Neural Comput & Applic
DOI 10.1007/s00521-016-2727-4

ORIGINAL ARTICLE

Further improvements on extreme learning machine for interval neural network

Li-fen Yang[1] · **Chong Liu**[2] · **Hao Long**[3] · **Rana Aamir Raza Ashfaq**[4] · **Yu-lin He**[3]

Received: 22 June 2016 / Accepted: 17 November 2016

Abstract The interval extreme learning machine (IELM) (Yang et al. in Neural Comput Appl 27(1):3–8, 2016) is a newly proposed regression algorithm to deal with the data with interval-valued inputs and interval-valued output. In this paper, we firstly analyze the disadvantages of IELM and further point out that IELM is actually a slight variant of fuzzy regression analysis using neural networks (Ishibuchi and Tanaka in Fuzzy Sets Syst 50(3):257–265, 1992). Then, we propose a new interval-valued ELM (IVELM) model to handle the interval-valued data regression. IVELM does not require any iterative adjustment to network weights and thus has the extremely fast training speed. The experimental results on data sets used in (Yang et al. 2016) demonstrate the feasibility and effectiveness of IVELM which obtains the better predictive performance and faster learning speed than IELM.

Keywords Interval-valued data · ELM · Interval ELM · Generalized inverse

✉ Chong Liu
cschongliu@126.com

✉ Yu-lin He
csylhe@126.com; yulinhe@szu.edu.cn

Li-fen Yang
cslfyang@126.com

Hao Long
longhao1@email.szu.edu.cn

Rana Aamir Raza Ashfaq
aamir@bzu.edu.pk

[1] The Foundation Department, Shijiazhuang Vocational College of Finance and Economics, Shijiazhuang 050061, China

[2] Office of Scientific Research and Practical Training, Cangzhou Technical College, Cangzhou 061001, China

[3] College of Computer Science and Software Engineering, Shenzhen University, Shenzhen 518060, China

[4] Department of Computer Science, Bahauddin Zakariya University, Multan, Pakistan

1 Brief introduction to IELM

Interval extreme learning machine (IELM) [7] is used to handle the regression problem based on interval-valued data set S as shown in Table 1,[1] where $s = \langle m, r\rangle$ represents an interval-valued number, m is the midpoint of s, r is the radius of s, $m, r \in \mathrm{R}$, N is the number of instances in S, and D is the number of interval-valued inputs. In order to construct a mapping between interval-valued input space and interval-valued output space, IELM used the gradient descent method to train a single-hidden-layer feed-forward neural network (SLFN) with D input-layer nodes, K hidden-layer nodes, and 1 output-layer node. IELM randomly initialized the input-layer weight matrix

$$\mathrm{W}_{D\times K} = \begin{bmatrix} w_{11} & w_{21} & \cdots & w_{K1} \\ w_{12} & w_{22} & \cdots & w_{K2} \\ \vdots & \vdots & \ddots & \vdots \\ w_{1D} & w_{2D} & \cdots & w_{KD} \end{bmatrix}$$

and hidden-layer bias vector

[1] Although the authors described the regression problem with multiple interval-valued outputs, in fact, they only used IELM to handle the regression problem with single interval-valued output. The experimental data sets used in IELM also confirmed this fact. Thus, there is only one interval-valued output which is considered in the following discussion.

Published online: 08 December 2016

Table 1 Data set S with the interval-valued inputs and interval-valued output in IELM

S	Interval-valued inputs				Interval-valued output
	A_1	A_2	$\cdots$	A_D	Y
x_1	$x_{11} = \langle m_{11}, r_{11} \rangle$	$x_{12} = \langle m_{12}, r_{12} \rangle$	$\cdots$	$x_{1D} = \langle m_{1D}, r_{1D} \rangle$	$y_1 = \langle m_1, r_1 \rangle$
x_2	$x_{21} = \langle m_{21}, r_{21} \rangle$	$x_{22} = \langle m_{22}, r_{22} \rangle$	$\cdots$	$x_{2D} = \langle m_{2D}, r_{2D} \rangle$	$y_2 = \langle m_2, r_2 \rangle$
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
x_N	$x_{N1} = \langle m_{N1}, r_{N1} \rangle$	$x_{N2} = \langle m_{N2}, r_{N2} \rangle$	$\cdots$	$x_{ND} = \langle m_{ND}, r_{ND} \rangle$	$y_N = \langle m_N, r_N \rangle$

$$\mathrm{b} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_K \end{bmatrix},$$

and optimized the output-layer weight vector

$$\beta = \begin{bmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_K \end{bmatrix}$$

with the gradient descent method as

$$\beta^{(i+1)} = \beta^{(i)} + \Delta\beta^{(i)}, \tag{1}$$

where

$$\Delta\beta^{(i)} = -\eta \frac{\partial E\left(\beta^{(i)}\right)}{\partial \beta} \tag{2}$$

is the output-layer weight vector update in the ith iteration, $\eta > 0$ is a constant learning rate,

$$E(\beta) = \sum_{n=1}^{N} \left[(m_n - m'_n)^2 + (r_n - r'_n)^2 \right] \tag{3}$$

is the cost function, m'_n and r'_n are, respectively, the predictive midpoint and radius corresponding to the nth training instance x_n $(n = 1, 2, \ldots, N)$:

$$m'_n = \sum_{k=1}^{K} \beta_k h_{nk}^{(m)} \tag{4}$$

and

$$r'_n = \sum_{k=1}^{K} |\beta_k| h_{nk}^{(r)}, \tag{5}$$

$$\mathrm{H}_{N\times K}^{(m)} = \begin{bmatrix} h_{11}^{(m)} & h_{12}^{(m)} & \cdots & h_{1K}^{(m)} \\ h_{21}^{(m)} & h_{22}^{(m)} & \cdots & h_{2K}^{(m)} \\ \vdots & \vdots & \ddots & \vdots \\ h_{N1}^{(m)} & h_{N2}^{(m)} & \cdots & h_{NK}^{(m)} \end{bmatrix}$$

is the midpoint matrix of hidden-layer output,

$$\mathrm{H}_{N\times K}^{(r)} = \begin{bmatrix} h_{11}^{(r)} & h_{12}^{(r)} & \cdots & h_{1K}^{(r)} \\ h_{21}^{(r)} & h_{22}^{(r)} & \cdots & h_{2K}^{(r)} \\ \vdots & \vdots & \ddots & \vdots \\ h_{N1}^{(r)} & h_{N2}^{(r)} & \cdots & h_{NK}^{(r)} \end{bmatrix}$$

is the radius matrix of hidden-layer output,

$$h_{nk}^{(m)} = \frac{g\left[f_{nk}^{(m)} + f_{nk}^{(r)}\right] + g\left[f_{nk}^{(m)} - f_{nk}^{(r)}\right]}{2}, \tag{6}$$

$$h_{nk}^{(r)} = \frac{g\left[f_{nk}^{(m)} + f_{nk}^{(r)}\right] - g\left[f_{nk}^{(m)} - f_{nk}^{(r)}\right]}{2}, \tag{7}$$

$$f_{nk}^{(m)} = g\left[\sum_{d=1}^{D} (w_{kd} m_{nd} + b_k)\right], \tag{8}$$

$$f_{nk}^{(r)} = g\left[\sum_{d=1}^{D} (|w_{kd}| r_{nd})\right] \tag{9}$$

$(k = 1, 2, \ldots, K)$, K is the number of hidden-layer nodes of SLFN, and $g(u) = \frac{1}{1+\exp(-u)}$, $u \in (-\infty, +\infty)$ is the sigmoid activation function.

2 Analysis to IELM

From above-mentioned descriptions, we can see that the nature of IELM [7] is different from ELM [3] which avoids the iterative optimization of network weights and uses the randomization and analytic solution to determine the network weights. However, the output-layer weights in IELM are optimized based on the gradient descent method which is time-consuming and easy to cause the over-fitting of IELM.

In fact, Ishibuchi and Tanaka [5] have designed a neural network based interval-valued regression model. They used two neural networks to estimate the lower and upper boundaries of interval-valued numbers, respectively. The input-layer and output-layer weights for each neural network were trained by error back propagation (BP) algorithm [6]. IELM is only a slight variant of Ishibuchi and Tanaka's networks (simplified as ITNetworks). The main differences between IELM and ITNetworks are summarized as follows:

1. ITNetworks used the BP algorithm to update the input-layer and output-layer weights, while there were only the output-layer weights which were updated in IELM;
2. ITNetworks tried to estimate the lower and upper boundaries of interval-valued numbers, while IELM fitted the midpoints and radii of interval-valued numbers. Generally, an interval-valued number s has two different representation forms, i.e., lower-upper boundary form $s = [l, u]$ and midpoint-radius form $s = \langle m, r \rangle$, where.
$$m = \frac{u+l}{2} \quad \text{and} \quad r = \frac{u-l}{2}.$$
3. ITNetworks used two different neural networks, while there was only one SLFN in IELM. Although IELM used one SLFN to construct the interval-valued regression model, in fact, the following nonlinear optimization problem was required to solve in IELM:
$$\begin{array}{ll} \text{minimize:} & \left\|\tilde{\mathrm{H}}\tilde{\beta} - \tilde{\mathrm{Y}}\right\|^2 \\ \text{Subject to:} & [\mathrm{I}_{K\times K} \quad 0_{K\times K}]\tilde{\beta} = \beta \\ & [0_{K\times K} \quad \mathrm{I}_{K\times K}]\tilde{\beta} = |\beta| \end{array}, \tag{10}$$
where
$$\tilde{\mathrm{H}}_{2N\times 2K} = \begin{bmatrix} \mathrm{H}^{(m)}_{N\times K} & 0 \\ 0 & \mathrm{H}^{(r)}_{N\times K} \end{bmatrix}$$
is the hidden-layer output matrix of SLFN,
$$\tilde{\beta}_{2K\times 1} = \begin{bmatrix} \beta \\ |\beta| \end{bmatrix}$$
is the output-layer weights of SLFN, and
$$\tilde{\mathrm{Y}}_{2N\times 1} = \begin{bmatrix} m_1 \\ \vdots \\ m_N \\ r_1 \\ \vdots \\ r_N \end{bmatrix}$$
is the real output vector of SLFN. From Eq. (10), we can find that IELM actually had also used two SLFNs: one with the output-layer weight β is to fit the midpoints and another with the output-layer weight $|\beta|$ to fit the radii.

The aforementioned analysis indicates that there is no obvious difference between IELM and ITNetworks. These two methods all used BP algorithm to update the network weights and the architectures of networks used in IELM and ITNetworks are nearly same. Consequently, the essence of extreme learning, i.e., randomly initializing input-layer weights and analytically solving output-layer weights, cannot be reflected in IELM.

3 An improved interval-valued ELM (IVELM)

In this section, we give an improved interval-valued ELM algorithm simplified as IVELM which uses the principle of ELM to train a SLFN to deal with the interval-valued analysis problem. IVELM trains the interval-valued SLFN based on the interval-valued data set as shown in Table 2, where

$$l_{nd} = m_{nd} - r_{nd} \quad \text{and} \quad u_{nd} = m_{nd} + r_{nd}, \quad d = 1, 2, \ldots, D, \tag{11}$$

$$l_n = m_n - r_n \quad \text{and} \quad u_n = m_n + r_n. \tag{12}$$

The main difference between Tables 1 and 2 is that IELM uses the midpoint-radius form to represent interval-valued number, while IVELM treats the interval-valued number as lower-upper boundary form.

The cost function of IVELM is defined as

$$\bar{E}(\bar{\beta}) = \sum_{n=1}^{N}\left[(l_n - l'_n)^2 + (u_n - u'_n)^2\right], \tag{13}$$

where

$$\bar{\beta} = \begin{bmatrix} \bar{\beta}_1 \\ \bar{\beta}_2 \\ \vdots \\ \bar{\beta}_K \end{bmatrix}$$

is the output-layer weights of IVELM,

$$l'_n = \sum_{k=1}^{K}\bar{\beta}_k h^{(l)}_{nk} = \sum_{k=1}^{K}\left[\bar{\beta}_k g\left[\sum_{d=1}^{D}(w_{kd}l_{nd} + b_k)\right]\right] \tag{14}$$

is the predictive lower boundary of the nth $(n = 1, 2, \ldots, N)$ training instance x_n,

$$u'_n = \sum_{k=1}^{K}\bar{\beta}_k h^{(u)}_{nk} = \sum_{k=1}^{K}\left[\bar{\beta}_k g\left[\sum_{d=1}^{D}(w_{kd}u_{nd} + b_k)\right]\right] \tag{15}$$

is the predictive upper boundary of x_n,

$$\mathrm{H}^{(l)} = \begin{bmatrix} h^{(l)}_{11} & h^{(l)}_{12} & \cdots & h^{(l)}_{1L} \\ h^{(l)}_{21} & h^{(l)}_{22} & \cdots & h^{(l)}_{2L} \\ \vdots & \vdots & \ddots & \vdots \\ h^{(l)}_{N1} & h^{(l)}_{N2} & \cdots & h^{(l)}_{NL} \end{bmatrix}$$

Table 2 Data set S with the interval-valued inputs and interval-valued output in IVELM

S	Interval-valued inputs				Interval-valued output
	A_1	A_2	$\cdots$	A_D	Y
x_1	$\bar{x}_{11} = [l_{11}, u_{11}]$	$\bar{x}_{12} = [l_{12}, u_{12}]$	$\cdots$	$\bar{x}_{1D} = [l_{1D}, u_{1D}]$	$\bar{y}_1 = [l_1, u_1]$
x_2	$\bar{x}_{21} = [l_{21}, u_{21}]$	$\bar{x}_{22} = [l_{22}, u_{22}]$	$\cdots$	$\bar{x}_{2D} = [l_{2D}, u_{2D}]$	$\bar{y}_2 = [l_2, u_2]$
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
x_N	$\bar{x}_{N1} = [l_{N1}, u_{N1}]$	$\bar{x}_{N2} = [l_{N2}, u_{N2}]$	$\cdots$	$\bar{x}_{ND} = [l_{ND}, u_{ND}]$	$\bar{y}_N = [l_N, u_N]$

is the lower boundary matrix of hidden-layer output, and

$$\mathrm{H}^{(u)} = \begin{bmatrix} h_{11}^{(u)} & h_{12}^{(u)} & \cdots & h_{1L}^{(u)} \\ h_{21}^{(u)} & h_{22}^{(u)} & \cdots & h_{2L}^{(u)} \\ \vdots & \vdots & \ddots & \vdots \\ h_{N1}^{(u)} & h_{N2}^{(u)} & \cdots & h_{NL}^{(u)} \end{bmatrix}$$

is the upper boundary matrix of hidden-layer output. Equation (13) can be transformed into another form:

$$\begin{aligned} \bar{E}(\bar{\beta}) &= \sum_{n=1}^{N} (l_n - l'_n)^2 + \sum_{n=1}^{N} (u_n - u'_n)^2 \\ &= \left\|\mathrm{H}^{(l)}\bar{\beta} - \mathrm{Y}^{(l)}\right\|^2 + \left\|\mathrm{H}^{(u)}\bar{\beta} - \mathrm{Y}^{(u)}\right\|^2 \end{aligned}, \tag{16}$$

where

$$\mathrm{Y}^{(l)} = \begin{bmatrix} l_1 \\ l_2 \\ \vdots \\ l_N \end{bmatrix} \quad \text{and} \quad \mathrm{Y}^{(u)} = \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_N \end{bmatrix}.$$

Let $\mathrm{H}^{(l)}\bar{\beta} - \bar{\mathrm{Y}}^{(l)} = 0$ and $\mathrm{H}^{(u)}\bar{\beta} - \bar{\mathrm{Y}}^{(u)} = 0$ in IVELM. Then, we can get

$$\left[\mathrm{H}^{(l)} + \mathrm{H}^{(u)}\right]\bar{\beta} = \left[\mathrm{Y}^{(l)} + \mathrm{Y}^{(u)}\right]. \tag{17}$$

Equation (17) is equivalent to

$$\left[\mathrm{H}^{(l)} + \mathrm{H}^{(u)}\right]^{\mathrm{T}}\left[\mathrm{H}^{(l)} + \mathrm{H}^{(u)}\right]\bar{\beta} = \left[\mathrm{H}^{(l)} + \mathrm{H}^{(u)}\right]^{\mathrm{T}}\left[\mathrm{Y}^{(l)} + \mathrm{Y}^{(u)}\right]. \tag{18}$$

Then, we can solve $\bar{\beta}$ from Eq. (18) as

$$\bar{\beta} = \left[\left[\mathrm{H}^{(l)} + \mathrm{H}^{(u)}\right]^{\mathrm{T}}\left[\mathrm{H}^{(l)} + \mathrm{H}^{(u)}\right]\right]^{-1}\left[\mathrm{H}^{(l)} + \mathrm{H}^{(u)}\right]^{\mathrm{T}}\left[\mathrm{Y}^{(l)} + \mathrm{Y}^{(u)}\right]. \tag{19}$$

Let $\bar{\mathrm{H}} = \mathrm{H}^{(l)} + \mathrm{H}^{(u)}$ and $\bar{\mathrm{Y}} = \mathrm{Y}^{(l)} + \mathrm{Y}^{(u)}$, Eq. (19) can be changed as

$$\bar{\beta} = \left(\bar{\mathrm{H}}^{\mathrm{T}}\bar{\mathrm{H}}\right)^{-1}\bar{\mathrm{H}}^{\mathrm{T}}\bar{\mathrm{Y}}. \tag{20}$$

In order to reduce the likelihood that $\bar{\mathrm{H}}^{\mathrm{T}}\bar{\mathrm{H}}$ becomes a singular matrix [2], we introduce a regularization factor $C > 0$ into Eq. (20) and rewrite Eq. (20) into

$$\bar{\beta} = \left(\bar{\mathrm{H}}^{\mathrm{T}}\bar{\mathrm{H}} + C\mathrm{I}_{K\times K}\right)^{-1}\bar{\mathrm{H}}^{\mathrm{T}}\bar{\mathrm{Y}}. \tag{21}$$

Equation (21) is the updating rule for the output-layer weights of our proposed IVELM.

For the unseen instance

$$\begin{aligned} \hat{\mathrm{x}} &= (\bar{x}_1, \bar{x}_2, \ldots, \bar{x}_D) \\ &= \left[\left[\hat{l}_1, \hat{u}_1\right], \left[\hat{l}_2, \hat{u}_2\right], \ldots, \left[\hat{l}_D, \hat{u}_D\right]\right] \end{aligned},$$

IVELM predicts its output as

$$\begin{aligned} \bar{y} &= [l', u'] \\ &= \left[\left[h_1^{(l)}, h_2^{(l)}, \ldots, h_K^{(l)}\right]\bar{\beta}, \left[h_1^{(u)}, h_2^{(u)}, \ldots, h_K^{(u)}\right]\bar{\beta}\right] \end{aligned}, \tag{22}$$

where

$$h_k^{(l)} = g\left[\sum_{d=1}^{D}\left[w_{kd}\hat{l}_d + b_k\right]\right] \tag{23}$$

and

$$h_k^{(u)} = g\left[\sum_{d=1}^{D}\left[w_{kd}\hat{u}_d + b_k\right]\right]. \tag{24}$$

It should be noted that we derive $\bar{\beta}$ as shown in Eq. (10) based on the assumption that $\bar{\beta}_k > 0$. In fact, the true calculation of $\bar{\beta}$ may include the negative component, i.e., $\exists k' \in \{1, 2, \ldots, K\}$, $\bar{\beta}_{k'} < 0$, which will cause $l' > u'$. In this case, we use the following method studied by Ishibuchi et al. [4] and He et al. [1] to adjust the predictive interval-valued output:

$$l' = \min\{l', u'\} \quad \text{and} \quad u' = \max\{l', u'\}.$$

4 Experimental validation

This section provides the experimental comparison to demonstrate the effectiveness of our proposed IVELM. Four interval-valued data sets which were used in [7] to test the predictive performances of IELM are also employed in this study. These four interval-valued data sets are transformed from four standard UCI regression data sets. For each data set, each attribute A which should be firstly normalized into $[0, 1]$

Table 3 The predictive performances of IVELM on four interval-valued data sets used in IELM [7]

Data set	Training RMSE	Testing RMSE	Training time	Testing time	(*K*, *C*)
Baskball	0.0290 ± 0.0007	0.0291 ± 0.0076	0.0046	0.0001	(160, −5)
Pyrim	0.0179 ± 0.0021	0.0210 ± 0.0130	0.0025	0.0007	(70, −5)
Bodyfat	0.0003 ± 0.0000	0.0002 ± 0.0002	0.0073	0.0010	(180, −8)
Housing	0.0121 ± 0.0009	0.0128 ± 0.0056	0.0070	0.0012	(120, −8)

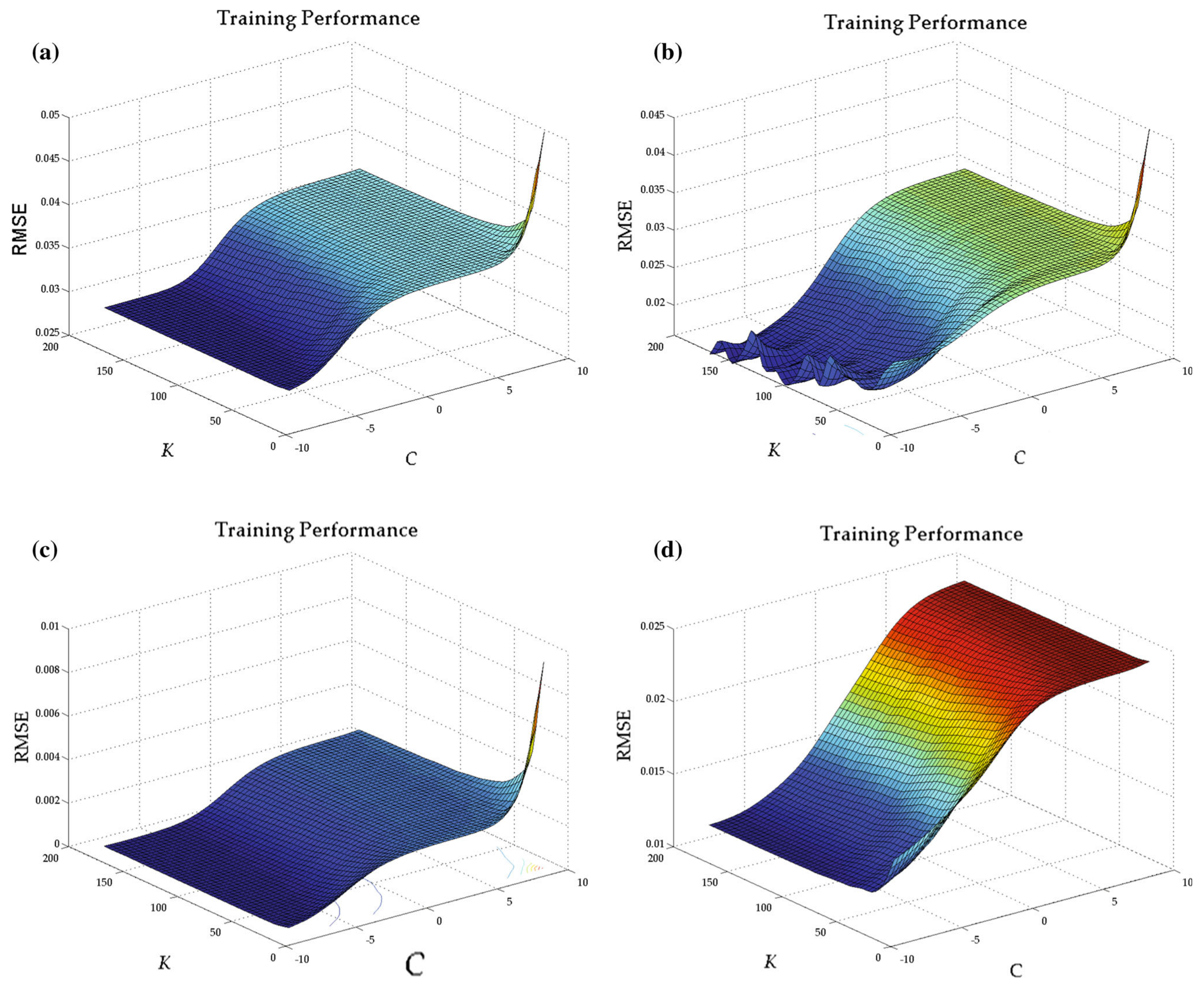

Fig. 1 Impact of different pairs of (*K*, *C*) on training performances of IVELM. **a** On *Baskball* data set. **b** On *Pyrim* data set. **c** On *Bodyfat* data set. **d** On *Housing* data set

is extended into an interval $[A - r_A, A + r_A]$, where the radius r_A is randomly selected from $[0, 0.1]$.

For the data set S, the predictive performance of IVELM is measured with root mean square error (RMSE) which is calculated as

$$\bar{E} = \sqrt{\frac{\sum_{n=1}^{N}\left[(l_n - l'_n)^2 + (u_n - u'_n)^2\right]}{N}}, \tag{25}$$

where l_n and u_n are the real lower and upper boundaries of the nth $(n = 1, 2, \ldots, N)$ training instance x_n, l'_n and u'_n are the predictive lower and upper boundaries. The procedure of ten times threefold cross-validation is used to obtain the experimental results as shown in Table 3, i.e., training RMSE, testing RMSE, training time, and testing time. For our IVELM, we test its predictive performances corresponding to 400 pairs of parameters (K, C), i.e., $K = \{10, 20, \ldots, 190, 200\}$ and $C = \{2^{-9}, 2^{-8}, \ldots, 2^{9}, 2^{10}\}$, and

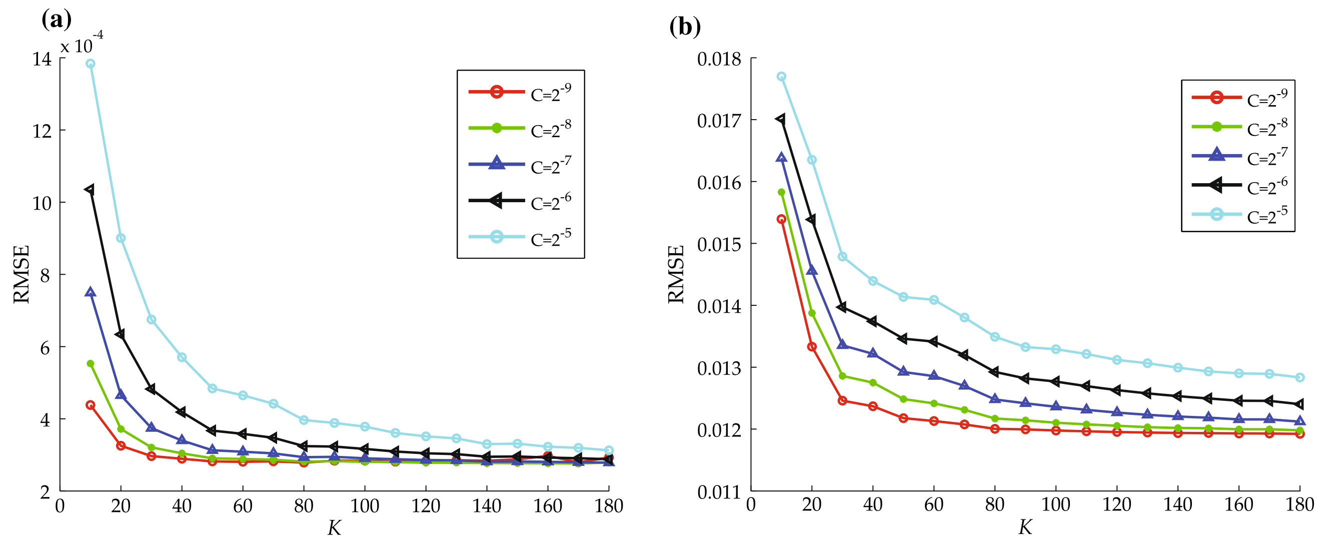

Fig. 2 Convergence of IVELM's training performances. **a** On *Bodyfat* data set. **b** On *Housing* data set

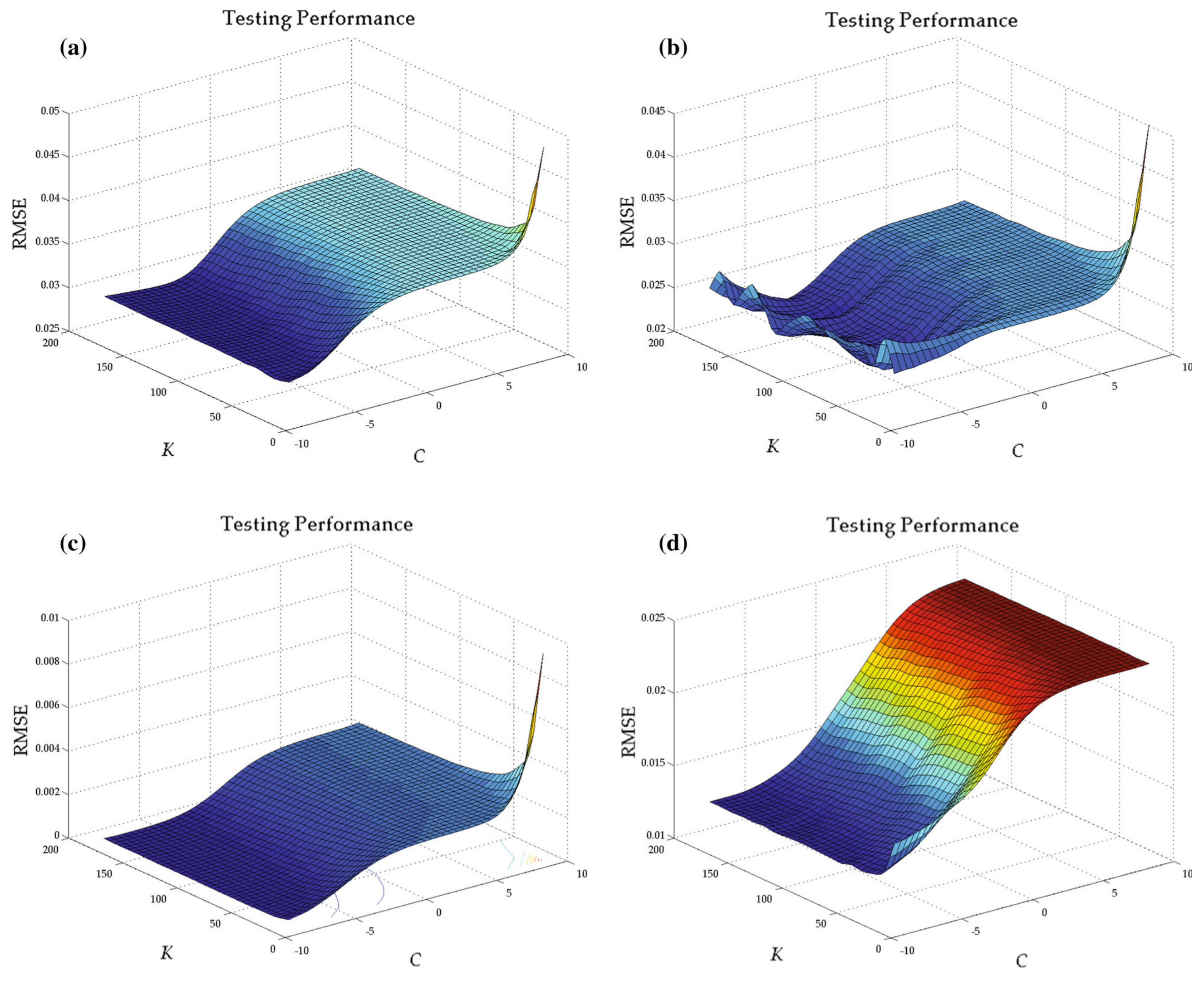

Fig. 3 Impact of different pairs of $(K,\ C)$ on testing performances of IVELM. **a** On *Baskball* data set. **b** On *Pyrim* data set. **c** On *Bodyfat* data set. **d** On *Housing* data set

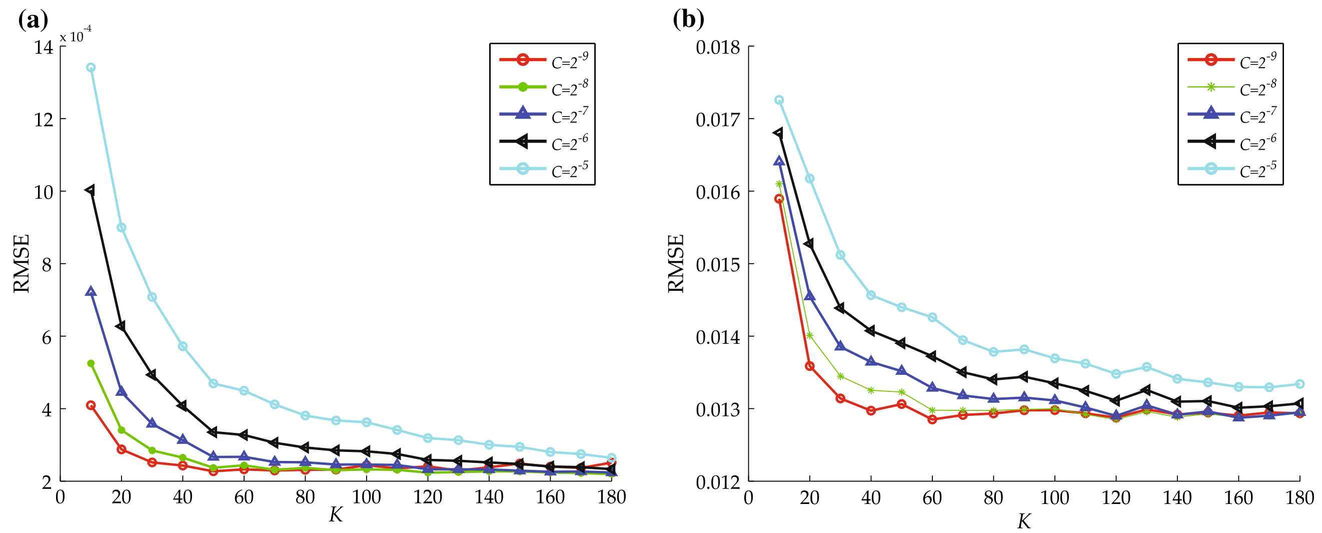

Fig. 4 Convergence of IVELM's testing performances. **a** On *Bodyfat* data set. **b** On *Housing* data set

select the best combinations of (K, C) for each interval-valued data set. The input-layer weights and hidden-layer biases of IVELM are selected from the random interval $[0, 1]$. Figures 1 and 3 show the impacts of different pairs of (K, C) on the training and testing RMSEs, respectively. From Figs. 1 and 3, we can see that the smaller C usually leads to the better training and testing RMSEs for IVELM. The experimental results as shown in Table 3 indicate that IVELM obtains the better testing RMSEs than IELM [7]: 0.0291 versus 0.1375 on *Baskball*, 0.0210 versus 0.1818 on *Pyrim*, 0.0002 versus 0.1219 on *Bodyfat*, and 0.0128 versus 0.2151 on *Housing*. In addition, the training time of IVELM is far less than IELM: 0.0046 versus 0.9220 on *Baskball*, 0.0025 versus 7.5940 on *Pyrim*, 0.0073 versus 1.0630 on *Bodyfat*, and 0.0070 versus 3.4220 on *Housing*. The main reason IVELM is faster than IELM is that IVELM does not need any iterative adjustment to the network weights (Fig. 2).

In addition, we also check the convergence of IVELM in our experiments. The experimental results are summarized in Figs. 2 and 4. We fix C and test RMSE with the change of K, i.e., the number of hidden-layer nodes in IVELM, on two representative data sets. In Figs. 2 and 4, we can see that the training and testing RMSEs gradually decrease with the increase of hidden-layer nodes. This observation demonstrates that IVELM is convergent, i.e., when $K \to +\infty$, $\bar{E} \to 0$ for the given C.

5 Conclusion and future works

In this paper, we firstly analyzed the disadvantages of IELM [7] which is a newly proposed interval-valued regression algorithm. Our main finding is that IELM does not integrate the essence of ELM into the design of regression model and still uses the gradient descent method to train the interval-valued neural network. Consequently, IELM is only a slight variant of fuzzy regression analysis using neural networks [5]. Secondly, we used a single-hidden-layer feed-forward neural network (SLFN) to deal with the interval-valued regression problems and proposed a new interval-valued ELM (IVELM) to train this SLFN. IVELM randomly selected the input-layer weights/hidden-layer biases of SLFN and analytically solved the output-layer weights with the newly derived updating rule. Finally, we conducted the experimental comparison to validate the feasibility and effectiveness of IVELM. The experimental results showed that our proposed IVELM can obtain the better predictive performances and faster learning speed than IELM. Our future works along this direction include (1) proving the convergence of IVELM and (2) extending IVELM to handle the multi-output interval-valued regression problem.

Acknowledgements We thank the editor-in-chief and two anonymous reviewers for their valuable comments and suggestions. This work is supported by National Natural Science Foundations of China (61503252) and China Postdoctoral Science Foundations (2015M572361 and 2016T90799).

References

1. He YL, Wang XZ, Huang JZX (2016) Fuzzy nonlinear regression analysis using a random weight network. Inf Sci 364–365:222–240
2. Hoerl AE, Kennard RW (2000) Ridge regression: biased estimation for nonorthogonal problems. Technometrics 42(1):80–86
3. Huang GB, Zhu QY, Siew CK (2006) Extreme learning machine: theory and applications. Neurocomputing 70(1):489–501

4. Ishibuchi H, Kwon K, Tanaka H (1995) A learning algorithm of fuzzy neural networks with triangular fuzzy weights. Fuzzy Sets Syst 71(3):277–293
5. Ishibuchi H, Tanaka H (1992) Fuzzy regression analysis using neural networks. Fuzzy Sets Syst 50(3):257–265
6. Rumelhart DE, Hinton GE, Williams RG (1986) Learning representations by back-propagating errors. Nature 323:533–536
7. Yang D, Li Z, Wu W (2016) Extreme learning machine for interval neural networks. Neural Comput Appl 27(1):3–8