Journal of Southeast University (English Edition) Vol. 29, No. 4, pp. 400 - 406 Dec. 2013 ISSN 1003—7985

RBF neural network regression model based on fuzzy observations

Zhu Hongxia[1,2] Shen Jiong[1] Su Zhigang[1]

([1]School of Energy and Environment, Southeast University, Nanjing 210096, China)

([2]School of Energy and Power Engineering, Nanjing Institute of Technology, Nanjing 211167, China)

Abstract: A fuzzy observations-based radial basis function neural network (FORBFNN) is presented for modeling nonlinear systems in which the observations of response are imprecise but can be represented as fuzzy membership functions. In the FORBFNN model, the weight coefficients of nodes in the hidden layer are identified by using the fuzzy expectation-maximization (EM) algorithm, whereas the optimal number of these nodes as well as the centers and widths of radial basis functions are automatically constructed by using a data-driven method. Namely, the method starts with an initial node, and then a new node is added in a hidden layer according to some rules. This procedure is not terminated until the model meets the preset requirements. The method considers both the accuracy and complexity of the model. Numerical simulation results show that the modeling method is effective, and the established model has high prediction accuracy.

Key words: radial basis function neural network (RBFNN); fuzzy membership function; imprecise observation; regression model

doi: 10.3969/j.issn.1003-7985.2013.04.009

Radial basis function (RBF) neural networks have been applied and evaluated in a wide variety of fields[1-7]. Most of the recent RBF neural networks assume a perfect knowledge of the values of the response for learning samples. That is to say, the observations are supposed to be precise (i.e., point-valued). However, in many real-life situations, such standard observations cannot be obtained. Information about the response is usually obtained through measuring devices or sensors with limited precision. Therefore, it is necessary to extend the RBF neural networks to deal with imprecise data and propose a new methodology in the imprecise setting.

Received 2013-08-13.

Biographies: Zhu Hongxia (1980—), female, graduate; Shen Jiong (corresponding author), male, doctor, professor, shenj@seu.edu.cn.

Foundation items: The National Natural Science Foundation of China (No. 51106025, 51106027, 51036002), Specialized Research Fund for the Doctoral Program of Higher Education (No. 20130092110061), the Youth Foundation of Nanjing Institute of Technology (No. QKJA201303).

Citation: Zhu Hongxia, Shen Jiong, Su Zhigang. RBF neural network regression model based on fuzzy observations[J]. Journal of Southeast University (English Edition), 2013, 29(4): 400 - 406. [doi: 10.3969/j.issn.1003-7985.2013.04.009]

Up to date, there is little literature on extending the RBF neural networks to deal with imprecise data. Cheng and Lee[1] proposed a fuzzy version of the RBF neural network, in which the weight coefficients are assumed to be fuzzy. In this regard, the output of such a fuzzy RBF neural network is fuzzy, and the application of fuzzy weight coefficients usually leads to learning complexity. In practice, it is more appropriate to obtain precise prediction in some sense, although the training samples can only be imprecise. In this paper, we suppose that the imprecise data are represented by fuzzy membership functions and investigate the RBF network regression with crisp inputs and fuzzy output. Unlike the existing family fuzzy RBF neural networks, our proposed method does not require the weight coefficients to be fuzzy, which reduces the learning complexity, and the prediction output is precise point value.

There exist two obstacles preventing the classical RBF neural networks to deal with imprecise data. The first one is how to determine the radial basis functions (i.e., the centers and widths of nodes in the hidden layer) when the response is a fuzzy membership function. The second one is how to identify the linear functions (i.e., the weight coefficients of nodes in the hidden layer) when observations (of responses) are fuzzy membership functions. To solve the first problem, we propose a data-driven automatic method. This method treats the input data and output data separately, but it considers both the structure of input data and the performance of the RBF neural networks so as to find the optimal number of nodes in the hidden layer with an acceptable accuracy. To identify final linear behaviors, a novel algorithm for estimating parameters in a fuzzy setting is needed. Recently, a significant contribution is the extension of the expectation-maximization (EM) algorithm[8] to fuzzy data, i.e., the so-called fuzzy EM algorithm[9]. Using the fuzzy EM algorithm, the weight coefficients in RBF neural networks can be identified when observations are fuzzy membership functions. Therefore, we propose a fuzzy observations-based RBF neural network (FORBFNN) regression model and it can be automatically data-driven.

1 Fuzzy EM Algorithm

Let $\boldsymbol{X}$, referred to as the complete-data vector, be a

random vector, taking value in sample space χ and describing the result of a random experiment. The probability density function (pdf) of $\boldsymbol{X}$ is denoted by $g(\boldsymbol{x}, \boldsymbol{\psi})$, where $\boldsymbol{\psi} = \{\psi_1, \psi_2, \ldots, \psi_d\}^{\mathrm{T}}$ is a column vector of unknown parameters with parameter space Ω.

If $\boldsymbol{x}$, a realization of $\boldsymbol{X}$, is known exactly, we can compute the maximum likelihood estimate (MLE) of $\boldsymbol{\psi}$ as any value maximizing the complete-data likelihood function:

$$L(\boldsymbol{\psi}; \boldsymbol{x}) = g(\boldsymbol{x}; \boldsymbol{\psi}) \tag{1}$$

However, $\boldsymbol{x}$ is usually not observed precisely, e. g., only partial information about $\boldsymbol{x}$ is available in the form of a fuzzy subset $\tilde{\boldsymbol{x}}$ of Ω_X. Therefore, the complete-data likelihood function (1) should be extended. Given $\tilde{\boldsymbol{x}}$ and assuming its membership function to be Borel measurable, the probability of fuzzy set $\tilde{\boldsymbol{x}}$ can be computed according to Zadeh's definition of the probability of a fuzzy event [10]. Thus, the observed-data likelihood in the imprecise setting can be defined as

$$L(\boldsymbol{\psi}; \tilde{\boldsymbol{x}}) = P(\tilde{\boldsymbol{x}}; \boldsymbol{\psi}) = \int_{\Omega_x} \mu_{\tilde{x}}(\boldsymbol{x})\, g(\boldsymbol{x}; \boldsymbol{\psi})\, \mathrm{d}\boldsymbol{x} \tag{2}$$

In the special case where the complete data $\boldsymbol{x} = \{x_1, x_2, \ldots, x_n\}^{\mathrm{T}}$ is a realization of an independent identically distributed (i. i. d.) random vector $\boldsymbol{X} = \{X_1, X_2, \ldots, X_n\}^{\mathrm{T}}$, assuming the joint membership function $\mu_{\tilde{x}}$ to be decomposed in the product of $\mu_{\tilde{x}_i}(i = 1, 2, \ldots, n)$, i. e.,

$$\mu_{\tilde{x}}(\boldsymbol{x}) = \prod_{i=1}^{n} \mu_{\tilde{x}_i}(x_i) \tag{3}$$

the likelihood function (2) can be written as a product of n items,

$$L(\boldsymbol{\psi}; \tilde{\boldsymbol{x}}) = \prod_{i=1}^{n} \int \mu_{\tilde{x}_i}(\boldsymbol{x})\, g(\boldsymbol{x}; \boldsymbol{\psi})\, \mathrm{d}\boldsymbol{x} \tag{4}$$

and the observed-data log likelihood is

$$\log L(\boldsymbol{\psi}; \tilde{\boldsymbol{x}}) = \sum_{i=1}^{n} \log \int \mu_{\tilde{x}_i}(\boldsymbol{x})\, g(\boldsymbol{x}; \boldsymbol{\psi})\, \mathrm{d}\boldsymbol{x} \tag{5}$$

The fuzzy EM algorithm[9] approaches the problem of maximizing the observed-data log likelihood $\log L(\boldsymbol{\psi}, \tilde{\boldsymbol{x}})$ by proceeding iteratively with the complete-data likelihood $\log L(\boldsymbol{\psi}, \boldsymbol{x}) = \log g(\boldsymbol{x}, \boldsymbol{\psi})$. Each iteration of the fuzzy EM algorithm involves two steps, the expectation step (E-step) and the maximization step (M-step).

The E-step consists in the calculation of

$$Q(\boldsymbol{\psi}, \boldsymbol{\psi}^{(q)}) = E_{\boldsymbol{\psi}^{(q)}}(\log[L(\boldsymbol{\psi}; \boldsymbol{X})] \mid \tilde{\boldsymbol{x}}) = \frac{\int \mu_{\tilde{x}} \log[L(\boldsymbol{\psi}; \boldsymbol{x})]\, g(\boldsymbol{x}, \boldsymbol{\psi}^{(q)})\, \mathrm{d}\boldsymbol{x}}{L(\boldsymbol{\psi}^{(q)}; \tilde{\boldsymbol{x}})} \tag{6}$$

where the expectation of $\log L(\boldsymbol{\psi}, \boldsymbol{X})$ is taken with respect to the conditional pdf of $\boldsymbol{x}$ given $\tilde{\boldsymbol{x}}$, using parameter vector $\boldsymbol{\psi}^{(q)}$.

$$g(\boldsymbol{x} \mid \tilde{\boldsymbol{x}}; \boldsymbol{\psi}^{(q)}) = \frac{\mu_{\tilde{x}}(\boldsymbol{x})\, g(\boldsymbol{x} \mid \boldsymbol{\psi}^{(q)})}{\int \mu_{\tilde{x}}(\boldsymbol{u})\, g(\boldsymbol{u} \mid \boldsymbol{\psi}^{(q)})\, \mathrm{d}\boldsymbol{u}}$$

The M-step requires the maximization of $Q(\boldsymbol{\psi}, \boldsymbol{\psi}^{(q)})$ with respect to $\boldsymbol{\psi}$ over the parameter space Ω, i. e., finding $\boldsymbol{\psi}^{(q+1)}$ such that

$$Q(\boldsymbol{\psi}^{(q+1)}, \boldsymbol{\psi}^{(q)}) \geqslant Q(\boldsymbol{\psi}, \boldsymbol{\psi}^{(q)}) \qquad \boldsymbol{\psi} \in \Omega$$

The fuzzy EM algorithm alternately repeats the E- and M-steps until the increment of observed-data likelihood becomes smaller than some threshold.

2 Proposed RBF Neural Network Based on Fuzzy Observations

2.1 Identification of radial basis functions

The basic topology of the RBF neural network comprises in sequence a hidden layer and a linear processing unit forming the output layer. Fig. 1 depicts this topology of a multi-input single-output network, where c represents the number of nodes in the hidden layer. The set of input-output data pairs can be symbolized as $T = \{(\boldsymbol{u}_i, x_i) \in \mathbf{R}^p \times \mathbf{R} \mid x_i = f(\boldsymbol{u}_i),\ i = 1, 2, \ldots, n\}$, where n is the number of training samples, $\boldsymbol{u}_i = \{u_{i1}, u_{i2}, \ldots, u_{ip}\}^{\mathrm{T}}$ is the i-th p-dimensional input vector and x_i is the i-th output variable. The Gaussian type RBF functions with the following form are selected:

$$h_k(\boldsymbol{u}_i) = \exp\left(-\frac{\|\boldsymbol{u}_i - \boldsymbol{v}_k\|^2}{s_k^2}\right) \tag{7}$$

where $\boldsymbol{v}_k$ is the p-dimensional center and s_k is the width of the k-th ($k = 1, 2, \ldots, c$) hidden unit of the RBF; $\|\boldsymbol{u}_i - \boldsymbol{v}_k\|$ is an Euclidean distance between the input vector and the center. The estimated output of the RBF neural network can be calculated using the following linear regression functional:

$$\hat{x}_i = \hat{f}(\boldsymbol{u}_i) = \sum_{k=1}^{c} w_k h_k(\boldsymbol{u}_i) \tag{8}$$

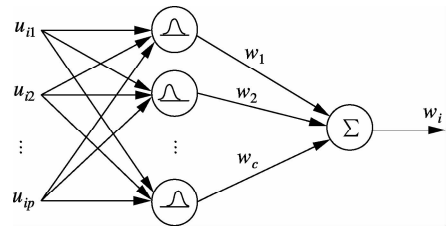

Fig. 1 Basic topology of an RBF neural network

This section presents the strategy to identify radial basis functions when the observations are in the following form:

$$T = \{\boldsymbol{e}_i \mid \boldsymbol{e}_i = (\boldsymbol{u}_i, \tilde{x}_i),\ i = 1, 2, \ldots, n\} \tag{9}$$

where $\tilde{x}_i$ is the imprecisely observed values of response x_i represented as a fuzzy membership function.

To address the proposed strategy, a new performance measure is first needed, which is called mean square fuzzy expectation error and defined as

$$\mathrm{MSE}_{\mathrm{fuzzy}} = \frac{1}{n}\sum_{i=1}^{n}\mathrm{MSE}_{\mathrm{fuzzy},i} = \frac{1}{n}\sum_{i=1}^{n}\|E[\tilde{x}_i] - x_i\|^2 \tag{10}$$

where $E[\tilde{x}_i] = \int_{\Omega_x}\omega\mu_{\tilde{x}_i}(\omega)\,\mathrm{d}\omega$ is the fuzzy expectation associated to $\tilde{x}_i$. When $\tilde{x}_i$ degenerate to be crisp point values, criteria (10) will degenerate to be the traditional mean square error (MSE).

The proposed strategy is an iterative procedure with two termination conditions: 1) The approximate accuracy is not higher than a given acceptable performance ε, i.e., $\mathrm{MSE}_{\mathrm{fuzzy}} \leqslant \varepsilon$; 2) The number of nodes in the hidden layer (i.e., node base) is bigger than a given maximum number $R_{\max}$, i.e., $c > R_{\max}$. Either condition 1) or condition 2) is satisfied, the iteration is then terminated. These two conditions can ensure a desirable tradeoff between the accuracy and the size of the node base according to the designer's intuition or expertise.

First, an initial node in the hidden layer is generated. The initial node is extracted by a simple method. The center and width of such node are determined by

$$v_{1j} = \frac{1}{n}\sum_{i=1}^{n}u_{ij} \qquad j = 1, 2, \ldots, p \tag{11}$$

$$s_1 = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}\|\boldsymbol{u}_i - \boldsymbol{v}_1\|^2} \tag{12}$$

In this original initial node base, the weight coefficients are identified using the fuzzy EM algorithm, which will be detailed in the following section.

Secondly, a new node in the hidden layer is constructed. The vector that has the worst $\mathrm{MSE}_{\mathrm{fuzzy},i}$, denoted by $\boldsymbol{u}_{i'}$, is considered as the candidate center of this new node:

$$\boldsymbol{u}_{i'} = \{\boldsymbol{u}_j \mid \mathrm{MSE}_{\mathrm{fuzzy},j} = \max_{i=1,2,\ldots,n}\{\mathrm{MSE}_{\mathrm{fuzzy},i}\}\} \tag{13}$$

Because the candidate center is only based on performance error, it is possible for an outlier to be considered as a new center. Although the preprocessing of data maybe detects and eliminates the outliers, it is still needed to reduce the effects of the noisy data and exclude the chance of an outlier to become a center. In addition, we do not want the new candidate center to be too close to the existing centers. Therefore, the following conditions should be satisfied:

$$\sum_{i=1}^{n}\mu_{i,i'}^2 \geqslant \Delta_1 \tag{14a}$$

$$\min_{k=1,2,\ldots,c}\|\boldsymbol{u}_{i'} - \boldsymbol{v}_k\|^2 \geqslant \Delta_2 \tag{14b}$$

where Δ_1 and Δ_2 are constants; $\mu_{i,i'}$ is the membership degree of the i-th data belonging to the i'-th cluster, determined in the following way [11]:

$$\mu_{i,i'} = \left(\sum_{k=1}^{c}\frac{d_{i,i'}^2 - \min d_{i,*}^2}{d_{i,k}^2 - \min d_{i,*}^2}\right)^{-1} \tag{15}$$

where $d_{i,k}$ is the distance between the i-th sample and the k-th center, defined as $d_{ik} = \|\boldsymbol{u}_i - \boldsymbol{v}_k\|$, and the item $\min d_{i,*}^2$ is defined as $\min d_{i,*}^2 = \min_i\{d_{i,k}^2 \mid k = 1, 2, \ldots, c\} - \gamma$. The bigger the γ, the wider the width of the membership function. Usually, we have $\gamma > 0$.

The role of condition (14a) is to prevent an outlier to be a new center, and the condition (14b) ensures that the new center is not located very close to the other existing centers. Hence, the constants Δ_1 and Δ_2 can be defined as

$$\Delta_1 = \frac{\eta}{c}\sum_{k=1}^{c}\sum_{i=1}^{n}\mu_{i,k}^2 \tag{16}$$

$$\Delta_2 = \frac{\sum_{i=1}^{n}\mu_{i,i'}^2\|\boldsymbol{u}_i - \boldsymbol{u}_{i'}\|^2}{\sum_{i=1}^{n}\mu_{i,i'}^2} \tag{17}$$

where $0 \leqslant \eta \leqslant 1$ is a soft factor used to control the effects of the average membership degrees of all data over all centers. Through our experiments, we find that sometimes there are no points in the data set satisfying $\sum_{i=1}^{n}\mu_{i,i'}^2 \geqslant \frac{1}{c}\sum_{k=1}^{c}\sum_{i=1}^{n}\mu_{i,k}^2$. In other words, we usually can only obtain one node, i.e., the initial node base. In this case, the constraint can be softened by using a small η.

If the selected vector $\boldsymbol{u}_{i'}$ satisfies (14), then it is declared as the center of a new node. Otherwise, it is marked as an outlier and the process of selecting the vector that has the worse performance is repeated without considering the outliers. When none of the existing vectors satisfies (14), the procedure is terminated to avoid over-fitting. The center and width of the new node are defined as

$$v_{\mathrm{new},j} = u_{i',j} \qquad j = 1, 2, \ldots, p \quad \text{or} \quad \boldsymbol{v}_{\mathrm{new}} = \boldsymbol{u}_{i'} \tag{18}$$

$$s_{\mathrm{new}} = \sqrt{\frac{\sum_{i=1}^{n}\mu_{i,i'}^2\|\boldsymbol{u}_i - \boldsymbol{v}_{\mathrm{new}}\|^2}{\sum_{i=1}^{n}\mu_{i,i'}^2}} \tag{19}$$

Finally, once the new node is added to the node base, the node number increases one, i.e., $c = c + 1$, and we have $\boldsymbol{v}_c = \boldsymbol{v}_{\mathrm{new}}$, $s_c = s_{\mathrm{new}}$. Due to the added node, the node base should be updated. The centers $\boldsymbol{v}_k$ of the previous $(c-1)$ nodes existing in the node base can be maintained whereas their widths $s_k (k = 1, 2, \ldots, c-1)$ can be updated according to Eq. (19) only by replacing index i' with the index k $(k = 1, 2, \ldots, c-1)$.

From the above interpretations, it is evident that the

computational complexity of the proposed strategy is $nO(R_{\max})$.

2.2 Identification of weight coefficients of RBF by fuzzy EM algorithm

For the following discussion, we first transform the estimated output in Eq. (8) to the following vector or matrix form:

$$\hat{x}_i = \boldsymbol{h}_k^{\mathrm{T}}\boldsymbol{w} \qquad \text{or} \qquad \hat{\boldsymbol{x}} = \boldsymbol{H}^{\mathrm{T}}\boldsymbol{w} \tag{20}$$

where

$$\boldsymbol{h}_k = \begin{Bmatrix} h_1(\boldsymbol{u}_i) \\ h_2(\boldsymbol{u}_i) \\ \vdots \\ h_c(\boldsymbol{u}_i) \end{Bmatrix}, \quad \boldsymbol{w} = \begin{Bmatrix} w_1 \\ w_2 \\ \vdots \\ w_c \end{Bmatrix}$$

$$\boldsymbol{H} = \begin{bmatrix} h_1(\boldsymbol{u}_1) & h_2(\boldsymbol{u}_1) & \cdots & h_c(\boldsymbol{u}_1) \\ h_1(\boldsymbol{u}_2) & h_2(\boldsymbol{u}_2) & \cdots & h_c(\boldsymbol{u}_2) \\ \vdots & \vdots & & \vdots \\ h_1(\boldsymbol{u}_n) & h_2(\boldsymbol{u}_n) & \cdots & h_c(\boldsymbol{u}_n) \end{bmatrix}$$

To solve the above regression with fuzzy membership functions as output, it can be assumed that each component x_i of the complete-data vector $\boldsymbol{x}$ is a realization of a normal random variable X_i with mean $\boldsymbol{h}_i^{\mathrm{T}}\boldsymbol{w}$ and standard deviations σ, and an observer encodes his/her partial and uncertain knowledge of x_i in the form of fuzzy membership functions $\tilde{x}_i$. With such assumption, the complete parameter vector is thus $\psi = (\boldsymbol{w}^{\mathrm{T}}, \sigma)^{\mathrm{T}}$ that should be identified in the case where only $\tilde{x}_i$ can be observed.

According to the above interpretations, the complete-data pdf can be defined as

$$g(\boldsymbol{x};\boldsymbol{\psi}) = \prod_{i=1}^{n} g(x_i;\boldsymbol{\psi}) = \prod_{i=1}^{n} \frac{1}{\sigma\sqrt{2\pi}}\exp\left(\frac{-(x_i - \boldsymbol{h}_i^{\mathrm{T}}\boldsymbol{w})^2}{2\sigma^2}\right) \tag{21}$$

By using the complete-data pdf, the complete-data log likelihood is computed as

$$\log L(\boldsymbol{\psi};\boldsymbol{x}) = \sum_{i=1}^{n} \log g(x_i;\boldsymbol{\psi}) = -\frac{n}{2}\log(2\pi) - n\log\sigma - \frac{1}{2\sigma^2}(\boldsymbol{x}^{\mathrm{T}}\boldsymbol{x} - 2\boldsymbol{w}^{\mathrm{T}}\boldsymbol{H}\boldsymbol{x} + \boldsymbol{w}^{\mathrm{T}}\boldsymbol{H}\boldsymbol{H}^{\mathrm{T}}\boldsymbol{w}) \tag{22}$$

Taking the expectation of $\log L(\boldsymbol{\psi}, \boldsymbol{x})$ conditionally on the observed $\tilde{x}_i$ and using the approximation of $\boldsymbol{\psi}$, $\boldsymbol{\psi}^{(q)}$, to perform the E-step, we can obtain

$$Q(\boldsymbol{\psi}, \boldsymbol{\psi}^{(q)}) = E_{\boldsymbol{\psi}^{(q)}}(\log[L(\boldsymbol{\psi};\boldsymbol{x})] \mid \tilde{\boldsymbol{x}}) = -\frac{n}{2}\log(2\pi) - n\log\sigma - \frac{1}{2\sigma^2}\left(\sum_{i=1}^{n}\alpha_i^{(q)} - 2\boldsymbol{w}^{\mathrm{T}}\boldsymbol{H}\boldsymbol{\beta}^{(q)} + \boldsymbol{w}^{\mathrm{T}}\boldsymbol{H}\boldsymbol{H}^{\mathrm{T}}\boldsymbol{w}\right) \tag{23}$$

where $\alpha_i^{(q)} = E_{\boldsymbol{\psi}^{(q)}}(X_i^2 \mid \tilde{x}_i)$, $\beta_i^{(q)} = E_{\boldsymbol{\psi}^{(q)}}(X_i \mid \tilde{x}_i)$ and $\boldsymbol{\beta}^{(q)} = \{E_{\boldsymbol{\psi}^{(q)}}(\boldsymbol{X} \mid \tilde{\boldsymbol{x}}) = \{E_{\boldsymbol{\psi}^{(q)}}(\boldsymbol{X}_1 \mid \tilde{\boldsymbol{x}}_1), \ldots, E_{\boldsymbol{\psi}^{(q)}}(X_n \mid \tilde{x}_n)\}^{\mathrm{T}}$. $\alpha_i^{(q)}$ and $\beta_i^{(q)}$ can be computed using the following equations.

For the sake of simplification, let us assume that P is the distribution of a univariate normal random variable X with mean m, standard deviation σ and pdf $g(x)$. Let $\tilde{x} = (a, b, c)$ be a triangular fuzzy number, with membership function

$$\mu_{\tilde{x}}(x) = \begin{cases} \dfrac{x-a}{b-a} & a \leqslant x \leqslant b \\ \dfrac{c-x}{c-b} & b \leqslant x \leqslant c \\ 0 & \text{otherwise} \end{cases} \tag{24}$$

Denoting by $g(x)$ the pdf of X, the probability of $\tilde{x}$ can be calculated as

$$P(\tilde{x}) = E[\mu_{\tilde{x}}(X)] = \frac{1}{b-a}\int_a^b x g(x)\,\mathrm{d}x - \frac{a}{b-a}(\Phi(b^*) - \Phi(a^*)) + \frac{c}{c-b}(\Phi(c^*) - \Phi(b^*)) - \frac{1}{c-b}\int_b^c x g(x)\,\mathrm{d}x \tag{25}$$

where $\Phi(\cdot)$ denotes the cumulative distribution function (cdf) of the standard normal distribution, and x^* denotes $(x-m)/\sigma$ for all x. It is easy to obtain that

$$\int_a^b x g(x)\,\mathrm{d}x = \frac{\sigma}{\sqrt{2\pi}}\left[\exp\left(-\frac{a^{*2}}{2}\right) - \exp\left(-\frac{b^{*2}}{2}\right)\right] + m(\Phi(b^*) - \Phi(a^*)) \tag{26}$$

Eq. (26) makes it possible to complete the calculation of $P(\tilde{x})$ using Eq. (25).

Let us now compute the expectation of X given $\tilde{x}$ for the conditional density of X. We have

$$E(X \mid \tilde{x}) = \frac{\int \mu_{\tilde{x}}(x) x g(x)\,\mathrm{d}x}{P(\tilde{x})} \tag{27}$$

where the denominator is given by Eq. (25). The numerator is

$$\int \mu_{\tilde{x}}(x) x g(x)\,\mathrm{d}x = \frac{1}{b-a}\int_a^b x^2 g(x)\,\mathrm{d}x - \frac{a}{b-a}\int_a^b x g(x)\,\mathrm{d}x + \frac{c}{c-b}\int_b^c x g(x)\,\mathrm{d}x - \frac{1}{c-b}\int_b^c x^2 g(x)\,\mathrm{d}x \tag{28}$$

which can be computed using Eq. (26) and

$$\int_a^b x^2 g(x)\,\mathrm{d}x = \frac{\sigma^2}{\sqrt{2\pi}}\left[a^*\exp\left(-\frac{a^{*2}}{2}\right) - b^*\exp\left(-\frac{b^{*2}}{2}\right)\right] + \frac{2\sigma m}{\sqrt{2\pi}}\left[\exp\left(-\frac{a^{*2}}{2}\right) - \exp\left(-\frac{b^{*2}}{2}\right)\right] + (m^2 + \sigma^2)(\Phi(b^*) - \Phi(a^*)) \tag{29}$$

We finally compute

$$E(X^2 \mid \tilde{x}) = \frac{\int \mu_{\tilde{x}}(x) x^2 g(x)\,\mathrm{d}x}{P(\tilde{x})} \tag{30}$$

The numerator is

$$\int \mu_{\tilde{x}}(x)x^2 g(x)\mathrm{d}x = \frac{1}{b-a}\int_a^b x^3 g(x)\mathrm{d}x - \frac{a}{b-a}\int_a^b x^2 g(x)\mathrm{d}x + \frac{c}{c-b}\int_b^c x^2 g(x)\mathrm{d}x - \frac{1}{c-b}\int_b^c x^3 g(x)\mathrm{d}x \quad (31)$$

which can be computed using Eq. (29) and

$$\int_a^b x^3 g(x)\mathrm{d}x = \frac{\sigma^3}{\sqrt{2\pi}}\left[(2+a^*)\exp\left(-\frac{a^{*2}}{2}\right) - (2+b^*)\exp\left(-\frac{b^{*2}}{2}\right)\right] + \frac{3\sigma^2 m}{\sqrt{2\pi}}\left[a^*\exp\left(-\frac{a^{*2}}{2}\right) - b^*\exp\left(-\frac{b^{*2}}{2}\right) + \sqrt{2\pi}(\Phi(b^*) - \Phi(a^*))\right] + \frac{3\sigma m^2}{\sqrt{2\pi}}\left[\exp\left(-\frac{a^{*2}}{2}\right) - \exp\left(-\frac{b^{*2}}{2}\right)\right] + m^3(\Phi(b^*) - \Phi(a^*)) \quad (32)$$

The M-step requires maximizing $Q(\psi, \psi^{(q)})$ with respect to ψ. This can be achieved by differentiating $Q(\psi, \psi^{(q)})$ with respect to $\boldsymbol{w}$ and σ, which results in

$$\frac{\partial Q(\psi, \psi^{(q)})}{\partial \boldsymbol{w}} = -\frac{1}{\sigma^2}(-\boldsymbol{H}\boldsymbol{\beta}^{(q)} + \boldsymbol{H}\boldsymbol{H}^{\mathrm{T}}\boldsymbol{w})$$

$$\frac{\partial Q(\psi, \psi^{(q)})}{\partial \sigma} = -\frac{n}{\sigma} + \frac{1}{\sigma^3}\left(\sum_{i=1}^{n}\alpha_i^{(q)} - 2\boldsymbol{w}^{\mathrm{T}}\boldsymbol{H}\boldsymbol{\beta}^{(q)} + \boldsymbol{w}^{\mathrm{T}}\boldsymbol{H}\boldsymbol{H}^{\mathrm{T}}\boldsymbol{w}\right)$$

Equating these derivatives to zero and solving for $\boldsymbol{w}$ and σ, we obtain the following unique solution:

$$\boldsymbol{w}^{(q+1)} = (\boldsymbol{H}\boldsymbol{H}^{\mathrm{T}})^{-1}\boldsymbol{H}\boldsymbol{\beta}^{(q)} \quad (33)$$

$$\sigma^{(q+1)} = \sqrt{\frac{1}{n}\left(\sum_{i=1}^{n}\alpha_i^{(q)} - 2(\boldsymbol{w}^{(q+1)})^{\mathrm{T}}\boldsymbol{H}\boldsymbol{\beta}^{(q)} + (\boldsymbol{w}^{(q+1)})^{\mathrm{T}}\boldsymbol{H}\boldsymbol{H}^{\mathrm{T}}\boldsymbol{w}^{(q+1)}\right)} \quad (34)$$

When the iteration terminates, we can obtain the regression weight coefficients $\boldsymbol{w}$ and thus obtain the final RBF neural network regression model with crisp inputs and fuzzy membership output.

3 Simulations

In this section, we validate the performance of FORBFNN by using a numerical simulation, in which the behavior of a nonlinear system is defined as

$$x = u\sin u \qquad u \in [0, 10] \quad (35)$$

To model the situation where response x can only be imprecisely observed, triangular fuzzy membership function (see Eq. (24)) is adopted. The core and support of such kind of fuzzy membership functions are generated according to the following two-step strategy:

Step 1 Generate the cores x_i of fuzzy observations, $x_i = f(u_i) + \varepsilon_i$, where $\varepsilon_i \sim N(0, \delta_{\max})$.

Step 2 The supports of the fuzzy membership function $\tilde{x}_i$ are defined as $[x_i - \delta_i, x_i + \delta_i]$, where $\delta_i \sim \mathrm{rand}[\delta_{\min}, \delta_{\max}]$.

In the simulation, four different study cases for deviation δ_i, i.e., $\delta_i \in \{[0, 0.01], [0, 1], [0, 2], [0, 3]\}$ are considered. Note that too wide range of imprecision is not considered, because too wide range of imprecision leads to useless training samples about the given system. In each study case, the size of training samples $n = 21$, accuracy threshold $\varepsilon = 10^{-5}$ and maximum node number $R_{\max} = 4, 5, 6, 7$. In addition, we consider that there are not outliers existing in the data sets. Therefore, the parameter η in Eq. (16) can be set to be zero.

To validate the performance, there are 101 testing samples produced according to

$$u_i^t = 0.1(i-1), \; x_i^t = u_i^t \sin u_i^t \qquad i = 1, 2, \cdots, 101 \quad (36)$$

In each study case, 100 data sets $T^{(l)}$ ($l = 1, 2, \cdots, 100$) are generated. The FORBFNN model is identified for each training set $T^{(l)}$. To measure the prediction accuracy of the identified model on each training set, we regularly generate the number of testing samples n_t from the input domain according to Eq. (27). The error is computed as the mean squared difference between the true output x_i and model prediction $\hat{x}_i^{(l)}$:

$$\mathrm{MSE}^{(l)} = \frac{1}{n_t}\sum_{i=1}^{n_t}(x_i - \hat{x}_i^{(l)})^2 \quad (37)$$

The numerical results are shown in Tab. 1, and four graphical results randomly selected from the 100 trials in each study case are shown in Fig. 2.

Tab. 1 Approximation and prediction errors (mean plus or minus one standard deviation) in different ranges of imprecision

Imprecision	Error type	Maximum node number $R_{\max}$				Remark
		4	5	6	7	
$\delta_i \in [0, 3]$	Approximation	6.306 1 ± 2.844 3	4.491 5 ± 2.326 4	4.609 5 ± 2.722 1	5.058 1 ± 2.490 7	$c = 5$
	Prediction	3.532 8 ± 1.452 5	2.437 3 ± 1.512 7	2.461 4 ± 1.686 2	2.625 2 ± 1.546 7	Over-fitting
$\delta_i \in [0, 2]$	Approximation	4.137 8 ± 1.668 2	3.629 8 ± 2.146 9	3.448 5 ± 2.269 3	3.344 4 ± 2.180 8	$c = 7$
	Prediction	2.598 7 ± 1.070 2	2.364 6 ± 1.551 1	2.139 8 ± 1.591 8	2.090 2 ± 1.498 4	
$\delta_i \in [0, 1]$	Approximation	3.459 1 ± 1.181 7	2.437 0 ± 1.944 7	2.447 0 ± 1.914 4	2.307 4 ± 1.641 2	$c = 7$
	Prediction	2.483 0 ± 0.883 9	1.688 8 ± 1.429 5	1.730 1 ± 1.436 3	1.587 3 ± 1.129 5	
$\delta_i \in [0, 0.01]$	Approximation	3.305 7 ± 0.005 4	0.265 1 ± 0.001 5	0.265 0 ± 0.001 7	0.264 9 ± 0.001 5	$c = 5, 6, 7$
	Prediction	2.321 7 ± 0.000 2	0.197 9 ± 0.000 3	0.197 9 ± 0.000 4	0.197 9 ± 0.000 4	

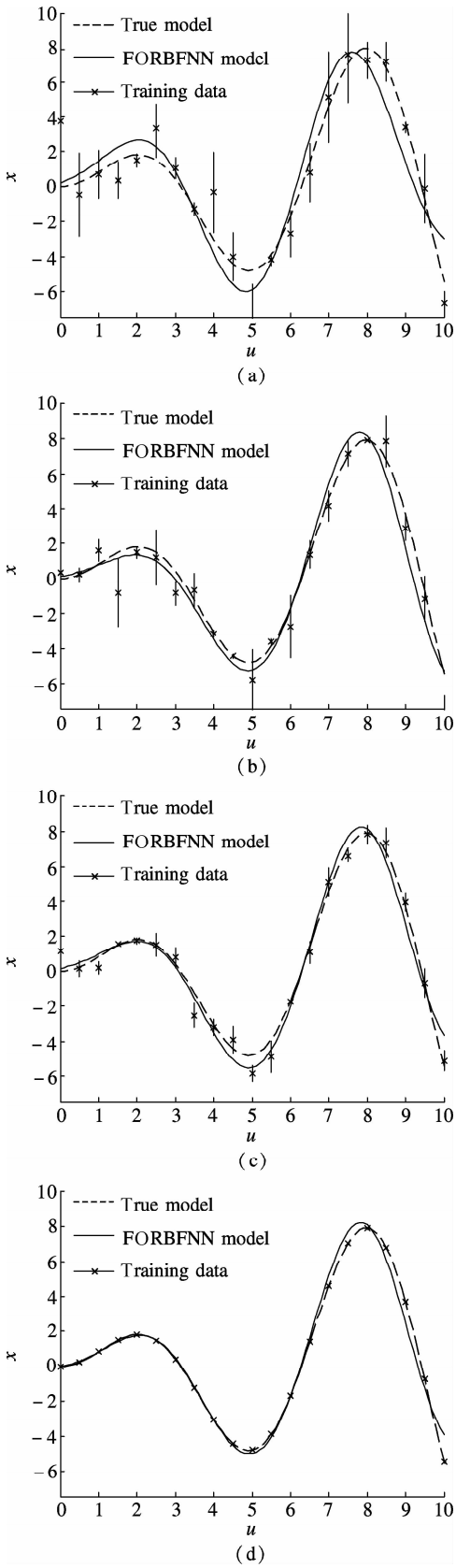

Fig. 2 Four data sets and prediction results randomly selected from 100 trials for four study cases. (a) $\delta_i \in [0,3]$; (b) $\delta_i \in [0,2]$; (c) $\delta_i \in [0,1]$; (d) $\delta_i \in [0,0.01]$

Fig. 2 illustrates the prediction results of the FORBFNN model in different ranges of imprecision. It can be seen that the predicted curves can approach the true behavior. The difference between the predicted curves and true behavior becomes smaller with the decrease in imprecision. Especially, such difference approaches zero in the precise and certain case.

Tab. 1 presents the approximate and prediction accuracies when the maximum number of nodes in the hidden layer $R_{\max}$ takes different values in different ranges of imprecision. They numerically show the performance of the FORBFNN model. For a given range of imprecision, the $R_{\max}$ corresponding to the highest approximate accuracy is determined as the node number without considering over-fitting. For instance in the first case $\delta_i \in [0,3]$ in Tab. 1, the highest approximate accuracy appears when $R_{\max} = 5$; therefore, the number of nodes in node base is 5, i.e., $c = 5$. We call a model over-fitting if its approximate accuracy becomes small whereas its associated prediction accuracy is high, see the case when $\delta_i \in [0,3]$ in Tab. 1. The over-fitting always occurs in the cases when high imprecision exists. In this regard, it suggests constructing the FORBFNN with small size of node base in the high imprecision cases. In addition, we can see that the performance of the FORBFNN can be improved when the number of nodes in the hidden layer increases to a limit.

In a word, the FORBFNN can deal with imprecise data, and its performance is determined by the ranges of imprecision. The lower the imprecision, the higher the approximate and predicated accuracies.

4 Conclusion

This paper proposes a fuzzy observations-based RBF neural network used to deal with problems when the response of a system can be represented by fuzzy membership functions. In this approach, the weight coefficients used to combine the outputs of the nodes in the hidden layer are identified by the fuzzy EM algorithm, and both the performance accuracy and the size of node number in node base (i.e., the complexity of the produced model) are considered simultaneously. The performance of the FORBFNN is illustrated by using some simulations.

There are still some further works that need to be studied for the extensive applications of our proposed method, such as how to establish fuzzy data from running data. If such an issue is solved, it can be widely used in engineering practice.

References

[1] Cheng C B, Lee E S. Fuzzy regression with radial basis function network[J]. *Fuzzy Sets and Systems*, 2001, **119**(2): 291 - 301.

[2] Lu S W, Basar T. Robust nonlinear system identification using neural-network models[J]. *IEEE Transactions on Neural Networks*, 1998, **9**(3): 407 - 429.

[3] Li Y, Qiang S, Zhuang X, et al. Robust and adaptive backstepping control for nonlinear systems using RBF neural networks[J]. *IEEE Transactions on Neural Networks*, 2004, **15**(3): 693 - 701.
[4] Panda S S, Chakraborty D, Pal S K. Flank wear prediction in drilling using back propagation neural network and radial basis function network[J]. *Applied Soft Computing*, 2008, **8**(2): 858 - 871.
[5] Rivas V M, Merelo J J, Castillo P A, et al. Evolving RBF neural networks for time-series forecasting with EvRBF[J]. *Information Sciences*, 2004, **165**(3/4): 207 - 220.
[6] Wei H K, Song W Z, Li Q. A RBF network based online modeling method for real-time cost model in power plant[J]. *Proceedings of the CSEE*, 2004, **24**(7): 246 - 252. (in Chinese)
[7] Kumar R, Ganguli R, Omkar S N. Rotorcraft parameter estimation using radial basis function neural network[J]. *Applied Mathematics and Computation*, 2010, **216**(2): 584 - 597.
[8] Dempster A P, Laird N M, Rubin D B. Maximum likelihood from incomplete data via EM algorithm[J]. *Journal of the Royal Statistical Society B*, 1977, **39**(1): 1 - 38.
[9] Denoeux T. Maximum likelihood estimation from fuzzy data using the EM algorithm[J]. *Fuzzy Sets and Systems*, 2011, **183**(1): 72 - 91.
[10] Zadeh L A. Probability measures of fuzzy events[J]. *Journal of Mathematical Analysis and Applications*, 1968, **23**(2): 421 - 427.
[11] Hoppner F, Klawonn F. Improved fuzzy partitions for fuzzy regression model[J]. *International Journal of Approximate Reasoning*, 2003, **32**(2/3): 85 - 102.

基于模糊观测数据的 RBF 神经网络回归模型

朱红霞[1,2] 沈 炯[1] 苏志刚[1]

([1] 东南大学能源与环境学院,南京 210096)
([2] 南京工程学院能源与动力工程学院,南京 211167)

摘要:提出了一种基于模糊观测数据的 RBF 神经网络(FORBFNN),用于解决一类输出不可精确测量但可用模糊隶属度来表征的非线性系统建模问题.神经网络模型中各隐层神经单元的权重系数采用一种新的模糊 EM 算法辨识获得;隐层神经单元的数量及径向基函数的中心和宽度基于一种数据驱动的方法自适应确定,即首先初始生成一个隐层单元,然后根据一定的规则逐步加入新的单元,该过程不断迭代直到模型满足预设要求.该方法同时考虑了模型的复杂度及预测精度.数值模拟实验结果表明该建模方法是有效的,且建立的模型具有较高的预测精度.

关键词:RBF 神经网络;模糊隶属度;不精确观测值;回归模型
中图分类号:TP183