Neural Comput & Applic (2013) 22:521–529
DOI 10.1007/s00521-011-0804-2

Model selection of extreme learning machine based on multi-objective optimization

Wentao Mao · Mei Tian · Xizheng Cao · Jiucheng Xu

Received: 3 August 2011 / Accepted: 29 December 2011 / Published online: 12 January 2012

Abstract As a novel learning algorithm for single-hidden-layer feedforward neural networks, extreme learning machines (ELMs) have been a promising tool for regression and classification applications. However, it is not trivial for ELMs to find the proper number of hidden neurons due to the nonoptimal input weights and hidden biases. In this paper, a new model selection method of ELM based on multi-objective optimization is proposed to obtain compact networks with good generalization ability. First, a new leave-one-out (LOO) error bound of ELM is derived, and it can be calculated with negligible computational cost once the ELM training is finished. Furthermore, the hidden nodes are added to the network one-by-one, and at each step, a multi-objective optimization algorithm is used to select optimal input weights by minimizing this LOO bound and the norm of output weight simultaneously in order to avoid over-fitting. Experiments on five UCI regression data sets are conducted, demonstrating that the proposed algorithm can generally obtain better generalization performance with more compact network than the conventional gradient-based back-propagation method, original ELM and evolutionary ELM.

Keywords Extreme learning machine · Multi-objective optimization · Leave-one-out error bound

W. Mao (✉) · X. Cao · J. Xu
College of Computer and Information Technology,
Henan Normal University, Xinxiang 453007, Henan, China
e-mail: maowt.mail@gmail.com

M. Tian
Management Institute, Xinxiang Medical University,
Xinxiang 453003, Henan, China

1 Introduction

As an important branch of neural network, extreme learning machines (ELMs), introduced by Huang et al. [1], plays an important role in the fields of pattern recognition and regression. ELMs extend single-hidden layer feedforward neural network (SLFN) to "generalized" hidden node case, and different from the conventional gradient-based learning algorithms, for example, back-propagation (BP) methods, ELMs can analytically determine the output weights via a simple matrix inversion procedure as soon as the input weights and hidden layer biases are generated randomly and then obtain good generalization performance with very high learning speed [2]. Recently, ELMs have demonstrated impressive performance in solving a wide range of real-world problems [3–5].

However, there may exist some redundant hidden nodes in ELM in many cases. It is very important to select most significant hidden nodes which achieve most contribution to improve the generalization ability of ELM. This problem is also called model selection of ELM [6]. From the incremental learning point of view, Feng et al. [7] proposed an error minimized extreme learning machine which measured the residual error caused by adding a new added hidden node in an incremental manner. On the basis of this research, Lan et al. [8] applied a random search method to select most important nodes. On the contrary, some researches adopted forward methods to select most significant nodes. Typically, Li et al. [9] used orthogonal least squares to measure the newly added neuron's contribution without repeating the whole training process. In spite of little computational costs, this method tends to fall into local minima. Lan et al. [10] added a refinement stage that used leave-one-out (LOO) error to evaluate the neuron's significance in each backward step. In a sense, these

methods are similar to feature selection or variable ranking.

The nonoptimal or unnecessary input weights and hidden biases could cause redundant hidden neurons. If the input weights, biases and the number of hidden nodes can be optimized uniformly, a more efficient ELM model will be obtained. As a pioneering research, Zhu et al. [11] proposed a new evolutionary ELM(E-ELM). This algorithm proposed a modified form of differential evolutionary algorithm to optimize the input weights and hidden biases while increasing hidden node sequentially. However, only RMSE on validation set is used as the fitness value, and $\|\boldsymbol{\beta}\|$ that determines generalization performance of ELM is only used to separate two individuals with similar fitness values. As a result, the fitness function in [11] cannot provide an accurate statement about the generalization ability of ELM, and the optimal number of hidden neurons could not be determined definitively.

According to the discussion above, there are two important factors for model selection of ELM: accurate evaluation of generalization performance and a proper selection strategy which can take generalization error and $\|\boldsymbol{\beta}\|$ into consideration simultaneously. The evaluation of generalization performance should not only be computationally inexpensive but also cover the whole training data set. However, to our best knowledge, the generalization ability of ELM is mainly estimated via validation error [11], which is perhaps inaccurate and only uses part of data set. Another research [10] utilized the block matrix inversion to derive a LOO bound, but it is merely suited to backward iteration.

This paper tries to supply a simple and efficient solution for these two factors. Fortunately, Huang et al. [12] extended ELM to support vector network, which provides a new idea for us: calculate the LOO bound of ELM by introducing the method used in support vector machine (SVM). Therefore, this paper first derives a new leave-one-out error bound of ELM from the standard optimization method point of view. This bound is only a byproduct of single training procedure and hence needs very little computational costs. Furthermore, from the perspective of evolutionary optimization, the LOO bound of ELM and $\|\boldsymbol{\beta}\|$ are minimized simultaneously to determine optimal input weights and biases while adding sequentially a hidden neuron. This work is mainly inspired by [11–13]. The rest of this paper is organized as follows. In Sect. 2, a brief review to ELM is given. In Sect. 3, a theoretical derivation about ELM LOO error bound is presented. Section 4 further proposes a new ELM model selection method based on multi-objective optimization. Section 5 is devoted to experiments, followed by a conclusion in the last section.

2 Brief introduction of ELM

As the theoretical foundation of ELM, Huang and Babri [14] studied the learning performance of SLFN on small-size data set and found that SLFN with at most N hidden neurons can learn N distinct samples with zero error by adopting any bounded nonlinear activation function. On the basis of this concept, Huang et al. [2] pointed out that ELM can obtain good generalization performance with high learning speed via a simple matrix inversion procedure once the input weights and hidden layer biases are generated randomly. Here, a brief summary of ELM is provided.

Given a set of *i.i.d* training samples $\{(\mathbf{x}_1, \mathbf{t}_1), \ldots, (\mathbf{x}_N, \mathbf{t}_N)\} \subset \mathbb{R}^n \times \mathbb{R}^m$, standard SLFNs with $\tilde{N}$ hidden nodes are mathematically formulated as Huang et al. [2]:

$$\sum_{i=1}^{\tilde{N}} \beta_i g_i(\mathbf{x}_j) = \sum_{i=1}^{\tilde{N}} \beta_i g_i(\mathbf{w}_i \cdot \mathbf{x}_j + b_i) = \mathbf{o}_j, \quad j = 1, \ldots, N \tag{1}$$

where $g(x)$ is activation function, $\mathbf{w}_i = [w_{i1}, w_{i2}, \ldots, w_{in}]^T$ is input weight vector connecting input nodes and the ith hidden node, $\boldsymbol{\beta}_i = [\beta_{i1}, \beta_{i2}, \ldots, \beta_{im}]^T$ is the output weight vector connecting output nodes and the ith hidden node, b_i is bias of the ith hidden node. Huang et al. [2] has rigorously proved that for N arbitrary distinct samples and any $(\mathbf{w}_i, b_i)$ randomly chosen from $\mathbb{R}^n \times \mathbb{R}^m$ according to any continuous probability distribution, the hidden layer output matrix $\mathbf{H}$ of a standard SLFN with N hidden nodes and is invertible and $\|\mathbf{H}\boldsymbol{\beta} - \mathbf{T}\| = 0$ with probability one if the activation function $g : \mathbb{R} \mapsto \mathbb{R}$ is infinitely differentiable in any interval. Then given $(\mathbf{w}_i, b_i)$, training a SLFN equals finding a least-squares solution of the following equation (2):

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

where:

$$\mathbf{H}(\mathbf{w}_1, \ldots, \mathbf{w}_{\tilde{N}}, b_1, \ldots, b_{\tilde{N}}, \mathbf{x}_1, \ldots, \mathbf{x}_{\tilde{N}}) = \begin{bmatrix} g(\mathbf{w}_1 \cdot \mathbf{x}_1 + b_1) & \cdots & g(\mathbf{w}_{\tilde{N}} \cdot \mathbf{x}_1 + b_{\tilde{N}}) \\ \vdots & \cdots & \vdots \\ g(\mathbf{w}_1 \cdot \mathbf{x}_N + b_1) & \cdots & g(\mathbf{w}_{\tilde{N}} \cdot \mathbf{x}_N + b_{\tilde{N}}) \end{bmatrix}_{N \times \tilde{N}}$$

$$\boldsymbol{\beta} = [\beta_1, \ldots, \beta_{\tilde{N}}]^T$$

$$\mathbf{T} = [\mathbf{t}_1, \ldots, \mathbf{t}_N]^T$$

Considering most cases that $\tilde{N} \ll N$, $\boldsymbol{\beta}$ cannot be computed through the direct matrix inversion. Therefore, Huang et al. [2] calculated the *smallest norm* least-squares solution of (2):

$$\hat{\boldsymbol{\beta}} = \mathbf{H}^{\dagger}\mathbf{T} \tag{3}$$

where $\mathbf{H}^{\dagger}$ is the Moore-Penrose generalized inverse of matrix $\mathbf{H}$ [2]. According to Bartlett's theory [20] that the generalization performance of SLFN will be improved by minimizing training errors as well as the norm of output weights, $\hat{\boldsymbol{\beta}}$ can theoretically pledge the generalization ability of SLFN.

Based on the above analysis, Huang et al. [2] proposed ELM whose framework is as follows:

Step 1. Randomly generate input weight and bias $(\mathbf{w}_i, b_i), \quad i = 1, \ldots, \tilde{N}$.

Step 2. Compute the hidden layer output matrix $\mathbf{H}$.

Step 3. Compute the output weight $\hat{\boldsymbol{\beta}} = \mathbf{H}^{\dagger}\mathbf{T}$.

Therefore, the output of SLFN can be calculated by $(\mathbf{w}_i, b_i)$ and $\hat{\boldsymbol{\beta}}$:

$$f(\mathbf{x}_j) = \sum_{i=1}^{\tilde{N}} \hat{\beta}_i g_i(\mathbf{w}_i \cdot \mathbf{x}_j + b_i) = \hat{\boldsymbol{\beta}} \cdot h(\mathbf{x}_j)$$

3 Leave-one-out error bound of ELM

Huang et al. [12] showed that ELM can be linearly extended to SVM with less optimization constraints and simpler random kernel. Cawley and Talbot [13] used block matrix inversion to construct LOO bound for least-squares SVM(LS-SVM). Inspired by [12, 13], in this section we will first build up a regularization formulation for ELM and then extend the approach of model selection in [13] to ELM. Although the work is similar to [13] to some extent, this bound is still specific for ELM.

As stated above, original ELM seeks the solution with zero training error, for example, $\|\boldsymbol{\beta} \cdot h(\mathbf{x}_i) - t_i\| = 0$. However, considering the acceptable minimal training error [13], for example, $\boldsymbol{\beta} \cdot h(\mathbf{x}_i) = t_i - \varepsilon_i$, the objective of ELM can be rewritten as minimizing training error as well as output weights' norm [12]:

$$\min \sum_{i=1}^{N} \|\boldsymbol{\beta} \cdot h(\mathbf{x}_i) - t_i\| \tag{4}$$

and

$$\min \|\boldsymbol{\beta}\| \tag{5}$$

From the standard optimization method point of view, (4) and (5) can be combined into the following regularization formulation:

$$\begin{aligned} \min \quad & \tfrac{1}{2}\|\boldsymbol{\beta}\|^2 + C\sum_{i=1}^{N}\varepsilon_i^2 \\ \text{s.t.} \quad & \varepsilon_i = t_i - \boldsymbol{\beta} \cdot h(\mathbf{x}_i), \quad i = 1, \ldots, N \end{aligned} \tag{6}$$

where C is regularization parameter which controls the tradeoff between the training error and generalization ability. Note that (6) is the regularized least-squares form of neural network [15, 16] with random kernel [17] employed. To a certain extent, (6) is similar to LS-SVM except $h(\mathbf{x}_i)$ is generated randomly and the bias b in LS-SVM is not required here.

After applying the method of Lagrange multipliers to (6), the regularized loss function is obtained as follows:

$$L = \frac{1}{2}\|\boldsymbol{\beta}\|^2 + C\sum_{i=1}^{N}\varepsilon_i^2 - \sum_{i=1}^{N}\alpha_i\{\boldsymbol{\beta} \cdot h(\mathbf{x}_i) + \varepsilon_i - t_i\} \tag{7}$$

where $\boldsymbol{\alpha} = \{\alpha_1, \alpha_2, \ldots, \alpha_N\}$ is a vector of Lagrange multipliers. After calculating the partial derivative of the parameters, respectively, the following equations are obtained:

$$\frac{\partial L}{\partial \boldsymbol{\beta}} = 0 : \boldsymbol{\beta} = \sum_{i=1}^{N}\alpha_i h(\mathbf{x}_i) \tag{8}$$

$$\frac{\partial L}{\partial \varepsilon_i} = 0 : \alpha_i = 2C\varepsilon_i, \quad \forall i \in \{1, 2, \ldots, N\} \tag{9}$$

After substituting (8) and (9) into equality constraint in (6), the following equation can be obtained:

$$\sum_{j=1}^{N}\alpha_j h(\mathbf{x}_j)h(\mathbf{x}_i) + \frac{\alpha_i}{2C} = t_i, \quad \forall i \in \{1, 2, \cdots, N\} \tag{10}$$

In [12], an valid ELM kernel has be defined as $K(\mathbf{x}_i, \mathbf{x}_j) = h(\mathbf{x}_j)h(\mathbf{x}_i)$. Therefore, the system of linear equations (10) can be rewritten as:

$$\left[\mathbf{K} + \frac{1}{2C}I\right]\boldsymbol{\alpha} = \mathbf{T} \tag{11}$$

where K is kernel matrix, and I is the $N \times N$ identity matrix.

The following derivation of LOO bound adopts the similar method of block matrix inversion in [13]. Denote by $\boldsymbol{\alpha}^{(-i)}$ the parameter vector during the ith iteration of the LOO procedure. For simplicity, the matrix on the left-hand side of (11) can be rewritten as:

$$\left[\mathbf{K} + \frac{1}{2C}I\right] = \begin{bmatrix} \mathrm{p}_{11} & \mathbf{p}_1^T \\ \mathbf{p}_1 & \mathbf{P}_1 \end{bmatrix} = \mathbf{P} \tag{12}$$

Then after removing the first training sample, the parameter $\boldsymbol{\alpha}^{(-1)} = \mathbf{P}_1^{-1}[\mathbf{t}_2, \ldots, \mathbf{t}_N]^T$. As a result, the prediction of the first sample using other samples is:

$$\begin{aligned} f(\mathbf{x}_1)^{(-1)} &= \mathbf{p}_1^T\boldsymbol{\alpha}^{(-1)} \\ &= \mathbf{p}_1^T\mathbf{P}_1^{-1}[\mathbf{p}_1\mathbf{P}_1]\boldsymbol{\alpha}^T \\ &= \mathbf{p}_1^T\mathbf{P}_1^{-1}\mathbf{p}_1\alpha_1 + \mathbf{p}_1^T[\alpha_2, \cdots, \alpha_N]^T \end{aligned} \tag{13}$$

From (11), we have:

$$\mathbf{t}_1 = \mathrm{p}_{11}\alpha_1 + \mathbf{p}_1^T[\alpha_2, \cdots, \alpha_N]^T \tag{14}$$

Substituting (14) into (13), we have:

$$\mathbf{t}_1 - f(\mathbf{x}_1)^{(-1)} = \alpha_1\left(\mathrm{p}_{11} - \mathbf{p}_1^T\mathbf{P}_1^{-1}\mathbf{p}_1\right) \tag{15}$$

The block matrix inversion lemma [13] shows that the first element of the inversion of (12) is $\left(\mathrm{p}_{11} - \mathbf{p}_1^T\mathbf{P}_1^{-1}\mathbf{p}_1\right)^{-1}$. Without loss of generality, the LOO error of the ith iteration is:

$$e_i^{(-i)} = \mathbf{t}_i - f(\mathbf{x}_i)^{(-i)} = \frac{\alpha_i}{\mathbf{P}_{ii}^{-1}} \tag{16}$$

Applying the predicted residual sum of squares(PRESS) statistics [18], ELM LOO bound is:

$$\mathrm{LOO}_{\mathrm{ELM}} = \frac{1}{N}\sum_{i=1}^{N}\left[e_i^{(-i)}\right]^2 \tag{17}$$

Equation (16) shows that the LOO error bound can be calculated simply from model parameter $\boldsymbol{\alpha}$ and the principal diagonal of inversion of (12), which means the generalization performance of ELM can be efficiently evaluated once a single training procedure ended.

It is worth noting that, although (16) has the same literal form with the LOO bound of LS-SVM in [13], there still exist two apparent differences: $\mathbf{P}$ in (12) is more simple than the corresponding matrix in [13] (without considering bias parameter b), and the kernel matrix $\mathbf{K}$ can be computed directly from inner product between any two nonlinear piecewise continuous functions whose tunable parameters can be randomly initialized according to any continuous probability distribution [12]. Therefore, (17) applies especially to ELM. According to Kohavi [19], ELM LOO bound is approximately unbiased and its computational complexity is $O(N)$ [13].

4 Model selection based on multi-objective optimization

Model selection can be essentially considered as a global optimization problem where the evaluation of generalization ability performs as fitness function. Therefore, one key issue of model selection is selection strategy. Different from the common researches [7–10], Zhu et al. [11] presents a new selection strategy that combines the optimization of the input weights and biases with selection of hidden nodes. This strategy can reduce the number of hidden nodes caused by nonoptimal or unnecessary input weights and biases. On the other hand, as pointed out by [13], LOO bounds tend to cause over-fitting. According to [20], the smaller the norm of output weights is, the better generalization performance the ELM tends to have. Therefore, if the LOO bound proposed in Sect. 3 and the norm of output weights are simultaneously minimized, the over-fitting can be avoided. On the basis of this analysis, a novel multi-objective optimization algorithm, named multi-objective comprehensive learning PSO (MOCLPSO) [21], is introduced to find the optimal input weights, biases for each added hidden neuron. The target of this research is to get good generalization performance for ELM with compact network.

4.1 Optimization strategy

To a certain extent, the LOO bound and the norm of output weights are conflicting, which will be verified by experimental results in Sect. 5. That indicates values of LOO bound can be increased merely at the cost of worse values of output weights' norm. A solution is called nondominated or parieto-optimal if no objective can be improved without getting worse at least one of the others [21]. Due to the ability of finding a set of nondominated solutions instead of single global best individual as in the single objective problem, multi-objective optimization is chosen as selection strategy of this research. In particular, PSO is chosen as the basic algorithm because of its advantages such as simple structure, few parameters and high convergence speed. More importantly, because PSO can search solutions in multiple inner parallel directions, it is easy to be extended for multi-objective problems.

Here MOCLPSO [21] is chosen as practical realization. A brief summary of MOCLPSO is supplied. On the basis of CLPSO that can effectively use all particles' historical best information, and MOCLPSO adopts a crowding distance-based archive maintenance strategy to update the particles' velocity and position. Therefore, better diversity of the swarm is obtained and leads to avoiding premature convergence efficiently. Its basic idea is described as follows [21]: MOCLPSO uses an external archive B to store the set of nondominated solutions obtained at each generation, and uses an archive A to store the best solutions found so far. Then solutions in two sets are compared one by one. If the solution x in B is dominated by a member of A, reject x; If x dominates a subset C of A, then $\mathrm{A} = \mathrm{A}\backslash\mathrm{C}$, $\mathrm{A} = \mathrm{A} \cup \{x\}$. The detailed derivation of this algorithm can be found in [21].

4.2 Fitness function

As stated above, there are two conflicting objectives of optimization. Each individual has two fitness values: the LOO bound shown in (17), and the norm of output weights $\|\boldsymbol{\beta}\|$ in (5).

4.3 Algorithm description

Same with [11], the individual in the population consists of a set of input weights and hidden biases:

$$\theta = \left[\omega_{11}, \omega_{12}, \ldots, \omega_{1\tilde{N}}, \omega_{21}, \ldots, \omega_{2\tilde{N}}, \ldots, \omega_{N1}, \ldots, \omega_{N\tilde{N}}, b_1, b_2, \ldots, b_{\tilde{N}}\right] \quad (18)$$

Therefore, model selection of ELM transforms into finding the θ^0 which minimize the LOO bound and $\|\boldsymbol{\beta}\|$ simultaneously with smallest $\tilde{N}$, and can be described as follows:

$$\theta^0 = \arg\min_{\theta} E(\theta) = \arg\min_{\theta}(\mathrm{LOO}_{\mathrm{ELM}}(\theta), \|\boldsymbol{\beta}\|) \quad (19)$$

The algorithm is described in three steps. Step 1, the population is randomly generated with one neuron. Each individual of the population is randomly initialized within the range of $[-1, 1]$. The fitness of each individual is evaluated. Once the MP generalized inverse in ELM has been done, the output weights and LOO bound can be calculated directly from (3) and (17), respectively. Step 2, MOCLPSO is applied to find the set of nondominated solutions. The one corresponding to lowest LOO bound is the preferred solution. Once the new population is generated in iteration, the fitness of each individual should be generated again. Step 3, add sequentially the hidden neuron, and go to step 1. After a set of solutions with different number of neurons has been obtained, the best input weights and biases with the best generalization performance are selected as the final one. The flowchart of the proposed model selection method is depicted in Fig. 1.

5 Experimental results

In this section, the proposed model selection method is compared with the traditional BP neural network, ELM [22] and E-ELM [11]. Specially, Levenberg–Marquardt algorithm, called LM, is adopted to train the BP neural network. LM has the fastest training speed among the common BP algorithms and has been implemented in the neural network box of MATLAB. For simplicity, the proposed model selection method in this paper is called here M-ELM. In ELM, E-ELM and M-ELM, the sigmoidal function: $g(x) = \frac{1}{1+\exp(-x)}$ is employed as activation function. The parameters of E-ELM are set as follows: population size (*NP*), *F* and *CR* is set to 200, 1, 0.8, respectively, and the maximum iteration number is 20. In order to compare the generalization error reasonably, M-ELM is set to run the same number of fitness evaluations of E-ELM. UCI regression data sets [23] are used to evaluate our method's effectiveness. All the experimental results are the mean values of 30 trails. All programs are carried out in MATLAB7.04 environment running in a Core2, 2.66 GHz CPU

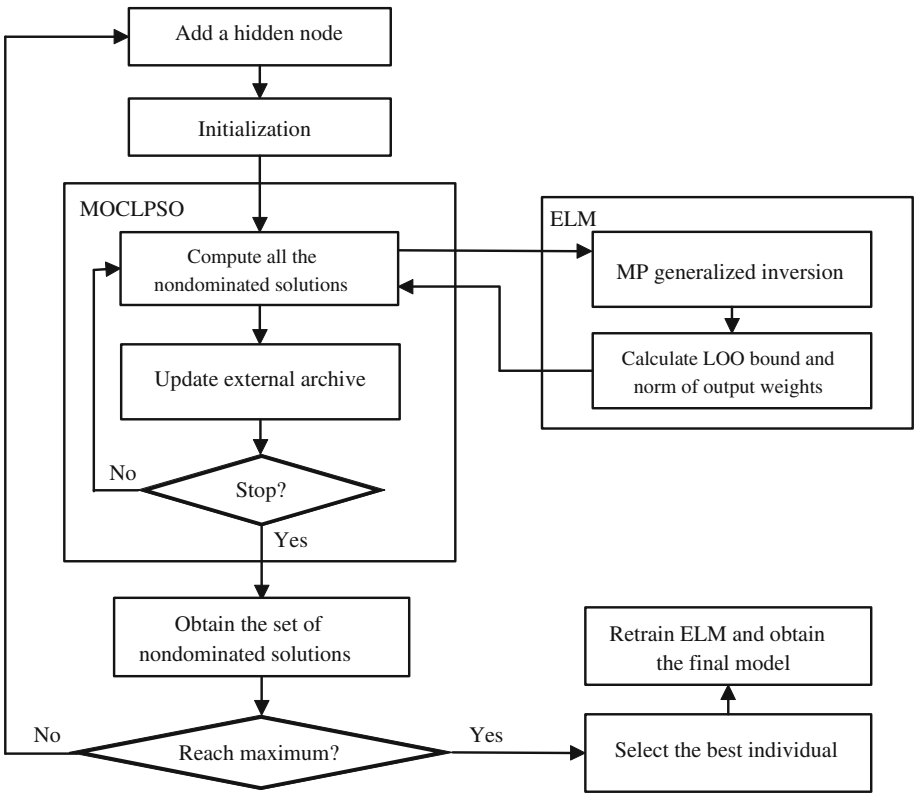

Fig. 1 Flowchart of model selection of ELM based on MOCLPSO

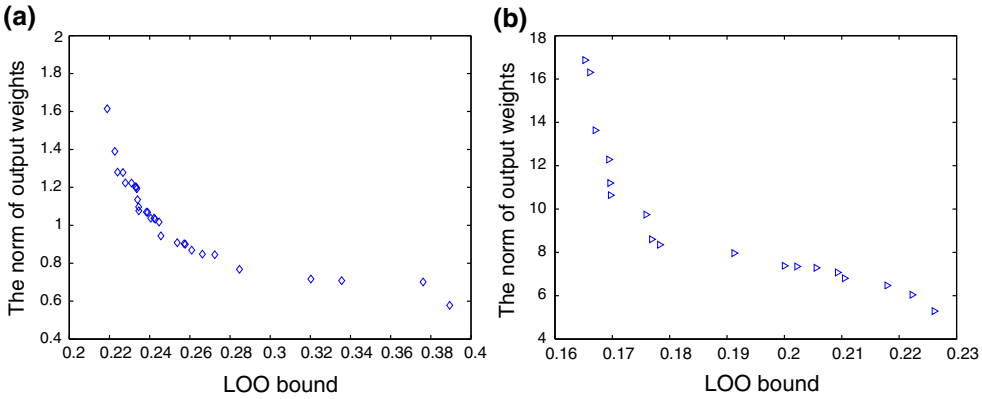

Fig. 2 Distribution of solutions obtained by M-ELM on Boston Housing data set where the number of hidden neurons is **a** 5 and **b** 35, respectively

and 3.37 GB RAM. Each of the input and output variables are rescaled linearly to the range $[-1, +1]$.

First, the existing of contradiction between the LOO bound and $\|\boldsymbol{\beta}\|$ is verified. Taking a typical UCI regression data set, Boston Housing, as an example, we apply M-ELM to find the best individuals whose distributions are shown in Fig. 2. As shown in Fig. 2, the solutions construct an *approximate* Pareto-optimal front. That means there exists apparent contradiction between two objectives even though various hidden neurons are selected. Note that "*approximate*" is probably caused by the fact that the LOO bound and $\|\boldsymbol{\beta}\|$ do not necessarily or completely contradict to each other, namely, the increase in one objective values may not lead to the decrease in another objective value in some cases. All of other numbers of hidden neurons give very similar results. Moreover, it is obvious that the number of solutions and the values of LOO bound both decreases meanwhile the number of hidden neurons increases.

Secondly, the effectiveness of M-ELM is tested. Here a noisy sinc function [24] is introduced in favor of graphical comparison. The training set including 100 samples $\{(x_i, y_i)\}$ is generated with x_i drawn from $[-3, +3]$ uniformly, and $y_i = \sin(\pi x_i)/(\pi x_i) + e_i$ where e_i is a Gaussian noise term with zero mean and a variance of 0.1. We take 70% of the data for training and the remaining for test. Then M-ELM and ELM are applied on this data set, respectively, and the corresponding decision curves are plotted in Fig. 3. As a priori knowledge, the sinc curve plotted by red line in Fig. 3 has best generalization performance. As shown in Fig. 3, the decision curve obtained by M-ELM is more close to the sinc curve than ELM in the section $[-3,-1]$ and nearby the peak. Although in other sections M-ELM and ELM have similar performance which results in somewhat unclear comparison in Fig. 3, the curve of M-ELM is more smooth than ELM, and at few points ELM tends to overfit. That implies M-ELM can improve the generalization performance of ELM. It has also been proved by the numerical results that the training error(RMSE) of M-ELM is 0.0280 while that of ELM is 0.0390, and the RMSE on test set of M-ELM is 0.0242 while that of ELM is 0.0411.

The parameter C in (6) is the only tunable parameter in M-ELM. To evaluate the influence of C, we fix the number of hidden neurons and calculate the RMSE on test set of noisy sinc function with different values of C. Figure 4 illustrates the variation tendency of generalization error with different values of C(in log scale). As shown in Fig. 4, when we choose 10 and 15 hidden neurons, the RMSEs on test set both vary in a very little fluctuation with large values of C. Therefore, for simplicity, we will set C as 100 in the following experiments.

Finally, we compare M-ELM with E-ELM, ELM and LM on five UCI regression data sets [23]. For ELM and LM, we gradually increase the hidden neurons in each simulation and select the best results as the final one. Some statistics of these five data sets are listed in Table 1. These data sets are divided into training, validation and test set. Note that only E-ELM needs validation set to calculate the validation RMSE as fitness function. To get a reliable comparison, two groups of experiments are conducted.

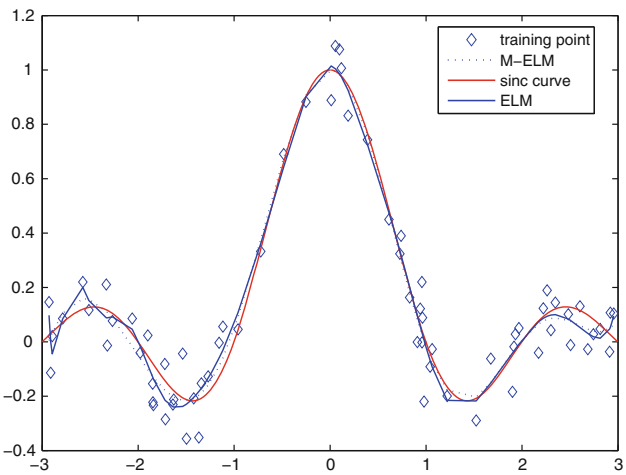

Fig. 3 Decision curves obtained by M-ELM and ELM on noisy sinc data set

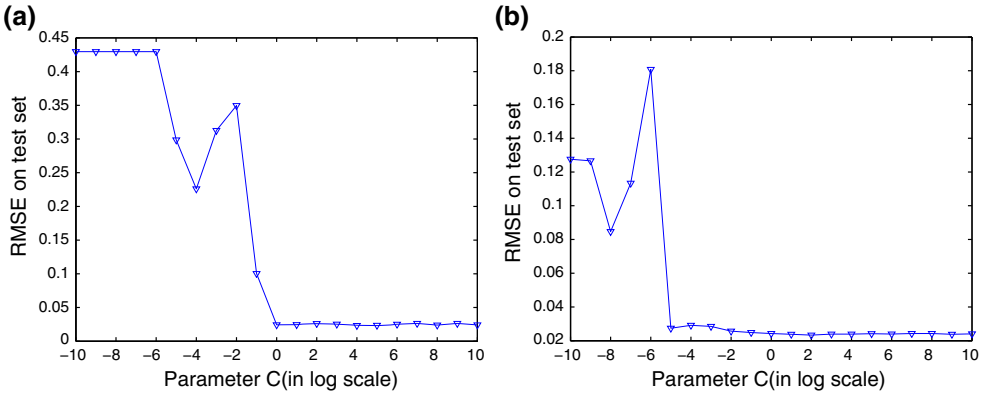

Fig. 4 RMSE on test set of noisy sinc function with different values of C where the number of hidden neurons is set to **a** 10 and **b** 15, respectively

First, for each data set, a fixed training/test split is produced, and the mean of RMSEs on training and test sets of 30 trials is calculated. Second, the experiments are implemented 30 times with random partitions of data set. Here we calculate the mean and standard error of 30 RMSEs. The ration of fixed and random partition are both same as the statistics listed in Table 1. In each trial, the training and test set are absolutely same for all four algorithms. Experimental results are listed in Tables 2 and 3, respectively. It is worthy to note that the simulation results in Table 3 which are calculated on a number of randomly partitioned data sets are more accurate and preferable than the results in Table 2 . Here we conduct the experiments on fixed split just to observe the comparative results at the microlevel.

From Table 2, M-ELM greatly outperforms the other three algorithms with fixed split of all five data sets except *Mpg* data set in terms of test error and gets a little bit improvement on *Mpg* data set. And from Table 3, M-ELM reduces the test error on all five data set with 30 random splits. The apparent reduction in test error suggests that the LOO bound can provide an accurate evaluation of generalization ability rather than the RMSE on validation set used as fitness function in [11], and then M-ELM using the norm of output weights can effectively prevent the overfitting caused by LOO bound. As a result, the generalization performance of ELM is greatly improved by the proposed model selection method. Note that the comparison in Table 2 is more apparent than in Table 3. We guess it is caused by the different split of data set. We also find that LM and ELM sometimes obtain very low training error, and M-ELM and E-ELM both get lower test error than LM and ELM on most data sets. It can be drawn a conclusion that LM and ELM will lead to over-fitting in some cases, and model selection methods such as M-ELM and E-ELM will prevent it via introducing the accurate evaluation of generalization ability. Note that ELM also yields a lower test error than LM.

Here the network architecture is also checked. From Table 2, we could not find the regular pattern of neuron's number for a fixed split of data set. But from Table 3, M-ELM tends to get smaller number of hidden neurons than ELM except *Triazines* data set. The comparative results demonstrate that M-ELM can exclude the unnecessary hidden neurons and get optimal input weights and biases for the resulting compact network. The probable reason is that the inaccuracy of fitness function in [11], maybe, leads to imprecise hidden neurons. Obviously, E-ELM also get smaller number of hidden nodes than LM and ELM in general, which has been proved in [11].

Moreover, the computational costs of all four algorithms are tested. From Tables 2 and 3, we find that M-ELM needs a little bit more training time than E-ELM in most cases. It is because that MOCLPSO used in M-ELM needs more operations than differential evolution algorithm in E-ELM. Although M-ELM is somewhat computationally expensive, the training time of M-ELM is still reasonable and acceptable. In addition, we also find that the number of hidden neurons can affect the training time greatly. Due to (18), the number of hidden neurons determines the individual in M-ELM and E-ELM and further influences the computational cost of whole algorithm. In spite of many hidden neurons required, ELM only needs very little training time. Note that the training time of M-ELM, E-ELM and ELM are all much less than LM even though it is one of the fastest gradient-based BP network algorithms.

Table 1 Specifications of five UCI regression data sets

Name	Attributes	Number of observations		
		Training	Validation	Test
Housing	13	354	76	76
Pyrim	27	51	11	12
Triazines	60	130	26	30
Mpg	7	274	59	59
Bodyfat	14	176	38	38

Table 2 Comparison between M-ELM, LM, ELM and E-ELM with a fixed split of data sets

Problem	Algorithm	Training time (s)	RMSE		Hidden
			Training	Test	
Housing	LM	187.34	2.8716	3.0298	58
	ELM	0.0718	2.5247	3.3777	76
	E-ELM	22.658	3.3284	2.6539	43
	M-ELM	34.742	0.9683	0.9097	47
Pyrim	LM	1023.8	0.0113	0.6764	85
	ELM	0.1463	1.24E–15	0.1622	96
	E-ELM	2.3157	0.1176	0.0858	12
	M-ELM	2.8032	0.0227	0.0133	7
Triazines	LM	341.73	0.1146	0.6271	35
	ELM	0.0479	0.0501	0.5864	43
	E-ELM	4.0071	0.1351	0.1341	12
	M-ELM	25.179	0.0382	0.0349	32
Mpg	LM	384.16	1.8566	3.4571	43
	ELM	0.0625	2.3799	2.9755	47
	E-ELM	16.174	2.7830	2.5265	27
	M-ELM	33.53	2.4885	2.2617	23
Bodyfat	LM	339.75	6.18e–5	0.0561	53
	ELM	0.4421	0.0007	0.0093	41
	E-ELM	9.6039	0.0027	0.0083	30
	M-ELM	24.874	3.49E–05	0.0026	27

Table 3 Comparison between M-ELM, LM, ELM and E-ELM with 30 random splits of data sets

Problem	Algorithm	Training time (s)	RMSE		Hidden
			Training	Test	
Housing	LM	535.22	2.6451 (1.14e−1)	4.3415 (8.72e−1)	61
	ELM	0.0343	2.8217 (1.50e−1)	4.0524 (6.77e−1)	72
	E-ELM	30.096	3.2456 (1.41e−1)	3.9861 (6.15e−1)	56
	M-ELM	23.514	1.5058 (1.98e−1)	1.5635 (5.40e−1)	18
Pyrim	LM	641.36	0.0345 (1.23e−3)	0.1562 (5.49e−2)	62
	ELM	0.0468	0.0818 (1.03e−2)	0.0982 (4.93e−2)	51
	E-ELM	2.1421	0.0383 (2.72e−3)	0.0949 (6.86e−2)	9
	M-ELM	3.0296	0.0775 (2.85e−3)	0.0650 (3.12e−3)	5
Triazines	LM	323.34	0.0822 (5.32e−3)	0.3584 (6.34e−2)	46
	ELM	0.0625	0.1373 (8.99e−3)	0.1442 (2.73e−2)	58
	E-ELM	21.698	0.1282 (1.12e−2)	0.1404 (2.42e−2)	30
	M-ELM	39.787	0.1156 (1.26e−2)	0.1259 (2.57e−2)	36
Mpg	LM	1046.8	2.3646 (1.29e−1)	2.6945 (4.56e−1)	49
	ELM	0.0732	2.4852 (1.38e−1)	2.4941 (3.69e−1)	54
	E-ELM	20.473	2.3946 (1.84e−1)	2.5475 (4.81e−1)	45
	M-ELM	29.136	2.3701 (1.71e−1)	2.2887 (2.33e−1)	31
Bodyfat	LM	115.23	0.0019 (3.45e−4)	0.0067 (4.46e−3)	43
	ELM	0.3182	0.0028 (8.42e−4)	0.0045 (2.28e−3)	55
	E-ELM	11.807	0.0022 (4.95e−4)	0.0042 (2.65e−3)	29
	M-ELM	19.509	0.0022 (9.23e−4)	0.0034 (2.39e−3)	13

Standard error of 30 RMSEs is listed in bracket

6 Conclusions

This paper addresses the problem of model selection for ELM from the optimization perspective. Different from current researches of ELM, this paper tries to determine the optimal network architecture and input weights at the same time. The key issue of model selection is how to evaluate generalization ability in an accurate and computationally inexpensive manner. To solve this problem, we first propose an ELM LOO bound. This bound can be calculated once a single training procedure ends and hence is selected as a minimization objective. To prevent the over-fitting caused by LOO bound, we introduce the norm of output weights as another objective and utilize a multi-objective optimization algorithm, MOCLPSO, to minimize these two objectives simultaneously. In obtained nondominant solutions, we choose the one leading to lowest generalization error as best solution. Experimental results on five UCI data sets show that the proposed model selection method has faster learning speed than traditional BP network and can greatly improve the generalization performance than other methods including LM, ELM and E-ELM.

Acknowledgments This work was supported by the National Natural Science Foundation of China (60873104). We also thank the author Suganthan of [21] for providing implementation of MOCLPSO.

References

1. Huang GB, Zhu QY, Siew C (2004) Extreme learning machine: a new learning scheme of feedforward neural networks. In: Proceedings of International Joint Conference on Neural Networks (IJCNN2004), vol 2, pp 25–29
2. Huang GB, Zhu QY, Siew C (2006) Extreme learning machine: theory and applications. Neurocomputing 70:489–501
3. Chen FL, Ou TY (2011) Sales forecasting system based on gray extreme learning machine with taguchi method in retail industry. Expert Syst Appl 38:1336–1345
4. Minhas R, Baradarani A, Seifzadeh S, Wu QMJ (2010) Human action recognition using extreme learning machine based on visual vocabularies. Neurocomputing 73:1906–1917
5. Mohammed A, Wu QMJ, Sid-Ahmed M (2010) Application of wave atoms decomposition and extreme learning machine for fingerprint classification. Lect Notes Comput Sci 6112:246–256
6. Huang GB, Wang D, Lan Y (2011) Extreme learning machines: a survey. Int J Mach Learn Cybern 2:107–122
7. Feng GR, Huang GB, Lin QP, Gay R (2009) Error minimized extreme learning machine with growth of hidden nodes and incremental learning. IEEE Trans Neural Netw 20:1352–1357
8. Lan Y, Soh Y, Huang GB (2010) Random search enhancement of error minimized extreme learning machine. In: Proceedings of European Symposium on Artificial Neural Networks (ESANN 2010), pp 327–332
9. Li K, Huang GB, Ge SS (2010) Fast construction of single hidden layer feedforward networks. In: Rozenberg G, Bäck T, Kok JN (eds) Handbook of Natural Computing. Springer, Berlin
10. Lan Y, Soh Y, Huang GB (2006) Two-stage extreme learning machine for regression. Neurocomputing 73:3028–3038
11. Zhu QY, Qin AK, Suganthan P, Huang GB (2005) Evolutionary extreme learning machine. Pattern Recognit 38:1759–1763
12. Huang GB, Ding XJ, Zhou H (2010) Optimization method based extreme learning machine for classification. Neurocomputing 74:155–163
13. Cawley G, Talbot N (2007) Preventing over-fitting in model selection via bayesian regularisation of the hyper-parameters. J Mach Learn Res 8:841–861
14. Huang GB, Babri H (1998) Upper bounds on the number of hidden neurons in feedforward networks with arbitrary bounded nonlinear activation functions. IEEE Trans Neural Netw 9:224–229
15. Girosi F, Jones M, Poggio T (1995) Regularization theory and neural network architectures. Neural Comput 7:219–269
16. Evgeniou T, Pontil M, Poggio T (2000) Regularization networks and support vector machines. Adv Comput Math 13:1–50
17. Frénay B, Verleysen M (2010) Using SVMs with randomised feature spaces: an extreme learning approach. In: Proceedings of The 18th European Symposium on Artificial Neural Networks (ESANN 2010), pp 315–320
18. Myers R (1990) Classical and modern regression with applications. Duxbury Press, CA
19. Kohavi R (1995) A study of cross-validation and bootstrap for accuracy estimation and model selection. In: Proceedings of the Fourteenth International Conference on Artificial Intelligence (IJCAI), pp 1137–1143
20. Bartlett P (1998) The sample complexity of pattern classification with neural networks: the size of the weights is more important than the size of the network. IEEE Trans Inf Theory 44:525–536
21. Huang VL, Suganthan P, Liang JJ (2006) Comprehensive learning particle swarm optimizer for solving multiobjective optimization problems. Int J Intell Syst 21:209–226
22. Zhu QY, Huang GB (2004) Source codes of ELM algorithm. In: http://www.ntu.edu.sg/home/egbhuang/, School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore
23. Newman D, Hettich S, Blake C (1998) UCI Repository of machine learning databases. In: http://www.ics.uci.edu/mlearn/MLRepository.html, Department of Information and Computer Science, University of California, Irvine, CA
24. Wang G, Yeung D, Lochovsky F (2008) A new solution path algorithm in support vector regression. IEEE Trans Neural Netw 19:1753–1767