Journal of Zhejiang University-SCIENCE C (Computers & Electronics)
ISSN 1869-1951 (Print); ISSN 1869-196X (Online)
www.zju.edu.cn/jzus; www.springerlink.com
E-mail: jzus@zju.edu.cn

A pruning algorithm with $L_{1/2}$ regularizer for extreme learning machine*

Ye-tian FAN[†1], Wei WU[†‡1], Wen-yu YANG[2], Qin-wei FAN[1], Jian WANG[1]

([1]*School of Mathematical Sciences, Dalian University of Technology, Dalian 116023, China*)

([2]*College of Science, Huazhong Agricultural University, Wuhan 430070, China*)

[†]E-mail: fanyetian@mail.dlut.edu.cn; wuweiw@dlut.edu.cn

Received July 22, 2013; Revision accepted Oct. 14, 2013; Crosschecked Jan. 15, 2014

Abstract: Compared with traditional learning methods such as the back propagation (BP) method, extreme learning machine provides much faster learning speed and needs less human intervention, and thus has been widely used. In this paper we combine the $L_{1/2}$ regularization method with extreme learning machine to prune extreme learning machine. A variable learning coefficient is employed to prevent too large a learning increment. A numerical experiment demonstrates that a network pruned by $L_{1/2}$ regularization has fewer hidden nodes but provides better performance than both the original network and the network pruned by L_2 regularization.

Key words: Extreme learning machine (ELM), $L_{1/2}$ regularizer, Network pruning
doi:10.1631/jzus.C1300197 **Document code:** A **CLC number:** TP312

1 Introduction

The neural network has been extensively used in many fields, such as artificial intelligence, robots, pattern recognition, and CAD/CAM (Wu, 2003; Yan *et al.*, 2004; Xu and Zhou, 2010). The slowness and low accuracy of its learning method, however, are two of its drawbacks.

In recent years, extreme learning machine (ELM), an emergent technology which overcomes these challenges faced by other techniques such as back propagation neural networks and impulsive neural networks, has attracted the attention of more and more researchers (Soria-Olivas *et al.*, 2011; Horata *et al.*, 2013; Zhang and Ji, 2013). Specifically, the computational time for actual training of ELM often has a dramatic decline, compared to some classical methods (Hornik *et al.*, 1989; Bishop, 1995; Rasmussen and Williams, 2006).

Basically, the ELM method needs sufficient hidden nodes to guarantee its accuracy. Therefore, the number of hidden nodes or the size of the system becomes an important issue: For learning the data, a too small system with few hidden nodes will not be sufficient for the accuracy, while a too large system with too many hidden nodes will cause the learning to be very slow and have bad generalization performance. To this end, a pruning ELM method, TROP-ELM, was proposed (Miche *et al.*, 2011), which is an improvement of the optimally pruned extreme learning machine (OP-ELM) (Miche *et al.*, 2010). TROP-ELM works well in some numerical examples. It uses least angle regression (LARS), leave-one-out validation, singular value decomposition (SVD), etc., and is rather complicated to implement. Moreover, the computational time significantly increases when the numbers of samples and hidden neurons are very large.

In this paper, the $L_{1/2}$ regularization method is combined with ELM to prune ELM. Our approach is

‡ Corresponding author

* Project supported by the National Natural Science Foundation of China (No. 11171367) and the Fundamental Research Funds for the Central Universities, China

simple to implement, and very effective. Preprocessing is performed to determine the suitable number of hidden nodes before running the usual ELM. The computational time of our method does not increase very much when the numbers of samples and hidden neurons are very large. In particular, a variable learning coefficient, which involves the inverse of the gradient norm of the error function, is proposed to prevent too large a learning increment. The advantage of our $L_{1/2}$ regularization over usual L_2 regularization is shown in the numerical examples.

Regularization methods are often used as feasible approaches to pruning the networks. In general, a regularization method has the form

$$\min\left\{\frac{1}{n}\sum_{i=1}^{n} l(y_i, f(x_i)) + \lambda\|f\|_k^k\right\}, \tag{1}$$

where $l(.,.)$ is a loss function, $\{(x_i, y_i)_{i=1}^n\}$ is a data set, and λ is the regularization parameter. When f is in linear form and the loss function is square loss, $\|f\|_k$ is normally taken as the norm of the coefficient of the linear model. The popular L_0, L_1, L_2, and L_∞ regularizers can be viewed as special forms of this regularization framework, corresponding to $k = 0, 1, 2,$ and ∞, respectively (Tibshirani, 1996; Chen *et al.*, 2001; Donoho and Huo, 2001).

Xu *et al.* (2010) proposed an $L_{1/2}$ regularizer to solve the following sparse linear model:

$$\boldsymbol{Y} = \boldsymbol{X}^{\mathrm{T}}\boldsymbol{\beta} + \boldsymbol{\epsilon},\ E(\boldsymbol{\epsilon}) = \boldsymbol{0},\ \mathrm{Cov}(\boldsymbol{\epsilon}) = \sigma^2\boldsymbol{I}, \tag{2}$$

where $\boldsymbol{Y} = (Y_1, Y_2, \cdots, Y_n)^{\mathrm{T}}$ is an $n \times 1$ response vector, $\boldsymbol{X} = (\boldsymbol{X}_1, \boldsymbol{X}_2, \cdots, \boldsymbol{X}_n)^{\mathrm{T}}$ ($\boldsymbol{X}_i^{\mathrm{T}} = (x_{i1}, x_{i2}, \cdots, x_{iL}), i = 1, 2, \cdots, n$) is an input vector, and $\boldsymbol{\beta} = (\beta_1, \beta_2, \cdots, \beta_L)^{\mathrm{T}}$ is a vector of L unknown parameters. $\boldsymbol{\epsilon}$ is the random error and σ^2 is a positive constant. The $L_{1/2}$ regularization method for this problem is in the following form:

$$\hat{\boldsymbol{\beta}}_{L_{1/2}} = \arg\min_{\boldsymbol{\beta}}\left\{\frac{1}{n}\sum_{i=1}^{n}(Y_i, \boldsymbol{X}_i^{\mathrm{T}}\boldsymbol{\beta})^2 + \lambda\sum_{i=1}^{p}|\beta_i|^{\frac{1}{2}}\right\}, \tag{3}$$

where λ is the tuning parameter. The $L_{1/2}$ regularizer has been successfully applied to the variable selection problem (Xu *et al.*, 2010) and the compressive sensing problem (Xu, 2010).

Generally speaking, a problem with L_2 or L_1 regularizer leads to a convex optimization problem that is easy to solve, but it does not yield a sufficiently sparse solution. While the solution of the L_0 regularizer is the most sparse, it is a combinatory optimization problem and is difficult to solve. As shown in Xu *et al.* (2010), the $L_{1/2}$ regularizer gives a good compromise between L_0 and L_1 regularizers: It is easier to solve than the L_0 regularizer, and more sparse than the L_1 regularizer.

We observe from the convergence theorems for ELM that the number of hidden nodes has to be large enough for an ELM to approximate a given function $f(x)$ (Huang *et al.*, 2011). In practice, however, it is difficult to know how large L has to be, and too many hidden nodes will lead to high cost and bad generalization performance. Therefore, an adaptive scheme to determine the suitable number of hidden nodes for ELM is desirable.

2 Extreme learning machine

ELM is an algorithm of establishing a single hidden layer feedforward neural network (Huang and Siew, 2004; Huang *et al.*, 2004; 2006). The hidden node weights of ELM are randomly given and the output weights can be calculated by solving a linear system using the least square method. This strategy is quite different from traditional learning methods for feedforward networks.

For N pairs of input-output samples $(\boldsymbol{x}_i, \boldsymbol{o}_i) \in \mathbb{R}^n \times \mathbb{R}^m$, the ELM with L hidden nodes and activation function $G(\boldsymbol{a}_i, b_i, \boldsymbol{x})$ can be expressed as follows:

$$f_L(\boldsymbol{x}_j) = \sum_{i=1}^{L}\beta_i G(\boldsymbol{a}_i, b_i, \boldsymbol{x}_j) = \boldsymbol{o}_j,\quad j = 1, 2, \cdots, N, \tag{4}$$

where weights $\boldsymbol{a}_i \in \mathbb{R}^n$ and bias $b_i \in \mathbb{R}$ are randomly chosen, and β_i is the weight connecting the ith hidden node with the output node.

Eq. (4) can be expressed in a condensed form as

$$\boldsymbol{H}\boldsymbol{\beta} = \boldsymbol{O}, \tag{5}$$

where

$$\begin{aligned}\boldsymbol{H} &= \boldsymbol{H}(\boldsymbol{a}_1, \cdots, \boldsymbol{a}_L, b_1, \cdots, b_L, \boldsymbol{x}_1, \cdots, \boldsymbol{x}_N)\\ &= \begin{bmatrix} G(\boldsymbol{a}_1, b_1, \boldsymbol{x}_1) & \cdots & G(\boldsymbol{a}_L, b_L, \boldsymbol{x}_1)\\ \vdots & & \vdots\\ G(\boldsymbol{a}_1, b_1, \boldsymbol{x}_N) & \cdots & G(\boldsymbol{a}_L, b_L, \boldsymbol{x}_N)\end{bmatrix}_{N\times L},\end{aligned} \tag{6}$$

$$\boldsymbol{\beta}=\begin{bmatrix}\boldsymbol{\beta}_1^{\mathrm{T}}\\ \vdots\\ \boldsymbol{\beta}_L^{\mathrm{T}}\end{bmatrix}_{L\times m},\qquad \boldsymbol{O}=\begin{bmatrix}\boldsymbol{o}_1^{\mathrm{T}}\\ \vdots\\ \boldsymbol{o}_N^{\mathrm{T}}\end{bmatrix}_{N\times m}. \tag{7}$$

Using the Moore-Penrose generalized inverse $\boldsymbol{H}^{\dagger}$ of $\boldsymbol{H}$, the least squares solution of Eq. (5) can be written as

$$\hat{\boldsymbol{\beta}}=\boldsymbol{H}^{\dagger}\boldsymbol{O}. \tag{8}$$

3 Extreme learning machine with $L_{1/2}$ regularizer

To prune the network, we need to identify the unimportant weights and remove them. Naturally, we assume that the larger the norm of the weight vector of a hidden node is, the more important the role it plays. Therefore, we try to find the $\boldsymbol{\beta}$ such that

$$\hat{\boldsymbol{\beta}}=\arg\min_{\boldsymbol{\beta}}\left\{\|\boldsymbol{H}\boldsymbol{\beta}-\boldsymbol{O}\|_2^2+\lambda\|\boldsymbol{\beta}\|_{1/2}^{1/2}\right\}, \tag{9}$$

where $\boldsymbol{\beta}\in\mathbb{R}^L\times\mathbb{R}^m$. The error function with $L_{1/2}$ regularization has the following form:

$$E(\boldsymbol{\beta})=\sum_{k=1}^{N}(\boldsymbol{H}\boldsymbol{\beta}-\boldsymbol{O})_{k*}^{\mathrm{T}}(\boldsymbol{H}\boldsymbol{\beta}-\boldsymbol{O})_{k*}+\lambda\sum_{i=1}^{L}\sum_{j=1}^{m}|\beta_{ij}|^{\frac{1}{2}}, \tag{10}$$

where $(\boldsymbol{H}\boldsymbol{\beta}-\boldsymbol{O})_{k*}$ represents the kth row of the matrix $\boldsymbol{H}\boldsymbol{\beta}-\boldsymbol{O}$. It is difficult to solve the problem directly due to the involvement of the $L_{1/2}$ regularizer. So, we use the gradient descent method to minimize the error function. For $i=1,2,\cdots,L$ and $j=1,2,\cdots,m$, we calculate

$$\frac{\partial E(\boldsymbol{\beta})}{\beta_{ij}}=\sum_{k=1}^{N}2H_{ki}(\boldsymbol{H}_{k*}\boldsymbol{\beta}_{*j}-O_{kj})+\lambda\frac{\mathrm{sgn}(\beta_{ij})}{2|\beta_{ij}|^{\frac{1}{2}}}. \tag{11}$$

We denote the Jacobian matrix as

$$\Delta\boldsymbol{\beta}=\left(\frac{\partial E(\boldsymbol{\beta})}{\beta_{ij}}\right)_{L\times m},\quad i=1,2,\cdots,L,\ j=1,2,\cdots,m. \tag{12}$$

First, we randomly take a weight around zero as the initial value $\boldsymbol{\beta}^0$. Then, the weights $\boldsymbol{\beta}^n$ are updated iteratively by

$$\boldsymbol{\beta}_{i*}^{n+1}=\boldsymbol{\beta}_{i*}^{n}-\eta\frac{\Delta\boldsymbol{\beta}_{i*}^{n}}{\|\Delta\boldsymbol{\beta}_{i*}^{n}\|}. \tag{13}$$

Here, we have used a variable coefficient $\eta/\|\Delta\boldsymbol{\beta}_{i*}^{n}\|$ rather than a constant coefficient η, so as to prevent too fast increase of the weights during the learning procedure. Due to use of the $L_{1/2}$ regularizer, it is expected that the absolute values of the weights connecting relatively important hidden nodes become relatively large.

The learning scheme of our extreme learning machine with $L_{1/2}$ regularizer (ELMR) can be described as follows:

1. Randomly take a weight around zero as the initial value $\boldsymbol{\beta}^0$, and update it iteratively using Eq. (13) until many components of $\boldsymbol{\beta}^n$ become very near to zero.

2. Delete all the hidden nodes with small weights and all the weights connected to them.

3. Keep the weights connecting the remaining hidden nodes and the input layer unchanged, and run the usual ELM again to obtain the final $\boldsymbol{\beta}$.

In computations, there is no general criterion about whether a weight is small or not. In this study, we use a threshold in the following fashion:

$$\gamma=\frac{a}{L}\sum_{i=1}^{L}\|\boldsymbol{\beta}_{i*}\|, \tag{14}$$

where $\boldsymbol{\beta}_{i*}$ represents the weights that connect the ith hidden node with the outputs, and a is an adjustment coefficient. In this study, we set $a=1$. Now, a weight $\boldsymbol{w}$ is small if $\|\boldsymbol{w}\|\leq\gamma$.

It can be seen that the first step of our method is the main part of the computation, determining the suitable number of hidden nodes. We note that there is no need to compute the Moore-Penrose generalized inverse or SVD in this step. Hence, for a large sample set and a large number of hidden nodes, the amount of computations can be reduced. After pruning ELM, the number of hidden nodes becomes smaller. Then it is easier to solve the problem by using ELM in the third step of our method.

4 Simulation results

Some experiments have been conducted to compare the performances of ELM and ELMR. In Sections 4.1 and 4.2, we compare their accuracies in approximating two simple functions and a few real-world problems, respectively. Section 4.3 demonstrates the sparsity properties of ELMR and ELM with the L_2 regularizer.

4.1 Function regression problems

First, we compare the performance of the proposed ELMR with that of ELM on two regression problems: sinc function and Gabor function. The sinc function is defined by (Fig. 1)

$$y(x) = \begin{cases} \sin x/x, & x \neq 0, \\ 1, & x = 0. \end{cases} \tag{15}$$

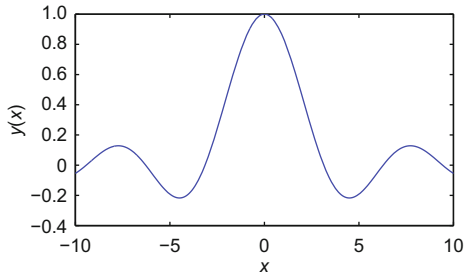

Fig. 1 Sinc function

There is 5000 data in both the training set and test set, created by randomly chosen x uniformly distributed in the interval $(-10, 10)$. All the training data is polluted by noise uniformly distributed in $[-0.2, 0.2]$, while the test data remains noise-free. ELM has 50 nodes and the activation function is a radial basis function (RBF).

Figs. 2a and 2b show that the curve plotted by ELMR is smoother than that by ELM. The average results of 50 experiments (Table 1) show that ELMR has fewer hidden nodes than ELM, but better accuracy, measured by root mean square error (RMSE):

$$\text{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(f_L(x_i) - o_i)^2}, \tag{16}$$

where $f_L(x_i)$ is the actual output and o_i is the ideal output.

Next, we test the performance of ELMR for a multi-dimensional Gabor function (Fig. 3):

$$h(x, y) = \frac{1}{2\pi(0.5)^2} \exp\left(-\frac{x^2 + y^2}{2(0.5)^2}\right) \cos(2\pi(x + y)). \tag{17}$$

First, we randomly select 441 training patterns from an evenly spaced 21×21 grid on the square $-0.5 \leq x \leq 0.5$ and $-0.5 \leq y \leq 0.5$. Noise uniformly distributed in $[-0.1, 0.1]$ has been added to all the training data. The 441 test patterns are selected similarly but without noise. The original ELM has 100 nodes and the activation function is an RBF. Then we use the ELMR algorithm to prune the network.

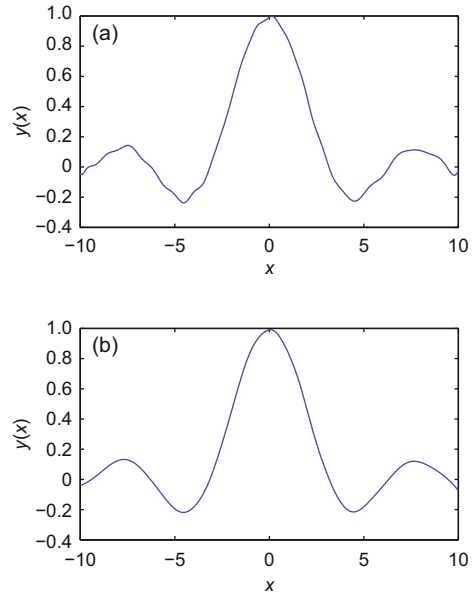

Fig. 2 The sinc function plotted by the ELM algorithm (a) and the ELMR algorithm (b)

Table 1 Performance on the sinc data set

Method	RMSE		Average number of nodes
	Training	Test	
ELM	0.1150	0.0107	50
ELMR	0.1151	0.0088	30.02

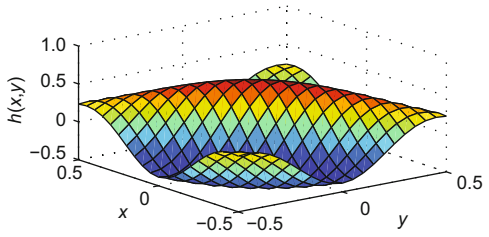

Fig. 3 Gabor function

Second, to see the performance of ELMR more clearly, we randomly select 2601 test patterns from an evenly spaced 51×51 grid on the square $-0.5 \leq x \leq 0.5$, $-0.5 \leq y \leq 0.5$. The original ELM has 100 nodes and the activation function is an RBF. Then we use the ELMR algorithm to prune the network.

Figs. 4 and 5 show that the network after pruning has better generalization performance. The test

curves plotted by ELMR are smoother than those by ELM.

Fifty trials have been conducted for both ELM and ELMR. Table 2 shows the average results.

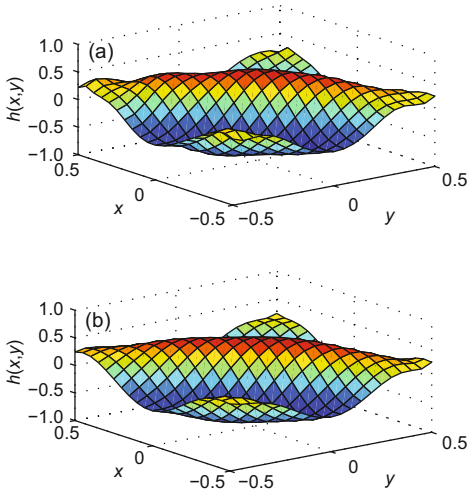

Fig. 4 The curves of the Gabor funtion plotted by ELM (a) and ELMR (b) with 441 test patterns

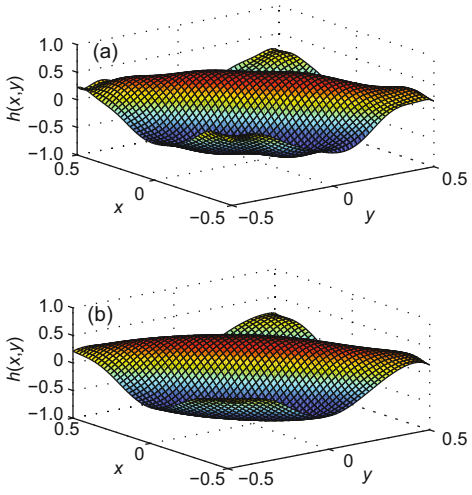

Fig. 5 The curves of the Gabor function plotted by ELM (a) and ELMR (b) with 2601 test patterns

4.2 Real-world problems

Now, we compare the performances of ELM and ELMR in dealing with a few real-world problems, including four regression problems, six binary classification problems, and six multiclass classification problems. These data sets are described in Tables 3–5.

Table 3 Specifications of regression problems with the RBF activation function

Data set	Number of samples		Number of features
	Training	Test	
Bodyfat	168	84	14
Cloud	1048	1000	9
Housing	337	169	13
Pyrim	49	25	27

Table 4 Specifications of binary classification problems with the sigmoid activation function

Data set	Number of samples		Number of features
	Training	Test	
Australian credit	460	230	6
Diabetes	512	256	8
Heart	170	100	13
Ionosphere	130	100	34
Sonar	108	100	60
Wisconsin	400	283	9

Table 5 Specifications of multiclass classification problems with the sigmoid activation function

Data set	Number of samples		Number of features	Number of classes
	Training	Test		
Glass	142	72	9	6
Iris	100	50	4	3
Landsat	3000	3435	36	6
Olitos	80	40	25	4
Segment	1310	1000	19	7
Wine	100	78	13	3

Table 2 Performance on the Gabor function

Testing pattern	RMSE				Average number of nodes	
	Training		Test			
	ELM	ELMR	ELM	ELMR	ELM	ELMR
21×21 grid	0.0518	0.0537	0.0252	0.0215	100	61.68
51×51 grid	0.0514	0.0535	0.0247	0.0203	100	62.12

Take the Iris data set as an example. It contains three classes: Versicolor, Virginica, and Setosa. The data set consists of 150 samples in $\mathbb{R}^4$. We randomly select 100 data as the training samples, and the remaining 50 data as the test samples. The activation function is the sigmoid function. As indicated in Tables 3–5, all the activation functions for the regression problems are chosen as RBF, while those for the classification problems are sigmoid functions.

We conduct 50 experiments for each data set. Then we use the ELMR algorithm to prune the network. The average results are shown in Tables 6–8.

Tables 6–8 show that for most of the data sets, ELMR has a little better performance than ELM. For the rest of the data sets, ELMR with fewer nodes has similar performance to ELM.

4.3 Sparsity of $L_{1/2}$ and L_2 regularizers

We select some of the above data sets to compare the sparsity of the $L_{1/2}$ regularizer and L_2 regularizer. ELM with the L_2 regularizer is to find the hidden-to-output weight $\boldsymbol{\beta} \in \mathbb{R}^L \times \mathbb{R}^m$ such that

$$\hat{\boldsymbol{\beta}} = \arg\min_{\boldsymbol{\beta}} \left\{ \|\boldsymbol{H}\boldsymbol{\beta} - \boldsymbol{O}\|_2^2 + \lambda \|\boldsymbol{\beta}\|_2^2 \right\}. \tag{18}$$

Its least square solution is

$$\hat{\boldsymbol{\beta}} = (\boldsymbol{H}^{\mathrm{T}}\boldsymbol{H} + \lambda \boldsymbol{I})^{-1}\boldsymbol{H}^{\mathrm{T}}\boldsymbol{O}. \tag{19}$$

Although sparsity has been widely studied in recent years, no unified criterion is available to measure the sparsity of a problem. In our study, we choose two indexes to compare the sparsity of two weight matrixes. The first index, denoted by AME, is the average of the minimum n elements, in absolute value, of both matrixes for a fixed number n.

Table 6 Performance on four regression problems

Data set	RMSE				Average number of nodes	
	Training		Test			
	ELM	ELMR	ELM	ELMR	ELM	ELMR
Bodyfat	0.0743	0.0827	0.1571	0.1139	50	29.98
Cloud	0.0065	0.0074	0.0126	0.0104	200	121.84
Housing	0.0533	0.0700	0.1017	0.1006	100	59.44
Pyrim	0.0063	0.0063	0.1481	0.1598	1000	623.30

Table 7 Performance on six binary classification problems

Data set	Accuracy				Average number of nodes	
	Training		Test			
	ELM	ELMR	ELM	ELMR	ELM	ELMR
Australian credit	84.02%	81.67%	73.54%	75.63%	100	56.56
Diabetes	83.41%	81.28%	74.21%	76.16%	100	60.50
Heart	91.02%	88.09%	78.04%	81.32%	50	28.80
Ionosphere	94.91%	90.28%	80.42%	82.32%	50	27.12
Sonar	100.00%	100.00%	79.36%	78.58%	1000	620.82
Wisconsin	97.72%	97.42%	96.47%	96.69%	50	28.4

Table 8 Performance on six multiclass classification problems

Data set	Accuracy				Average number of nodes	
	Training		Test			
	ELM	ELMR	ELM	ELMR	ELM	ELMR
Glass	85.94%	77.45%	62.47%	64.39%	50	27.44
Iris	99.40%	98.84%	92.56%	95.68%	50	27.24
Landsat	89.65%	87.65%	87.09%	86.00%	200	112.40
Olitos	99.65%	95.93%	70.90%	77.95%	50	26.48
Segment	97.59%	95.51%	94.62%	93.76%	200	107.72
Wine	100.00%	99.92%	94.77%	96.92%	50	27.44

The other index, denoted by NES, is the number of elements whose absolute values are smaller than a threshold. In this experiment, $n = 20$ and the threshold is taken as the absolute value of the 20th smallest element of the matrix for the L_2 regularizer. Fifty experiments are carried out for each data set. Table 9 shows the average results. The $L_{1/2}$ regularizer has better sparsity than the L_2 regularizer.

Table 9 Comparison between the L_2 regularizer and $L_{1/2}$ regularizer with n=20

Data set	AME		NES	
	L_2	$L_{1/2}$	L_2	$L_{1/2}$
Gabor	0.2884	0.0507	20	86.10
Bodyfat	0.0925	0.0809	20	23.34
Ionosphere	0.0965	0.0243	20	68.86
Sonar	0.0064	0.0053	20	22.44
Glass	0.0378	0.0151	20	46.56
Iris	0.1045	0.0350	20	57.90
Landsat	0.0198	0.0078	20	47.98
Wine	0.0734	0.0326	20	42.18

5 Conclusions

In this paper, we combine the $L_{1/2}$ regularization method with extreme learning machine, to prune neural networks by choosing a suitable number of hidden nodes. A variable learning coefficient is introduced to prevent too large a learning increment. The experiments for both regression and classification problems show that a network pruned by $L_{1/2}$ regularization has fewer hidden nodes but better performance than both the original network and the network pruned by L_2 regularization. In future work we want to investigate how to more efficiently implement ELMR methods.

References

Bishop, C.M., 1995. Neural Networks for Pattern Recognition. Oxford University Press, UK.

Chen, S., Donoho, D.L., Saunders, M.A., 2001. Atomic decomposition by basis pursuit. *SIAM Rev.*, **43**(1):129-159. [doi:10.1137/S003614450037906X]

Donoho, D.L., Huo, X., 2001. Uncertainty principles and ideal atomic decomposition. *IEEE Trans. Inform. Theory*, **47**(7):2845-2862. [doi:10.1109/18.959265]

Horata, P., Chiewchanwattana, S., Sunat, K., 2013. Robust extreme learning machine. *Neurocomputing*, **102**:31-44. [doi:10.1016/j.neucom.2011.12.045]

Hornik, K., Stinchcombe, M., White, H., 1989. Multilayer feedforward networks are universal approximators. *Neur. Networks*, **2**(5):359-366. [doi:10.1016/0893-6080(89)90020-8]

Huang, G.B., Siew, C.K., 2004. Extreme learning machine: RBF network case. Proc. 8th Int. Conf. on Control, Automation, Robotics and Vision, p.1029-1036. [doi:10.1109/ICARCV.2004.1468985]

Huang, G.B., Zhu, Q.Y., Siew, C.K., 2004. Extreme learning machine: a new learning scheme of feedforward neural networks. Proc. IEEE Int. Joint Conf. on Neural Networks, p.985-990. [doi:10.1109/IJCNN.2004.1380068]

Huang, G.B., Zhu, Q.Y., Siew, C.K., 2006. Extreme learning machine: theory and applications. *Neurocomputing*, **70**(1-3):489-501. [doi:10.1016/j.neucom.2005.12.126]

Huang, G.B., Wang, D.H., Lan, Y., 2011. Extreme learning machines: a survey. *Int. J. Mach. Learn. Cybern.*, **2**(2):107-122. [doi:10.1007/s13042-011-0019-y]

Miche, Y., Sorjamaa, A., Bas, P., *et al.*, 2010. OP-ELM: optimally pruned extreme learning machine. *IEEE Trans. Neur. Networks*, **21**(1):158-162. [doi:10.1109/TNN.2009.2036259]

Miche, Y., van Heeswijk, M., Bas, P., *et al.*, 2011. TROP-ELM: a double-regularized ELM using LARS and Tikhonov regularization. *Neurocomputing*, **74**(16):2413-2421. [doi:10.1016/j.neucom.2010.12.042]

Rasmussen, C.E., Williams, C.K.I., 2006. Gaussian Processes for Machine Learning. MIT Press, Cambridge, MA.

Soria-Olivas, E., Gomez-Sanchis, J., Jarman, I.H., *et al.*, 2011. BELM: Bayesian extreme learning machine. *IEEE Trans. Neur. Networks*, **22**(3):505-509. [doi:10.1109/TNN.2010.2103956]

Tibshirani, R., 1996. Regression shrinkage and selection via the Lasso. *J. R. Stat. Soc. B*, **58**(1):267-288.

Wu, W., 2003. Computation of Neural Networks. Higher Education Press, Beijing (in Chinese).

Xu, G.B., Zhou, D.H., 2010. Fault prediction for state-dependent fault based on online learning neural network. *J. Zhejiang Univ. (Eng. Sci.)*, **44**(7):1251-1254 (in Chinese).

Xu, Z.B., 2010. Data modeling: visual psychology approach and $L_{1/2}$ regularization theory. Proc. Int. Congress of Mathmaticians, p.3151-3184.

Xu, Z.B., Zhang, H., Wang, Y., *et al.*, 2010. $L_{1/2}$ regularization. *Sci. China Inform. Sci.*, **53**(6):1159-1169. [doi:10.1007/s11432-010-0090-0]

Yan, G.F., Zhang, S.L., Liu, M.Q., 2004. Standard neural network model and its application. *J. Zhejiang Univ. (Eng. Sci.)*, **38**(3):297-301 (in Chinese).

Zhang, W.B., Ji, H.B., 2013. Fuzzy extreme learning machine for classification. *Electron. Lett.*, **49**(7):448-450. [doi:10.1049/el.2012.3642]