Neural Networks 20 (8) (2009) 1352–1357.
[16] G.B. Huang, D.H. Wang, Y. Lan, Extreme learning machines: a survey, Int. J. Mach. Learn. Cybern. 2 (2) (2011) 107–122.
[17] X.Z. Wang, A.X. Chen, H.M. Feng., Upper integral network with extreme learning mechanism, Neurocomputing 74 (16) (2011) 2520–2525.
[18] J. Wu, S.T. Wang, F.L. Chung, Positive and negative fuzzy rule system, extreme learning machine and image classification, Int. J. Mach. Learn. Cybern. (2011)http://dx.doi.org/10.1007/s13042-011-0024-1.
[19] D. Serre, Matrices: Theory and Applications, Springer, New York, 2002.
[20] W. Hoeffding, Probability inequalities for sums of bounded random variables, J. Am. Stat. Assoc. 58 (1) (1963) 13–30.
[21] W.W.Y. Ng, D.S. Yeung, Selection of weight quantization accuracy for radial basis function neural network using stochastic sensitivity measure, Electron. Lett. 39 (10) (2003) 787–789.
[22] W.W.Y. Ng, D.S. Yeung, X.Z. Wang, I. Cloete. A study of the difference between partial derivative and stochastic neural network sensitivity analysis for applications in supervised pattern classification problems, Proceedings of 2004 International Conference on Machine Learning and Cybernetics, 7, pp: 4283–4288.
[23] C.L. Blake, C.J. Merz. UCI Repository of machine learning databases. 〈http://www.ics.uci.edu/~mlearn/MLRepository.html〉.
[24] T. Hastie, R. Tibshirani, J. Friedman, The Element of Statistical Learning, Springer-Verlag, New York, 2001.
[25] M. Heeswijk, Y. Miche, E. Oja, A Lendasse., GPU-accelerated and parallelized ELM ensembles for large scale regression, Neurocomputing 74 (16) (2011) 2430–2437.
[26] D.L. Tong, R. Mintram, Genetic algorithm-neural network (GANN): a study of neural network activation functions and depth of genetic algorithm search applied to feature selection, Int. J. Mach. Learn. Cybern. 1 (1–4) (2010) 75–87.
[27] X.Z. Wang, C.G. Li., A definition of partial derivative of random functions and its application to RBFNN sensitivity analysis, Neurocomputing 71 (7–9) (2008) 1515–1526.
[28] S. Geman, E. Bienenstock, Neural networks and the bias/variance dilemma, Neural Comput. 4 (1992) 1–58.
[29] G.-B. Huang, L. Chen, Enhanced random search based incremental extreme learning machine, Neurocomputing 71 (2008) 3460–3468.
[30] G.-B. Huang, H. Zhou, X. Ding, and R. Zhang, Extreme learning mnachine for regression and multi-class classification. accepted by IEEE Trans. Syst. Man Cybern.
[31] Y. Miche, A. Sorjamaa, P. Bas, et al., OP-ELM: optimally pruned extreme learning machine, IEEE Trans. Neural Networks 21 (1) (2010) 158–162.
[32] Y. Lan, Y.C. Soh, G.B. Huang, Constrctive hidden nodes selection of extreme learning machine for regression, Neurocomputing 73 (2010) 3191–3199.

Xi-zhao Wang, born in 1963. Ph.D. supervisor. He is a professor of the College of Mathematics and Computer Science, Hebei University. He received his Ph.D.degree in computer science from Harbin Institute of Technology in 1998. He is an editor-in-chief for the International Journal of Machine Learning and Cybernetics, and associated editor for Information Sciences, IEEE Transactions on SMC Part (B), and International Journal of Pattern Recognition and Artificial Intelligence, respectively. His main research interests include machine learning and computational intelligence, neural networks, fuzzy measures and fuzzy integral.

Qing-yan Shao, born in 1987. He is a M.S. degree candidate in computer application technology, College of Mathematics and Computer Science, Hebei University. His research interests are in the area of machine learning and pattern recognition.

Qing Miao, born in 1986. She is a M.S. degree candidate in computer application technology, College of Mathematics and Computer Science, Hebei University. Her research interests are in the area of machine learning and pattern recognition.

Jun-hai Zhai, born in 1964. Ph.D. He is an associate professor of the College of Mathematics and Computer Science, Hebei University. He received his M.S. Degree in Computing Mathematics from Lanzhou University in June 2000, and Ph.D. Degree in Optical Engineering from Hebei University in June 2010, respectively. His research interests are in the area of machine learning and computational intelligence, rough sets, pattern recognition and wavelet analysis.