Information Sciences 364–365 (2016) 222–240

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Fuzzy nonlinear regression analysis using a random weight network

Yu-Lin He, Xi-Zhao Wang*, Joshua Zhexue Huang*

Big Data Institute, College of Computer Science & Software Engineering, Shenzhen University, Shenzhen 518060, Guangdong, China

ARTICLE INFO

Article history:
Received 28 May 2015
Revised 11 December 2015
Accepted 12 January 2016
Available online 28 January 2016

Keywords:
α-cut set
Fuzzy-in fuzzy-out
Fuzzy nonlinear regression
Random weight network
Triangular fuzzy number

ABSTRACT

Modeling a fuzzy-in fuzzy-out system where both inputs and outputs are uncertain is of practical and theoretical importance. Fuzzy nonlinear regression (FNR) is one of the approaches used most widely to model such systems. In this study, we propose the use of a Random Weight Network (RWN) to develop a FNR model called FNR_{RWN}, where both the inputs and outputs are triangular fuzzy numbers. Unlike existing FNR models based on back-propagation (BP) and radial basis function (RBF) networks, FNR_{RWN} does not require iterative adjustment of the network weights and biases. Instead, the input layer weights and hidden layer biases of FNR_{RWN} are selected randomly. The output layer weights for FNR_{RWN} are calculated analytically based on a derived updating rule, which aims to minimize the integrated squared error between α-cut sets that correspond to the predicted fuzzy outputs and target fuzzy outputs, respectively. In FNR_{RWN}, the integrated squared error is solved approximately by Riemann integral theory. The experimental results show that the proposed FNR_{RWN} method can effectively approximate a fuzzy-in fuzzy-out system. FNR_{RWN} obtains better prediction accuracy in a lower computational time compared with existing FNR models based on BP and RBF networks.

1. Introduction

Fuzzy regression analysis is a powerful method for forecasting the fuzzy outputs of an uncertain system. According to Wang and Tsaur in their study entitled "*Insight of A Fuzzy Regression Model*", *fuzzy regression can be quite useful in estimating the relationships among variables where the available data are very limited and imprecise, and variables are interacting in an uncertain, qualitative, and fuzzy way* [40]. During recent decades, fuzzy regression has found many applications in various areas, such as business management, engineering, economics, sociology, and biological science. Fuzzy regression can be divided into two categories: fuzzy linear regression (FLR) and fuzzy nonlinear regression (FNR).

FLR was first studied by Tanaka et al. [35] in 1982, who used a fuzzy linear function with crisp inputs, fuzzy outputs, and fuzzy coefficients to approximate an uncertain system. Subsequently, their FLR model was extended by [35] to handle fuzzy regression tasks with fuzzy inputs and fuzzy outputs. Improvements were made in two areas: constructing fuzzy linear functions with crisp coefficients [2,7,8,12,20,21] and making fuzzy linear functions with fuzzy coefficients [8,15,30,31,38,42,43,45]. Linear programming and least squares are used widely to solve the crisp or fuzzy coefficients for FLR models. However, for a crisp-in fuzzy-out or fuzzy-in fuzzy-out system, the relationship between the inputs and outputs is usually nonlinear in

* Corresponding authors
E-mail addresses: csylhe@126.com, yulinhe@szu.edu.com (Y.-L. He), xzwang@szu.edu.cn (X.-Z. Wang), zx.huang@szu.edu.cn (J. Z.-X. Huang).

http://dx.doi.org/10.1016/j.ins.2016.01.037
0020-0255/

many applications, i.e., the fuzzy outputs cannot be expressed simply as the weighted sum of the crisp or fuzzy inputs; thus, a more sophisticated FNR approach is required. Due to their better capacity for approximating nonlinear functions, feed-forward neural networks are usually employed for constructing complex FNR models.

FNR models based on feed-forward neural networks include the following two categories:

1. Back-propagation (BP) network-based FNR models. In 1992, Ishibuchi and Tanaka [17] proposed an FNR model ($FNR_{BP\text{-}I}$) that uses two BP networks to fit the upper and lower bounds of interval-valued fuzzy numbers. These two BP networks with crisp weights and biases are trained using the standard BP algorithm [29]. Both the inputs and outputs of $FNR_{BP\text{-}I}$ are interval-valued fuzzy numbers. In 1993, Ishibuchi and Tanaka proposed another BP network-based FNR model ($FNR_{BP\text{-}II}$) [18] that handles the FNR problem using crisp inputs and fuzzy outputs. Unlike $FNR_{BP\text{-}I}$, there is only one BP network in $FNR_{BP\text{-}II}$. The weights and biases in this BP network are interval-valued fuzzy numbers. In 1995, Ishibuchi et al. [19] proposed the use of a BP network with triangular fuzzy number (TFN) weights to conduct the fuzzy regression analysis ($FNR_{BP\text{-}III}$), where the inputs and outputs are interval-valued fuzzy numbers.
2. Radial basis function (RBF) network-based FNR models. In 2001, Cheng and Lee [6] used an RBF network to design an FNR model ($FNR_{RBF\text{-}I}$). In $FNR_{RBF\text{-}I}$, the inputs and centers of the input layer nodes are crisp, whereas the outputs and output layer weights are TFNs. Compared with previous BP network-based FNR models, $FNR_{RBF\text{-}I}$ obtained a faster convergence rate. A fuzzified RBF network-based FNR model ($FNR_{RBF\text{-}II}$) was described by Zhang et al. [49] in 2005. In $FNR_{RBF\text{-}II}$, the inputs, outputs, centers, deviations, and weights are Left–Right (L–R) fuzzy numbers. $FNR_{RBF\text{-}II}$ can serve as a universal function approximation for any continuous fuzzy function defined on a compact set.

The main problems with these FNR models based on BP and RBF networks include their high training complexity, local minima, and complex parameter tuning (e.g., learning rate, learning epochs, and stopping criteria). Optimization of the neural network parameters, e.g., the weights, biases, node centers, and node deviations in $FNR_{BP\text{-}I}$, $FNR_{BP\text{-}II}$, $FNR_{BP\text{-}III}$, $FNR_{RBF\text{-}I}$, and $FNR_{RBF\text{-}II}$, is based on gradient descent approaches where the learning speed is relatively slow. Moreover, the gradient-based methods may readily converge to a local minimum. In addition, there are no widely accepted methods for determining the optimal learning rate, learning epochs, and stopping criteria for BP and RBF networks at present. Thus, trial-and-error methods are often used to select these parameters. These methods require large periods of computational time to establish BP and RBF network-based FNR models.

Thus, in the present study, we develop a new FNR learning algorithm that is faster with high generalization performance, as well as avoiding many of the difficulties that affect the traditional gradient-based FNR models. In contrast to the conventional training algorithms for BP and RBF networks, Random Weight Networks (RWNs) [3,5,34,51] do not require iterative adjustments of the network weights and there is no learning parameter to determine. Thus, the training speed of RWNs can be thousands of times faster than traditional gradient descent algorithms. In addition, the good generalization capacity of RWNs has been demonstrated in recent studies [3,5,51]. Therefore, we propose a RWN-based FNR model called FNR_{RWN} in the present study. FNR_{RWN} is a single hidden layer feed-forward neural network, where the inputs and outputs are TFNs. The input layer weights and hidden layer biases of FNR_{RWN} are selected randomly. In order to analytically calculate the output layer weights, we define a new computational paradigm to minimize the integrated squared error between α-cut sets that correspond to the predicted fuzzy outputs and target fuzzy outputs. Our simulation results indicate that FNR_{RWN} has better generalization performance as well as requiring less training time compared with $FNR_{BP\text{-}III}$ and $FNR_{RBF\text{-}II}$. Overall, our results demonstrate that FNR_{RWN} can effectively approximate a fuzzy-in fuzzy-out system.

The remainder of this paper is organized as follows. In Section 2, we provide a brief introduction to TFNs. In Section 3, we describe the BP network and RBF network-based FNR models. The RWN-based FNR model (FNR_{RWN}) is presented in Section 4. In Section 5, we report experimental comparisons that demonstrate the feasibility and effectiveness of FNR_{RWN}. Finally, we give our conclusions and suggestions for further research in Section 6.

2. TFNs and their mathematical operations

2.1. Definition of a TFN

Definition 1 [13,23]**.** A fuzzy number A is defined as a fuzzy set on the domain of real numbers $\Re$, which satisfies the following conditions:

1. $\exists\ x_0 \in \Re,\ \mu_A(x_0) = 1$, where $\mu_A(x)$ is the membership function of fuzzy set A;
2. $\forall\ \alpha \in (0, 1],\ A_\alpha = \{x | x \in \Re, \mu_A(x) \geq \alpha\}$ is a finite closed interval.

The TFN A is the most popular fuzzy number, which is represented by two endpoints a_1 and a_3 and one peak-point a_2

$$A = (a_1, a_2, a_3).$$

It can be interpreted as a membership function

Table 1
Data set $\mathbb{D}$ where the inputs and outputs of instances are represented by TFNs.

$\mathbb{D}$	Fuzzy inputs				Fuzzy outputs
D_1	$X_{11}=(x_{111},x_{112},x_{113})$	$X_{12}=(x_{121},x_{122},x_{123})$	$\cdots$	$X_{1D}=(x_{1D1},x_{1D2},x_{1D3})$	$Y_1=(y_{11},y_{12},y_{13})$
D_2	$X_{21}=(x_{211},x_{212},x_{213})$	$X_{22}=(x_{221},x_{222},x_{223})$	$\cdots$	$X_{2D}=(x_{2D1},x_{2D2},x_{2D3})$	$Y_2=(y_{21},y_{22},y_{23})$
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
D_N	$X_{N1}=(x_{N11},x_{N12},x_{N13})$	$X_{N2}=(x_{N21},x_{N22},x_{N23})$	$\cdots$	$X_{ND}=(x_{ND1},x_{ND2},x_{ND3})$	$Y_N=(y_{N1},y_{N2},y_{N3})$

$$\mu_A(x)=\begin{cases}\dfrac{x-a_1}{a_2-a_1}, & a_1\le x\le a_2\\ \dfrac{a_3-x}{a_3-a_2}, & a_2\le x\le a_3\\ 0, & x>a_3 \text{ or } x<a_1\end{cases}. \tag{1}$$

The α-cut operation on TFN A can generate an α-cut interval

$$A_\alpha=[a_L,a_U],$$

where $a_L=\alpha a_2+(1-\alpha)a_1$ and $a_U=\alpha a_2+(1-\alpha)a_3$. In this study, we focus on handling the FNR problem on the data set $\mathbb{D}$, as shown in Table 1, where each instance is represented as a fuzzy vector with D fuzzy inputs $(X_{n1},X_{n2},\ldots,X_{nD})$ and one fuzzy output Y_n, and both $X_{nj}(n=1,2,\ldots,N;\ j=1,2,\ldots,D)$ and Y_n are TFNs.

2.2. Mathematical operations using TFNs

Some important operations using TFNs are summarized as follows. Assume that there are two TFNs $A=(a_1,a_2,a_3)$ and $B=(b_1,b_2,b_3)$.

- Addition:

$$A(+)B=(a_1,a_2,a_3)(+)(b_1,b_2,b_3)=(a_1+b_1,a_2+b_2,a_3+b_3); \tag{2}$$

- Subtraction:

$$A(-)B=(a_1,a_2,a_3)(-)(b_1,b_2,b_3)=(a_1-b_3,a_2-b_2,a_3-b_1); \tag{3}$$

- Multiplication by a real number r:

$$r(\times)A=r(\times)(a_1,a_2,a_3)=\begin{cases}(ra_1,ra_2,ra_3), & r\ge 0\\ (ra_3,ra_2,ra_1), & r<0\end{cases}; \tag{4}$$

- Multiplication of A and B: The multiplication operation of two TFNs can be performed by using their α-cut intervals. The product $A\ (\times)\ B$ is a L-R fuzzy number. From above-mentioned analysis, we know that the α-cut intervals of A and B are

$$A_\alpha=[\alpha a_2+(1-\alpha)a_1,\alpha a_2+(1-\alpha)a_3]$$

and

$$B_\alpha=[\alpha b_2+(1-\alpha)b_1,\alpha b_2+(1-\alpha)b_3],$$

respectively. Then, the product of A_α and B_α is

$$\begin{aligned}A_\alpha(\times)B_\alpha&=[[\alpha a_2+(1-\alpha)a_1][\alpha b_2+(1-\alpha)b_1],[\alpha a_2+(1-\alpha)a_3][\alpha b_2+(1-\alpha)b_3]]\\&=\left[\gamma_{L1}\alpha^2+\gamma_{L2}\alpha+\gamma_{L3},\gamma_{U1}\alpha^2+\gamma_{U2}\alpha+\gamma_{U3}\right],\quad a_i,b_i>0,\quad i=1,2,3,\end{aligned}$$

where $\gamma_{L1}=(a_2-a_1)(b_2-b_1)$, $\gamma_{L2}=a_1b_2+a_2b_1-2a_1b_1$, $\gamma_{L3}=a_1b_1$, $\gamma_{U1}=(a_2-a_3)(b_2-b_3)$, $\gamma_{U2}=a_3b_2+a_2b_3-2a_3b_3$, and $\gamma_{U3}=a_3b_3$. Let $x=\gamma_{L1}\alpha^2+\gamma_{L2}\alpha+\gamma_{L3}$, we can calculate the left reference function of $A\ (\times)\ B$

$$\alpha=\frac{-\gamma_{L2}+\sqrt{\gamma_{L2}^2-4\gamma_{L1}(\gamma_{L3}-x)}}{2\gamma_{L1}}\ge 0,\quad a_1b_1\le x<a_2b_2.$$

Similarly, let $x=\gamma_{U1}\alpha^2+\gamma_{U2}\alpha+\gamma_{U3}$, we can get the right reference function of $A\ (\times)\ B$

$$\alpha=\frac{-\gamma_{U2}-\sqrt{\gamma_{U2}^2-4\gamma_{U1}(\gamma_{U3}-x)}}{2\gamma_{U1}}\le 1,\quad a_2b_2\le x\le a_3b_3.$$

Then, the membership function of A $(\times)$ B can be represented as

$$\mu_{A(\times)B}(x)=\begin{cases}\dfrac{-\gamma_{L2}+\sqrt{\gamma_{L2}^2-4\gamma_{L1}(\gamma_{L3}-x)}}{2\gamma_{L1}}, & a_1b_1\le x<a_2b_2\\ \dfrac{-\gamma_{U2}-\sqrt{\gamma_{U2}^2-4\gamma_{U1}(\gamma_{U3}-x)}}{2\gamma_{U1}}, & a_2b_2\le x\le a_3b_3\end{cases} \tag{5}$$

From Eq. (5), we know that A $(\times)$ B is not a TFN.

- Division of A and B: We also use the α-cut intervals of A and B to determine the membership function of $A(\div)B$ which is a L-R fuzzy number. The quotient of A_α and B_α is

$$A_\alpha(\div)B_\alpha=\left[\frac{\alpha a_2+(1-\alpha)a_1}{\alpha b_2+(1-\alpha)b_3},\frac{\alpha a_2+(1-\alpha)a_3}{\alpha b_2+(1-\alpha)b_1}\right],\quad a_i,b_i>0,\quad i=1,2,3.$$

Let $x=\frac{\alpha a_2+(1-\alpha)a_1}{\alpha b_2+(1-\alpha)b_3}$, we can derive the left reference function of $A(\div)B$

$$\alpha=\frac{b_3x-a_1}{(a_2-a_1)+(b_3-b_2)x},\quad \frac{a_1}{b_3}\le x<\frac{a_2}{b_2}.$$

Similarly, let $x=\frac{\alpha a_2+(1-\alpha)a_3}{\alpha b_2+(1-\alpha)b_1}$, we can get the right reference function of A $(\div)$ B

$$\alpha=\frac{b_1x-a_3}{(a_2-a_3)+(b_1-b_2)x},\quad \frac{a_2}{b_2}\le x\le\frac{a_3}{b_1}.$$

Then, the membership function of $A(\div)B$ can be expressed as

$$\mu_{A(\div)B}(x)=\begin{cases}\dfrac{b_3x-a_1}{(a_2-a_1)+(b_3-b_2)x}, & \dfrac{a_1}{b_3}\le x<\dfrac{a_2}{b_2}\\ \dfrac{b_1x-a_3}{(a_2-a_3)+(b_1-b_2)x}, & \dfrac{a_2}{b_2}\le x\le\dfrac{a_3}{b_1}\end{cases}. \tag{6}$$

One can see from Eq. (6) that $A(\div)B$ is also not a TFN.

3. FNR analysis

A FNR model attempts to find the functional relationship between fuzzy inputs $X_{n1}, X_{n2}, \ldots, X_{nD}$ and the fuzzy output Y_n, i.e., learning an optimal fuzzy function $f^*(\bullet)$ that minimizes the total error between the predicted fuzzy outputs and target fuzzy outputs. This process can be expressed as the following equation:

$$f^*=\arg\min_f\frac{1}{N}\sum_{n=1}^{N}\|f(X_{n1},X_{n2},\ldots,X_{nD})-Y_n\|^2. \tag{7}$$

In Eq. (7), the error between two TFNs $A=(a_1,a_2,a_3)$ and $B=(b_1,b_2,b_3)$ is measured as follows [2,7]:

$$\|A-B\|^2=(a_1-b_1)^2+(a_2-b_2)^2+(a_3-b_3)^2. \tag{8}$$

Neural networks have the capacity to fit complex nonlinear functions, so they are often selected to construct the fuzzy function $f(\bullet)$ in Eq. (7), where representative methods include BP network-based FNR ($\text{FNR}_{\text{BP-III}}$ [19]) and RBF network-based FNR ($\text{FNR}_{\text{RBF-II}}$ [49]).

1. $\text{FNR}_{\text{BP-III}}$ is an improved version of $\text{FNR}_{\text{BP-I}}$, which uses a fuzzy BP network to handle the FNR problem with TFN inputs and TFN outputs. The weights and biases of the fuzzy BP network in $\text{FNR}_{\text{BP-III}}$ are symmetric TFNs. Let $W_{ji}=(w_{ji1},w_{ji2},w_{ji3})$ denote the weight between the ith $(1\le i\le D)$ input layer node and the jth $(1\le j\le K)$ hidden layer node, $\beta_j=(\beta_{j1},\beta_{j2},\beta_{j3})$ is the weight between the jth hidden layer node and output layer node, $b_j=(b_{j1},b_{j2},b_{j3})$ is the bias of the jth hidden layer node, and $b_0=(b_{01},b_{02},b_{03})$ is the bias of output layer node. $[A]_h=[[A]_h^L,[A]_h^U]$ is the h-level set of TFN A [19] (i.e., h-cut set, $0<h\le 1$). For the nth $(1\le n\le N)$ instance in data set $\mathbb{D}$, the input and output of the jth hidden layer node are

$$[R_j]_h=\sum_{i=1}^{D}[X_{ni}]_h[W_{ji}]_h+[b_j]_h=\left[\sum_{i=1}^{D}\left[[X_{ni}]_h[W_{ji}]_h\right]^L+[b_j]_h^L,\sum_{i=1}^{D}\left[[X_{ni}]_h[W_{ji}]_h\right]^U+[b_j]_h^U\right]=\left[[R_j]_h^L,[R_j]_h^U\right] \tag{9}$$

and

$$[H_j]_h=g\left[[R_j]_h\right]=g\left[[R_j]_h^L,[R_j]_h^U\right]=\left[g\left[[R_j]_h^L\right],g\left[[R_j]_h^U\right]\right]=\left[[H_j]_h^L,[H_j]_h^U\right], \tag{10}$$

respectively, where $g(u) = \frac{1}{1+e^{-u}}$ is the sigmoid activation function and

$$\begin{cases} \left[[X_{ni}]_h[W_{ji}]_h\right]^L = \min\left\{[X_{ni}]_h^L[W_{ji}]_h^L, [X_{ni}]_h^L[W_{ji}]_h^U, [X_{ni}]_h^U[W_{ji}]_h^L, [X_{ni}]_h^U[W_{ji}]_h^U\right\} \\ \left[[X_{ni}]_h[W_{ji}]_h\right]^U = \max\left\{[X_{ni}]_h^L[W_{ji}]_h^L, [X_{ni}]_h^L[W_{ji}]_h^U, [X_{ni}]_h^U[W_{ji}]_h^L, [X_{ni}]_h^U[W_{ji}]_h^U\right\} \end{cases}.$$

The predicted output of the fuzzy BP network is

$$[T_n]_h = \sum_{j=1}^{K}[H_j]_h[\beta_j]_h + [b_0]_h = \left[\sum_{j=1}^{K}\left[[H_j]_h[\beta_j]_h\right]^L + [b_0]_h^L, \sum_{j=1}^{K}\left[[H_j]_h[\beta_j]_h\right]^U + [b_0]_h^U\right] = \left[[T_n]_h^L, [T_n]_h^U\right], \tag{11}$$

where

$$\begin{cases} \left[[H_j]_h[\beta_j]_h\right]^L = \min\left\{[H_j]_h^L[\beta_j]_h^L, [H_j]_h^L[\beta_j]_h^U, [H_j]_h^U[\beta_j]_h^L, [H_j]_h^U[\beta_j]_h^U\right\} \\ \left[[H_j]_h[\beta_j]_h\right]^U = \max\left\{[H_j]_h^L[\beta_j]_h^L, [H_j]_h^L[\beta_j]_h^U, [H_j]_h^U[\beta_j]_h^L, [H_j]_h^U[\beta_j]_h^U\right\} \end{cases}.$$

The error between the predicted outputs and target outputs is measured as

$$E = \frac{h}{2}\sum_{n=1}^{N}\|[T_n]_h - [Y_n]_h\|^2 = \frac{h}{2}\sum_{n=1}^{N}\left[\left[[T_n]_h^L - [Y_n]_h^L\right]^2 + \left[[T_n]_h^U - [Y_n]_h^U\right]^2\right]. \tag{12}$$

By solving equations $\frac{\partial E}{\partial w_{ji1}} = 0$, $\frac{\partial E}{\partial w_{ji3}} = 0$, $\frac{\partial E}{\partial \beta_{j1}} = 0$, $\frac{\partial E}{\partial \beta_{j3}} = 0$, $\frac{\partial E}{\partial b_{j1}} = 0$, $\frac{\partial E}{\partial b_{j3}} = 0$, $\frac{\partial E}{\partial b_{01}} = 0$, and $\frac{\partial E}{\partial b_{03}} = 0$, Ishibuchi et al. [19] derived updating rules for the weights W_{ji}, β_j and biases b_j, b_0 of the fuzzy BP network.

2. $\text{FNR}_{\text{RBF-II}}$ uses a fuzzy RBF network to fit the data set with fuzzy inputs and fuzzy outputs, which are L-R fuzzy numbers. The membership function of L-R fuzzy number $\tilde{A} = (\tilde{a}_1, \tilde{a}_2, \tilde{a}_3)$ is

$$\mu_{\tilde{A}}(x) = \begin{cases} L\left(\dfrac{\tilde{a}_2 - x}{\tilde{a}_1}\right), & x \le \tilde{a}_2, \tilde{a}_1 > 0 \\ R\left(\dfrac{x - \tilde{a}_2}{\tilde{a}_3}\right), & x \ge \tilde{a}_2, \tilde{a}_3 > 0 \end{cases},$$

where $L(\bullet)$ and $R(\bullet)$ are reference functions [49]. From the definition of TFN in Eq. (1), we know that TFN is a L-R fuzzy number and we introduce $\text{FNR}_{\text{RBF-II}}$ based on the TFN data set $\mathbb{D}$, as shown in Table 1. The output of the jth $(1 \le j \le K)$ hidden layer node in Zhang et al.'s fuzzy RBF network [49] is

$$h_j(X_{n1}, X_{n2}, \ldots, X_{nD}) = \exp\left[-\left[\frac{d\left((X_{n1}, X_{n2}, \ldots, X_{nD}), C_j\right)}{\sigma_j}\right]\right], \tag{13}$$

where $C_j = \left(C_{j1}, C_{j2}, \ldots, C_{jD}\right)$ is the center of the jth hidden layer node, $C_{ji} = \left(c_{ji1}, c_{ji2}, \ .c_{ji3}\right)$ $(1 \le i \le D)$ is a TFN, σ_j is the deviation of the jth hidden layer node. Note that $d\left((X_{n1}, X_{n2}, \ldots, X_{nD}), C_j\right)$, σ_j, and $h_j(X)$ are real numbers, where

$$d\left((X_{n1}, X_{n2}, \ldots, X_{nD}), C_j\right) = \sqrt{\sum_{i=1}^{D}\left[d\left(X_{ni}, C_{ji}\right)\right]^2}. \tag{14}$$

The predicted output of the fuzzy RBF network is

$$T_n = \sum_{j=1}^{K} R_j(X_{n1}, X_{n2}, \ldots, X_{nD})\beta_j, \tag{15}$$

where $R_j(X_{n1}, X_{n2}, \ldots, X_{nD}) = \frac{h_j(X_{n1}, X_{n2}, \ldots, X_{nD})}{\sum_{j=1}^{K} h_j(X_{n1}, X_{n2}, \ldots, X_{nD})}$, $\beta_j = (\beta_{j1}, \beta_{j2}, \beta_{j3})$ is the weight between the jth hidden layer node and the output layer node. The estimated error of the fuzzy RBF network is measured as

$$J_2 = \frac{1}{N}\sum_{n=1}^{N}[d(T_n, Y_n)]^2. \tag{16}$$

In [49], Zhang et al. defined the distance between two TFNs A and B as follows:

$$d(A, B) = \sqrt{\int_0^1\left[\left|[A]_\alpha^L - [B]_\alpha^L\right|^2 + \left|[A]_\alpha^U - [B]_\alpha^U\right|^2\right]d\alpha}, \tag{17}$$

where $[[A]_\alpha^L, [A]_\alpha^U]$ and $[[B]_\alpha^L, [B]_\alpha^U]$ are α-cut sets $(0 < \alpha \le 1)$ of A and B, respectively. $C_j = ((c_{j11}, c_{j12}, c_{j13}).$, $(c_{j21}, c_{j22}, c_{j23})$, $\ldots$, $.(c_{jD1}, c_{jD2}, c_{jD3}))$, σ_j, and $\beta_j = (\beta_{j1}, \beta_{j2}, \beta_{j3})$ are adjustable parameters determined by functions of $\frac{\partial J_2}{\partial c_{jil}} = 0$ $(j = 1, 2, \ldots, K;\ i = 1, 2, \ldots, D;\ l = 1, 2, 3)$, $\frac{\partial J_2}{\partial \beta_l} = 0$, and $\frac{\partial J_2}{\partial \sigma_j} = 0$. In $\text{FNR}_{\text{RBF-II}}$ algorithm, C_j is selected randomly from the TFN data set and σ_j is determined using a k-nearest neighbor heuristic.

4. RWN-based FNR model

In this section, we first introduce a simple and fast training mechanism for a single hidden layer feed-forward neural network, i.e., RWN. We then present the RWN-based FNR model called $\mathrm{FNR}_{\mathrm{RWN}}$.

4.1. RWN

Iterative and time-consuming parameter adjustment is required by the BP and RBF networks proposed in [19,49], which generates high computational complexity when constructing FNR models. Thus, a non-iterative approach to train neural networks and the further development of a FNR model based on the trained network would be of practical and theoretical significance. There are two non-iterative methods for training neural networks by randomization, i.e., RWNs [3–5,34,51] and a Random Vector Functional-Link Networks (RVFLNs) [1,16,27,32,33,50,54]. RWNs were originally proposed by Schmidt et al. in [34] and developed further by Cao et al. [3–5,51]. Pao and Takefuji first introduced RVFLNs in [27] and further investigations of RVFLNs were provided in [1,16,32,33,50]. One of the main differences between RWNs and RVFLNs is that RWNs lack direct links from input layer nodes to output layer nodes whereas RVFLNs possess direct links. Given the simplicity of the network architecture employed, we selected RWN to construct our FNR model. For the data set with the crisp input

$$\mathrm{x} = \begin{bmatrix} \mathrm{x}_1 \\ \mathrm{x}_2 \\ \vdots \\ \mathrm{x}_N \end{bmatrix} = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1D} \\ x_{21} & x_{22} & \cdots & x_{2D} \\ \vdots & \vdots & \ddots & \vdots \\ x_{N1} & x_{N2} & \cdots & x_{ND} \end{bmatrix}$$

and crisp output

$$\mathrm{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix},$$

the RWN calculates the output layer weights as follows [3–5,34,51]:

$$\beta = \mathrm{H}^{\dagger}\mathrm{y}, \tag{18}$$

where H† is the Moore–Penrose generalized inverse of the hidden layer output matrix

$$\mathrm{H} = \begin{bmatrix} g(\mathrm{w}_1\mathrm{x}_1 + b_1) & g(\mathrm{w}_2\mathrm{x}_1 + b_2) & \cdots & g(\mathrm{w}_K\mathrm{x}_1 + b_K) \\ g(\mathrm{w}_1\mathrm{x}_2 + b_1) & g(\mathrm{w}_2\mathrm{x}_2 + b_2) & \cdots & g(\mathrm{w}_K\mathrm{x}_2 + b_K) \\ \vdots & \vdots & \ddots & \vdots \\ g(\mathrm{w}_1\mathrm{x}_N + b_1) & g(\mathrm{w}_2\mathrm{x}_N + b_2) & \cdots & g(\mathrm{w}_K\mathrm{x}_N + b_K) \end{bmatrix}, \tag{19}$$

where $g(v) = \frac{1}{1+e^{-v}}$ is sigmoid activation function,

$$\mathrm{w} = \begin{bmatrix} \mathrm{w}_1 \\ \mathrm{w}_2 \\ \vdots \\ \mathrm{w}_K \end{bmatrix}^{\mathrm{T}} = \begin{bmatrix} w_{11} & w_{21} & \cdots & w_{K1} \\ w_{12} & w_{22} & \cdots & w_{K2} \\ \vdots & \vdots & \ddots & \vdots \\ w_{1D} & w_{2D} & \cdots & w_{KD} \end{bmatrix}$$

are the input layer weights, and

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

is the hidden layer bias. w and b in RWNs are considered to be random variables that are selected randomly and independently [5].

4.2. FNR_{RWN}

Fig. 1 shows the architecture of the neural network used to construct the RWN-based FNR model called $\mathrm{FNR}_{\mathrm{RWN}}$, where both the inputs and outputs are TFNs. The input layer weights and hidden layer biases of $\mathrm{FNR}_{\mathrm{RWN}}$ are random numbers selected from the interval [0, 1]. The crisp output layer weights need to be determined. Next, we described how to use

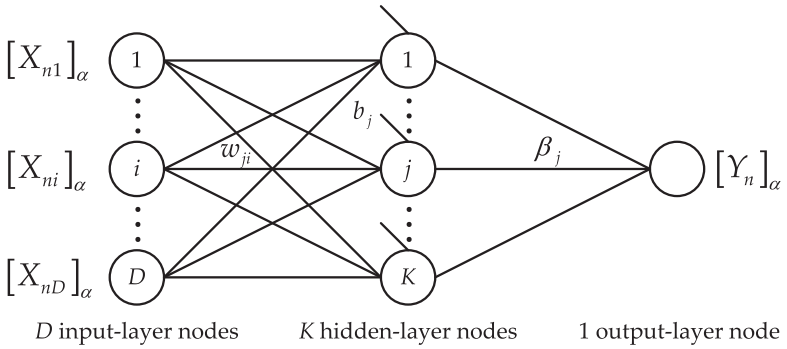


Fig. 1. RWN architecture for constructing FNR_{RWN}. $[A]_\alpha = \left[[A]_\alpha^L, [A]_\alpha^U\right]$ is α-cut set of TFN A, $0 < \alpha \le 1$).

RWNs to handle the FNR with fuzzy inputs and fuzzy outputs. For the nth $(1 < n < N)$ instance $(X_{n1}, X_{n2}, \ldots, X_{nD})$ in Table 1, the corresponding input and output of the jth $(1 < j < K)$ hidden layer node are

$$R_{nj} = (r_{nj1}, r_{nj2}, r_{nj3}) = \left(\sum_{i=1}^{D} w_{ji}x_{ni1}, \sum_{i=1}^{D} w_{ji}x_{ni2}, \sum_{i=1}^{D} w_{ji}x_{ni3}\right) \tag{20}$$

and

$$H_{nj} = \left(h_{nj1}, h_{nj2}, h_{nj3}\right) = \left(\frac{1}{1+\exp\left(-r_{nj1}\right)}, \frac{1}{1+\exp\left(-r_{nj2}\right)}, \frac{1}{1+\exp\left(-r_{nj3}\right)}\right), \tag{21}$$

respectively, where the α-cut set of H_{nj} is

$$[H_{nj}]_\alpha = \left[[H_{nj}]_\alpha^L, [H_{nj}]_\alpha^U\right] = \left[\frac{1}{1+\exp\left[-\left[\alpha r_{nj2} + (1-\alpha)r_{nj1}\right]\right]}, \frac{1}{1+\exp\left[-\left[\alpha r_{nj2} + (1-\alpha)r_{nj3}\right]\right]}\right]. \tag{22}$$

Eq. (22) can be derived as follows. After TFN R_{nj} is activated by the sigmoid function, the membership degree function of H_{nj} is

$$\mu_{H_{nj}}(x) = \begin{cases} \dfrac{\ln\left(\frac{x}{1-x}\right) - r_{nj1}}{r_{nj2} - r_{nj1}}, & \dfrac{1}{\exp\left(-r_{nj1}\right)} \le x \le \dfrac{1}{\exp\left(-r_{nj2}\right)} \\ \dfrac{r_{nj3} - \ln\left(\frac{x}{1-x}\right)}{r_{nj3} - r_{nj2}}, & \dfrac{1}{\exp\left(-r_{nj2}\right)} \le x \le \dfrac{1}{\exp\left(-r_{nj3}\right)} \\ 0, & x < \dfrac{1}{\exp\left(-r_{nj1}\right)} \quad \text{or} \quad x > \dfrac{1}{\exp\left(-r_{nj3}\right)} \end{cases}, \tag{23}$$

where $\ln(\bullet)$ is the natural logarithm function. Let $\frac{\ln\left(\frac{x}{1-x}\right)-r_{nj1}}{r_{nj2}-r_{nj1}} = \alpha$. The lower bound of the α-cut set of H_{nj} can be solved as

$$x = [H_{nj}]_\alpha^L = \frac{1}{1+\exp\left[-\left[\alpha r_{nj2} + (1-\alpha)r_{nj1}\right]\right]}. \tag{24}$$

Similarly, we can obtain the upper bound of the α-cut set of H_{nj} from $\frac{r_{nj3}-\ln\left(\frac{x}{1-x}\right)}{r_{nj3}-r_{nj2}} = \alpha$

$$x = [H_{nj}]_\alpha^U = \frac{1}{1+\exp\left[-\left[\alpha r_{nj2} + (1-\alpha)r_{nj3}\right]\right]}. \tag{25}$$

Then, we can calculate the α-cut set for the predicted output of the RWN corresponding to the fuzzy input $(X_{n1}, X_{n2}, \ldots, X_{nD})$

$$[T_n]_\alpha = \left[[T_n]_\alpha^L, [T_n]_\alpha^U\right] = \left[\sum_{j=1}^{K} \beta_j [H_{nj}]_\alpha^L, \sum_{j=1}^{K} \beta_j [H_{nj}]_\alpha^U\right], \quad \beta_j > 0 \text{ for any } j. \tag{26}$$

It should be noted that β_j is unknown and it needs to be determined. We assume that $\beta_j > 0$ $(j = 1, 2, \ldots, K)$ and we use this assumption as a heuristic to derive the updating rule for the output layer weights β.

For a given α, the estimated error between the predicted output $[T_n]_\alpha$ and target output $[Y_n]_\alpha$ can be measured as

$$e_n(\alpha) = \|[T_n]_\alpha - [Y_n]_\alpha\|^2 = \left[\sum_{j=1}^{K} \beta_j [H_{nj}]_\alpha^L - [Y_n]_\alpha^L\right]^2 + \left[\sum_{j=1}^{K} \beta_j [H_{nj}]_\alpha^U - [Y_n]_\alpha^U\right]^2. \tag{27}$$

From the perspective of the membership degree function, the estimated error on D_n is

$$E_n = \int_0^1 e_n(\alpha)d\alpha = \int_0^1 \|[T_n]_\alpha - [Y_n]_\alpha\|^2 d\alpha. \tag{28}$$

Then, the estimated errors on the data set $\mathbb{D}$ that correspond to a given α and $\alpha \in (0, 1]$ are

$$e(\alpha) = \sum_{n=1}^{N} e_n(\alpha) = \left\|\mathbb{H}_\alpha^L \beta - Y_\alpha^L\right\|^2 + \left\|\mathbb{H}_\alpha^U \beta - Y_\alpha^U\right\|^2 \tag{29}$$

and

$$\begin{aligned} E = \sum_{n=1}^{N} E_n = \sum_{n=1}^{N} \int_0^1 \|[T_n]_\alpha - [Y_n]_\alpha\|^2 d\alpha &= \int_0^1 \left[\sum_{n=1}^{N} \|[T_n]_\alpha - [Y_n]_\alpha\|^2\right] d\alpha \\ &= \int_0^1 \left\|\mathbb{H}_\alpha^L \beta - Y_\alpha^L\right\|^2 d\alpha + \int_0^1 \left\|\mathbb{H}_\alpha^U \beta - Y_\alpha^U\right\|^2 d\alpha \end{aligned}, \tag{30}$$

respectively, where

$$\mathbb{H}_\alpha^L = \begin{bmatrix} [H_{11}]_\alpha^L & [H_{12}]_\alpha^L & \cdots & [H_{1K}]_\alpha^L \\ [H_{21}]_\alpha^L & [H_{22}]_\alpha^L & \cdots & [H_{2K}]_\alpha^L \\ \vdots & \vdots & \ddots & \vdots \\ [H_{N1}]_\alpha^L & [H_{N2}]_\alpha^L & \cdots & [H_{NK}]_\alpha^L \end{bmatrix}$$

is called the lower bound matrix of the hidden layer output,

$$\mathbb{H}_\alpha^U = \begin{bmatrix} [H_{11}]_\alpha^U & [H_{12}]_\alpha^U & \cdots & [H_{1K}]_\alpha^U \\ [H_{21}]_\alpha^U & [H_{22}]_\alpha^U & \cdots & [H_{2K}]_\alpha^U \\ \vdots & \vdots & \ddots & \vdots \\ [H_{N1}]_\alpha^U & [H_{N2}]_\alpha^U & \cdots & [H_{NK}]_\alpha^U \end{bmatrix}$$

is the upper bound matrix of the hidden layer output,

$$\mathbb{Y}_\alpha^L = \begin{bmatrix} [Y_1]_\alpha^L \\ [Y_2]_\alpha^L \\ \vdots \\ [Y_N]_\alpha^L \end{bmatrix}$$

is the lower bound vector of the target output, and

$$\mathbb{Y}_\alpha^U = \begin{bmatrix} [Y_1]_\alpha^U \\ [Y_2]_\alpha^U \\ \vdots \\ [Y_N]_\alpha^U \end{bmatrix}$$

is the upper bound vector of the target output. $\mathrm{FNR}_{\mathrm{RWN}}$ selects the β that minimizes the overall estimated error in Eq. (30). However, it is difficult to derive the updating rule for β directly from Eq. (30) through the first derivative of E with respect to β, i.e., solving β from $\frac{dE}{d\beta} = 0$.

Thus, we use the following method to approximately calculate the hidden layer weight of $\mathrm{FNR}_{\mathrm{RWN}}$. We assume that there are M α-cut points: $0 = \alpha_0 < \alpha_1 < \alpha_2 < \cdots < \alpha_M = 1$ and $\alpha_m - \alpha_{m-1} = \frac{1}{M}$, $m = 1, 2, \ldots, M$. The estimated error on the data set $\mathbb{D}$ with respect to these M α-cut points can be calculated as

$$E' = \frac{1}{M^2} \sum_{m=1}^{M} \sum_{n=1}^{N} e_n(\alpha_m) = \frac{1}{M^2} \sum_{m=1}^{M} \left[\left\|\mathbb{H}_{\alpha_m}^L \beta - \mathbb{Y}_{\alpha_m}^L\right\|^2 + \left\|\mathbb{H}_{\alpha_m}^U \beta - \mathbb{Y}_{\alpha_m}^U\right\|^2\right]. \tag{31}$$

If we let $\frac{dE'}{d\beta} = 0$, the we can obtain

$$\beta = \left[\frac{1}{M}\sum_{m=1}^{M}\left[\mathbb{H}_{\alpha_m}^{L} + \mathbb{H}_{\alpha_m}^{U}\right]\right]^{\dagger}\left[\frac{1}{M}\sum_{m=1}^{M}\left[\mathbb{Y}_{\alpha_m}^{L} + \mathbb{Y}_{\alpha_m}^{U}\right]\right]. \tag{32}$$

According to the definition of a Riemann integral, when $\frac{1}{M} \to 0$, we can derive

$$\lim_{M\to\infty}\frac{1}{M}\sum_{m=1}^{M}\left[\mathbb{H}_{\alpha_m}^{L} + \mathbb{H}_{\alpha_m}^{U}\right] = \int_0^1 \left(\mathbb{H}_{\alpha}^{L} + \mathbb{H}_{\alpha}^{U}\right)d\alpha \tag{33}$$

and

$$\lim_{M\to\infty}\frac{1}{M}\sum_{m=1}^{M}\left[\mathbb{Y}_{\alpha_m}^{L} + \mathbb{Y}_{\alpha_m}^{U}\right] = \int_0^1 \left(\mathbb{Y}_{\alpha}^{L} + \mathbb{Y}_{\alpha}^{U}\right)d\alpha. \tag{34}$$

By substituting Eqs. (33) and (34) for the corresponding terms in Eq. (32), the output layer weight β of $\mathrm{FNR}_{\mathrm{RWN}}$ can be expressed as

$$\beta = \left[\int_0^1 \left(\mathbb{H}_{\alpha}^{L} + \mathbb{H}_{\alpha}^{U}\right)d\alpha\right]^{\dagger}\left[\int_0^1 \left(\mathbb{Y}_{\alpha}^{L} + \mathbb{Y}_{\alpha}^{U}\right)d\alpha\right], \tag{35}$$

where

$$\int_0^1 \mathbb{H}_{\alpha}^{L}d\alpha = \begin{bmatrix} \int_0^1 [H_{11}]_{\alpha}^{L}d\alpha & \int_0^1 [H_{12}]_{\alpha}^{L}d\alpha & \cdots & \int_0^1 [H_{1K}]_{\alpha}^{L}d\alpha \\ \int_0^1 [H_{21}]_{\alpha}^{L}d\alpha & \int_0^1 [H_{22}]_{\alpha}^{L}d\alpha & \cdots & \int_0^1 [H_{2K}]_{\alpha}^{L}d\alpha \\ \vdots & \vdots & \ddots & \vdots \\ \int_0^1 [H_{N1}]_{\alpha}^{L}d\alpha & \int_0^1 [H_{N2}]_{\alpha}^{L}d\alpha & \cdots & \int_0^1 [H_{NK}]_{\alpha}^{L}d\alpha \end{bmatrix} \tag{36}$$

is the definite integral of the lower bound matrix $\mathbb{H}_{\alpha}^{L}$,

$$\int_0^1 \mathbb{H}_{\alpha}^{U}d\alpha = \begin{bmatrix} \int_0^1 [H_{11}]_{\alpha}^{U}d\alpha & \int_0^1 [H_{12}]_{\alpha}^{U}d\alpha & \cdots & \int_0^1 [H_{1K}]_{\alpha}^{U}d\alpha \\ \int_0^1 [H_{21}]_{\alpha}^{U}d\alpha & \int_0^1 [H_{22}]_{\alpha}^{U}d\alpha & \cdots & \int_0^1 [H_{2K}]_{\alpha}^{U}d\alpha \\ \vdots & \vdots & \ddots & \vdots \\ \int_0^1 [H_{N1}]_{\alpha}^{U}d\alpha & \int_0^1 [H_{N2}]_{\alpha}^{U}d\alpha & \cdots & \int_0^1 [H_{NK}]_{\alpha}^{U}d\alpha \end{bmatrix} \tag{37}$$

is the definite integral of the upper bound matrix $\mathbb{H}_{\alpha}^{U}$,

$$\int_0^1 \mathbb{Y}_{\alpha}^{L}d\alpha = \begin{bmatrix} \int_0^1 [Y_1]_{\alpha}^{L}d\alpha \\ \int_0^1 [Y_2]_{\alpha}^{L}d\alpha \\ \vdots \\ \int_0^1 [Y_N]_{\alpha}^{L}d\alpha \end{bmatrix} = \begin{bmatrix} \frac{1}{2}(y_{11} + y_{12}) \\ \frac{1}{2}(y_{21} + y_{22}) \\ \vdots \\ \frac{1}{2}(y_{N1} + y_{N2}) \end{bmatrix} \tag{38}$$

is the definite integral to the lower bound vector $\mathbb{Y}_{\alpha}^{L}$, and

$$\int_0^1 \mathbb{Y}_{\alpha}^{U}d\alpha = \begin{bmatrix} \int_0^1 [Y_1]_{\alpha}^{U}d\alpha \\ \int_0^1 [Y_2]_{\alpha}^{U}d\alpha \\ \vdots \\ \int_0^1 [Y_N]_{\alpha}^{U}d\alpha \end{bmatrix} = \begin{bmatrix} \frac{1}{2}(y_{12} + y_{13}) \\ \frac{1}{2}(y_{22} + y_{23}) \\ \vdots \\ \frac{1}{2}(y_{N2} + y_{N3}) \end{bmatrix} \tag{39}$$

is the definite integral to the upper bound vector $\mathbb{Y}_{\alpha}^{U}$.

The term $\int_0^1 [H_{ni}]_{\alpha}^{L}d\alpha$ in $\int_0^1 \mathbb{H}_{\alpha}^{L}d\alpha$ can be calculated as

$$\begin{aligned} \int_0^1 [H_{ni}]_{\alpha}^{L}d\alpha &= \int_0^1 \frac{1}{1+\exp\left[-\left[\alpha r_{nj2} + (1-\alpha)r_{nj1}\right]\right]}d\alpha \\ &= \left.\frac{\left(r_{nj2} - r_{nj1}\right)\alpha + r_{nj1} + \ln\left[1+\exp\left[-\left[\left(r_{nj2} - r_{nj1}\right)\alpha + r_{nj1}\right]\right]\right]}{r_{nj2} - r_{nj1}}\right|_0^1 = 1 + \frac{\ln\frac{1+\exp\left[-r_{nj2}\right]}{1+\exp\left[-r_{nj1}\right]}}{r_{nj2} - r_{nj1}}. \end{aligned} \tag{40}$$

Similarly, the term $\int_0^1 [H_{ni}]_\alpha^U d\alpha$ in $\int_0^1 \mathbb{H}_\alpha^U d\alpha$ is calculated as

$$\int_0^1 [H_{ni}]_\alpha^U d\alpha = 1 + \frac{\ln \frac{1+\exp[-r_{nj2}]}{1+\exp[-r_{nj3}]}}{r_{nj2} - r_{nj3}}. \tag{41}$$

Thus, the output layer weight β of $\mathrm{FNR_{RWN}}$ can be represented as

$$\beta = \left[\int_0^1 \left(\mathbb{H}_\alpha^L + \mathbb{H}_\alpha^U\right) d\alpha\right]^\dagger \begin{bmatrix} \frac{1}{2}y_{11} + y_{12} + \frac{1}{2}y_{13} \\ \frac{1}{2}y_{21} + y_{22} + \frac{1}{2}y_{23} \\ \vdots \\ \frac{1}{2}y_{N1} + y_{N2} + \frac{1}{2}y_{N3} \end{bmatrix}. \tag{42}$$

Let $\mathbb{H} = \int_0^1 \left(\mathbb{H}_\alpha^L + \mathbb{H}_\alpha^U\right) d\alpha$ and

$$\mathbb{Y} = \begin{bmatrix} \frac{1}{2}y_{11} + y_{12} + \frac{1}{2}y_{13} \\ \frac{1}{2}y_{21} + y_{22} + \frac{1}{2}y_{23} \\ \vdots \\ \frac{1}{2}y_{N1} + y_{N2} + \frac{1}{2}y_{N3} \end{bmatrix},$$

where

$$\int_0^1 \mathbb{H}_\alpha^L d\alpha = \begin{bmatrix} 1 + \frac{\ln \frac{1+\exp(-r_{112})}{1+\exp(-r_{111})}}{r_{112} - r_{111}} & 1 + \frac{\ln \frac{1+\exp(-r_{122})}{1+\exp(-r_{121})}}{r_{122} - r_{121}} & \cdots & 1 + \frac{\ln \frac{1+\exp(-r_{1K2})}{1+\exp(-r_{1K1})}}{r_{1K2} - r_{1K1}} \\ 1 + \frac{\ln \frac{1+\exp(-r_{212})}{1+\exp(-r_{211})}}{r_{212} - r_{211}} & 1 + \frac{\ln \frac{1+\exp(-r_{222})}{1+\exp(-r_{221})}}{r_{222} - r_{221}} & \cdots & 1 + \frac{\ln \frac{1+\exp(-r_{2K2})}{1+\exp(-r_{2K1})}}{r_{2K2} - r_{2K1}} \\ \vdots & \vdots & \ddots & \vdots \\ 1 + \frac{\ln \frac{1+\exp(-r_{N12})}{1+\exp(-r_{N11})}}{r_{N12} - r_{N11}} & 1 + \frac{\ln \frac{1+\exp(-r_{N22})}{1+\exp(-r_{N21})}}{r_{N22} - r_{N21}} & \cdots & 1 + \frac{\ln \frac{1+\exp(-r_{NK2})}{1+\exp(-r_{NK1})}}{r_{NK2} - r_{NK1}} \end{bmatrix} \tag{43}$$

and

$$\int_0^1 \mathbb{H}_\alpha^U d\alpha = \begin{bmatrix} 1 + \frac{\ln \frac{1+\exp(-r_{112})}{1+\exp(-r_{113})}}{r_{112} - r_{113}} & 1 + \frac{\ln \frac{1+\exp(-r_{122})}{1+\exp(-r_{123})}}{r_{122} - r_{123}} & \cdots & 1 + \frac{\ln \frac{1+\exp(-r_{1K2})}{1+\exp(-r_{1K3})}}{r_{1K2} - r_{1K3}} \\ 1 + \frac{\ln \frac{1+\exp(-r_{212})}{1+\exp(-r_{213})}}{r_{212} - r_{213}} & 1 + \frac{\ln \frac{1+\exp(-r_{222})}{1+\exp(-r_{223})}}{r_{222} - r_{223}} & \cdots & 1 + \frac{\ln \frac{1+\exp(-r_{2K2})}{1+\exp(-r_{2K3})}}{r_{2K2} - r_{2K3}} \\ \vdots & \vdots & \ddots & \vdots \\ 1 + \frac{\ln \frac{1+\exp(-r_{N12})}{1+\exp(-r_{N13})}}{r_{N12} - r_{N13}} & 1 + \frac{\ln \frac{1+\exp(-r_{N22})}{1+\exp(-r_{N23})}}{r_{N22} - r_{N23}} & \cdots & 1 + \frac{\ln \frac{1+\exp(-r_{NK2})}{1+\exp(-r_{NK3})}}{r_{NK2} - r_{NK3}} \end{bmatrix}. \tag{44}$$

$\mathbb{H}_{N\times K}^\dagger$ denotes the Moore–Penrose generalized inverse of the matrix $\mathbb{H} = \int_0^1 \left(\mathbb{H}_\alpha^L + \mathbb{H}_\alpha^U\right) d\alpha$. $\mathrm{FNR_{RWN}}$ uses the orthogonal projection method to calculate $\mathbb{H}_{N\times K}^\dagger$ as

$$\mathbb{H}^\dagger = \begin{cases} (\mathbb{H}^{\mathrm{T}}\mathbb{H} + C\mathrm{I}_{K\times K})^{-1}\mathbb{H}^{\mathrm{T}}, & \text{if } N \geq K \\ \mathbb{H}^{\mathrm{T}}(\mathbb{H}\mathbb{H}^{\mathrm{T}} + C\mathrm{I}_{N\times N})^{-1}, & \text{if } N < K \end{cases}, \tag{45}$$

where $C > 0$ is a regularization factor and I is the identity matrix. According to ridge regression theory [14], the role of C is to reduce the likelihood that $\mathbb{H}^{\mathrm{T}}\mathbb{H}$ or $\mathbb{H}\mathbb{H}^{\mathrm{T}}$ becomes a singular matrix.

Thus, we have derived the updating formula for the output layer weight of RWN in our fuzzy regression model $\mathrm{FNR_{RWN}}$. For an unseen fuzzy vector

$$(X_1, X_2, \ldots, X_D) = ((x_{11}, x_{12}, x_{13}), (x_{21}, x_{22}, x_{23}), \ldots, (x_{D1}, x_{D2}, x_{D3})),$$

the α-cut set of the predicted fuzzy output with $\mathrm{FNR_{RWN}}$ can be calculated as

$$[T]_\alpha = \left[\left[[H_1]_\alpha^L, [H_1]_\alpha^U\right], \left[[H_2]_\alpha^L, [H_2]_\alpha^U\right], \ldots, \left[[H_K]_\alpha^L, [H_K]_\alpha^U\right]\right]\beta, \tag{46}$$

where

$$\left[[H_j]_\alpha^L, [H_j]_\alpha^U\right] = \left[\frac{1}{1 + \exp\left[-\left[\alpha r_{j2} + (1-\alpha) r_{j1}\right]\right]}, \frac{1}{1 + \exp\left[-\left[\alpha r_{j2} + (1-\alpha) r_{j3}\right]\right]}\right] \tag{47}$$

and

$$\left(r_{j1}, r_{j2}, r_{j3}\right) = \left(\sum_{i=1}^D w_{ji} x_{i1}, \sum_{i=1}^D w_{ji} x_{i2}, \sum_{i=1}^D w_{ji} x_{i3}\right) \tag{48}$$

Table 2
Details of the 12 data sets.

Data set	Number of inputs	Number of instances	Source
Concrete compressive strength	8	1030	KEEL
Daily Electricity Energy (DEE)	6	365	KEEL
Electrical-maintenance (Ele-2)	4	1056	KEEL
Energy efficiency	8	768	UCI
Friedman	5	1200	KEEL
Housing	14	506	UCI
Laser generated	4	993	KEEL
Mortgage	15	1049	KEEL
Stock prices	9	950	KEEL
Treasury	15	1049	KEEL
Weather Ankara	9	1609	KEEL
Weather Izmir	9	1461	KEEL

for any $j \in \{1, 2, \ldots, K\}$. It should be noted that β in Eq. (46) is derived based on the assumption that $\beta_j > 0$, $j = 1, 2, \ldots, K$. In fact, the true calculation of β should always include the negative component, i.e., $\exists\, j \in \{1, 2, \ldots, K\}$, $\beta_j < 0$, which will lead to $[T]_\alpha^L > [T]_\alpha^U$. In this case, we refer to the study by Ishibuchi et al. [19] and use the following heuristics to adjust the predicted fuzzy output:

$$[T]_\alpha^L = \min\left\{\sum_{j=1}^{K}[H_j]_\alpha^L\beta_j, \sum_{j=1}^{K}[H_j]_\alpha^U\beta_j\right\} \quad \text{and} \quad [T]_\alpha^U = \max\left\{\sum_{j=1}^{K}[H_j]_\alpha^L\beta_j, \sum_{j=1}^{K}[H_j]_\alpha^U\beta_j\right\}.$$

5. Experimental comparisons

In this section, we compare the regression performance of FNR_{RWN} with two existing FNR models, i.e., BP network-based FNR ($\text{FNR}_{\text{BP-III}}$ [19]) and RBF network-based FNR ($\text{FNR}_{\text{RBF-II}}$ [49]). $\text{FNR}_{\text{BP-III}}$, $\text{FNR}_{\text{RBF-II}}$, and FNR_{RWN} were implemented on a Thinkpad SL410K PC with Windows 8 running on a Pentium Dual-core T4400 2.20 GHz processor with 4 GB RAM. In our experiments, the estimated errors corresponding to a given α and whole interval (0, 1] were measured with Eqs. (29) and (31), respectively. In order to measure the overall error on interval (0, 1], we set α from 0.0001 to 1 with a step of 0.0001.

5.1. Fuzzification of crisp numbers

In previous studies, the dimensions of the fuzzy input data used in experimental comparisons were usually low, e.g., a one-dimensional fuzzy input for $\text{FNR}_{\text{BP-III}}$ [19] and a two-dimensional fuzzy input for $\text{FNR}_{\text{RBF-II}}$ [49]. These types of data sets are insufficient to validate the approximation capacity of neural network-based fuzzy regression models to establish the nonlinear relationship between fuzzy inputs and fuzzy outputs. Therefore, in this study, we selected 2 UCI[1] and 10 KEEL[2] regression data sets containing at least four inputs to compare the performance of $\text{FNR}_{\text{BP-III}}$, $\text{FNR}_{\text{RBF-II}}$, and FNR_{RWN}. The details of these 12 data sets are summarized in Table 2.

However, the inputs and one crisp output of these data sets are all crisp; thus, we give a new fuzzification method to transform the crisp number into TFN in this section. Assume that there is a real number data set

$$\text{Data}_{\text{real}} = \begin{bmatrix} m_1 \\ m_2 \\ \vdots \\ m_N \end{bmatrix},$$

$m_i \in \Re$, $i = 1, 2, \ldots, N$. The objective of our fuzzification method is to obtain a TFN data set

$$\text{Data}_{\text{TFN}} = \begin{bmatrix} (l_1, m_1, r_1) \\ (l_2, m_2, r_2) \\ \vdots \\ (l_N, m_N, r_N) \end{bmatrix},$$

where (l_i, m_i, r_i) is a TFN, and l_i and r_i are the left and right endpoints of the TFN, respectively. For simplicity, we assume that $m_1 \le m_2 \le \cdots \le m_N$ and $\text{Data}_{\text{real}}$ obey a normal distribution of

$$p(m) = \frac{1}{\sqrt{2\pi}\sigma}\exp\left[-\frac{1}{2}\left(\frac{m-\mu}{\sigma}\right)^2\right], \tag{49}$$

[1] http://archive.ics.uci.edu/ml/datasets.html
[2] http://sci2s.ugr.es/keel/datasets.php

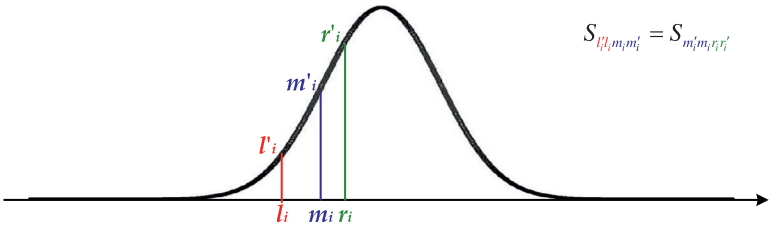


Fig. 2. Explanation of Eq. (50): l_i and r_i are two endpoints that make the squares of l'_i-l_i-m_i-m'_i and m'_i-m_i-r_i-r'_i equal.

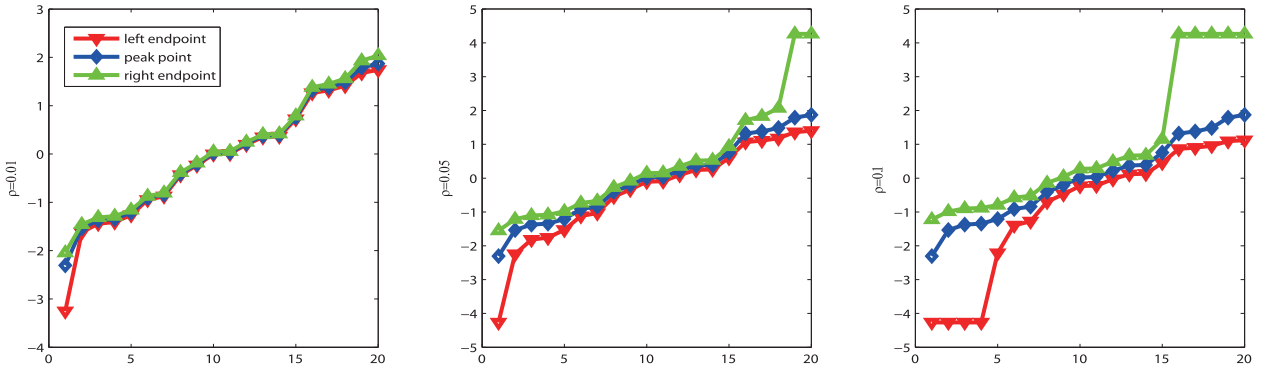


Fig. 3. Impact of ρ on fuzzification, where $\text{Data}_{\text{real}} = \{-2.306, -1.535, -1.374, -1.345, -1.209, -0.909, -0.839, -0.406, -0.209, 0.020, 0.033, 0.221, 0.376, 0.391, 0.756, 1.320, 1.381, 1.482, 1.789, 1.870\}$.

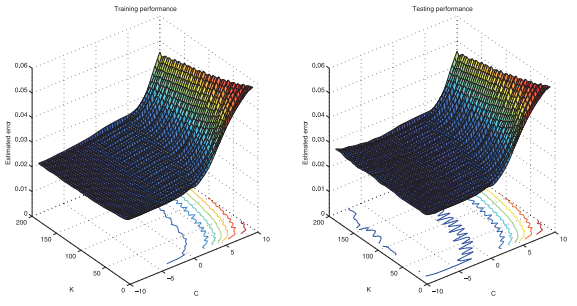
where $\mu = \frac{\sum_{i=1}^{N} m_i}{N}$ and $\sigma^2 = \frac{\sum_{i=1}^{N} (m_i - \mu)^2}{N}$ are the mean value and variance of $\text{Data}_{\text{real}}$, respectively. l_i and r_i can be determined by the following expression:

$$\text{P}\{l_i < m \le m_i\} = \text{P}\{m_i < m \le r_i\} = \rho, \tag{50}$$

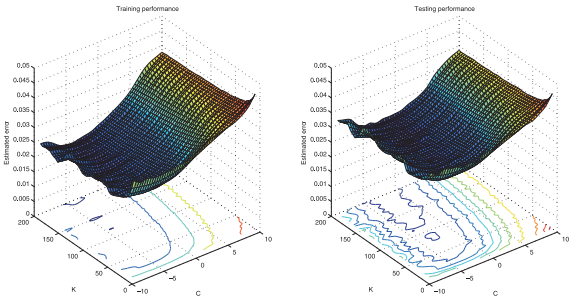
where ρ is a pre-defined parameter. A direct explanation of Eq. (50) is given in Fig. 2, which shows that l_i and r_i are two endpoints that make the squares of l'_i-l_i-m_i-m'_i and m'_i-m_i-r_i-r'_i equal. ρ usually affects the fuzzification on m_i. We provide a specific example to demonstrate this impact. We randomly generated 10 real numbers that followed a standard normal distribution and we then used three different ρ values (i.e., 0.01, 0.05 and 0.1) to conduct fuzzification. The TFNs obtained are presented in Fig. 3, which indicates that as ρ increases, the distance between l_i and r_i becomes larger. In addition, a higher value of ρ increases the possibility that different values of m_i have the same l_i or r_i, which is shown clearly in the third plot in Fig. 3. Thus, we decided to select a smaller ρ for use in applications. In this study, we let $\rho = 0.05$. For a given data set, the crisp inputs and outputs were first normalized on the interval $[-1, 1]$ and then fuzzified using the approach described above.

5.2. Impacts of the regularization factor C and the number of hidden layer nodes K on the performance of FNR_{RWN}

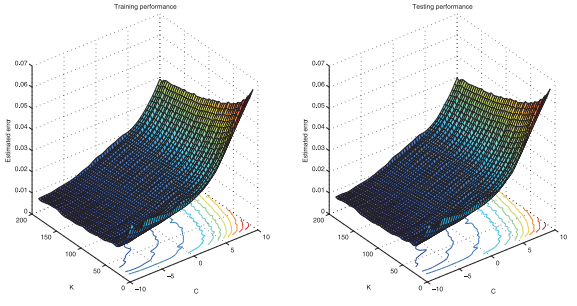
Our proposed FNR_{RWN} method has two parameters that need to be determined by users, i.e., C in Eq. (45) and the number of hidden layer nodes K in the RWN. Using the *DEE*, *Housing*, *Mortgage*, and *Treasury* data sets, we tested the overall training and testing errors for FNR_{RWN} with respect to 400 different pairs of (C, K), where $C = \{2^{-9}, 2^{-8}, \ldots, 2^{9}, .2^{10}\}$ and $K = \{10, .\ 20, \ldots, 190, .200\}$. The input layer weights and hidden layer biases of FNR_{RWN} were initialized with random numbers from a *uniform* distribution over the interval [0, 1]. For any given parameter pair of (C, K), the estimated error was obtained based on a 10-times 10-fold cross-validation procedure [22,25,26,48,52,53]. Fig. 4 shows the influence of different user-specified parameters on the regression performance with FNR_{RWN}. In order to more clearly illustrate the variations in the estimated errors with changes in C and K, Fig. 5 shows the two-dimensional learning curves for FNR_{RWN} on the *Energy*, *Stock*, *Weather Ankara*, and *Weather Izmir* data sets. This figure shows that: (1) as K increases, the training and testing errors of FNR_{RWN} gradually decrease; and (2) a smaller value for C usually leads to a lower estimated error and a faster convergence speed. The first observation demonstrates that our proposed FNR_{RWN} method is convergent, i.e., when $K \to +\infty$, $E \to 0$ for any given C. This indicates that FNR_{RWN} has a good approximation capability. The second observation provides a primary parameter selection criterion for the regularization factor C in applications, i.e., a small C is more favorable. In addition, our proposed FNR_{RWN} method effectively controls the over-fitting, i.e., the training and testing errors all decrease as the number of hidden layer nodes increases.



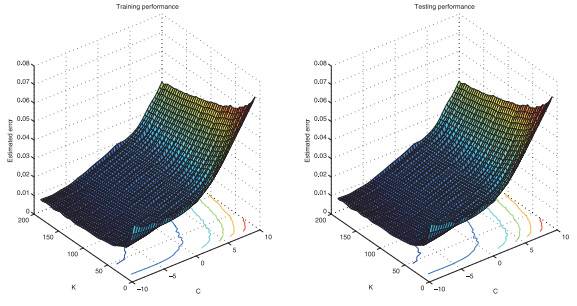
(a) On *DEE* data set



(b) On *Housing* data set

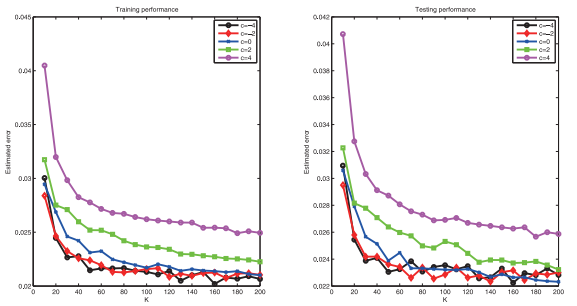


(c) On *Mortgage* data set

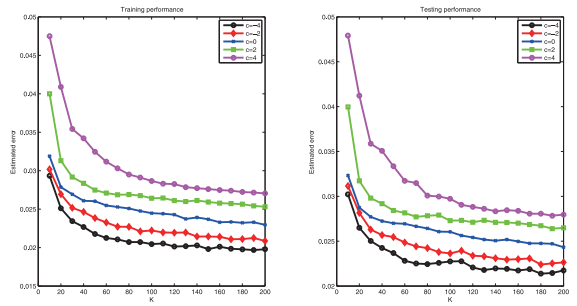


(d) On *Treasury* data set

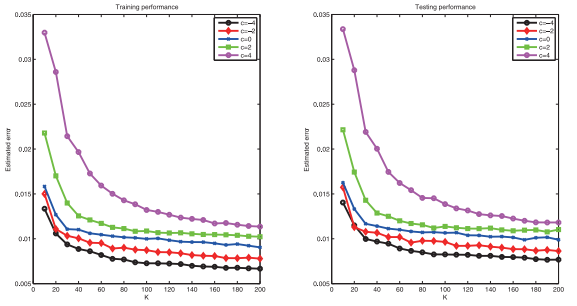
Fig. 4. Impact of parameter pair (C, K) on the training and testing errors of FNR_{RWN} using four representative data sets. The initialization interval for random weights and biases was [0, 1].



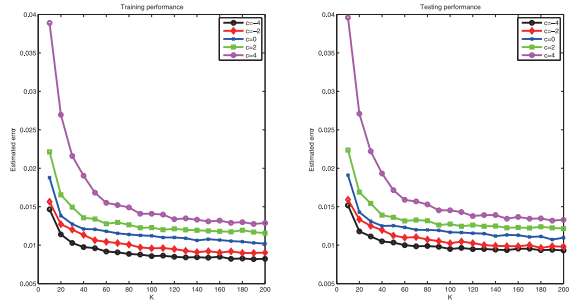
(a) On *Energy* data set



(b) On *Stock* data set

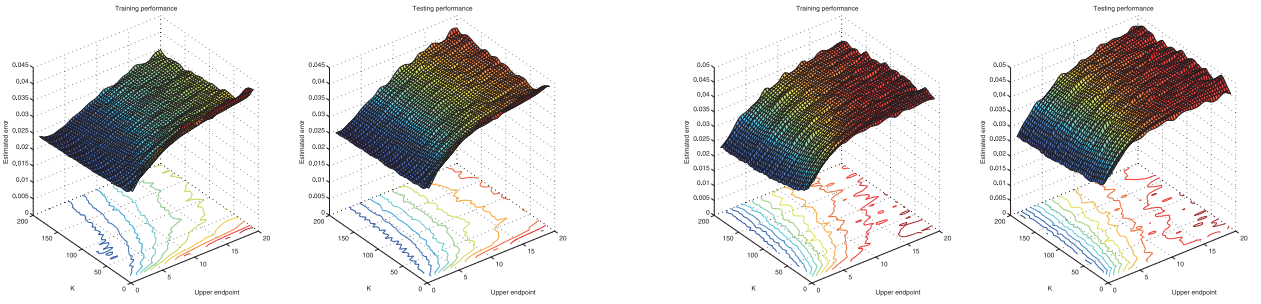


(c) On *Weather Ankara* data set



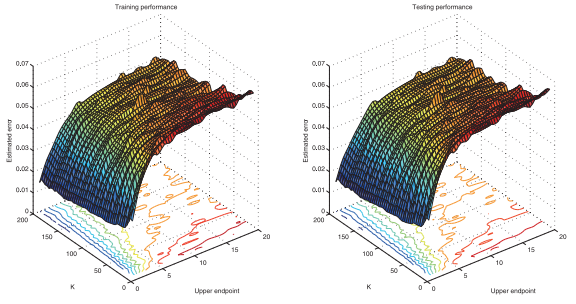
(d) On *Weather Izmir* data set

Fig. 5. Convergence of FNR_{RWN} with changes in the parameters C and K using four representative data sets. The initialization interval for the random weights and biases was [0, 1].

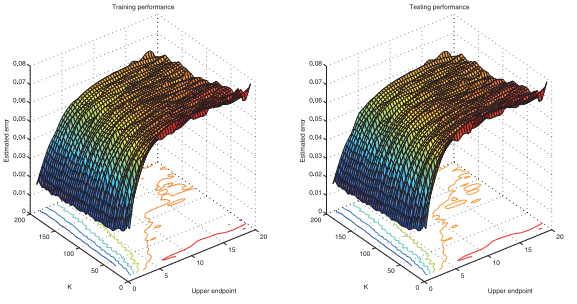


(a) On *DEE* data set ($C = 2^0$)

(b) On *Housing* data set ($C = 2^0$)



(c) On *Mortgage* data set ($C = 2^0$)



(d) On *Treasury* data set ($C = 2^0$)

Fig. 6. Impact of different initialization intervals on the training and testing errors of FNR_{RWN} using four representative data sets.

5.3. Impacts of different initialization intervals on the performance of FNR_{RWN}

In this experiment, we tested the impact of different initialization intervals on the overall training and testing errors of FNR_{RWN}. The initialization interval in this experiment was [0, Upper endpoint], where Upper endpoint = {1, 2, ..., 19, 20}. Thus, the input layer weights and hidden layer biases of FNR_{RWN} were random numbers that followed a *uniform* distribution over the interval [0, Upper endpoint]. The regularization factors on the *DEE*, *Housing*, *Mortgage*, and *Treasury* data sets were all $C = 2^0$. Fig. 6 shows the training and testing errors corresponding to 400 different pairs of (Upper endpoint, K), where K = {10, . 20, ..., 190, .200}. In order to more clearly illustrate the variations in the estimated errors with changes in the upper endpoint of the initialization interval, Fig. 7 shows the two-dimensional learning curves for FNR_{RWN} on the *Energy*, *Stock*, *Weather Ankara*, and *Weather Izmir* data sets. This figure shows that the training and testing errors of FNR_{RWN} gradually increase as the width of the initialization interval increases. A smaller initialization interval, i.e., [0, 1], leads to lower training and testing errors. In addition, the initialization intervals with a smaller width make the predictions more stable with FNR_{RWN}, e.g., the black lines in Fig. 7 are smoother.

5.4. Comparisons of $FNR_{BP\text{-}III}$, $FNR_{RBF\text{-}II}$, and FNR_{RWN}

We compared the training errors, testing errors, training times, and testing times with $\text{FNR}_{\text{BP-III}}$, $\text{FNR}_{\text{RBF-II}}$, and FNR_{RWN} using different values for α, i.e., $\alpha = 0.1$, 0.3, 0.5, 0.7, and 0.9, respectively. A 10-times 10-fold cross-validation procedure was also used in this experimental comparison. The experimental parameters were set as follows.

1. For the BP network in $\text{FNR}_{\text{BP-III}}$, the input layer weights, hidden layer biases, and output layer weights were initialized with random numbers that followed a *uniform* distribution over the interval [0, 1], where the number of hidden layer nodes was 100, the maximum iteration number was 1000, the learning constant was 0.01, the momentum constant was 0.9, and the training error threshold was 0.001.
2. For the RBF network in $\text{FNR}_{\text{RBF-II}}$, the center of the RBF was selected randomly for training the TFN inputs, where the output layer weights were initialized with random numbers that followed a *uniform* distribution over the interval [0, 1], the number of hidden layer nodes was 100, the maximum epoch number was 1000, and the learning constant was 0.01.
3. For the RWN in FNR_{RWN}, we set the regularization factor as $C = 2^{-9}$ and the initialization interval was [0, 1] based on the experimental results described in Sections 5.2 and 5.3. The number of hidden layer nodes in the RWN was $K = 10$.

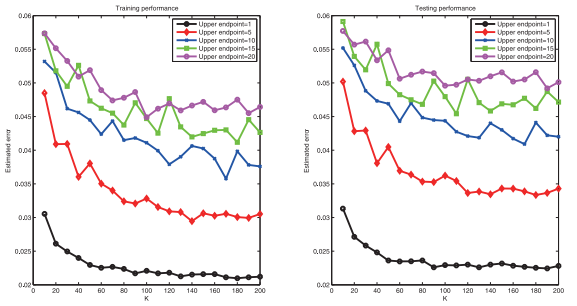
Table 3 and 4 compare the results obtained in terms of the errors and times, respectively. These experimental results shows that FNR_{RWN} clearly obtained smaller training and testing errors on the 10 data sets for given values of $C = 2^{-9}$ and

Table 3
Training and testing errors (mean $\times 10^{-2} \pm$ standard deviation $\times 10^{-2}$) for $FNR_{BP\text{-}III}$, $FNR_{RBF\text{-}II}$, and FNR_{RWN}.

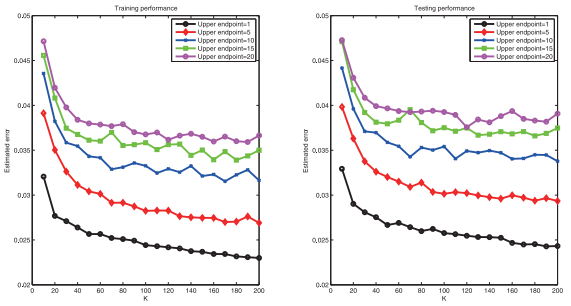
		$FNR_{BP\text{-}III}$					$FNR_{RBF\text{-}II}$					FNR_{RWN}				
		$\alpha = 0.1$	$\alpha = 0.3$	$\alpha = 0.5$	$\alpha = 0.7$	$\alpha = 0.9$	$\alpha = 0.1$	$\alpha = 0.3$	$\alpha = 0.5$	$\alpha = 0.7$	$\alpha = 0.9$	$\alpha = 0.1$	$\alpha = 0.3$	$\alpha = 0.5$	$\alpha = 0.7$	$\alpha = 0.9$
Training error	1	16.74 ± 0.39	11.03 ± 0.10	10.62 ± 0.17	8.91 ± 0.05	14.26 ± 0.02	9.10 ± 0.00	8.26 ± 0.00	7.55 ± 0.00	6.96 ± 0.00	6.50 ± 0.00	7.39 ± 0.53	5.28 ± 0.26	4.13 ± 0.08	3.30 ± 0.06	2.83 ± 0.06
	2	12.66 ± 0.15	8.47 ± 0.04	6.86 ± 0.13	9.70 ± 0.06	11.80 ± 0.05	4.62 ± 0.00	4.11 ± 0.00	3.70 ± 0.00	3.39 ± 0.00	3.18 ± 0.00	3.63 ± 0.07	2.84 ± 0.03	2.30 ± 0.01	1.90 ± 0.01	1.65 ± 0.02
	3	11.10 ± 0.04	7.03 ± 0.15	4.80 ± 0.13	5.65 ± 0.13	9.42 ± 0.34	2.29 ± 0.00	2.18 ± 0.00	2.11 ± 0.00	2.09 ± 0.00	2.12 ± 0.00	0.74 ± 0.02	0.34 ± 0.01	0.17 ± 0.00	0.09 ± 0.00	0.08 ± 0.00
	4	17.85 ± 1.10	10.35 ± 1.18	7.84 ± 0.63	11.44 ± 0.23	14.52 ± 0.81	3.60 ± 0.00	2.96 ± 0.00	2.44 ± 0.00	2.04 ± 0.00	1.76 ± 0.00	12.97 ± 3.12	8.41 ± 1.40	4.13 ± 0.61	2.34 ± 0.25	1.41 ± 0.05
	5	19.19 ± 0.03	13.30 ± 0.01	12.49 ± 0.01	15.23 ± 0.05	25.27 ± 0.01	5.13 ± 0.00	4.60 ± 0.00	4.17 ± 0.00	3.83 ± 0.00	3.60 ± 0.00	4.17 ± 0.07	3.18 ± 0.03	2.49 ± 0.04	2.03 ± 0.04	1.77 ± 0.04
	6	12.86 ± 0.17	9.10 ± 0.26	7.91 ± 0.52	6.30 ± 0.13	11.07 ± 0.29	5.21 ± 0.00	4.83 ± 0.00	4.52 ± 0.00	4.31 ± 0.00	4.18 ± 0.00	8.20 ± 0.50	6.13 ± 0.26	4.81 ± 0.20	3.86 ± 0.18	3.39 ± 0.25
	7	8.88 ± 0.74	6.09 ± 0.20	5.69 ± 0.06	4.07 ± 0.06	8.36 ± 0.02	5.40 ± 0.00	5.04 ± 0.00	4.75 ± 0.00	4.53 ± 0.00	4.39 ± 0.00	4.49 ± 0.54	2.61 ± 0.31	1.63 ± 0.10	1.02 ± 0.04	0.75 ± 0.05
	8	3.52 ± 0.20	2.48 ± 0.17	2.03 ± 0.05	1.53 ± 0.13	3.19 ± 0.65	0.78 ± 0.00	0.74 ± 0.00	0.71 ± 0.00	0.70 ± 0.00	0.71 ± 0.00	6.15 ± 0.62	3.73 ± 0.22	2.09 ± 0.22	1.08 ± 0.12	0.74 ± 0.05
	9	6.90 ± 0.75	5.89 ± 0.72	5.13 ± 0.85	5.08 ± 0.67	7.32 ± 0.22	4.45 ± 0.00	4.11 ± 0.00	3.85 ± 0.00	3.68 ± 0.00	3.60 ± 0.00	8.14 ± 0.53	5.37 ± 0.26	3.65 ± 0.16	2.81 ± 0.13	2.35 ± 0.14
	10	4.77 ± 0.19	2.91 ± 0.01	2.53 ± 0.08	2.81 ± 0.16	3.28 ± 0.05	0.54 ± 0.00	0.52 ± 0.00	0.52 ± 0.00	0.53 ± 0.00	0.56 ± 0.00	4.67 ± 0.37	3.01 ± 0.42	1.83 ± 0.22	1.05 ± 0.13	0.71 ± 0.11
	11	10.78 ± 1.12	6.98 ± 1.17	6.30 ± 0.43	6.82 ± 0.41	6.36 ± 0.21	3.33 ± 0.00	3.12 ± 0.00	2.96 ± 0.00	2.83 ± 0.00	2.74 ± 0.00	2.55 ± 0.25	1.53 ± 0.19	1.08 ± 0.13	0.80 ± 0.09	0.63 ± 0.07
	12	9.08 ± 0.84	6.83 ± 0.42	5.45 ± 0.97	5.16 ± 0.10	5.82 ± 0.62	4.12 ± 0.00	3.83 ± 0.00	3.59 ± 0.00	3.41 ± 0.00	3.28 ± 0.00	1.88 ± 0.19	1.23 ± 0.23	0.80 ± 0.06	0.55 ± 0.07	0.47 ± 0.06
Testing error	1	21.57 ± 0.20	12.32 ± 0.10	11.69 ± 0.10	9.97 ± 0.00	16.56 ± 0.09	9.30 ± 0.02	8.43 ± 0.02	7.69 ± 0.01	7.08 ± 0.01	6.59 ± 0.01	7.69 ± 0.52	5.51 ± 0.27	4.36 ± 0.12	3.43 ± 0.11	2.94 ± 0.09
	2	18.88 ± 0.25	12.28 ± 0.18	9.34 ± 0.16	14.70 ± 0.11	13.60 ± 0.15	4.87 ± 0.02	4.30 ± 0.03	3.86 ± 0.01	3.51 ± 0.02	3.28 ± 0.02	4.11 ± 0.21	3.23 ± 0.12	2.58 ± 0.08	2.15 ± 0.03	1.85 ± 0.05
	3	12.20 ± 0.16	9.25 ± 0.12	6.01 ± 0.07	6.70 ± 0.06	9.36 ± 0.23	2.34 ± 0.00	2.22 ± 0.01	2.14 ± 0.00	2.12 ± 0.00	2.15 ± 0.01	0.80 ± 0.03	0.37 ± 0.02	0.18 ± 0.01	0.10 ± 0.00	0.08 ± 0.01
	4	19.64 ± 0.11	12.68 ± 0.30	8.51 ± 0.12	13.32 ± 0.14	15.90 ± 0.32	3.82 ± 0.01	3.13 ± 0.02	2.57 ± 0.01	2.14 ± 0.01	1.84 ± 0.01	13.29 ± 2.46	8.81 ± 1.74	4.22 ± 0.64	2.39 ± 0.24	1.46 ± 0.05
	5	24.14 ± 0.06	14.58 ± 0.05	14.62 ± 0.01	16.56 ± 0.03	27.19 ± 0.01	5.33 ± 0.01	4.76 ± 0.01	4.28 ± 0.01	3.91 ± 0.01	3.65 ± 0.01	4.28 ± 0.15	3.23 ± 0.09	2.53 ± 0.05	2.07 ± 0.05	1.78 ± 0.06
	6	14.99 ± 0.21	10.12 ± 0.13	10.96 ± 0.27	9.13 ± 0.86	20.49 ± 0.36	5.72 ± 0.06	5.25 ± 0.06	4.94 ± 0.03	4.67 ± 0.03	4.55 ± 0.08	8.57 ± 0.68	6.31 ± 0.32	5.07 ± 0.35	4.04 ± 0.20	3.54 ± 0.31
	7	10.16 ± 0.21	6.16 ± 0.19	6.17 ± 0.19	6.39 ± 0.23	8.85 ± 0.19	5.44 ± 0.01	5.07 ± 0.01	4.78 ± 0.01	4.55 ± 0.01	4.41 ± 0.01	4.55 ± 0.45	2.65 ± 0.34	1.71 ± 0.12	1.09 ± 0.07	0.79 ± 0.04
	8	3.62 ± 1.37	2.69 ± 0.40	2.04 ± 0.06	1.60 ± 0.24	3.33 ± 0.47	0.81 ± 0.00	0.76 ± 0.00	0.73 ± 0.00	0.71 ± 0.00	0.72 ± 0.00	6.37 ± 0.78	3.84 ± 0.19	2.16 ± 0.26	1.09 ± 0.15	0.76 ± 0.07
	9	8.04 ± 0.60	5.99 ± 0.00	5.92 ± 0.16	4.87 ± 0.10	7.92 ± 0.15	4.53 ± 0.01	4.18 ± 0.01	3.91 ± 0.01	3.74 ± 0.01	3.65 ± 0.01	8.38 ± 0.50	5.32 ± 0.38	3.74 ± 0.15	2.86 ± 0.14	2.42 ± 0.18
	10	4.75 ± 0.17	3.06 ± 0.45	2.57 ± 0.10	2.81 ± 0.16	3.54 ± 0.11	0.55 ± 0.00	0.53 ± 0.00	0.53 ± 0.00	0.54 ± 0.00	0.57 ± 0.00	4.58 ± 0.46	3.11 ± 0.43	1.81 ± 0.24	1.07 ± 0.14	0.71 ± 0.10
	11	15.19 ± 0.38	10.80 ± 2.55	10.40 ± 0.42	13.42 ± 8.46	10.26 ± 1.92	3.59 ± 0.02	3.33 ± 0.02	3.12 ± 0.02	2.95 ± 0.01	2.83 ± 0.01	2.68 ± 0.26	1.73 ± 0.31	1.23 ± 0.17	0.88 ± 0.15	0.67 ± 0.07
	12	10.87 ± 3.36	8.18 ± 1.19	6.06 ± 2.01	5.13 ± 0.13	7.03 ± 0.76	4.16 ± 0.00	3.87 ± 0.00	3.62 ± 0.00	3.43 ± 0.00	3.30 ± 0.00	1.95 ± 0.26	1.25 ± 0.27	0.82 ± 0.06	0.57 ± 0.07	0.48 ± 0.06

Table 4
Training and testing times (seconds) for $FNR_{BP\text{-}III}$, $FNR_{RBF\text{-}II}$, and FNR_{RWN}.

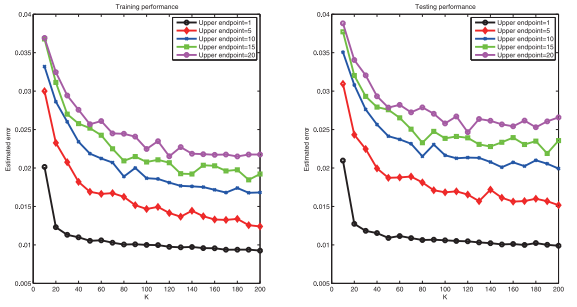
		$FNR_{BP\text{-}III}$					$FNR_{RBF\text{-}II}$					FNR_{RWN}				
		$\alpha = 0.1$	$\alpha = 0.3$	$\alpha = 0.5$	$\alpha = 0.7$	$\alpha = 0.9$	$\alpha = 0.1$	$\alpha = 0.3$	$\alpha = 0.5$	$\alpha = 0.7$	$\alpha = 0.9$	$\alpha = 0.1$	$\alpha = 0.3$	$\alpha = 0.5$	$\alpha = 0.7$	$\alpha = 0.9$
Training time	1	72.44531	69.48438	67.57813	66.12500	66.89063	1.51969	1.53391	1.49234	1.53563	1.54625	0.12984	0.12250	0.12609	0.14594	0.13734
	2	50.07813	48.99219	48.14063	46.70313	47.09375	0.28844	0.28922	0.27797	0.27531	0.27297	0.01594	0.01609	0.01578	0.01609	0.01578
	3	66.07031	66.27344	65.42188	66.07031	65.82813	1.21719	1.21234	1.21203	1.21313	1.21109	0.14047	0.14438	0.14516	0.14516	0.14266
	4	60.70313	59.89844	57.05469	56.82031	57.17969	0.90844	0.98531	1.04906	1.03594	1.05672	0.08437	0.07531	0.07125	0.07750	0.08594
	5	67.52344	67.53906	67.43750	67.96875	68.98438	2.16406	2.33344	2.56125	2.39687	2.27250	0.09094	0.09094	0.09016	0.10594	0.10422
	6	49.77344	49.04688	49.38281	49.20313	50.00000	0.67125	0.58688	0.61641	0.68344	0.68031	0.01781	0.01547	0.01531	0.01906	0.01688
	7	66.07031	65.18750	63.96875	64.64063	63.67188	1.28766	1.28875	1.28609	1.30625	1.29391	0.05281	0.05234	0.05328	0.05172	0.05078
	8	69.52344	66.29688	68.01563	65.19531	66.25000	2.17266	2.13844	2.19078	2.18438	2.19375	0.05922	0.05922	0.05891	0.06125	0.06062
	9	64.14063	62.28906	62.34375	62.23438	62.12500	1.59484	1.71203	1.77031	1.56969	1.66031	0.04453	0.04016	0.04281	0.04328	0.04266
	10	69.24219	67.17188	63.95313	65.75000	67.05469	2.53672	2.39125	2.13406	2.14063	2.13797	0.06719	0.06828	0.07031	0.06453	0.06922
	11	48.02344	47.10938	47.21875	44.95313	45.88281	0.26359	0.26266	0.26063	0.26437	0.31047	0.00719	0.00797	0.00703	0.00656	0.00672
	12	90.67969	85.08594	84.23438	82.06250	82.85938	3.42797	3.34719	3.36016	3.32969	3.38891	0.14766	0.14328	0.15594	0.18078	0.14219
Testing time	1	0.07813	0.06250	0.03125	0.06250	0.04688	0.16813	0.16531	0.15891	0.16563	0.16422	0.00125	0.00219	0.00109	0.00156	0.00172
	2	0.03906	0.03125	0.03906	0.03125	0.05469	0.03203	0.04625	0.04375	0.04234	0.04250	0.00078	0.00047	0.00063	0.00063	0.00063
	3	0.04688	0.05469	0.03906	0.04688	0.07031	0.12984	0.12984	0.12906	0.12750	0.13094	0.00203	0.00172	0.00141	0.00266	0.00203
	4	0.0625	0.03125	0.03125	0.04688	0.03125	0.09703	0.10906	0.11234	0.10750	0.11719	0.00172	0.00109	0.00109	0.00125	0.00125
	5	0.03125	0.04688	0.03125	0.0625	0.03125	0.23344	0.24859	0.26828	0.25250	0.24188	0.00156	0.00125	0.00172	0.00234	0.00281
	6	0.02344	0.03125	0.04688	0.03125	0.04688	0.06562	0.05828	0.05891	0.06688	0.06406	0.00063	0.00047	0.00063	0.00031	0.00031
	7	0.05469	0.06250	0.03125	0.06250	0.04688	0.13594	0.13578	0.13719	0.14016	0.13656	0.00109	0.00141	0.00109	0.00078	0.00094
	8	0.05469	0.03906	0.05469	0.03125	0.03125	0.22906	0.23109	0.23328	0.23563	0.23266	0.00125	0.00156	0.00094	0.00172	0.00094
	9	0.04688	0.03906	0.03906	0.03125	0.03125	0.16672	0.18156	0.18500	0.16703	0.17047	0.00094	0.00063	0.00094	0.00078	0.00125
	10	0.05469	0.03906	0.02344	0.05469	0.03906	0.26656	0.25156	0.22859	0.22922	0.22969	0.00125	0.00078	0.00078	0.00063	0.00109
	11	0.03906	0.03125	0.03125	0.03125	0.03125	0.02688	0.02688	0.02719	0.0275	0.03313	0.00031	0.00047	0.00031	0.00031	0.00047
	12	0.05469	0.03125	0.03906	0.03125	0.07813	0.36641	0.35859	0.35891	0.35547	0.36188	0.00203	0.00203	0.00266	0.00172	0.00188



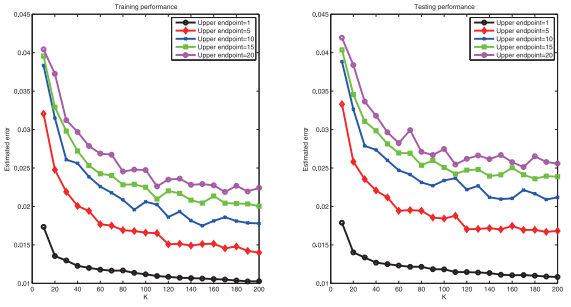
(a) On *Energy* data set



(b) On *Stock* data set



(c) On *Weather Ankara* data set



(d) On *Weather Izmir* data set

Fig. 7. Convergence of $\mathrm{FNR_{RWN}}$ with changes in the parameters of the upper endpoint and K using four representative data sets. The regularization factor was $C = 2^0$.

$K = 10$. In addition, $\mathrm{FNR_{RWN}}$ could learn thousands of times faster than $\mathrm{FNR_{BP\text{-}III}}$ and hundreds of times faster than $\mathrm{FNR_{RBF\text{-}II}}$. This indicates that our proposed $\mathrm{FNR_{RWN}}$ method is effective and efficient. According to these experimental results, we can summarize the advantages of $\mathrm{FNR_{RWN}}$ as follows: a simpler network architecture (only 10 hidden layer nodes), fewer learning parameters (C and K), faster training speed (less than 1 s), and better forecasting performance (smaller training and testing errors).

In addition, we found that $\mathrm{FNR_{RWN}}$ obtained better fits to the larger α-cut set of the TFN output. Thus, we provide the following analysis for this experimental observation. The error between $[T]_\alpha^L$ and $[Y]_\alpha^L$ can be represented as a function with respect to α, i.e.,

$$\begin{aligned} e_L(\alpha) &= [T]_\alpha^L - [Y]_\alpha^L \\ &= \sum_{j=1}^{K} \frac{\beta_j}{1+\exp\left[-\left[\alpha r_{j2} + (1-\alpha) r_{j1}\right]\right]} - \left[\alpha y_{j2} + (1-\alpha) y_{j1}\right] \\ &= \sum_{j=1}^{K} \frac{\beta_j}{1+\exp\left[-\left[\alpha\left(r_{j2} - r_{j1}\right) + r_{j1}\right]\right]} - \left[\alpha\left(y_{j2} - y_{j1}\right) + y_{j1}\right] \end{aligned}. \tag{51}$$

Without loss of generality, we assume that the RWN contains only one hidden layer node and we simplify Eq. (53) as

$$e_L(\alpha) = \frac{1}{1+\exp(-\alpha)} - \alpha, \ \ \alpha \in (0, 1]. \tag{52}$$

We can calculate the derivative of $e_L(\alpha)$ as

$$e_L'(\alpha) = -\frac{1+\exp(-\alpha) + [\exp(-\alpha)]^2}{[1+\exp(-\alpha)]^2} < 0. \tag{53}$$

This indicates that $e_L(\alpha)$ is a decreasing function. In a similar manner, we can obtain $e_U(\alpha) = [T]_\alpha^U - [Y]_\alpha^U$, which is also a decreasing function. This is the main reason why the training and testing errors decreased as α increased.

6. Conclusions and further work

In this study, we used RWNs to design the FNR model called FNR_{RWN}, which handles regression problem with fuzzy inputs and fuzzy outputs represented as TFNs. In FNR_{RWN}, no interaction is required to tune the weights and the biases of the RWN. The input layer weights and hidden layer biases are selected randomly, and the output layer weights are calculated analytically based on a new updating rule that minimizes the integrated squared error between the α-cut sets of the predicted fuzzy outputs and target fuzzy outputs. FNR_{RWN} uses a Riemann integral to approximately solve the integrated squared error. Our experimental results demonstrate the feasibility and effectiveness of FNR_{RWN}, i.e., FNR_{RWN} is convergent and it can obtain better regression performance with a simple network architecture, as well as a faster learning speed, compared with existing FNR models based on BP and RBF networks.

In our future research, we will address the following five issues. First, we will prove the universal convergence of FNR_{RWN}. Second, we will theoretically analyze the influence of random initialization on the approximation power of FNR_{RWN}. Third, we will develop the fuzzy RWNs based on granular computing paradigm [28] and maximum uncertainty [36,37,39,41]. Fourth, we will find some practical applications for FNR_{RWN}, e.g., image processing [24,44,55], sentiment analysis for Chinese text [9–11], and protein-protein interaction prediction [46,47]. Fifth, we will construct a RWN-based FNR model to handle more general types of fuzzy data.

Acknowledgments

We thank the Editor-in-Chief, Guest Editor, and two anonymous reviewers, whose valuable comments and suggestions help us to improve this paper significantly after five rounds of review. This study was supported by China Postdoctoral Science Foundation (2015M572361), Basic Research Project of Knowledge Innovation Program in Shenzhen (JCYJ20150324140036825), and National Natural Science Foundations of China (61503252, 71371063, 61473194 and 61473111).

References

[1] M. Alhamdoosh, D.H. Wang, Fast decorrelated neural network ensembles with random weights, Inf. Sci. 264 (2014) 104–117.
[2] A.R. Arabpour, M. Tata, Estimating the parameters of a fuzzy linear regression model, Iran. J. Fuzzy Syst. 5 (2) (2008) 1–19.
[3] F.L. Cao, Y.P. Tan, M.M. Cai, Sparse algorithms of random weight networks and applications, Expert Syst. Appl. 41 (5) (2014) 2457–2462.
[4] F.L. Cao, D.H. Wang, F.L Cao, H.Y. Zhu, Y.G. Wang, An iterative learning algorithm for feedforward neural networks with random weights, Inf. Sci. 328 (2016) 546–557.
[5] F.L. Cao, H.L. Ye, D.H. Wang, A probabilistic learning algorithm for robust modeling using neural networks with random weights, Inf. Sci. 313 (2015) 62–78.
[6] C.B. Cheng, E.S. Lee, Fuzzy regression with radial basis function network, Fuzzy Sets Syst. 119 (2) (2001) 291–301.
[7] P. Diamond, Fuzzy least squares, Inf. Sci. 46 (3) (1988) 141–157.
[8] P. D'Urso, Linear regression analysis for fuzzy/crisp input and fuzzy/crisp output data, Comput. Stat. Data Anal. 42 (1) (2003) 47–72.
[9] X.H. Fu, J.Q. Li, K. Yang, L.Z. Cui, L. Yang, Dynamic online HDP model for discovering evolutionary topics from Chinese social texts, Neurocomputing 171 (2016) 412–424.
[10] X.H. Fu, G. Liu, Y.Y. Guo, Z.Q. Wang, Multi-aspect sentiment analysis for Chinese online social reviews based on topic modeling and HowNet lexicon, Knowl. Based Syst. 37 (2013) 186–195.
[11] X.H. Fu, K. Yang, J.Z.X. Huang, L.Z. Cui, Dynamic non-parametric joint sentiment topic mixture model, Knowl. Based Syst. 82 (2015) 102–114.
[12] H. Hasanpour, H.R. Maleki, M.A. Yaghoubi, Fuzzy linear regression model with crisp coefficients: a goal programming approach, Iran. J. Fuzzy Syst. 7 (2) (2010) 19–39.
[13] C.M. He, Approximation of polygonal fuzzy neural networks in sense of Choquet integral norms, Int. J. Mach. Learn. Cybern. 5 (1) (2014) 93–99.
[14] A.E. Hoerl, R.W. Kennard, Ridge regression: biased estimation for nonorthogonal problems, Technometrics 42 (1) (2000) 80–86.
[15] D.H. Hong, S. Lee, H.Y. Do, Fuzzy linear regression analysis for fuzzy input-output data using shape-preserving operations, Fuzzy Sets Syst. 122 (3) (2001) 513–526.
[16] B. Igelnik, Y.H. Pao, Stochastic choice of basis functions in adaptive function approximation and the functional-link net, IEEE Trans. Neural Netw. 6 (6) (1995) 1320–1329.
[17] H. Ishibuchi, H. Tanaka, Fuzzy regression analysis using neural networks, Fuzzy Sets Syst. 50 (3) (1992) 257–265.
[18] H. Ishibuchi, H. Tanaka, An architecture of neural networks with interval weights and its application to fuzzy regression analysis, Fuzzy Sets Syst. 57 (1) (1993) 27–39.
[19] H. Ishibuchi, K. Kwon, H. Tanaka, A learning algorithm of fuzzy neural networks with triangular fuzzy weights, Fuzzy Sets Syst. 71 (3) (1995) 277–293.
[20] C. Kao, C.L. Chyu, A fuzzy linear regression model with better explanatory power, Fuzzy Sets Syst. 126 (3) (2002) 401–409.
[21] C. Kao, C.L. Chyu, Least-squares estimates in fuzzy regression analysis, Eur. J. Oper. Res. 148 (2) (2003) 426–435.
[22] R. Khemchandani, A. Karpatne, S. Chandra, Twin support vector regression for the simultaneous learning of a function and its derivatives, Int. J. Mach. Learn. Cybern. 4 (1) (2013) 51–63.
[23] H.L. Kwang, First Course on Fuzzy Theory and Applications, Springer Science & Business Media, 2006.
[24] F. Liu, D. Zhang, L.L. Shen, Study on novel curvature features for 3d fingerprint recognition, Neurocomputing 168 (2015) 599–608.
[25] A.B. Musa, Comparative study on classification performance between support vector machine and logistic regression, Int. J. Mach. Learn. Cybern. 4 (1) (2013) 13–24.
[26] A.B. Musa, A comparison of l_1-regularizion, PCA, KPCA and ICA for dimensionality reduction in logistic regression, Int. J. Mach. Learn. Cybern. 5 (6) (2014) 861–873.
[27] Y.H. Pao, Y. Takefuji, Functional-link net computing, IEEE Comput. 25 (5) (1992) 76–79.
[28] W. Pedrycz, Granular Computing: Analysis and Design of Intelligent Systems, CRC Press/Francis Taylor, Boca Raton, 2013.
[29] D.E. Rumelhart, G.E. Hinton, R.J. Williams, Learning representations by back-propagating errors, Nature 323 (1986) 533–536.
[30] M. Sakawa, H. Yano, Fuzzy linear regression analysis for fuzzy input output data, Inf. Sci. 63 (3) (1992a) 191–206.
[31] M. Sakawa, H. Yano, Multiobjective fuzzy linear regression analysis for fuzzy input-output data, Fuzzy Sets Syst. 47 (2) (1992b) 173–181.
[32] S. Scardapane, D. Comminiello, M. Scarpiniti, A. Uncini, A semi-supervised random vector functional-link network based on the transductive framework, Inf. Sci. (2016), in press. doi:10.1016/j.ins.2015.07.060
[33] S. Scardapane, D.H. Wang, M. Panella, A. Uncini, Distributed learning for random vector functional-link networks, Inf. Sci. 301 (2015b) 271–284.

[34] W.F. Schmidt, M.A. Kraaijveld, R.P.W. Duin, Feedforward neural networks with random weights, in: Proceedings of 11th IAPR International Conference on Pattern Recognition, Conference B: Pattern Recognition Methodology and Systems, II, 1992, pp. 1–4.
[35] H. Tanaka, S. Uejima, K. Asai, Linear regression analysis with fuzzy model. IEEE transactions on systems, Man Cybern. 12 (6) (1982) 903–907.
[36] X.Z. Wang, Uncertainty in learning from big data-editorial, J. Intell. Fuzzy Syst. 28 (5) (2015) 2329–2330.
[37] X.Z. Wang, R.A.R. Ashfaq, A.M. Fu, Fuzziness based sample categorization for classifier performance improvement, J. Intell. Fuzzy Syst. 29 (3) (2015) 1185–1196.
[38] X.Z. Wang, M.H. Ha, Fuzzy linear regression analysis, Fuzzy Sets Syst. 51 (2) (1992) 179–188.
[39] X.Z. Wang, J.R. Hong, On the handling of fuzziness for continuous-valued attributes in decision tree generation, Fuzzy Sets Syst. 99 (3) (1998) 283–290.
[40] H.F. Wang, R.C. Tsaur, Insight of a fuzzy regression model, Fuzzy Sets Syst. 112 (3) (2000) 355–369.
[41] X.Z. Wang, H.J. Xing, Y. Li, Q. Hua, C.R. Dong, W. Pedrycz, A study on relationship between generalization abilities and fuzziness of base classifiers in ensemble learning, IEEE Trans. Fuzzy Syst. 23 (5) (2015) 1638–1654.
[42] H.C. Wu, Linear regression analysis for fuzzy input and output data using the extension principle, Comput. Math. Appl. 45 (12) (2003) 1849–1859.
[43] M.S. Yang, T.S. Lin, Fuzzy least-squares linear regression analysis for fuzzy input-output data, Fuzzy Sets Syst. 126 (3) (2002) 389–399.
[44] M. Yang, P.F. Zhu, F. Liu, L.L. Shen, Joint representation and pattern learning for robust face recognition, Neurocomputing 168 (2015) 70–80.
[45] K.K. Yen, S. Ghoshray, G. Roig, A linear regression model using triangular fuzzy number coefficients, Fuzzy Sets Syst. 106 (2) (1999) 167–177.
[46] Z.H. You, J.Z. Yu, L. Zhu, S. Li, Z.K. Wen, A mapreduce based parallel SVM for large-scale predicting protein-protein interactions, Neurocomputing 145 (2014a) 37–43.
[47] Z.H. You, L. Zhu, C.H. Zheng, H.J. Yu, S.P. Deng, Z. Ji, Prediction of protein-protein interactions from amino acid sequences using a novel multi-scale continuous and discontinuous feature set, BMC Bioinform. 15 (2014b).(Suppl 15): S9
[48] Y.Q. Zhang, F.L. Cao, C.W. Yan, Learning rates of least-square regularized regression with strongly mixing observations, Int. J. Mach. Learn. Cybern. 3 (4) (2012) 277–283.
[49] D. Zhang, L.F. Deng, K.Y. Cai, A. So, Fuzzy nonlinear regression with fuzzified radial basis function network, IEEE Trans. Fuzzy Syst. 13 (6) (2005) 742–760.
[50] L. Zhang, P.N. Suganthan, A comprehensive evaluation of random vector functional link networks, Inf. Sci., (2015), in press. doi:10.1016/j.ins.2015.09.025.
[51] J.W. Zhao, Z.H. Wang, F.L. Cao, D.H. Wang, A local learning algorithm for random weights networks, Knowl. Based Syst. 74 (2015) 159–166.
[52] S.F. Zheng, Gradient descent algorithms for quantile regression with smooth approximation, Int. J. Mach. Learn. Cybern. 2 (3) (2011) 191–207.
[53] S.F. Zheng, A fast algorithm for training support vector regression via smoothed primal function minimization, Int. J. Mach. Learn. Cybern. 6 (1) (2015) 155–166.
[54] P. Zhou, M. Yuan, H. Wang, Z. Wang, T.Y. Chai, Multivariable dynamic modeling for molten iron quality using online sequential random vector functional-link networks with self-feedback connections, Inf. Sci. 325 (2015) 237–255.
[55] Z.X. Zhu, S. Jia, S. He, Y.W. Sun, Z. Ji, L.L. Shen, Three-dimensional Gabor feature extraction for hyperspectral imagery classification using a memetic framework, Inf. Sci. 298 (2015) 274–287.