Available online at www.sciencedirect.com
SCIENCE DIRECT®
Pattern Recognition 38 (2005) 1759–1763
PATTERN RECOGNITION
THE JOURNAL OF THE PATTERN RECOGNITION SOCIETY
www.elsevier.com/locate/patcog

Rapid and brief communication

Evolutionary extreme learning machine

Qin-Yu Zhu, A.K. Qin, P.N. Suganthan, Guang-Bin Huang*

School of Electrical and Electronic Engineering, Nanyang Technological University, Nanyang Avenue, Singapore 639798, Singapore

Received 21 March 2005; accepted 30 March 2005

Abstract

Extreme learning machine (ELM) [G.-B. Huang, Q.-Y. Zhu, C.-K. Siew, Extreme learning machine: a new learning scheme of feedforward neural networks, in: Proceedings of the International Joint Conference on Neural Networks (IJCNN2004), Budapest, Hungary, 25–29 July 2004], a novel learning algorithm much faster than the traditional gradient-based learning algorithms, was proposed recently for single-hidden-layer feedforward neural networks (SLFNs). However, ELM may need higher number of hidden neurons due to the random determination of the input weights and hidden biases. In this paper, a hybrid learning algorithm is proposed which uses the differential evolutionary algorithm to select the input weights and Moore–Penrose (MP) generalized inverse to analytically determine the output weights. Experimental results show that this approach is able to achieve good generalization performance with much more compact networks.

Keywords: Differential evolution; Minimum norm least square; Extreme learning machine

1. Introduction

Conventional gradient-based learning algorithms, such as back-propagation (BP) and it variant Levenberg–Marquardt (LM) method, have been extensively used in the training of multilayer feedforward neural networks. Although reasonable performance can be obtained when the networks are trained by BP, these gradient-based learning algorithms are still relatively slow in learning. These learning algorithms may also easily get stuck in a local minimum. Moreover, the activation functions used in these gradient-based tuning methods need to be differentiable.

A novel learning algorithm for single-hidden-layer feedforward neural networks (SLFNs) called extreme learning machine (ELM) [1,2] was proposed recently. In ELM, the input weights (linking the input layer to the hidden layer) and hidden biases are randomly chosen, and the output weights (linking the hidden layer to the output layer) are analytically determined by using Moore–Penrose (MP) generalized inverse. ELM not only learns much faster with higher generalization performance than the traditional gradient-based learning algorithms but also avoids many difficulties faced by gradient-based learning methods such as stopping criteria, learning rate, learning epochs, and local minima. However, it is also found that ELM tends to require more hidden neurons than conventional tuning-based algorithms in many cases.

Since evolutionary algorithms (EAs) are widely used as a global searching method for optimization, the hybrids of EA and analytical methods should be promising for network training. In the method proposed by Ghosh and Verma [3] namely GALS, the input weights are tuned by EA and the output weights are *iteratively* tuned/updated using the QR factorization method. In this paper, instead of iterative optimization of the output weights, a novel hybrid approach taking advantages of both ELM and the differential

* Corresponding author. Tel.: +65 67904489; fax: +65 67933318.
E-mail addresses: egbhuang@ntu.edu.sg (G.-B. Huang).
URLs: http://www.ntu.edu.sg/home/epnsugan (P.N. Suganthan), http://www.ntu.edu.sg/home/egbhuang/ (G.-B. Huang).

0031-3203/$30.00
doi:10.1016/j.patcog.2005.03.028

evolution (DE) [4] is proposed. DE is known for its ability and efficiency to locate global optimum over other EAs (cf. [4]). In the proposed algorithm, a modified DE is used to search for the optimal input weights and hidden biases, while the MP generalized inverse is used to analytically calculate the output weights.

2. Extreme learning machine

Extreme learning machine (ELM) was proposed in Huang, et al. [1]. Suppose we are training SLFNs with K hidden neurons and activation function $g(x)$ to learn N distinct samples $(\mathbf{x}_i, \mathbf{t}_i)$, where $\mathbf{x}_i = [x_{i1}, x_{i2}, \ldots, x_{in}]^{\mathrm{T}} \in \mathbf{R}^n$ and $\mathbf{t}_i = [t_{i1}, t_{i2}, \ldots, t_{im}]^{\mathrm{T}} \in \mathbf{R}^m$. In ELM, the input weights and hidden biases are randomly generated instead of tuned. By doing so, the nonlinear system has been converted to a linear system:

$$\mathbf{H}\boldsymbol{\beta} = \mathbf{T}, \tag{1}$$

where $\mathbf{H} = \{h_{ij}\}$ $(i = 1, \ldots, N$ and $j = 1, \ldots, K)$ is the hidden-layer output matrix, $h_{ij} = g(\mathbf{w}_j \cdot \mathbf{x}_i + b_j)$ denotes the output of jth hidden neuron with respect to $\mathbf{x}_i$; $\mathbf{w}_j = [w_{j1}, w_{j2}, \ldots, w_{jn}]^{\mathrm{T}}$ is the weight vector connecting jth hidden neuron and input neurons, and b_j denotes the bias of jth hidden neuron; $\mathbf{w}_j \cdot \mathbf{x}_i$ denotes the inner product of $\mathbf{w}_j$ and $\mathbf{x}_i$; $\boldsymbol{\beta} = [\boldsymbol{\beta}_1, \boldsymbol{\beta}_2, \ldots, \boldsymbol{\beta}_K]^{\mathrm{T}}$ is the matrix of output weights and $\boldsymbol{\beta}_j = [\beta_{j1}, \beta_{j2}, \ldots, \beta_{jm}]^{\mathrm{T}}$ $(j = 1, \ldots, K)$ denotes the weight vector connecting the jth hidden neuron and output neurons; $\mathbf{T} = [\mathbf{t}_1, \mathbf{t}_2, \ldots, \mathbf{t}_N]^{\mathrm{T}}$ is the matrix of targets (desired output).

Thus, the determination of the output weights (linking the hidden layer to the output layer) is as simple as finding the least-square solution to the given linear system. The minimum norm least-square (LS) solution to the linear system (1) is

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

where $\mathbf{H}^{\dagger}$ is the MP generalized inverse of matrix $\mathbf{H}$. The minimum norm LS solution is unique and has the smallest norm among all the LS solutions. As analyzed by Huang, et al. [1], ELM using such MP inverse method tends to obtain good generalization performance with dramatically increased learning speed.

3. Differential evolution

Differential evolution (DE) proposed by Storn and Price [4] is known as one of the most efficient evolutionary algorithms (EAs). The basic strategy of DE can be described as follows:

Given a set of parameter vectors $\{\boldsymbol{\theta}_{i,G} | i = 1, 2, \ldots, NP\}$ as a population at each generation G, we do:

Mutation: For each target vector $\boldsymbol{\theta}_{i,G+1}$, $i = 1, 2, \ldots, NP$, a mutant vector is generated according to

$$\mathbf{v}_{i,G+1} = \boldsymbol{\theta}_{r_1,G} + F \cdot (\boldsymbol{\theta}_{r_2,G} - \boldsymbol{\theta}_{r_3,G}) \tag{3}$$

with random and mutually different indices $r_1, r_2, r_3 \in \{1, 2, \ldots, NP\}$ and $F \in [0, 2]$. The constant factor F is used to control the amplification of the differential variation $(\boldsymbol{\theta}_{r_2,G} - \boldsymbol{\theta}_{r_3,G})$.

Crossover: In this step, the D-dimensional trial vector:

$$\boldsymbol{\mu}_{i,G+1} = (\boldsymbol{\mu}_{1i,G+1}, \boldsymbol{\mu}_{2i,G+1}, \ldots, \boldsymbol{\mu}_{Di,G+1})$$

is formed so that

$$\boldsymbol{\mu}_{ji,G+1} = \begin{cases} \mathbf{v}_{ji,G+1} & \text{if } rand\ b(j) \leqslant CR \text{ or } j = rnbr(i), \\ \boldsymbol{\theta}_{ji,G} & \text{if } rand\ b(j) > CR \text{ and } j \neq rnbr(i). \end{cases} \tag{4}$$

In Eq. (4), $rand\ b(j)$ is the jth evaluation of a uniform random number generator with outcome in $[0, 1]$. CR is the crossover constant in $[0, 1]$ which is determined by user. $rnbr(i)$ is a random chosen integer index $\in [1, D]$ which ensures that $\boldsymbol{\mu}_{i,G+1}$ gets at least one parameter from $\mathbf{v}_{ji,G+1}$.

Selection: If vector $\boldsymbol{\mu}_{i,G+1}$ is better than $\boldsymbol{\theta}_{i,G}$, then $\boldsymbol{\theta}_{i,G+1}$ is set to $\boldsymbol{\mu}_{i,G+1}$. Otherwise, the old value $\boldsymbol{\theta}_{i,G}$ is retained as $\boldsymbol{\theta}_{i,G+1}$.

4. Proposed evolutionary extreme learning machine (E-ELM)

Since ELM just randomly chooses the input weights and hidden biases, much of learning time traditionally spent in tuning these parameters is saved. However, as the output weights are computed based on the prefixed input weights and hidden biases, there may exist a set of nonoptimal or unnecessary input weights and hidden biases. ELM may require more hidden neurons than conventional tuning-based learning algorithms in some applications, which may make ELM respond slowly to unknown testing data. Thus, one may expect more compact networks in the applications which requires faster response of the trained networks.

In this section, a hybrid approach named E-ELM using DE and MP generalized inverse is proposed. Firstly, we randomly generate the population. Each individual in the population is composed of a set of input weights and hidden biases:

$$\boldsymbol{\theta} = [w_{11}, w_{12}, \ldots, w_{1K}, w_{21}, w_{22}, \ldots, w_{2K}, \ldots, w_{n1}, w_{n2}, \ldots w_{nK}, \ldots, b_1, b_2, \ldots, b_K]$$

All w_{ij} and b_j are randomly initialized within the range of $[-1, 1]$.

Secondly, for each individual (a set of weights and biases), the corresponding output weights are analytically computed by using the MP generalized inverse as done in ELM (cf. Eq. (2)) instead of any iterative tuning [3].

Then the fitness of each individual is evaluated. Suppose that the cost function (E) is root mean squared error (RMSE):

$$E = \sqrt{\frac{\sum_{j=1}^{N} \| \sum_{i=1}^{K} \boldsymbol{\beta}_i g(\mathbf{w}_i \cdot \mathbf{x}_j + b_i) - \mathbf{t}_j \|_2^2}{m \times N}}. \tag{5}$$

In the normal case, the RMSE on the whole training dataset is used as the fitness. However, it may cause the networks to overfit. In BP, a validation dataset, which normally has no overlap with the training dataset, is used to avoid the overfitting. In addition, as the $\boldsymbol{\beta}$ is the minimum norm least-square solution to the training dataset already, the training error with respect to different individual should be very similar. Therefore, in order to save time, we set the fitness to the RMSE on the validation set only instead of the whole training set as used in [3].

After the fitness of all individuals in the population is calculated, we apply the three steps of DE: mutation, crossover and selection. In DE, during selection, the mutated vectors are compared with the original ones, and the vectors with better fitness values are retained to the next generation. However, for neural networks training, using the fitness (validation RMSE) alone as the selection criteria is not appropriate. A small validation error does not necessarily lead to a small testing error, which largely depends on the distribution of the validation data. As analyzed by Bartlett [5], networks tend to have better generalization performance with smaller weights. Therefore, to further improve the generalization performance, we add one more criteria into the selection: the norm of output weights $\|\boldsymbol{\beta}\|$. In our new selection strategy, when the difference of the fitness between different individuals is small, the one resulting in smaller $\|\boldsymbol{\beta}\|$ is selected. The determination of new population $\boldsymbol{\theta}_{i,G+1}$ can be described as follows:

$$\boldsymbol{\theta}_{i,G+1} = \begin{cases} \boldsymbol{\mu}_{i,G} & \text{if } f(\boldsymbol{\theta}_{i,G}) - f(\boldsymbol{\mu}_{i,G}) > \varepsilon f(\boldsymbol{\theta}_{i,G}), \\ \boldsymbol{\mu}_{i,G} & \text{if } |f(\boldsymbol{\theta}_{i,G}) - f(\boldsymbol{\mu}_{i,G})| < \varepsilon f(\boldsymbol{\theta}_{i,G}) \\ & \quad \text{and } \|\boldsymbol{\beta}^{\boldsymbol{\mu}_{i,G}}\| < \|\boldsymbol{\beta}^{\boldsymbol{\theta}_{i,G}}\|, \\ \boldsymbol{\theta}_{i,G} & \text{else,} \end{cases} \quad (6)$$

where $f(\cdot)$ is the fitness function (validation error) and ε is a preset tolerance rate. Once the new population is generated, repeat the same DE process until the goal is met or a preset maximum learning epochs is completed.

5. Experimental results

In this section, we compare the E-ELM with LM, ELM,[1] GALS (cf. [3]) and another hybrid DE-LM. DE-LM is much like the proposed E-ELM except that the output weights are trained by LM learning algorithm instead of MP generalized inverse. All the programs are run in MATLAB 6.5 environment. The LM program is one of the fastest implementation of BP algorithms and is provided in the neural networks tools box of MATLAB. A validation set is used with LM to prevent the network from overfitting. The parameters of the DE used in E-ELM and DE-LM are set as follows: Population size NP is set to twice the number of networks' parameters; F and CR are set to 1 and 0.8 respectively. In every simulation, we gradually increase the hidden neurons, and select the results with best generalization performance as the final one. All the results shown in this paper are the mean values of 50 trails. The inputs of all cases are normalized into $[-1, 1]$.

We first simply try to reconstruct the sigmoid function $f(x) = 1/(1 + e^{-x})$ using the proposed E-ELM. Theoretically, only one hidden neuron is needed to learn this function. 100 observations are randomly generated for training, testing and validation datasets, respectively. The results can been seen in Table 1. E-ELM, LM, DE-LM and GALS are able to achieve a very small RMSE with single hidden neuron, whereas the learning time used by E-ELM and GALS is much less than LM and DE-LM. Although ELM can achieve a similar accuracy at the fastest speed, it needs 10 neurons.

The performance of E-ELM is also tested on four real benchmark classification problems. The specification of these problems are listed in Table 2. The training, testing and validation datasets are randomly regenerated at each trial of simulations according to Table 2 for all the algorithms. From Table 3, we can see that E-ELM outperforms the other three algorithms in all the simulations except Satellite Image in terms of testing accuracy. Obviously, DE helps to reduce the hidden neurons in both E-ELM and DE-LM and also helps to improve the generalization performance. Even DE–LM yields a higher testing accuracy than the pure LM. Although DE prolongs the training process, the training time of E-ELM is still reasonable which is still much less than the time needed by LM and GALS in all the simulations. One should keep it in mind that LM is one of the fastest gradient-based learning algorithms. Unfortunately, the training of GALS on Shuttle case could not be completed as it ran out of memory during the QR factorization step with 10 hidden neurons only, while all the other algorithms are successfully tested on this dataset with even higher number of hidden neurons.

6. Conclusions

In this paper, we proposed a novel learning algorithm named evolutionary extreme learning machine (E-ELM) which makes use of the advantages of both ELM and DE. It uses the fast minimum norm least-square scheme to analytically determine the output weights instead of tuning, and a modified form of DE is used to optimize the input weights and hidden biases. Unlike the gradient-based methods, the proposed E-ELM does not require the activation functions to be differentiable, implying that E-ELM can be used to train SLFNs with many nonlinear hidden units such as threshold units which are claimed to be easier for hardware implementation. Experimental results show that E-ELM generally achieves higher generalization performance than other algorithms including BP, GALS and the original ELM. The results also indicate that GALS is relatively slow and need more memory due to the large storage and computational complexity involved in computing the

[1] ELM Source Codes: http://www.ntu.edu.sg/home/egbhuang/.

Table 1
Results of regression problem: to approximate a sigmoid function

Algorithm	Training time (s)	RMSE		Hidden neurons
		Training	Testing	
E-ELM	1.18	$5.8193e^{-5} \pm 3.9384e^{-7}$	$8.6764e^{-5} \pm 5.1838e^{-7}$	1
ELM	0.0016	$6.8591e^{-5} \pm 3.7340e^{-7}$	$7.0865e^{-5} \pm 3.9776e^{-6}$	10
DE–LM	290.313	$2.2948e^{-4} \pm 1.9774e^{-5}$	$4.6584e^{-4} \pm 2.9837e^{-5}$	1
LM	12.297	$4.9961e^{-5} \pm 1.7083e^{-6}$	$7.6952e^{-5} \pm 8.2876e^{-6}$	1
GALS	1.328	$4.3848e^{-4} \pm 5.8382e^{-7}$	$4.4913e^{-4} \pm 1.8371e^{-5}$	1

Table 2
Specification of classification problems

Name	Attributes	Classes	Number of observations		
			Training	Testing	Validation
Diabetes	17	2	202	258	258
Satellite image	36	7	4400	1000	1000
Image segmentation	18	7	1500	410	410
Shuttle	10	7	43 500	7250	7250

Table 3
Results of classification problems

Problem	Algorithm	Training time (s)	Accuracy (%)		Hidden neurons
			Training	Testing	
Diabetes	E-ELM	1.2936	82.36 ± 1.28	80.69 ± 1.83	10
	ELM	0.0116	78.68 ± 1.18	78.2 ± 2.65	20
	DE–LM	864.31	79.47 ± 1.51	78.33 ± 2.09	10
	LM	3.0116	86.63 ± 1.7	74.73 ± 3.2	20
	GALS	86.76	78.45 ± 1.59	75.36 ± 2.61	10
Satellite image	E-ELM	1569.27	92.39 ± 0.97	88.46 ± 1.36	90
	ELM	14.9164	93.52 ± 1.46	89.04 ± 1.5	500
	DE–LM	36924	93.58 ± 1.26	85.31 ± 2.14	80
	LM	12561	95.26 ± 0.96	82.34 ± 1.25	100
	GALS	29209.4	88.48 ± 1.17	86.5 ± 2.06	80
Image segmentation	E-ELM	154.1	96.37 ± 0.49	95.27 ± 1.56	70
	ELM	1.4015	97.35 ± 0.32	95.01 ± 0.78	200
	DE–LM	13196.6	97.02 ± 0.86	88.65 ± 2.05	80
	LM	4745.7	97.35 ± 0.32	86.27 ± 1.80	100
	GALS	3423.7	95.67 ± 0.89	94.27 ± 1.95	80
Shuttle	E-ELM	1336.9	99.63 ± 0.11	99.53 ± 0.12	20
	ELM	5.740	99.65 ± 0.12	99.40 ± 0.12	100
	DE–LM	10892.3	99.72 ± 0.1	99.33 ± 0.09	20
	LM	6132.2	99.77 ± 0.1	99.27 ± 0.13	50
	GALS	Out of memory			

QR factorization. While doing simulations of GALS, the networks are easily overtrained because GALS only uses the training RMSE as the fitness, whereas E-ELM considers both validation RMSE and norm of weights during searching to achieve a better generalization performance. Hence, GALS is sensitive to the population size and the number of learning epochs. Compared with conventional gradient-based BP algorithms and GALS, E-ELM has faster learning speed and higher testing accuracy and compared with the ELM algorithm, E-ELM can obtain much more compact network architecture which would increase the response speed and be helpful in fast response (to unknown testing data) applications.

References

[1] G.-B. Huang, Q.-Y. Zhu, C.-K. Siew, Extreme learning machine: a new learning scheme of feedforward neural networks, in: Proceedings of the International Joint Conference on Neural Networks (IJCNN2004), Budapest, Hungary, 25–29 July 2004.

[2] M.-B. Li, G.-B. Huang, P. Saratchandran, N. Sundararajan, Fully complex extreme learning machine, Neurocomputing, 2005, to appear.

[3] R. Ghosh, B. Verma, A hierarchical method for finding optimal architecture and weights using evolutionary least square based learning, Int. J. Neural Syst. 12 (1) (2003) 13–24.

[4] R. Storn, K. Price, Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces, J. Global Optim. 11 (1997) 341–359.

[5] P.L. Bartlett, 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. Inform. Theory 44 (2) (1998) 525–536.