Neural Comput & Applic
DOI 10.1007/s00521-013-1519-3

Extreme learning machine for interval neural networks

Dakun Yang · Zhengxue Li · Wei Wu

Received: 16 September 2013 / Accepted: 16 November 2013

Abstract Interval data offer a valuable way of representing the available information in complex problems where uncertainty, inaccuracy, or variability must be taken into account. Considered in this paper is the learning of interval neural networks, of which the input and output are vectors with interval components, and the weights are real numbers. The back-propagation (BP) learning algorithm is very slow for interval neural networks, just as for usual real-valued neural networks. Extreme learning machine (ELM) has faster learning speed than the BP algorithm. In this paper, ELM is applied for learning of interval neural networks, resulting in an interval extreme learning machine (IELM). There are two steps in the ELM for usual feedforward neural networks. The first step is to randomly generate the weights connecting the input and the hidden layers, and the second step is to use the Moore–Penrose generalized inversely to determine the weights connecting the hidden and output layers. The first step can be directly applied for interval neural networks. But the second step cannot, due to the involvement of nonlinear constraint conditions for IELM. Instead, we use the same idea as that of the BP algorithm to form a nonlinear optimization problem to determine the weights connecting the hidden and output layers of IELM. Numerical experiments show that IELM is much faster than the usual BP algorithm. And the generalization performance of IELM is much better than that of BP, while the training error of IELM is a little bit worse than that of BP, implying that there might be an over-fitting for BP.

Keywords Interval computation · Interval neural network · Extreme learning machine · Interval extreme learning machine

1 Introduction

In real-life situations, available information is often uncertain, imprecise, and incomplete. In many such applications, it is natural to express and treat the information in the form of intervals rather than real values [1–4]. For instance, interval neural networks [5–11] have been proposed for handling interval data. BP algorithm [12, 13] was extended to the case of interval input vectors [5]. But as is well known, BP algorithm is usually very slow.

Recently, extreme learning machine (ELM) [14–18] proposed by Huang [19–21] becomes a popular learning algorithm for single hidden layer feedforward networks (SLFNs). There are two steps in the ELM for usual feedforward neural networks. The first step is to randomly generate the weights connecting the input and the hidden layers, and the second step is to use the Moore–Penrose generalized inversely to determine the weights connecting the hidden and output layers. The learning speed of ELM can be thousands of times faster than BP algorithm, since the hidden layer of SLFNs need not be tuned.

The aim of this paper is to apply ELM for the learning of single hidden layer interval feedforward networks (SLIFNs), of which the input and output are vectors with interval components, and the weights are real numbers. The SLIFN with ELM learning is called an interval extreme learning machine (IELM). The above-mentioned

D. Yang · Z. Li · W. Wu (✉)
School of Mathematical Sciences, Dalian University of Technology, Dalian, China
e-mail: wuweiw@dlut.edu.cn

D. Yang
e-mail: ydk1026@gmail.com

Published online: 05 December 2013

first step of ELM can be directly applied for interval neural networks. But the second step cannot, due to the involvement of nonlinear constraint conditions for IELM. Instead, we use the same idea as that of the BP algorithm [22–24] to form a nonlinear optimization problem to determine the weights connecting the hidden and output layers of IELM. Numerical experiments show that IELM is much faster than the usual BP algorithm. It is also shown that the generalization performance of IELM is much better than that of BP, while the training error of IELM is a little bit worse than that of BP. This observation implies that there might be an over-fitting for BP.

The remainder of this paper is organized as follows. Some basic notations of interval arithmetic are described in Sect. 2. ELM and SLIFNs are introduced in Sects. 3 and 4, respectively. IELM is proposed in Sect. 5. Supporting numerical experiments are provided in Sect. 6.

2 Interval arithmetic

Interval arithmetic as a tool appeared in numerical computing in the late 50s. Then, Moore [1] and Sunaga [2] introduced interval mathematic theory in the study of error control in numerical computation. Some fundamentals of the interval arithmetic used in this paper are described below.

Let us denote the intervals by uppercase letters such as A and the real numbers by lowercase letters such as a. An interval can be represented by its lower bounds L and upper bounds U as $A = [a^L, a^U]$, or equivalently by its midpoint C and radius R as $A = \langle a^C, a^R \rangle$, where

$$a^C = \frac{a^L + a^U}{2}, \tag{1}$$

$$a^R = \frac{a^U - a^L}{2}. \tag{2}$$

For intervals $A = [a^L, a^U]$ and $B = [b^L, b^U]$, the basic interval operations are defined by

$$A + B = \left[a^L + b^L, a^U + b^U\right], \tag{3}$$

$$A - B = \left[a^L - b^U, a^U - b^L\right], \tag{4}$$

$$k \cdot A = \begin{cases} [k \cdot a^L, k \cdot a^U], & k > 0, \\ [k \cdot a^U, k \cdot a^L], & k < 0, \end{cases} \tag{5}$$

where k is a constant. More generally, if $f : R^1 \to R^1$ is an increasing function, then we can generalize that it as a function mapping intervals to intervals in the following fashion:

$$f(A) = [f(a^L), f(a^U)]. \tag{6}$$

The weighted Euclidean distance for a pair of intervals A and B is defined by the following:

$$d(A, B) = \alpha(a^C - b^C)^2 + (1 - \alpha)(a^R - b^R)^2, \ \alpha \in (0, 1). \tag{7}$$

The parameter $\alpha \in [0, 1]$ facilitates giving more importance to the prediction of the output centers or to the prediction of the radii. For $\alpha = 1$, learning concentrates on the prediction of the output interval center and no importance is given to the prediction of its radius. For $\alpha = 0.5$, both predictions (centers and radii) have the same weights in the objective function. For our purpose, we assume $\alpha \in (0, 1)$.

3 Extreme learning machine

In this section, we describe the extreme learning machine [19]. For N arbitrary distinct samples $(\mathbf{x}_i, \mathbf{t}_i)$, where $\mathbf{x}_i = [x_{i1}, x_{i2}, \ldots, x_{in}]^T \in \mathbf{R}^n$ and $\mathbf{t}_i = [t_{i1}, t_{i2}, \ldots, t_{im}]^T \in \mathbf{R}^m$, a standard single hidden layer feedforward network (SLFN) with $\widetilde{N}$ hidden nodes and activation function $g(x)$ is mathematically modeled as

$$\sum_{i=1}^{\widetilde{N}} \beta_i g_i(\mathbf{x}_j) = \sum_{i=1}^{\widetilde{N}} \beta_i g(\mathbf{w}_i \cdot \mathbf{x}_j + b_i) = \mathbf{o}_j, \quad j = 1, \ldots, N, \tag{8}$$

where $\mathbf{w}_i = [w_{i1}, w_{i2}, \ldots, w_{in}]^T$ is the weight vector connecting the ith hidden node and the input nodes, $\beta_i = [\beta_{i1}, \beta_{i2}, \ldots, \beta_{im}]^T$ is the weight vector connecting the ith hidden node and the output nodes, and b_i is the threshold of the ith hidden node.

That standard SLFN with $\widetilde{N}$ hidden nodes and activation function $g(x)$ can approximate these N samples with zero error means that $\sum_{j=1}^{\widetilde{N}} \|\mathbf{o}_j - \mathbf{t}_j\| = 0$, i.e., there exist β_i, $\mathbf{w}_i$ and b_i such that

$$\sum_{i=1}^{\widetilde{N}} \beta_i g(\mathbf{w}_i \cdot \mathbf{x}_j + b_i) = \mathbf{t}_j, \quad j = 1, \ldots, N. \tag{9}$$

The above N equations can be written compactly as

$$H\beta = T, \tag{10}$$

where

$$H(\mathbf{w}_1,\ldots,\mathbf{w}_{\widetilde{N}},b_1,\ldots,b_{\widetilde{N}},\mathbf{x}_1,\ldots,\mathbf{x}_N) = \begin{bmatrix} g(\mathbf{w}_1\cdot\mathbf{x}_1+b_1) & \cdots & g(\mathbf{w}_{\widetilde{N}}\cdot\mathbf{x}_1+b_{\widetilde{N}}) \\ \vdots & \cdots & \vdots \\ g(\mathbf{w}_1\cdot\mathbf{x}_N+b_1) & \cdots & g(\mathbf{w}_{\widetilde{N}}\cdot\mathbf{x}_N+b_{\widetilde{N}}) \end{bmatrix}_{N\times\widetilde{N}}, \tag{11}$$

$$\beta = \begin{bmatrix} \beta_1^T \\ \vdots \\ \beta_{\widetilde{N}}^T \end{bmatrix}_{\widetilde{N}\times m}, \quad T = \begin{bmatrix} t_1^T \\ \vdots \\ t_N^T \end{bmatrix}_{N\times m}. \tag{12}$$

H is called the hidden layer output matrix of the neural network, and the ith column of H is the ith hidden node output with respect to inputs $\mathbf{x}_1,\mathbf{x}_2,\ldots,\mathbf{x}_N$.

Conventionally, hidden layer parameters need to be adjusted at least once based on the training samples. However, all the parameters of the hidden layer in the ELMs need not be tuned after randomly generated and can be independent of the training samples. So the hidden node parameters ($\mathbf{w}_i$, b_i) remain fixed after randomly generated. To train an SLFN is simply equivalent to finding a least-squares solution $\widehat{\beta}$ of the linear system $H\beta = T$ such that

$$\|H\widehat{\beta}-T\| = \min_{\beta}\|H\beta-T\|. \tag{13}$$

The smallest norm least-squares solution of the above linear system is

$$\widehat{\beta} = H^{\dagger}T, \tag{14}$$

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

4 Single hidden layer interval feedforward networks (SLIFNs)

Let us consider a SLIFNs, where the input and output are interval values, and the weights are real values. The numbers of neurons for the input, hidden and output layers are $n,\widetilde{N},m$, respectively. For $i=1,2,\ldots,\widetilde{N}$, let $\mathbf{w}_i = [w_{i1},w_{i2},\ldots,w_{in}]^T \in \mathbf{R}^n$ be the weight matrix connecting the input and the ith hidden layers. The weight vector connecting the ith hidden and the output layers is denoted by $\beta_i = (\beta_{i1},\beta_{i2},\ldots,\beta_{im})^T \in \mathbf{R}^m$, $i=1,2,\ldots,\widetilde{N}$. b_i denotes the threshold of the ith hidden node for $i=1,2,\ldots,\widetilde{N}$. In the SLIFN, a nonlinear activation function $g(x)$ is used for the hidden layer, and a linear activation function for the output layer.

Let $\mathbf{x} = (X_1,X_2,\ldots,X_n)$ be an interval-valued input to the network, where $X_i = \langle x_i^C, x_i^R\rangle$, $i=1,2,\ldots,n$. It generates an input to hidden node $l\,(l=1,2,\ldots,\widetilde{N})$ as follows:

$$S_l = \sum_{i=1}^{n}\mathbf{w}_{il}X_i + b_i = \left\langle \sum_{i=1}^{n} w_{il}x_i^C + b_i, \sum_{i=1}^{n}|w_{il}|x_i^R \right\rangle = \langle s_l^C, s_l^R\rangle. \tag{15}$$

Then, the output of the hidden neuron l is given by

$$H_l = g(S_l) = \left[g(s_l^C - s_l^R), g(s_l^C + s_l^R)\right] = \langle h_l^C, h_l^R\rangle = \left\langle \frac{g(s_l^C - s_l^R)+g(s_l^C+s_l^R)}{2}, \frac{g(s_l^C+s_l^R)-g(s_l^C-s_l^R)}{2} \right\rangle. \tag{16}$$

Finally, the output of the interval neuron in the output layer is given by $j=1,2,\ldots,m$,

$$O_j = \left\langle o_j^C, o_j^R \right\rangle, \tag{17}$$

where

$$o_j^C = \sum_{i=1}^{\widetilde{N}}\beta_{ij}h_i^C, \tag{18}$$

$$o_j^R = \sum_{i=1}^{\widetilde{N}}|\beta_{ij}|h_i^R. \tag{19}$$

5 Interval extreme learning machine

For N arbitrary distinct interval samples ($\mathbf{x}_i$, $\mathbf{t}_i$), where $\mathbf{x}_i = [x_{i1},x_{i2},\ldots,x_{in}]^T$, $x_{ij} = [x_{ij}^L, x_{ij}^U] = \left\langle x_{ij}^C, x_{ij}^R \right\rangle$, $j=1,2,\ldots,N$, and $\mathbf{t}_i = [t_{i1},t_{i2},\ldots,t_{im}]^T$, $t_{ij} = [t_{ij}^L, t_{ij}^U] = \left\langle t_{ij}^C, t_{ij}^R \right\rangle$, $j=1,2,\ldots,m$, that the SLIFNs with $\widetilde{N}$ hidden nodes with activation function $g(x)$ approximate these N samples with zero error means that $\sum_{j=1}^{N}\|\mathbf{o}_j - \mathbf{t}_j\| = 0$, i.e., there exist β_i, $\mathbf{w}_i$, b_i such that

$$\begin{aligned} \sum_{i=1}^{\widetilde{N}}\beta_i h_i^C(\mathbf{w}_i,b_i) &= \mathbf{t}_j^C, \\ \sum_{i=1}^{\widetilde{N}}|\beta_i|h_i^R(\mathbf{w}_i,b_i) &= \mathbf{t}_j^R, \end{aligned} \tag{20}$$

where $j=1,2,\ldots,N$, $|\beta_i| = [|\beta_{i1}|,|\beta_{i2}|,\ldots,|\beta_{im}|]^T$, $\mathbf{t}_j^C = [t_{j1}^C, t_{j2}^C,\ldots,t_{jm}^C]^T$ and $\mathbf{t}_j^R = [t_{j1}^R, t_{j2}^R,\ldots,t_{jm}^R]^T$.

Equation (20) can be written compactly as

$$\begin{aligned} H^C\beta &= T^C, \\ H^R|\beta| &= T^R, \end{aligned} \tag{21}$$

where

$$H^C = \begin{bmatrix} h_1^C(\mathbf{x}_1) & \cdots & h_{\widetilde{N}}^C(\mathbf{x}_1) \\ \vdots & \cdots & \vdots \\ h_1^C(\mathbf{x}_N) & \cdots & h_{\widetilde{N}}^C(\mathbf{x}_N) \end{bmatrix},$$
$$H^R = \begin{bmatrix} h_1^R(\mathbf{x}_1) & \cdots & h_{\widetilde{N}}^R(\mathbf{x}_1) \\ \vdots & \cdots & \vdots \\ h_1^R(\mathbf{x}_N) & \cdots & h_{\widetilde{N}}^R(\mathbf{x}_N) \end{bmatrix}, \tag{22}$$

$$|\beta| = \begin{bmatrix} |\beta_1^T| \\ \vdots \\ |\beta_{\widetilde{N}}^T| \end{bmatrix}, \quad T^C = \begin{bmatrix} \mathbf{t}_1^C \\ \vdots \\ \mathbf{t}_N^C \end{bmatrix}, \quad T^R = \begin{bmatrix} \mathbf{t}_1^R \\ \vdots \\ \mathbf{t}_N^R \end{bmatrix}. \tag{23}$$

We denote that

$$\widetilde{H} = \begin{bmatrix} H^C & \mathbf{0} \\ \mathbf{0} & H^R \end{bmatrix}, \quad \widetilde{\beta} = \begin{bmatrix} \beta \\ |\beta| \end{bmatrix}, \quad \widetilde{T} = \begin{bmatrix} T^C \\ T^R \end{bmatrix}, \tag{24}$$

then (21) can be written as

$$\widetilde{H}\widetilde{\beta} = \widetilde{T}. \tag{25}$$

In this paper, the proposed interval extreme learning machine is basically an ELM operating on interval-valued input and output data. So the hidden node parameters ($\mathbf{w}_i$, b_i) remain fixed after randomly generated. To train an SLIFN is equivalent to finding a best solution $\widehat{\beta}$ of the following nonlinear optimization problem

$$\|\widetilde{H}\widehat{\beta} - \widetilde{T}\| = \min_{\widetilde{\beta}} \|\widetilde{H}\widetilde{\beta} - \widetilde{T}\| \tag{26}$$

$$\text{s.t.} \quad \begin{aligned} [I_{\widetilde{N}} \ \mathbf{0}_{\widetilde{N}}]\widetilde{\beta} &= \beta, \\ [\mathbf{0}_{\widetilde{N}} \ I_{\widetilde{N}}]\widetilde{\beta} &= |\beta|, \end{aligned} \tag{27}$$

where $I_{\widetilde{N}}$ is a identity matrix of $\widetilde{N} \times \widetilde{N}$, and $\mathbf{0}_{\widetilde{N}}$ is a zero matrix of $\widetilde{N} \times \widetilde{N}$.

However, for the nonlinear optimization problem (26) and (27), we cannot obtain the solution $\widehat{\beta}$ as similar as the solution of (14), because the constraint conditions (27) are nonlinear and the Moore–Penrose generalized inverse of matrix $\widetilde{\beta}$ dose not exist. In this paper, the gradient-based learning algorithm is used to search the minimum of the nonlinear problem (26). Since (26) is equivalent to minimizing the cost function

$$E(\beta) = \sum_{j=1}^{N} \left((\mathbf{o}_j^C - \mathbf{t}_j^C)^2 + (\mathbf{o}_j^R - \mathbf{t}_j^R)^2 \right), \tag{28}$$

then the gradient of the error function $E(\beta)$ with respect to β are given by

$$\frac{\partial E(\beta)}{\partial \beta} = 2 \sum_{j=1}^{N} \left((\mathbf{o}_j^C - \mathbf{t}_j^C) \frac{\partial \mathbf{o}_j^C}{\partial \beta} + (\mathbf{o}_j^R - \mathbf{t}_j^R) \frac{\partial \mathbf{o}_j^R}{\partial \beta} \right), \tag{29}$$

where

$$\begin{aligned} \frac{\partial \mathbf{o}_j^C}{\partial \beta} &= h_j^C, \\ \frac{\partial \mathbf{o}_j^R}{\partial \beta} &= h_j^R sgn(\beta). \end{aligned} \tag{30}$$

In the learning procedure, the weights β are iteratively refined as follows:

$$\beta^{k+1} = \beta^k + \Delta\beta^k, \tag{31}$$

where

$$\Delta\beta^k = -\eta \frac{\partial E(\beta^k)}{\partial \beta}, \ k = 1, 2, \ldots, \tag{32}$$

and $\eta > 0$ is a constant learning rate.

6 Numerical experiment

In this section, the proposed interval extreme learning machine learning algorithm is applied for three numerical examples. A simple interval function with noise is approximated by interval extreme learning machine in the first example. In the second example, we consider some benchmark real-world regression problems. In Example 3, we consider the prediction of composite index of Shanghai stock market.

Example 1 A simple approximation problem

we consider the approximation of the following interval function with noise

$$y = 0.5 * x - \sin x. \tag{33}$$

In this example, 81 input points are evenly selected from the interval $[-4, 4]$ as the midpoints of the input samples, and their radii are selected randomly from [0, 0.1]. The midpoints of the output samples are the function values of the input points, and the noise is selected randomly from [0, 0.1] as the radius of the each output sample. The number of training iterations is 3,000, the number of neurons of the hidden layer is 50, the initial weight vector is selected randomly from $[-1, 1]$, and the fix learning rate is $\eta = 0.0006$.

From Figs. 1 and 2, respectively, we can see that for the interval extreme learning machine, both the value of error function, defined by (28), and the gradient norm of the error function with respect to the weights are convergent.

Example 2 Four real-world regression problems

In this example, we uses four data sets [21] to compare the performances of BP algorithm and the proposed

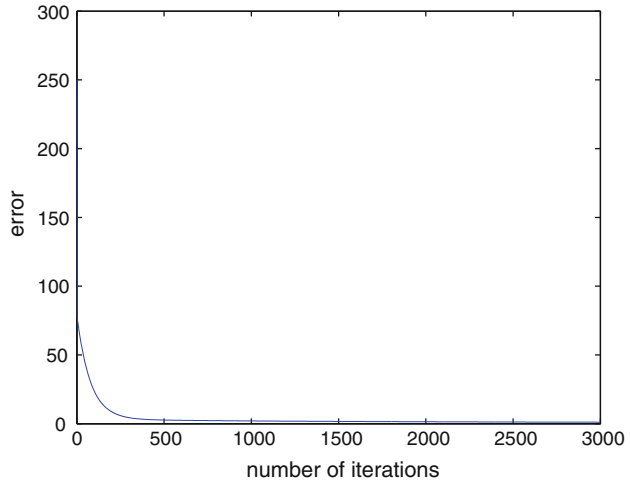

Fig. 1 Value of the error function for Example 1

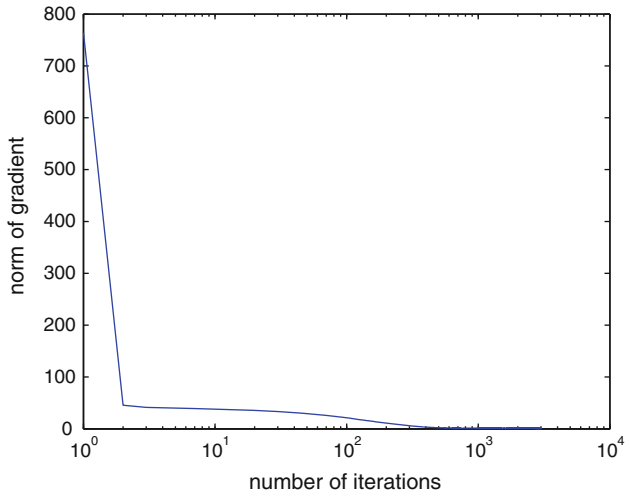

Fig. 2 Norm of gradient of the weights for Example 1

interval extreme learning machine. Some details of the four data sets are given in Table 1.

In order to apply our interval extreme learning machine, we first build the interval data sets from the four data sets. For each data set, each feature z is expanded as an interval $[z - r, z + r]$, where the parameter r is selected randomly from [0,0.1]. In order to let the parameter r to have the same effect for each feature z, all the data have been normalized into [0, 1] before the training process.

First, a special training process is carried out to determine the suitable number of hidden nodes. Following Huang's approach [19], we gradually increase the number of hidden nodes for BP and IELM, respectively, by an interval of 5 and finally select the best number of hidden nodes based on cross-validation method. The learning rates are also chosen to guarantee the convergence of the training process. These parameters are specified in Tables 2 and 3

Table 1 Specification of data sets in Example 2

Data sets	Train	Test	Features
Baskball	64	32	4
Pyrim	49	25	27
Bodyfat	168	84	14
Housing	337	169	13

Table 2 Learning performance of IELM for Example 2

Data sets	Time	Hidden nodes	Learning rate	Training RMSE	Testing RMSE
Baskball	0.9220	40	0.06	0.1260	0.1375
Pyrim	7.5940	100	0.06	0.1705	0.1818
Bodyfat	1.0630	20	0.06	0.1173	0.1219
Housing	3.4220	30	0.06	0.1742	0.2151

Table 3 Learning performance of BP for Example 2

Data sets	Time	Hidden nodes	Learning rate	Training RMSE	Testing RMSE
Baskball	1.4370	5	0.006	0.1234	0.1543
Pyrim	33.9840	10	0.006	0.0678	0.1865
Bodyfat	83.1560	10	0.003	0.0951	0.6484
Housing	185.9370	15	0.0006	0.0876	0.3092

for IELM and BP, respectively. We see that IELM allows bigger learning rate compared with BP.

Then, for each data set, 50 training processes are carried out for the interval extreme learning machine and the BP algorithm, respectively, in which the number of hidden nodes and the learning rate obtained in the first step are applied. The early stopping criterion is applied for BP. The performances of IELM and BP are shown in Tables 2 and 3, respectively. The average learning times of the 50 learning processes are given. From Tables 2 and 3, we can see that, similarly as the results for ELM in Huang [19], although the training root-mean-square error (RMSE) of IELM is a little bit worse than those of the BP algorithm, the training time is less and the testing RMSE is better than those of the BP algorithm. So for the IELM, the learning speed is faster and the generalization performance is better than those of the BP algorithm.

Example 3 A stock prediction problem

In this example, we use the interval neural network to predict Shanghai Securities Composite index by using the data from January 1, 2009 to April 13, 2013, downloaded from http://quotes.money.163.com/trade/lsjysj_zhishu_000001.html. In particular, the input intervals include six intervals: five intervals indicating the highest and lowest

Table 4 Learning performances of BP and IELM for Example 3

Algorithms	Hidden nodes	Time	Training RMSE	Testing RMSE
BP	5	3059.1	0.0649	0.0718
IELM	10	11.9850	0.0711	0.0474

values of successive five days data, respectively, and an interval indicating the mean values of the highest and lowest values of the five days data. The output is an interval indicating the highest and lowest values of the sixth day's index. About 688 samples of the data are chosen as the training samples, and the others 344 samples as testing samples. The choices of the hidden nodes number and the learning rate are the same as in Example 2. The early stopping criterion for BP is also applied.

Similar results as for Example 2 above are obtained for Example 3, as shown in Table 4.

7 Conclusion

The idea of extreme learning machine is applied for interval neural network in this paper, ending up with a so-called interval extreme learning machine (IELM), to deal with the interval data. The input-hidden weights are stochastically chosen as the original ELM. But due to the nonlinearity of the interval arithmetic, we have to use the same idea as that of the BP algorithm to solve a nonlinear optimization problem to determine the hidden-output weights. As observed from numerical experiments, the proposed IELM needs more hidden nodes than the BP algorithm, but it spends less time than the BP algorithm since only the hidden-output weights need to be tuned for IELM. For IELM, although the training RMSE is little bit worse than that of the BP algorithm, the testing RMSE is better than that of the BP algorithm.

Acknowledgments This work is supported by the National Natural Science Foundation of China (11171367) and the Fundamental Research Funds for the Central Universities of China.

References

1. Moore RE (1966) Interval analysis. Prentice-Hall, Englewood Cliffs
2. Sunaga T (1958) Theory of an interval algebra and its applications to numerical analysis. RAAG Mem 2:29–46
3. Yeung DS, Ng WWY, Wang DF, Tsang ECC, Wang XZ (2007) Localized generalization error model and its application to architecture selection for radial basis function neural network. IEEE Trans Neural Netw 18:1294–1305
4. Tsang ECC, Wang XZ, Yeung DS (2000) Improving learning accuracy of fuzzy decision trees by hybrid neural networks. IEEE Trans Fuzzy Syst 8:601–614
5. Ishibuchi H, Tanaka H (1991) An extension of the BP algorithm to interval input vectors. In: Proceedings of IJCNN'91, Singapore, pp 1588–1593
6. Hemandez CA, Espf J, Nakayama K (1993) Interval arithmetic backpropagation. In: Proceedings of 1993 international joint conference on neural networks, Nagoya, pp 375–378
7. Roque AMC, Mate C, Arroyo J, Sarabia A (2007) iMLPApplying multi-layer perceptrons to interval-valued data. Neural Process Lett 25:157–169
8. Chetwynd D, Worden K, Manson G (2006) An application of interval-valued neural networks to a regression problem. Proc R Soc A: Math Phys Eng Sci 462:3097–3114
9. Jeng JT, Chuang CC, Su SF (2003) Support vector interval regression networks for interval regression analysis. Fuzzy Sets Syst 138:283–300
10. Jiang FF, Shen JH, Li XD (2013) The LMI method for stationary oscillation of interval neural networks with three neuron activations under impulsive effects. Nonlinear Anal-Real World Appl 14(3):1404–1416
11. Yang DK, Wu W (2012) A smoothing interval neural network. Discret Dyn Nat Soc 2012. doi:10.1155/2012/456919
12. Rumelhart DE, Hinton GE, Williams RJ (1986) Learning representations by back-propagation errors. Nat Biotechnol 323:533–536
13. Rumelhart DE, McClelland JL et al (1986) Parallel distributed processing. The MIT Press, Cambridge
14. Cao JW, Lin ZP, Huang GB, Liu N (2012) Voting based extreme learning machine. Inf Sci 185:66–77
15. Zhai JH, Xu HY, Wang XZ (2012) Dynamic ensemble extreme learning machine based on sample entropy. Soft Comput 16:1493–1502
16. He Q, Shang TF, Zhuang FZ, Shi ZZ (2013) Parallel extreme learning machine for regression based on MapReduce. Neurocomputing 102:52–58
17. Chacko BP, Vimal Krishnan VR, Raju G, Babu Anto P (2012) Handwritten character recognition using wavelet energy and extreme learning machine. Int J Mach Learn Cybern 3:149–161
18. Wu J, Wang ST, Chung FL (2011) Positive and negative fuzzy rule system, extreme learning machine and image classification. Int J Mach Learn Cybern 2:261–271
19. Huang GB, Zhu QY, Siew CK (2006) Extreme learning machine: theory and applications. Neurocomputing 70:489–501
20. Huang GB, Wang DH (2011) Extreme learning machines: a survey. Int J Mach Learn Cybern 2:107–122
21. Huang GB, Zhou HM, Ding XJ, Zhang R (2012) Extreme learning machine for regression and multiclass classification. IEEE Trans Syst Man Cybern Part B Cybern 42:513–529
22. Shao HM, Zheng GF (2011) Convergence analysis of a back-propagation algorithm with adaptive momentum. Neurocomputing 74:749–752
23. Wu W, Wang J, Cheng MS, Li ZX (2011) Convergence analysis of online gradient method for BP neural networks. Neural Netw 24:91–98
24. Wang J, Yang J, Wu W (2011) Convergence of cyclic and almost-cyclic learning with momentum for feedforward neural networks. IEEE Trans Neural Netw 22:1297–1306