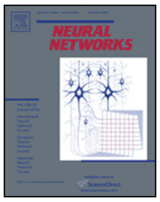
Contents lists available at ScienceDirect

Neural Networks

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



# An efficient self-organizing RBF neural network for water quality prediction

Hong-Gui Han [1], Qi-li Chen, Jun-Fei Qiao [*]

*College of Electronic and Control Engineering, Beijing University of Technology, Beijing, China*

ARTICLE INFO

*Article history:*
Received 31 May 2010
Received in revised form 20 February 2011
Accepted 25 April 2011

*Keywords:*
Flexibility structure
Self-organizing
Radial basis function (RBF)
Water quality prediction

ABSTRACT

This paper presents a flexible structure Radial Basis Function (RBF) neural network (FS-RBFNN) and its application to water quality prediction. The FS-RBFNN can vary its structure dynamically in order to maintain the prediction accuracy. The hidden neurons in the RBF neural network can be added or removed online based on the neuron activity and mutual information (MI), to achieve the appropriate network complexity and maintain overall computational efficiency. The convergence of the algorithm is analyzed in both the dynamic process phase and the phase following the modification of the structure. The proposed FS-RBFNN has been tested and compared to other algorithms by applying it to the problem of identifying a nonlinear dynamic system. Experimental results show that the FS-RBFNN can be used to design an RBF structure which has fewer hidden neurons; the training time is also much faster. The algorithm is applied for predicting water quality in the wastewater treatment process. The results demonstrate its effectiveness.

## 1. Introduction

Predicting water quality in the wastewater treatment process can provide a basis for water treatment plant management decisions that can minimize microbial risks and optimize the treatment operation (Ge & Frick, 2009). In many practical situations, however, it is difficult to predict accurately the quality of water in the treatment process due to a lack of knowledge of the parameters used in the process, or the presence of disturbances in the system. Thus, the predictor should take an appropriate action to counteract the presence of disturbances to which the system is subjected and should be able to adjust itself to the changing dynamics of the system. Radial basis function (RBF) neural networks have been successfully applied for solving dynamic system problems, because they can predict the behavior directly from input/output data (Ferrari, Bellocchio, Piuri, & Borghese, 2010; Lee & Ko, 2009; Wang & Yu, 2008). However, the number of hidden neurons in these RBF networks (Ferrari et al., 2010; Lee & Ko, 2009; Wang & Yu, 2008) is often assumed to be constant. In fact, if the number of hidden neurons is too large, the computational loading is heavy and, in general, the performance is poor. On the other hand, if the number of hidden neurons is too small, the learning performance may not be good enough to achieve the desired performance. For this reason, it is crucial to optimize the structure of RBF networks to improve performance. A brief review of the existing algorithms is given below.

The resource-allocating network (RAN), proposed by Platt (1991), was the first dynamic structure RBF neural network model. In the training procedure, new Gaussian neurons can be inserted into the hidden layer. Although the growing strategy usually results in a smaller network size than a fixed-size neural network, the RAN may become extremely large because insignificant hidden neurons are not pruned. Yingwei et al. introduced a strategy to prune hidden neurons, whose contribution is relatively small and incorporated this pruning strategy into the RAN, combining it with Extended Kalman Filter (EKF) (Li, Sundararajan, & Saratchandran, 1997). This network is referred to as the minimal resource-allocating network (MRAN); its applications are described in Li, Sundararajan, and Saratchandran (1998). MRAN is a popular tool used to design optimal RBF structures (Panchapakesan, Palaniswami, Ralph, & Manzie, 2002). However, the initial parameters of the new hidden neurons were not considered and the convergence of MRAN was not discussed (Lian, Lee, Sudhoff, & Stanislaw, 2008).

The self-organizing design of RBF neural networks has been discussed by several authors. Huang, Saratchandran, and Sundararajan (2004) proposed a simple sequential learning algorithm for RBF neural networks, which is referred to as the RBF growing and pruning algorithm (GAP-RBF). The original design of GAP-RBF was enhanced to produce a more advanced model known as GGAP-RBF (Huang, Saratchandran, & Sundararajan, 2005). Both GAP-RBF and GGAP-RBF neural networks use a pruning and growing strategy which is based on the "significance" of a neuron and links it to the learning accuracy. The structure of this RBF neural network is

* Corresponding author.
*E-mail addresses:* Rechardhan@sina.com, rechard112@emails.bjut.edu.cn (H.-G. Han), isibox@sina.com (J.-F. Qiao).
1 Fax: +86 10 67391631.

0893-6080/$ – see front matter Crown Copyright © 2011 Published by Elsevier Ltd. All rights reserved.
doi:10.1016/j.neunet.2011.04.006

simple and the computational time is less than the time used by a conventional RBF. However, when used in practice, GAP-RBF and GGAP-RBF require a complete set of training samples for the training process. In general, it is not possible for designers and technical domain experts to have *a priori* knowledge of the training samples prior to implementation. Recently, methods based on genetic algorithms (GA) have been used to change the number of hidden neurons (Gonzalez et al., 2003; Wu & Chow, 2007). The great advantage, in theory, of a GA is its ability to do global searching, but this comes at the cost of increased computation requirements. Feng (2006) proposed a self-organizing RBF neural network model based on a particle swarm optimization (PSO) method which attempts to solve the training time issue. The PSO method is used to construct the structure of the RBF neural network in order to simplify and speed up optimization. However, because the PSO method is a population-based evolutionary computation technique, the training time is still too long.

In this paper, a new algorithm, which is called the flexible structure RBF neural network (FS-RBFNN) is presented. This algorithm has several advantages: firstly, a neuron's average firing rate is used to determine whether new neurons should be inserted. The rate of firing used in the FS-RBFNN is similar to the spiking frequency of the presynaptic neuron in the biological neural system (Neves, Cooke, & Bliss, 2008). When the rate value of the hidden neuron is bigger than a given threshold value, new neurons will be inserted into the hidden layer.

Secondly, the connectivity of hidden neurons is estimated using an information-theoretic methodology. The connectivity between hidden neurons is obtained by measuring the mutual information (MI) (Krivov, Ulanowicz, & Dahiya, 2003) in the training process.

Thirdly, the convergence of the FS-RBFNN is analyzed both theoretically and experimentally. However, most of the self-organizing methods used for RBF structure design (Alexandridis, Sarimveis, & Bafas, 2003; Bortman & Aladjem, 2009; Shi, Yeung, & Gao, 2005) only analyze the convergence for learning process by experiments. In order to work in practice it is essential that the FS-RBFNN training process must be convergent. The FS-RBFNN has been specifically designed with this in mind.

Fourthly, the FS-RBFNN is particularly focused on reducing the retraining epochs after the structure has been modified by growing and pruning. The error required (ER) method is used to determine the initial values of the neurons inserted into the FS-RBFNN. It is well known that when a design algorithm adds (or prunes) hidden neurons to (or from) an existing RBF structure, it retrains the modified structure to adapt their connection weights.

The outline of this paper is as follows. Section 2 describes how the MI and the average firing rate are used to design the RBF; it also introduces the FS-RBFNN. Section 3 discusses and analyzes the algorithm. Section 4 presents experimental results which compare the performance of the algorithm with other similar algorithms. Section 5 concludes the paper.

## 2. Flexible structure radial basis function neural network (FS-RBFNN)

An RBF neural network has a simple neural network structure in terms of the direction of information flow. Since the performance of an RBF neural network is heavily dependent on its architecture, research has focused on self-organizing methods that can be used to design the architecture of three-layered RBF neural networks.

In order to design the structure of the RBF neural network automatically a dynamic tuning strategy is used in the FS-RBFNN. This strategy changes the topology of the RBF neural network by measuring the average firing rate of the neurons and the MI in the training process. The FS-RBFNN also uses an online learning algorithm to connect the weights in the RBF neural network training process.

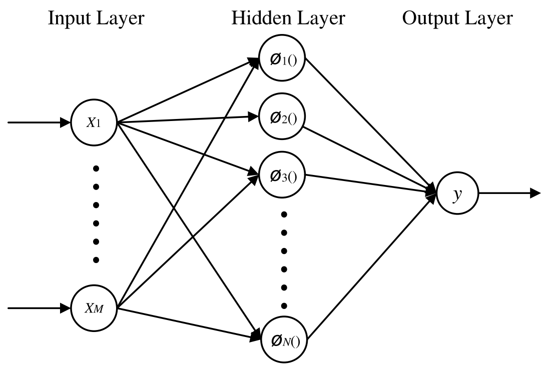


**Fig. 1.** The RBFNN structure.

### 2.1. *Radial basis function neural network (RBFNN)*

Fig. 1 shows the structure of a basic RBF network which consists of one input layer, one output layer and one hidden layer. In order to simplify the discussion, the RBF model used for analysis is multi-input and single output (MISO).

A single-output RBF neural network with $K$ hidden layer neurons can be described by

$$y = \sum_{k=1}^{K} w_k \theta_k(\mathbf{x}), \tag{1}$$

where $\mathbf{x} = (x_1, x_2, \ldots, x_M)^T$ and $y$ denote the input and output of the network, M is the number of input variables, $\mathbf{W} = [w_1, w_2, \ldots, w_K]$ is the connecting weights between the hidden neuron and the output layer, $\theta_k(\mathbf{x})$ is the output value of the $k$th hidden neuron, and

$$\theta_k(\mathbf{x}) = \mathrm{e}^{(-\|\mathbf{x}-\boldsymbol{\mu}_k\|/\sigma_k^2)}, \tag{2}$$

$\boldsymbol{\mu}_k$ denotes the center vector of the $k$th hidden neuron, and $\|\mathbf{x}-\boldsymbol{\mu}_k\|$ is the Euclidean distance between $\mathbf{x}$ and $\boldsymbol{\mu}_k$; $\sigma_k$ is the radius or width of the $k$th hidden neuron.

### 2.2. *Flexible structure radial basis function neural network (FS-RBFNN)*

The FS-RBFNN is based on the average firing rate of the hidden neurons and the MI intensities between the hidden layer and the output layer. Firstly, the activities of the hidden neuron are evaluated according to the average firing rate. The hidden neurons which have a high firing rate are divided into new neurons. Secondly the MI value is used to adjust the network structure i.e. the value of the MI is used as a measure of the connectivity between the hidden layer and the output layer; connections which have a small MI value will be pruned in order to simplify the structure of the RBFN. Finally the gradient-descent method, which is used to adjust the values of the parameters, ensures the exactitude of the FS-RBFNN. Two aspects of the flexible method used to design the structure of the RBFNN are discussed: the neuron splitting mechanism and the neuron adjusting mechanism used to design the structure.

*Case* 1: *Neuron splitting mechanism*

Consider the active firing (AF) of the hidden neurons described by the following equation:

$$Af_i = \rho \mathrm{e}^{-\|\mathbf{x}-\boldsymbol{\mu}_i\|} \frac{\theta_i(\mathbf{x})}{\sum_{i=1}^{K} \theta_i(\mathbf{x})}, \quad (i = 1, 2, \ldots, K) \tag{3}$$

where $Af_i$ is the active firing of the $i$th hidden neuron, $K$ is the number of hidden neurons, $\theta_i$ is the output value of the $i$th hidden neuron, $\rho \gg 1$ is a positive constant.

When $Af_i$ is larger than the activity threshold $Af_o$ ($0.05 < Af_o < 0.3$), the hidden neuron $i$ is the active neuron: there will be no connection between the $i$th hidden neuron and the output neuron. This hidden neuron will be divided; new neurons will be inserted to the hidden layer.

The centers and the radii of the new divided hidden neurons will be designed as

$$\boldsymbol{\mu}_{i,j} = \alpha_j \boldsymbol{\mu}_i + \beta_j \mathbf{x},$$
$$\sigma_{i,j} = \alpha_j \sigma_i, \quad j = 1, 2, \ldots, N_{new}, \tag{4}$$

where $0.95 < \alpha_j < 1.05$ and $0 < \beta_j < 0.1$ (in practice, in order to own better performance, $\alpha_j$ should be close to 1, and $\beta_j$ should be close to 0.05), $\boldsymbol{\mu}_i$ and $\sigma_i$ are the center and radius of the $i$th hidden neuron, $N_{new}$ ($N_{new} < 10$) is the number of the new inserted neurons which is decided by the rate of active firing $Af_i$.

The weights between the new neurons and the output layer are

$$w_{i,j} = r_j \frac{w_i \cdot \theta_i(\mathbf{x}) - e(\mathbf{x})}{\theta_{i,j}(\mathbf{x})},$$
$$\sum_{j=1}^{N_{new}} r_j = 1, \quad j = 1, 2, \ldots, N_{new}, \tag{5}$$

where $r_i$ is the allocating parameters for the new neurons, $\theta_i(x)$ is the output value of the $i$th hidden neuron, $\theta_{i,j}(x)$ is the output value of the new inserted $j$th hidden neuron. $w_i$ is weight of the $i$th hidden neuron, $e(\mathbf{x})$ is the current approximation error of the RBFNN

$$e(\mathbf{x}) = y(\mathbf{x}) - \widehat{y}(\mathbf{x}), \tag{6}$$

$y(\mathbf{x})$ is the output of the neural network and $\widehat{y}(\mathbf{x})$ is the required output for the current input sample $\mathbf{x}$ at time $t$.

*Case* 2: *Neuron adjusting mechanism*

MI, free of assumptions about the nature of the underlying joint distribution, provides a natural quantitative measure of the degree of statistical dependence between the stochastic variables (Krivov et al., 2003). We start with the definition of the MI between two stochastic variables $X_i$ and $Y$

$$M(X_i; Y) = \sum_{x,y} p(X_i, Y) \log_2 \frac{p(X_i, Y)}{p(X_i)p(Y)}, \tag{7}$$

where $p(X_i, Y)$ is the joint distribution, $p(X_i)$ and $p(Y)$ are the single-variable marginal. In this paper, $X_i$ is the output value of the $i$th hidden neuron; $Y$ is the network's output. If and only if the two variables $X_i$ and $Y$ are statistically independent—i.e., $p(X_i, Y) = p(X_i)p(Y)$.

The magnitude of the MI between two neurons is dependent on the information content of each neuron, in terms of their Shannon entropy $H(X_i)$ and $H(Y)$. This becomes clear through consideration of the relation

$$M(X_i; Y) = H(X_i) - H(X_i|Y) = H(Y) - H(Y|X_i) \tag{8}$$

where $H(X_i) = \ln(2\pi e)^K|\mathrm{COV}(X_i)|/2$, $\mathrm{COV}(X_i)$ is a standard covariance of $X_i$, $e$ is an irrational constant here, $K$ is the number of hidden neurons. And $H(Y) = \ln(2\pi e)|\mathrm{COV}(Y)|/2$.

The MI is semi-positive definite and equal to zero if and only if the two variables $X_i$ and $Y$ are statistically independent

$$M(X_i; Y) \le \min(H(X_i), H(Y)). \tag{9}$$

This bound, taken together with $M(X_i; Y) \ge 0$, implies that the normalized MI $m(X_i, Y)$

$$m(X_i; Y) = \frac{M(X_i; Y)}{\min(H(X_i), H(Y))}, \tag{10}$$

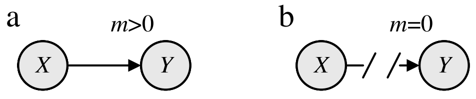


**Fig. 2.** The correlation between neurons $X_i$ and $Y$.

where $0 \le m(X_i; Y) \le 1$, in this spirit, more restricted measures of statistical dependence such as cross correlation have also been used to identify putative links between simultaneously recorded neurons.

In the RBFNN's structure, the neurons $X_i$ and $Y$ in the hidden and the output layer connect if and only if the normalized MI $m(X_i, Y)$ is positive (Fig. 2(a)); and if the normalized MI $m(X_i, Y)$ equals to zero which means neurons $X_i$ and $Y$ are independent (Fig. 2(b)). The details of the definition can be seen in Fig. 2.

The main steps of the proposed FS-RBFNN algorithm can be summarized as follows.

Step (1) Create an initial RBFNN consisting of three layers: an input layer, a hidden layer and an output layer. The number of neurons in the input and output layers is the same as the number of input and output variables in the problem that is being solved. The number of neurons in the hidden layer is randomly generated. Initialize all the parameters: the centers, radii, and connection weights of the RBFNN are all uniformly distributed with a small range.

Step (2) For the input sample $\mathbf{x}(t)$, train the RBFNN using training rule

$$\mathbf{W}(t+1) = \mathbf{W}(t) + \dot{\mathbf{W}}(t). \tag{11}$$

The centers and radii are adjusting by the gradient method (Qiao & Han, 2010). The mean squared error (MSE) is defined as

$$E = \frac{1}{2T} \sum_{t=1}^{T} \mathrm{e}^2(t), \tag{12}$$

$T$ is the number of the training samples.

Step (3) Compute the active firing rate, $AF$, of the hidden neurons using formula (3). New neurons are inserted according to the activity threshold $Af_o$. If $Af_i > Af_o$, go to step (4), else go to step (5).

Step (4) Split the $i$th hidden neuron and insert new hidden neurons. The initial parameters of the new inserted neurons are obtained from formulas (4) and (5).

Step (5) If $m(X_i; Y)$ is less than the threshold $m_0$ ($m_0 < 0.05$), go to step (6). Otherwise, go to step (7). The threshold $m_0$ is defined by the designers of the algorithm.

Step (6) Delete the connection between the hidden neuron $X_i$ and the output neuron $Y$; update the remaining RBFNN parameters. Find the neuron $X'$ in the hidden layer which has the minimal Euclidean distance between neuron $X_i$ and neuron $X'$. The parameters of the hidden neuron $X'$ are adjusted as follows

$$\boldsymbol{\mu}'_{X'} = \boldsymbol{\mu}_{X'},$$
$$\sigma'_{X'} = \sigma_{X'},$$
$$w'_{X'} = w_{X'} + w_{X_i} \frac{\theta_X(\mathbf{x}(t))}{\theta_{X'}(\mathbf{x}(t))}, \tag{13}$$

where $w_{X'}$ and $w'_{X'}$ are the connecting weight of the hidden neurons $X'$ before and after structure adjusting, $\boldsymbol{\mu}_{X'}$ and $\boldsymbol{\mu}'_{X'}$ are the center of the hidden neurons $X'$ before and after deleting neuron $X_i$, $\sigma_{X'}$ and $\sigma'_{X'}$ are the radius of the hidden neurons $X'$ before and after deleting neuron $X_i$.

Step (7) $t = t + 1$, go to Step (2). Stop when $t = T$.

It is now clear that FS-RBFNN can insert or prune the hidden neurons, depending on the hidden neurons' activities and the connecting MI. This execution behavior indicates that FS-RBFNN does not guide the RBFNN design process in a predefined and fixed way. The design scheme used in FS-RBFNN is a new efficient method for the RBFNN structure design. The essence of the proposed self-organizing algorithm is similar to the adaptive search strategy in the brains (Samanwoy & Hojjat, 2009). This character is useful for the applications especially when the systems with dynamic models such as the wastewater treatment.

## 3. Convergence discussion

For our proposed FS-RBFNN, the convergence of the algorithm with respect to the topology adjusting is an important issue and needs careful investigation. This is crucial for the successful applications. First, the convergence property of the case without structure changing is determined. Secondly, we investigate the convergence in the structure changing phase, the convergence of the constructive training algorithm will be considered. Furthermore, through this analysis one obtains a better understanding on the weights adjusting algorithms.

### 3.1. Structure non-changing phase

The convergence properties of the case without structure changing will be considered in this section. This analysis can be determinate by the convergence theorem.

The nonlinear continuous-time dynamical systems that we consider in this paper are modeled by the following vector differential equation:

$$\dot{\hat{y}}(t) = f(\hat{y}(t), \mathbf{x}(t)), \qquad \hat{y}(t_0) = \hat{y}_0, \tag{14}$$

where $\hat{y}(t) \in \Re^1$, $\mathbf{x}(t) \in \Re^M$. We assume that the function $f(\cdot,\cdot)$ is unknown to us. In order to discuss the convergence convenient, we then express (14) in the following form:

$$\dot{\hat{y}}(t) = -\hat{y}(t) + \hat{g}(\hat{y}(t), \mathbf{x}(t)), \qquad \hat{y}(t_0) = \hat{y}_0, \tag{15}$$

and

$$\begin{aligned} \hat{g}(\hat{y}(t), \mathbf{x}(t)) &= \dot{\hat{y}}(t) + \hat{y}(t) \\ &= \hat{y}(t) + f(\hat{y}(t), \mathbf{x}(t)). \end{aligned} \tag{16}$$

We now analyze an RBF network with $M$ inputs and 1 output that we will use to approximate the unknown function in real-time, equivalently, the unknown dynamical system (14). Let

$$y(t) = g(\hat{y}(t), \mathbf{x}(t)) = \mathbf{W}\xi(\hat{y}(t), \mathbf{x}(t)), \tag{17}$$

where $\mathbf{W} \in \Re^{1\times M}$ is a weight matrix and $\xi(\hat{y}(t), \mathbf{x}(t)) \in \Re^{M\times 1}$.

The RBF network has the following form:

$$\dot{y}(t) = -y(t) + g(y(t), \mathbf{x}(t)). \tag{18}$$

Note that the approximation error is defined as formula (6). Combining (14) and (18), we obtain the approximation error dynamics

$$\begin{aligned} \dot{e}(t) &= \dot{y}(t) - \dot{\hat{y}}(t) \\ &= -y(t) + g(\hat{y}(t), \mathbf{x}(t)) + \hat{y}(t) - \hat{g}(\hat{y}(t), \mathbf{x}(t)) \\ &= -(y(t) - \hat{y}(t)) + g(\hat{y}(t), \mathbf{x}(t)) - \hat{g}(\hat{y}(t), \mathbf{x}(t)) \\ &= -e(t) + g(\hat{y}(t), \mathbf{x}(t)) - \hat{g}(\hat{y}(t), \mathbf{x}(t)). \end{aligned} \tag{19}$$

For the operation of the RBF neural network, If there exists $\mathbf{W}^*$ such that

$$\hat{g}(\hat{y}(t), \mathbf{x}(t)) = \mathbf{W}^*\xi(\hat{y}(t), \mathbf{x}(t)), \tag{20}$$

where each element of $\mathbf{W}^*$ is a constant. We define the adaptation parameter error as $\Xi = \mathbf{W} - \mathbf{W}^*$.

Because $\Xi \in \Re^{M\times 1}$, its Frobenius norm satisfies

$$\|\Xi\|_F^2 = \sum_{i=1}^{M}\sum_{i=1}^{1} \vartheta_{ij}^2 = \mathrm{trace}(\Xi\Xi^T), \tag{21}$$

and we have

$$\frac{\mathrm{d}}{\mathrm{d}t}(\mathrm{trace}(\Xi\Xi^T)) = 2\sum_{I=1}^{M}\sum_{J=1}^{K} \vartheta_{ij}\dot{\vartheta}_{ij} = 2\mathrm{trace}(\Xi\dot{\Xi}^T), \tag{22}$$

where $\dot{\Xi} = \mathrm{d}\Xi/\mathrm{d}t = \dot{\mathbf{W}}$. The weight adaptation strategy is defined as

$$\dot{\mathbf{W}}^{\mathrm{T}} = \eta\xi(y(t), \mathbf{x}(t))e, \tag{23}$$

where, $\eta > 0$ is the learning rate for connecting weights.

To prove the convergence of the proposed algorithm, we consider the following Lyapunov function candidate

$$V = V(e, \Xi) = \frac{1}{2}\left(e^2 + \frac{1}{\eta}\mathrm{trace}(\Xi\Xi^T)\right). \tag{24}$$

The error dynamics model (6) can be represented as

$$\begin{aligned} \dot{e}(t) &= -e(t) + g(\hat{y}(t), \mathbf{x}(t)) - \hat{g}(\hat{y}(t), \mathbf{x}(t)) \\ &= -e(t) + \mathbf{W}\xi(\hat{y}(t), \mathbf{x}(t)) - \mathbf{W}^*\xi(\hat{y}(t), \mathbf{x}(t)) \\ &= -e(t) + (\mathbf{W} - \mathbf{W}^*)\xi(\hat{y}(t), \mathbf{x}(t)) \\ &= -e(t) - \Xi\xi(\hat{y}(t), \mathbf{x}(t)). \end{aligned} \tag{25}$$

Then, we evaluate the time derivative of $V$

$$\begin{aligned} \dot{V}(e, \Xi) &= e\dot{e} + \frac{1}{\eta}\mathrm{trace}(\Xi\dot{\Xi}^T) \\ &= e(-e - \Xi\xi(\hat{y}(t), \mathbf{x}(t))) + \frac{1}{\eta}\mathrm{trace}(\Xi\dot{\mathbf{W}}^{\mathrm{T}}) \\ &= -e^2 - e\Xi\xi(\hat{y}(t), \mathbf{x}(t)) + \frac{1}{\eta}\mathrm{trace}(\Xi\dot{\mathbf{W}}^{\mathrm{T}}). \end{aligned} \tag{26}$$

Using the definition and the property of the trace operator, we obtain

$$\begin{aligned} e\Xi\xi(\hat{y}(t), \mathbf{x}(t)) &= \mathrm{trace}(e\Xi\xi(\hat{y}(t), \mathbf{x}(t))) \\ &= \mathrm{trace}(\Xi\xi(\hat{y}(t), \mathbf{x}(t))e). \end{aligned} \tag{27}$$

Combining the weight adaptation strategy as formula (23), the formula (26) will be

$$\begin{aligned} \dot{V}(e, \Xi) &= -e^2 - \mathrm{trace}(\Xi\xi(\hat{y}(t), \mathbf{x}(t))e) + \frac{1}{\eta}\mathrm{trace}(\Xi\dot{\mathbf{W}}^{\mathrm{T}}) \\ &= -e^2 - \frac{1}{\eta}\mathrm{trace}(\Xi(\eta\xi(\hat{y}(t), \mathbf{x}(t))e - \dot{\mathbf{W}}^{\mathrm{T}})) \\ &= -e^2 - \frac{1}{\eta}\mathrm{trace}(\Xi(\eta\xi(\hat{y}(t), \mathbf{x}(t))e \\ &\quad - \Xi(\eta\xi(\hat{y}(t), \mathbf{x}(t))e))) \\ &= -e^2. \end{aligned} \tag{28}$$

Thus, $\dot{V}$ is negative semidefinite in the $(e, \Xi)$-space, so $e(t)$ and $\Xi(t)$ are bounded for $t \geq t_0$. Hence, by the Lyapunov-like lemma

$$\lim_{t\to\infty} e(t) = 0. \tag{29}$$

By now we have proved the convergence result during the structure non-changing phase.

**Remark.** The theorem is stated for the RBFNN with fixed structure. However, the theorem will be investigated for the RBFNN with varying structure, that is, for the case when the hidden neurons change with time. This is because the structure change will destroy the convergence of the RBFNN if the structure design method is not suitable.

### 3.2. Structure changing phase

This consists of two parts: (1) the neuron splitting mechanism in step (4); (2) the neuron adjusting mechanism described in step (6). It is suggested in this paper that the changes made to the structure influence the current error $e(\mathbf{x})$ for the input sample $\mathbf{x}(t)$ at time $t$ (in the following discussion $e(\mathbf{x})$ is written as $e(t)$); at time $t$ there are $K$ hidden neurons in the hidden layer. The convergence of each component is described below.
(1) The neuron splitting mechanism in step (4)
When the activity neuron (i.e. the $i$th neuron) has been divided into $N_{new}$ new hidden neurons, there are $K + N_{new-1}$ hidden neurons. The current error $e(t)$ at time $t$ will is given by

$$\begin{aligned} e_{K+N_{new-1}}(t) &= y(t) - \hat{y}(t) \\ &= \sum_{k=1}^{K+N_{new-1}} w_k\theta_k(\mathbf{x}(t)) - \hat{y}(t) \\ &= \left[\sum_{k=1}^{K} w_k\theta_k(\mathbf{x}(t)) - w_i\theta_i(\mathbf{x}(t)) \right. \\ &\quad \left. + \sum_{j=1}^{N_{new}} w_{i,j}\theta_j(\mathbf{x}(t))\right] - \hat{y}(t), \end{aligned} \tag{30}$$

where $y(t)$ is the output of the neural network and $\hat{y}(t)$ is the required output for the current input sample $\mathbf{x}(t)$ at time $t$.
Based on formulas (4) and (5), the following will be true

$$\sum_{j=1}^{N_{new}} w_{i,j}\theta_{i,j}(\mathbf{x}(t)) - w_i\theta_i(\mathbf{x}(t)) = -e_K(\mathbf{x}(t)). \tag{31}$$

Formula (30) can be rewritten as

$$\begin{aligned} e_{K+N_{new-1}}(t) &= \left[\sum_{k=1}^{K} w_k\theta_k(\mathbf{x}(t)) - w_i\theta_i(\mathbf{x}(t)) \right. \\ &\quad \left. + \sum_{j=1}^{N_{new}} w_{i,j}\theta_j(\mathbf{x}(t))\right] - \hat{y}(t) \\ &= \sum_{k=1}^{K} w_k\theta_k(\mathbf{x}(t)) - \hat{y}(t) - e_K(t) \\ &= e_K(t) - e_K(t) = 0. \end{aligned} \tag{32}$$

The error $e(t)$ will be changed to zero for the current input sample $\mathbf{x}$ at time $t$ based on the error required (ER) method. In this structure changing step, the output MSE of the RBFNN will converge rapidly for the learning process.
(2) The neuron adjusting mechanism of step (6)
Although the connection between the hidden neuron $X_i$ and the output neuron $Y$ is cut, the error $e(t)$ will be adjusted using formula (13).

$$\begin{aligned} e_{K-1}(t) &= \sum_{k=1,k\neq X'}^{K} w_k\theta_k(\mathbf{x}(t)) - \hat{y}(t) \\ &\quad - w_{X_i}\theta_{X_i}(\mathbf{x}(t)) + w'_{X'}\theta_{x'}(\mathbf{x}(t)) \\ &= \sum_{k=1,k\neq X'}^{K} w_k\theta_k(\mathbf{x}(t)) - \hat{y}(t) \\ &\quad - w_{X_i}\theta_{X_i}(\mathbf{x}(t)) + \left(w_{X'} + w_X\frac{\theta_X(\mathbf{x}(t))}{\theta_{X'}(\mathbf{x}(t))}\right)\theta_{X'}(\mathbf{x}(t)) \\ &= \sum_{k=1,k\neq X'}^{K} w_k\theta_k(\mathbf{x}(t)) - \hat{y}(t) \\ &\quad - w_{X_i}\theta_{X_i}(\mathbf{x}(t)) + w_{X'}\theta_{X'}(\mathbf{x}(t)) + w_X\theta_X(\mathbf{x}(t)) \\ &= \sum_{k=1}^{K} w_k\theta_k(\mathbf{x}(t)) - \hat{y}(t) \\ &= e_K(t). \end{aligned} \tag{33}$$

So the structure adjusting mechanism step will not influence the error $e(t)$ at time $t$, and the output MSE of the RBFNN will not change much in the following phases.
The following conditions hold:

(1) Each time new neurons are inserted, the network has good convergence; this property guarantees the convergence of the algorithm.
(2) Each time the connecting relations are cut, the convergence of the proposed algorithm will not be influenced.

As can be seen in the three phases presented above, the convergence of the proposed FS-RBFNN can be maintained or speeded up.

## 4. Experimental studies

The performance of the FS-RBFNN was verified by applying it to a nonlinear dynamic system: predicting water quality in the wastewater treatment process. The performance of the algorithm was evaluated by comparing the results with other similar self-organizing RBFNNs. All the simulations are programmed with Matlab version 7.01, and were run on a Pentium 4 with a clock speed of 2.6 GHz and 1 GB of RAM, under a Microsoft Windows XP environment.

In the following simulations the learning parameters for all the algorithms are set independently so that each network solution would, on average, obtain optimal performance. By the practical test, the learning parameters used for FS-RBFNN in all the following simulations were $Af_o = 0.2, \alpha_j = 1.02, \beta_j = 0.02$ ($j = 1, 2, \ldots, N_{new}$), $N_{new} = 2, r_1 = 0.4, r_2 = 0.6, m_0 = 0.02, \rho = 100$. We have chosen these parameters after some preliminary runs. They were not meant to be optimal. Moreover, the performance of the network is measured using the root-mean-square-error (RMSE) function which is defined as

$$\mathrm{RMSE}(t) = \sqrt{\frac{1}{2T}\sum_{t=1}^{T}(y(\mathbf{x}(t)) - \hat{y}(\mathbf{x}(t)))^2}, \tag{34}$$

where $T$ is the number of data pairs, $y(\mathbf{x}(t))$ and $\hat{y}(\mathbf{x}(t))$ are the $t$th calculated output and the desired output respectively.

### 4.1. Identification of a nonlinear dynamic system

Consider the following dynamic plant with long input delays:

$$\begin{aligned} y_p(t+1) &= 0.72y_p(t) + 0.025y_p(t-1)u(t-1) \\ &\quad + 0.01u^2(t-2) + 0.2u(t-3). \end{aligned} \tag{35}$$

This plant is the same as that used in (Lin & Chen, 2005). In this paper, there are only two input values, $y_p(t)$ and $u(t)$ which were fed into the FS-RBFNN to determine the output $y_p(t)$. The training inputs were sequenced uniformly over the interval $[-2, 2]$ for about half the training time whilst a single sinusoid signal of $1.05 \times \sin(t/45)$ was used for the remaining training time. The size of the training and testing sample was 1000. The check input signal, $u(t)$, was used to determine the identification results using the following equation:

$$u(t) = \begin{cases} \sin(\pi t/25), & 0 < t < 250, \\ 1.0, & 250 \le t < 500, \\ -1.0, & 500 \le t < 750, \\ 0.3\sin(\pi t/25) + 0.1(\pi t/32) & \\ \quad + 0.6(\pi t/10), & 750 \le t < 1000. \end{cases} \tag{36}$$

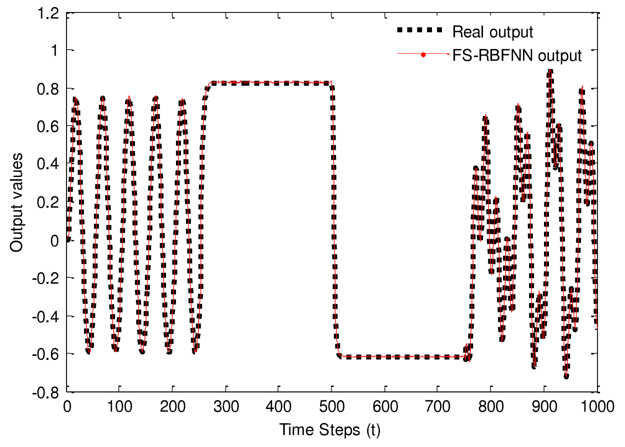


**Fig. 3.** The output values of the plant and the FS-RBFNN.

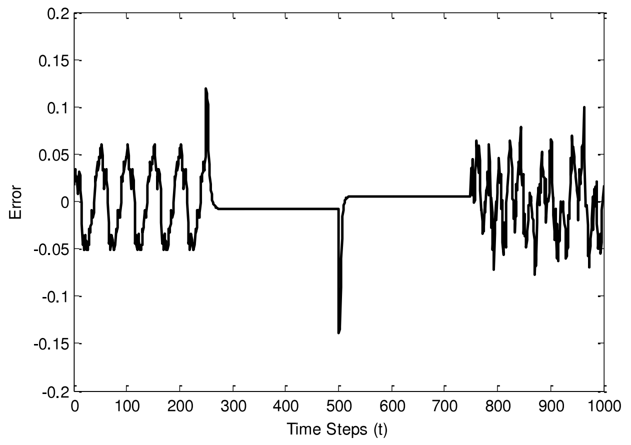


**Fig. 4.** The error between the plant output and the FS-RBFNN output.

In order to evaluate the performance of the FS-RBFNN, the results were compared with two other algorithms: the minimal resource-allocating network (MRAN) (Li et al., 1997) and the growing and pruning algorithm for RBF (GGAP-RBF) (Huang et al., 2005).

The initial architecture of the RBFNN consisted of two input variables (two nodes), one output variable (one node), and a hidden layer with 2 initial neurons. The stable MSE error was 0.01. Fig. 3 shows the outputs of the plant and the FS-RBFNN model. Fig. 4 illustrates the error between the desired output and the FS-RBFNN output. The number of hidden neurons in the training process is shown in Fig. 5. A number of parameters were selected in order to measure the performance over 20 independent trials: the number of preserved neurons in the hidden layer, the testing MSE value, the preserved number of hidden neurons, the CPU run time and the RMSE value used in testing. Table 1 shows the results with different initial hidden neurons in detail.

The results clearly show that the proposed FS-RBFNN can modify the architecture of the RBFNN automatically. Whatever the initial structures are, the final structure of the FS-RBFNN is more compact than the final structure obtained using the other algorithms. The mean testing CPU time (which is used to calculate the time by identifying the testing samples following the trained neural networks), the mean testing MSE and the mean testing RMSE are the smallest of all the values obtained. This example shows that the self-organizing method proposed in this paper can improve the ability of the final RBFNN.

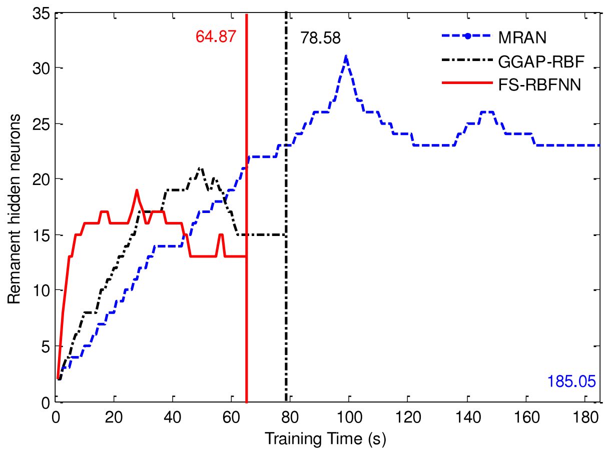


**Fig. 5.** The remanent hidden neurons in the training process.

### 4.2. Real-life date classification

In our second set of experiments, we apply the FS-RBFNN to two benchmark datasets provided by the UCI Repository of Machine Learning, namely, Glass and Iris (Asuncion & Newman, 2007).

The design was conducted using a stratified holdout procedure with 50 runs, where approximately 50% of the patterns were randomly selected for the training set and the remaining 50% for the test set. Table 2 shows the features for each data set.

The Table 2 shows the total number of instances in each data set, the number of instances in training and testing, the number of input variables, the number of classes (outputs). The results are compared with the Adaptive Merging and Growing Algorithm (AMGA) (Islam, Sattar, Amin, Yao, & Murase, 2009) and the sensitivity-based RBF (SenV-RBF) (Shi et al., 2005). The experiments were performed with the same data sets and training and testing partitions for all algorithms.

The initial number of hidden nodes was 2 (Case B.1) and 50 (Case B.2). The maximum training steps are $10^4$, and the expected MSE value is 0.001. The results reported in this paper are averaged over 50 trials of simulations for each case. The training and test sets are randomly regenerated before each trial of simulation. The results are shown in Table 3 which compares the performance to other self-organizing algorithms. The averaged values of the following parameters were used to measure the performance: the mean accumulated CPU time the training process took, the number of preserved nodes in the hidden layer, and the testing recognition rate (The testing recognition rate refers to the percentage of right classifications produced by the self-organizing neural networks on the testing set). Table 3 shows a detailed comparison of the results with two other self-organizing algorithms. By analyzing Tables 2 and 3, the following specific comments can be made.

(1) The FS-RBFNN can self-organize the network structure by the research objects. The final structure of FS-RBFNN is compact and steady over the 50 trials of simulations.
(2) The FS-RBFNN obtains the best accuracy values for Iris in Case B.1 and Case B.2. Besides, The FS-RBFNN gets better accuracy values for Glass in Case B.1 and Case B.2. But it yields a slight poor performance than SenV-RBF for Glass in the both cases.
(3) The mean training CPU time is much shorter than the other algorithms for Glass and Iris in Case B.2. The training time is little longer than the SenV-RBF for the datasets classification in Case B.1 due to the network structure adjusting, as shown in Table 3.

From the above experimental results on four classification problems, it can be seen that, for the Iris datasets, the recognition

**Table 1**
The performance comparison of different algorithms for the plant averaged in 20 independent runs.

| Initial hidden neurons | Algorithm | Hidden neurons | Mean CPU time (s) | | Testing MSE value | Testing RMSE value |
|---|---|---|---|---|---|---|
| | | | Training | Testing | | |
| 2 | MRAN | $23 \pm 4$ | 185.05 | 0.0076 | 0.0677 | 0.2602 |
| | GGAP-RBF | $15 \pm 3$ | 78.58 | 0.0063 | 0.0481 | 0.2193 |
| | **FS-RBFNN** | $\mathbf{13 \pm 3}$ | **64.87** | **0.0055** | **0.0268** | **0.1637** |
| 20 | MRAN | $21 \pm 2$ | 175.41 | 0.0068 | 0.0631 | 0.2512 |
| | GGAP-RBF | $14 \pm 1$ | 67.75 | 0.0058 | 0.0497 | 0.2229 |
| | **FS-RBFNN** | $\mathbf{13 \pm 1}$ | **58.53** | **0.0054** | **0.0263** | **0.1622** |
| 100 | MRAN | $24 \pm 5$ | 818.25 | 0.0089 | 0.0751 | 0.2740 |
| | GGAP-RBF | $15 \pm 3$ | 302.12 | 0.0065 | 0.0451 | 0.2124 |
| | **FS-RBFNN** | $\mathbf{14 \pm 2}$ | **141.23** | **0.0055** | **0.0223** | **0.1493** |

**Table 2**
Characteristics of UCI benchmarks and a qualitative analytical chemistry data set.

| Datasets | Patterns | Training patterns | Testing patterns | Input variables | Classes |
|---|---|---|---|---|---|
| Glass | 214 | 107 | 107 | 9 | 6 |
| Iris | 150 | 75 | 75 | 4 | 3 |

**Table 3**
Performance comparison of the different algorithms (averaged over 50 independent runs).

| Datasets | Algorithms | Case B.1: with 2 initial hidden neurons | | | Case B.2: with 50 initial hidden neurons | | |
|---|---|---|---|---|---|---|---|
| | | Mean training time (s) | Hidden neurons | Recognition rate (%) | Mean training time (s) | Hidden neurons | Recognition rate (%) |
| Glass | FS-RBFNN | 18.5 | $8 \pm 1$ | $\mathbf{69.71 \pm 2.02}$ | 26.8 | $8 \pm 1$ | $\mathbf{70.21 \pm 2.11}$ |
| | AMGA | 29.9* | $5 \pm 1$* | $66.84 \pm 3.24$* | 41.9* | $6 \pm 1$* | $67.88 \pm 2.68$* |
| | SenV-RBF | 16.3* | $8 \pm 1$* | $73.40 \pm 1.93$* | 255.4 | $9 \pm 1$* | $73.28 \pm 2.01$ |
| Iris | FS-RBFNN | 11.0 | 5 | $\mathbf{97.35 \pm 0.47}$ | 12.7 | $6 \pm 1$ | $\mathbf{97.97 \pm 0.33}$ |
| | AMGA | 9.9* | 3* | $96.72 \pm 0.71$* | 13.6* | 3* | $97.29 \pm 0.54$* |
| | SenV-RBF | 9.9* | $6 \pm 2$* | $97.30 \pm 0.87$* | 74.1 | $7 \pm 1$* | $97.12 \pm 0.74$ |

* The results are same as the original papers.

accuracy of the FS-RBFNN is better than the ones of the other self-organizing algorithms. The training time cost by the FS-RBFNN is also much shorter than the other self-organizing algorithms in Case B.2. However, for the Glass datasets, the recognition accuracy of the FS-RBFNN is slightly worse than the SenV-RBF. And the CPU time of the FS-RBFNN to train is slightly longer in Case B.1. On the basis of the analysis, the proposed FS-RBFNN is efficient for the Real-life date classification.

### *4.3. Predicting water quality in the wastewater treatment system*

In recent decades, wastewater has become one of the major environmental concerns facing national and local governments worldwide; treating wastewater at source has therefore become of critical environmental importance. In order to minimize microbial risk and optimize the treatment operation, many variables must be controlled. Biochemical oxygen demand (BOD) and chemical oxygen demand (COD), pH and nutrient levels are among the most critical. Although wastewater quality parameters can be measured by laboratory analysis, a significant time delay which may the range from a matter of minutes to a few days is usually unavoidable. This lack of suitable real-time process variable information limits the effective operation of effluent quality. Therefore, a water quality prediction model is essential to support water quality parameters. Before a water quality prediction model can be used in practice successfully, some key issues need to be addressed: (1) estimation of pollutant load; (2) estimation of model parameters; (3) assessment of the uncertainties in the development of the model and the application. Since an approach based on a neural network does not make any assumptions about the functional relationship between the dependent and independent variables, it is suitable for capturing functional relationships between bacterial levels and other variables.

In this section, the main objective is to develop a water quality prediction model that provides accurate predictions of the BOD in the wastewater treatment process using the proposed FS-RBFNN. The FS-RBFNN can be designed to update its input–output performance, resulting in continuous, online, self-correcting models. In the experiment, the most important influent water quality parameters were selected: Chemical Oxygen Demand (COD), Mixed Liquor Suspended Solid (MLSS), pH, Oil and $NH_3$–N as the input research variables, where COD measures the total amount of oxygen that can be combined with the chemical compounds in the water. MLSS is a measure of the concentration of dry solids in mg/L in the mixed liquor in an aeration tank. The PH measures the acidity or alkalinity of the influent water, Oil is the content of oil contamination, and $NH_3$–N delegates the content of nutritious contamination in the influent water. The network's output vector is the BOD of the water after passing through the treatment process. The data used in the experiment was collected from a wastewater treatment factory in Beijing, China.

The model used the COD, SS, pH, oil and $NH_3$–N data as inputs to estimate the settled sewage BOD. The input–output water quality data from a sewage treatment plant measured over the year 2006 was used as the data. After deleting abnormal data, 360 samples were obtained and normalized; sixty samples from October and November were used as testing data whilst the remaining 300 samples were used as training data. The water quality prediction model was obtained by training the FS-RBFNN. The error measures for this model are 3 mg/L confidence limits. The initial architecture for the FS-RBFNN is 5-4-1 (Case C.1) and 5-50-1 (Case C.2). The initial models used for the MRAN and the GAP-RBF networks are 5-4-1. The results of the FS-RBFNN model are given concurrently for the sake of comparison. The results in Case C.1 are shown Figs. 6–8. Fig. 6 shows the predicted results and the real values of BOD. Fig. 7 shows the error between the modeling results and the real values. Fig. 8 shows the number of hidden neurons that

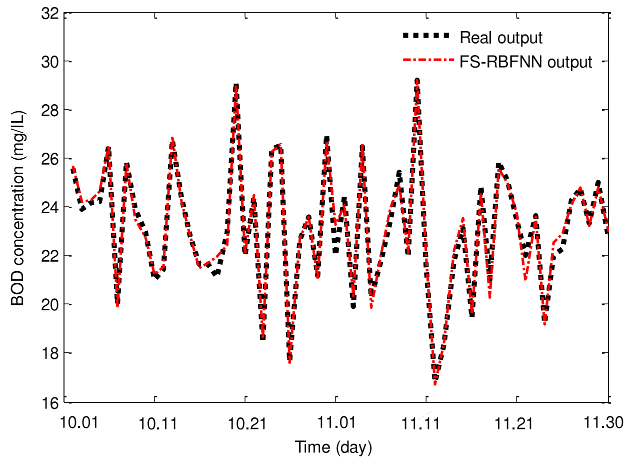


**Fig. 6.** The predicting results of BOD based on the FS-RBFNN (Case C.1).

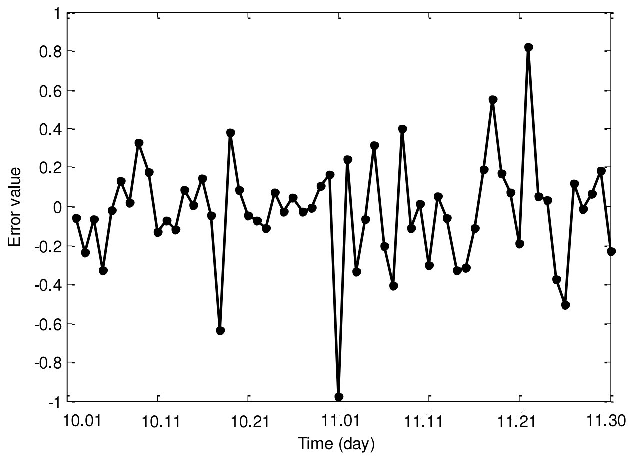


**Fig. 7.** The error between the predicting results and the real output (Case C.1).

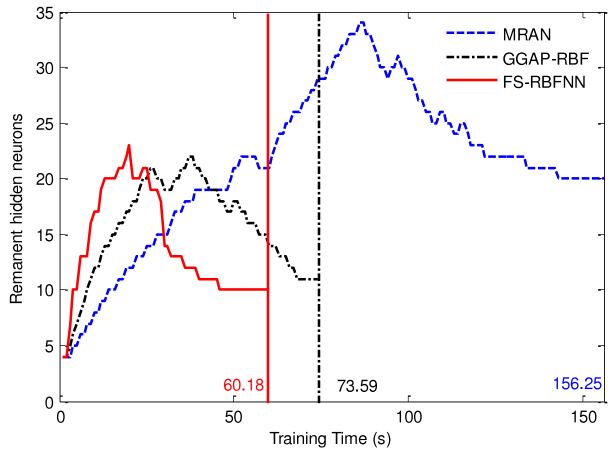


**Fig. 8.** The remanent hidden neurons in the training process (Case C.1).

remain after the training process. Table 4 shows a compares the performance between the algorithms.

Figs. 6 and 7 show that the proposed FS-RBFNN is a suitable and effective method for predicting the BOD value. The experimental results show that the proposed FS-RBFNN optimizes the structure of the RBFNN more efficiently than other comparable methods.

Table 4 shows a detailed comparison of the results with the other algorithms. By analyzing Table 4, the algorithm has the following advantages compared to other algorithms.

(1) The FS-RBFNN can self-organize the network structure in this example. The final structure of FS-RBFNN is compact and steady over the two initial structures.
(2) The FS-RBFNN obtains the best testing MSE and RMSE values for the BOD prediction in Case C.1 and Case C.2. Besides, The FS-RBFNN gets the best accuracy both for the minima and maxima values.
(3) The mean training CPU time is much shorter than the other algorithms for both cases. The training time used in Case C.2 is little longer than the GGAP-RBF used in Case C.1 due to the initial structure, as shown in Table 4.

From the above experimental results on the BOD prediction, the proposed FS-RBFNN is a suitable and effective method for predicting the BOD value. It can also be seen that, in this example, the FS-RBFNN is more accurate and has a faster training and test speed. The model is relatively straightforward to implement online and can be used to make real-time predictions of the BOD. The results demonstrate that the BOD trends in the settled sewage at the wastewater treatment could be predicted with acceptable accuracy using COD, SS, pH, Oil and $NH_3$–N data as model inputs. It is concluded that this is a significant feature of the FS-RBFNN model since the BOD value is the parameter most commonly used to determine requirements to achieve water quality standards.

### 4.4. Discussion

This section briefly explains some of the limitations of the FS-RBFNN algorithm. There is one major area that should be improved.

The major problem is the values of the parameters $Af_o$, $N_{new}$, $\alpha_j$, $\beta_j$, $r_i$, $m_0$, and $\rho$. For the practical test, the learning parameters used for FS-RBFNN in this paper were $Af_o = 0.2$, $\alpha_j = 1.02$, $\beta_j = 0.02 (j = 1, 2, \ldots, N_{new})$, $N_{new} = 2$, $r_1 = 0.4$, $r_2 = 0.6$, $m_0 = 0.02$, $\rho = 100$. They were not meant to be optimal. In Table 1, the mean testing CPU time (which is used to calculate the time by identifying the testing samples following the trained neural networks), the mean testing MSE and the mean testing RMSE are the smallest of all the values obtained. However, Table 3 shows that the FS-RBFNN obtains the best accuracy values for Iris and yields a slight poor performance than SenV-RBF for Glass in both Case B.1 and Case B.2. Moreover, the FS-RBFNN obtains the best accuracy values for Iris and yields a slight poor performance than SenV-RBF for Glass in Case B.1 and Case B.2. The FS-RBFNN algorithm is demonstrated to be able to self-organize the network structure by the proposed strategies. The final structure of FS-RBFNN is the most compact comparing with the other self-organizing algorithms. However, a limitation of the proposed FS-RBFNN algorithm is that we only supply the special values of the former parameters. For some other applications, the values of the former parameters should be fine tuning in their suitable bounds.

## 5. Conclusion

In this paper, a new algorithm is proposed for creating a self-organizing RBFNN whose architecture is automatically adapted based on neuron activity and mutual information (MI). The advantages of the proposed approach are that it can simplify and accelerate the structure optimization process of the RBFNN, and can solve practical problems for predicting water quality in the wastewater treatment process. The effectiveness and performance of the method are firstly demonstrated by using an example involving the identification of a nonlinear dynamic system. The experimental results signify that the algorithm can quickly identify a nonlinear dynamic system and outperforms comparable algorithms (MRAN and GGAP-RBF). The algorithm was then applied to classify the real-life date. The results demonstrate

**Table 4**
The performance comparison of different approaches averaged in 20 independent runs.

| Methods | Initial structure | Hidden neurons | Mean CPU time (s) | | Testing MSE value | Testing RMSE value | Accuracy (%) | |
|---|---|---|---|---|---|---|---|---|
| | | | Training | Testing | | | Min | Max |
| Model based (Brydon & Frodsham, 2001) | - | - | - | 0.0093 | 2.0802 | 1.4423 | 82.08 | 93.34 |
| Bayesian Approach (Liu, Yang, Hu, & Guo, 2008) | - | - | - | 0.0078 | 0.8427 | 0.9180 | 90.15 | 96.32 |
| Fixed MLP (Chandramouli, Brion, Neelakantan, & Lingireddy, 2007) | 5-50-1 | 50 | 355.87 | 0.0119 | 0.8191 | 0.9050 | 93.48 | 96.52 |
| MRAN | 5-4-1 | 20±3 | 156.25 | 0.0067 | 0.6980 | 0.8355 | 96.08 | 97.78 |
| GGAP-RBF | 5-4-1 | 11 ± 1 | 73.59 | 0.0049 | 0.3325 | 0.5766 | 97.01 | 98.31 |
| **FS-RBFNN** | **5-4-1** | **10 ± 1** | **60.18** | **0.0047** | **0.3038** | **0.5512** | 97.19 | 98.81 |
| **FS-RBFNN** | **5-50-1** | **11 ± 1** | **83.24** | **0.0049** | **0.3015** | **0.5491** | 97.05 | 98.90 |

that the proposed FS-RBFNN is efficient for the real-life date classification—the recognition accuracy of the FS-RBFNN is better than the ones of the other self-organizing algorithms (AMGA and SenV-RBF) for the Iris datasets and better than the AMGA for the Glass datasets. Finally, the algorithm was then applied to predicting wastewater quality: simulation results show that the FS-RBFNN model is able to predict the output-water quality more accurately than the other self-organizing RBFNN models. The control efficiency of the wastewater treatment system can be further improved by tuning the operating parameters according to the predicted results.

## Acknowledgments

The authors would like to thank Dr. R. Dale-Jones for reading the manuscript and providing valuable comments. The authors would also like to thank the anonymous reviewers for their valuable comments and suggestions, which helped improve this paper greatly. This work was supported by the National 863 Scheme Foundation of China under Grants 2009AA04Z155 and 2007AA04Z160, National Science Foundation of China under Grants 61034008 and 60873043, Ph.D. Program Foundation from Ministry of Chinese Education under Grant 200800050004, Beijing Municipal Natural Science Foundation under Grant 4092010 and Funding Project for Academic Human Resources Development under Grant PHR(IHLB)201006103.

## References

Alexandridis, A., Sarimveis, H., & Bafas, G. (2003). A new algorithm for online structure and parameter adaptation of RBF networks. *Neural Networks*, *16*(7), 1003–1017.
Asuncion, A., & Newman, D.J. (2007). UCI Machine Learning Repository, [Online]. Available http://www.ics.uci.edu/~mlearn/MLRepository.html.
Bortman, M., & Aladjem, M. (2009). A growing and pruning method for radial basis function networks. *IEEE Transactions on Neural Networks*, *20*(6), 1039–1045.
Brydon, D. A., & Frodsham, D. A. (2001). A model-based approach to predicting BOD5 in settled sewage. *Water Science and Technology*, *44*(2–3), 9–15.
Chandramouli, V., Brion, G., Neelakantan, T. R., & Lingireddy, S. (2007). Backfilling missing microbial concentrations in a riverine database using artificial neural networks. *Water Reasearch*, *41*(1), 217–227.
Feng, H. M. (2006). Self-generation RBFNs using evolutional PSO learning. *Neurocomputing*, *70*(1–3), 241–251.
Ferrari, S., Bellocchio, F., Piuri, V., & Borghese, N. A. (2010). A hierarchical RBF online learning algorithm for real-time 3-D scanner. *IEEE Transactions on Neural Networks*, *21*(2), 275–285.
Ge, Z. F., & Frick, W. E. (2009). Time–frequency analysis of beach bacteria variations and its implication for recreational water quality modeling. *Environmental Science & Technology*, *43*(4), 1128–1133.
Gonzalez, J., Rojas, I., Ortega, J., Pomares, H., Fernandez, F. J., & Diaz, A. F. (2003). Multi-objective evolutionary optimization of the size, shape, and position parameters of radial basis function networks for function approximation. *IEEE Transactions on Neural Networks*, *14*(6), 1478–1495.
Huang, G. B., Saratchandran, P., & Sundararajan, N. (2004). An efficient sequential learning algorithm for growing and pruning RBF (GAP-RBF) networks. *IEEE Transactions on Systems, Man, and Cybernetics – Part B: Cybernetics*, *34*(6), 2284–2292.
Huang, G. B., Saratchandran, P., & Sundararajan, N. (2005). A generalized growing and pruning RBF (GGAP-RBF) neural network for function approximation. *IEEE Transactions on Neural Networks*, *16*(1), 57–67.
Islam, M. M., Sattar, M. A., Amin, M. F., Yao, X., & Murase, K. (2009). A new adaptive merging and growing algorithm for designing artificial neural networks. *IEEE Transactions on Systems, Man, and Cybernetics – Part B: Cybernetics*, *39*(3), 705–722.
Krivov, S., Ulanowicz, R. E., & Dahiya, A. (2003). Quantitative measures of organization for multiagent systems. *Biosystems*, *69*(1), 39–54.
Lee, C. M., & Ko, C. N. (2009). Time series prediction using RBF neural networks with a nonlinear time-varying evolution PSO algorithm. *Neurocomputing*, *73*(1–3), 449–460.
Platt, J. (1991). A resource-allocating network for function interpolation. *Neural Computation*, *3*(2), 213–225.
Li, Y. W., Sundararajan, N., & Saratchandran, P. (1997). A sequential learning scheme for function approximation using minimal radial basis function (RBF) neural networks. *Neural Computation*, *9*(2), 461–478.
Li, Y. W., Sundararajan, N., & Saratchandran, P. (1998). Performance evaluation of a sequential minimal radial basis function (RBF) neural network learning algorithm. *IEEE Transactions on Neural Networks*, *9*(2), 308–318.
Lian, J. M., Lee, Y. G., Sudhoff, S. D., & Stanislaw, H. Z. (2008). Self-organizing radial basis function network for real-time approximation of continuous-time dynamical systems. *IEEE Transactions on Neural Networks*, *19*(3), 460–474.
Lin, C. J., & Chen, C. H. (2005). Identification and prediction using recurrent compensatory neuro-fuzzy systems. *Fuzzy Sets and Systems*, *150*(2), 307–330.
Liu, Y., Yang, P. J., Hu, C., & Guo, H. C. (2008). Water quality modeling for load reduction under uncertainty: a Bayesian approach. *Water Reasearch*, *42*(13), 3305–3314.
Neves, G., Cooke, S. F., & Bliss, T. V. P. (2008). Synaptic plasticity, memory and the hippocampus: a neural network approach to causality. *Nature Reviews Neuroscience*, *9*(1), 65–75.
Panchapakesan, C., Palaniswami, M., Ralph, D., & Manzie, C. (2002). Effects of moving the centers in an RBF network. *IEEE Transactions on Neural Networks*, *13*(6), 1299–1307.
Qiao, J. F., & Han, H. G. (2010). A repair algorithm for RBF neural network and its application to chemical oxygen demand modeling. *International Journal of Neural Systems*, *20*(1), 63–74.
Samanwoy, G. D., & Hojjat, A. (2009). A new supervised learning algorithm for multiple spiking neural networks with application in epilepsy and seizure detection. *Neural Networks*, *22*(10), 1419–1431.
Shi, D., Yeung, D. S., & Gao, J. (2005). Sensitivity analysis applied to the construction of radial basis function networks. *Neural Networks*, *18*(7), 951–957.
Wang, S. W., & Yu, D. L. (2008). Adaptive RBF network for parameter estimation and stable air–fuel ratio control. *Neural Networks*, *21*(1), 102–112.
Wu, S., & Chow, T. W. S. (2007). Self-organizing and self-evolving neurons: a new neural network for optimization. *IEEE Transactions on Neural Networks*, *18*(2), 385–396.