processes

MDPI

Article

A Novel Radial Basis Function Neural Network with High Generalization Performance for Nonlinear Process Modelling

Yanxia Yang [1,2,*], Pu Wang [1,2] and Xuejin Gao [1,2,*]

[1] Faculty of Information Technology, Beijing University of Technology, Beijing 100124, China; wangpu@bjut.edu.cn
[2] Engineering Research Center of Digital Community Ministry of Education, Beijing 100124, China
* Correspondence: yangyx@emails.bjut.edu.cn (Y.Y.); gaoxuejin@bjut.edu.cn (X.G.)

Abstract: A radial basis function neural network (RBFNN), with a strong function approximation ability, was proven to be an effective tool for nonlinear process modeling. However, in many instances, the sample set is limited and the model evaluation error is fixed, which makes it very difficult to construct an optimal network structure to ensure the generalization ability of the established nonlinear process model. To solve this problem, a novel RBFNN with a high generation performance (RBFNN-GP), is proposed in this paper. The proposed RBFNN-GP consists of three contributions. First, a local generalization error bound, introducing the sample mean and variance, is developed to acquire a small error bound to reduce the range of error. Second, the self-organizing structure method, based on a generalization error bound and network sensitivity, is established to obtain a suitable number of neurons to improve the generalization ability. Third, the convergence of this proposed RBFNN-GP is proved theoretically in the case of structure fixation and structure adjustment. Finally, the performance of the proposed RBFNN-GP is compared with some popular algorithms, using two numerical simulations and a practical application. The comparison results verified the effectiveness of RBFNN-GP.

Keywords: radial basis function neural network (RBFNN); generation performance; local generalization error bound; self-organizing structure method; convergence analysis

Citation: Yang, Y.; Wang, P.; Gao, X. A Novel Radial Basis Function Neural Network with High Generalization Performance for Nonlinear Process Modelling. *Processes* **2022**, *10*, 140. https://doi.org/10.3390/pr10010140

Academic Editors: Jie Zhang and Meihong Wang

Received: 16 December 2021
Accepted: 3 January 2022
Published: 10 January 2022

1. Introduction

In recent years, with the continuous development of artificial intelligence and intelligent algorithms, data-driven methods have been widely used as an effective modeling method because they do not require complex mathematical models and high maintenance costs. Among them, the radial basis function neural network (RBFNN) is widely used due to its simple structure and strong nonlinear function approximation ability, especially in the fields of pattern classification, industrial control, nonlinear system modeling and so on [1–4]. However, there are still some problems to be solved in practice, for example, how to extend the network performance from limited training set to invisible data set, that is, how to design RBFNN with a good generalization ability [5,6]. The generalization performance of RBFNN is usually measured by generalization error, which mainly includes the approximation error caused by the insufficient representation ability of network and estimation errors caused by a limited number of samples. In order to make the RBFNN learnable, the generalization error should be zero as the data tends to infinity.

Due to the limitation of sample data, the network model will produce an estimation error. In order to make the total generalization error close to zero, the number of parameters and samples should tend to infinity to ensure the learnability of the model. Among them, it is worth mentioning that references [5,7–10] deal with the problem of estimation error according to different assumptions. On this basis, the sample complexity of finite networks is studied to demonstrate that, when the data tends to infinity, the estimation

error tends to zero. In addition, due to the limited number of samples, even if the optimal parameter setting is obtained, it will produce functions far from the target, resulting in errors and a poor generalization performance [9]. To solve this problem, Barron et al. [10] introduced the concept of an approximation and estimation bound of artificial neural network, pointing out that, for a kind of common artificial neural network, the integral square error between the estimation network model and the objective function is bounded, and discussing the comprehensive influence of approximation error and estimation error as the objective function on the network accuracy. In addition, Yeung et al. [11] developed a new RBFNN local generalization error model to identify the classifier. By predefining the neighborhood of training samples in the local generalization error model, the upper bound of generalization error of invisible samples was derived. Sarraf [12] proposed a tight upper bound of the generalization error under the assumption of twice continuously differentiable, which was composed of the estimation error under the sample space mean and the expected sensitivity of the error to the input change, and showed how the given upper bound could be used to analyze the generalization error of a feedforward neural network. Although the above methods achieved good results through the generalization error bound based on sensitivity, they still faced challenges due to the computational complexity of partial derivatives, and the generalization error should not only be a function of the number of parameters; it is also important to find a better structure. In addition, Wu et.al. proposed a self-adaptive structural optimal algorithm-based fuzzy neural network (SASOA-FNN) in [13]. This network can improve the generalization ability of the network by minimizing the structural risk model with the number of samples. Terada et al. [14] derived the fast generalization error bound of deep learning under the framework developed in [15]. In the derivation process, they only focused on the minimization of empirical risk and eliminated the scale invariance assumption of activation function. The common feature of the above references is that they are based on risk minimization, and accelerate the convergence speed of the network, while ignoring the influence of the properties of different networks on the generalization error. For example, RBFNN is essentially a local learning method. Each hidden neuron captures the local information of a specific region in the input space by the center and width of its Gaussian activation function [16]. However, the training samples far away from the center of hidden neurons have no effect on the learning of hidden neurons. Therefore, for this local learning method, finding the optimal compromise between model accuracy and generalization error is an effective way to improve the generalization ability of the network.

Different from the estimation error caused by the insufficient samples mentioned above, the approximation error of the network is greatly affected by the network structure. Thus, how to obtain a suitable network structure has always been a hot topic. For instance, Zhou et al. [17] proposed a self-organizing fuzzy neural network with hierarchical pruning scheme (SOFNN-HPS). In SOFNN-HPS, the adaptive ability and robustness of the prediction model were improved through the effective combination of a hierarchical pruning strategy and adaptive allocation strategy. Finally, the accurate prediction of ammonia nitrogen, the key variable in the wastewater treatment process, is realized. To predict the outlet ferrous ion concentration on-line, Xie et al. [18] developed a self-adjusting structure radial basis function neural network (SAS-RBFNN). This algorithm uses the supervised clustering algorithm to initialize the RBFNN, and combines or segments the hidden neurons according to the clustering distance to realize the structural self-organization of RBFNN. In addition, Huang et al. [19] proposed a growing and pruning RBF (GAP-RBF) method based on the significance of a neuron. For the GAP-RBF, the number of neurons can be self-designed to realize a compact RBFNN by linking the significance of neurons and a desired accuracy. On this basis, an improved GAP-RBF algorithm (IGAP-RBF) for any arbitrary distribution of input training data was proposed in reference [20]. This algorithm only adjusts the parameters of the nearest neuron to reduce the computational complexity while ensuring the learning performance. A common feature of GAP-RBF and IGAP-RBF is that the self-organizing strategy is based on the contribution of hidden

neurons, and the training samples need to be known in advance. In reference [21], an adaptive gradient multi-objective particle swarm optimization algorithm was proposed to predict the biochemical oxygen demand, a key water quality parameter in the wastewater treatment process. This method adopts a multi-objective gradient method and adaptive flight parameter mechanism, which not only greatly reduces computational complexity, but also improves generalization performance; other structure self-organization methods are outlined in [22–25]. The advantage of the above algorithms is that they can adjust the network parameters while adjusting the network structure, and the disadvantage is that different parameter adjustment methods make the learning speed of the network different, which may affect the accuracy of the network.

Most of the existing methods focus on using self-organizing strategies to obtain appropriate structures, or using effective learning algorithms to obtain a higher accuracy. However, good training accuracy is not equal to good generalization performance. Therefore, designing an effective learning model to improve the generalization performance of RBFNN is still an urgent problem that needs to be solved. Based on the above analysis, a self-organizing RBFNN based on network sensitivity is proposed to improve the generalization performance. The main contributions of this method are as follows.

The generalization ability is quantified by network sensitivity. Then, an RBFNN-GP algorithm is constructed to improve the generalization performance:

1. The convergence of the RBFNN-GP is verified in theory, which ensures its successful application;
2. The effectiveness and feasibility of the RBFNN-GP are verified by predicting the key water quality parameters in wastewater treatment process.

The remainder of this paper is organized as follows. Section 2 briefly introduces the basic RBFNN and the local generalization error bound of the network. Then, the details of RBFNN-GP are given in Section 3. The convergence of RBFNN-GP is discussed in Section 4. Section 5 presents the experimental results of RBFNN-GP to demonstrate its advantages. The application field and future work direction of the proposed method are shown in Section 6. Finally, the conclusions are given in Section 7.

2. Materials and Methods

2.1. Radial Basis Function Neural Network (RBFNN)

In general, RBFNN consists of three layers: the input layer, the hidden layer and the output layer. A typical multiple-input, single-output RBFNN (MISO-RBFNN) is shown in Figure 1. The MISO-RBFNN is a k-m-1 network, and each neuron in the RBFNN hidden layer is constructed in the form of Gaussian function. The mathematical description of RBFNN output is as follows:

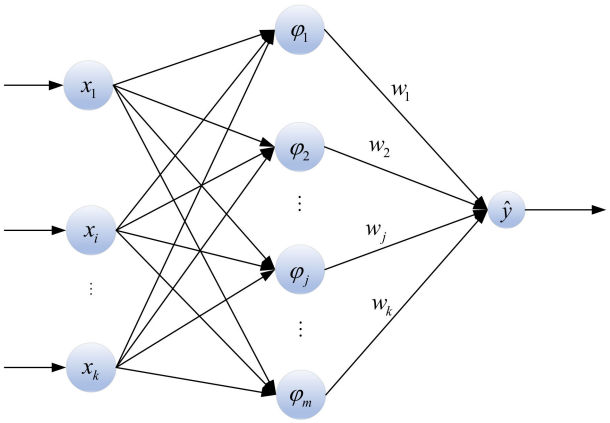


Figure 1. The structure of MISO-RBFNN.

$$\hat{y}(t) = \sum_{j=1}^{m} w_j(t)\theta_j(t), \tag{1}$$

where m is the number of hidden neurons, $w_j(t)$ is the weight between the jth hidden layer and the output layer at time t, and $\theta_j(t)$ is the output of the jth hidden layer neuron, described as:

$$\theta_j(t) = e^{-\frac{\|\mathbf{x}(t)-\mathbf{c}_j(t)\|^2}{2\sigma_j^2(t)}}, \tag{2}$$

$$\mathbf{c}_j(t) = \left[c_{1,j}(t), c_{2,j}(t), \cdots, c_{n,j}(t)\right]^T, \tag{3}$$

where $\mathbf{x}(t) = [x_1(t), x_2(t), \ldots, x_n(t)]^T$ is the input vector, $\mathbf{c}_j(t)$ is the center vector of the jth hidden neuron at time t, and n is the dimension of the input vector; $\|\mathbf{x}(t)-\mathbf{c}_j(t)\|$ is the Euclidean distance between $\mathbf{x}(t)$ and $\mathbf{c}_j(t)$, and $\sigma_j(t)$ is the width of the jth hidden neuron at time t.

2.2. Local Generalization Error Bound

The generalization error of the whole input space is defined as [11]:

$$E_{gen}(t) = \int_D [f(\mathbf{x}(t)) - F(\mathbf{x}(t))]^2 p(\mathbf{x}(t)) d\mathbf{x}(t), \tag{4}$$

where $\mathbf{x}(t)$ is the input vector in the input space and $p(\mathbf{x}(t))$ is the unknown probability density function of $\mathbf{x}(t)$. Given a training data set $D = (\mathbf{x}_a(t), F_a(t)), a = 1, 2, \cdots, N$, N is the number of pairs of input and output, namely the number of samples of training set. The empirical error of network can be defined as:

$$E_{emp}(t) = \frac{1}{N}\sum_{a=1}^{N}\left[\hat{f}(\mathbf{x}_a(t)) - F(\mathbf{x}_a(t))\right]^2, \tag{5}$$

where $\hat{f}(\mathbf{x}_a(t))$ and $F(\mathbf{x}_a(t))$represent the approximate and real mapping functions between the ath input and output in the input space, respectively. The ultimate goal of improving the generalization ability is minimize the approximation error, and the network can directly predict the unseen data.

Since RBFNN is a local method, for each sample $\mathbf{x}_a(t) \in D$, we can find a sample set:

$$\mathbf{X}_{S,\mathbf{x}_a}(t) = \{\mathbf{x}(t) | \mathbf{x}(t) = \mathbf{x}_a(t) + \Delta\mathbf{x}(t)\}, \tag{6}$$

where $\mathbf{X}_{S,\mathbf{x}_a}(t)$ defines an S-neighborhood of the training sample $\mathbf{x}_a(t)$, $\Delta\mathbf{x}(t) = [\Delta x_1(t), \cdots, \Delta x_n(t)]$ is regarded as perturbations, n denotes the number of input features, and S is a given number. The samples in $\mathbf{X}_{S,\mathbf{x}_a}(t)$ (except $\mathbf{x}_a(t)$) are regarded as unseen samples. For $0 \le S_1 \le \cdots \le S_k \le \infty$, the relationship holds$D \subseteq \mathbf{X}_{S_1}(t) \subseteq \cdots \subseteq \mathbf{X}_{S_k}(t) \subseteq I$, where I is the entire input space. By Hoeffding's inequality, we can derive the definition of local generalization error bound as follows [11,26]:

$$E_{gen,S}(t) = \sqrt{E_{emp}(t)} + \sqrt{E_{\mathbf{X}_S,\Delta y^2}(t)}, \tag{7}$$

with:

$$\begin{aligned} \Delta y(t) &= \hat{f}(\mathbf{x}_a(t)) - F(\mathbf{x}_a(t)) \\ E_{\mathbf{X}_s,\Delta y^2}(t) &= \frac{1}{N}\sum_{a=1}^{N}\int_{\mathbf{X}_s(\mathbf{x}_a(t))}\left(\Delta y^2(t)\right)\frac{1}{(2S)^a}dx(t), \end{aligned} \tag{8}$$

where $\Delta y(t)$ is the difference between the network output and the real value; the term $E_{\mathbf{X}_S,\Delta y^2}(t)$ is introduced in Section 3.1.

Remark 1. *Different from previous methods, to the best of our knowledge, this work is a new attempt to obtain looser generalization error bounds by eliminating high-order terms and reducing partial accuracy in exchange for better generalization ability.*

3. RBFNN with High Generation Performance (RBFNN-GP)

The proposed RBFNN-GP, which can improve the network generalization ability, is introduced in this section. It contains the following two parts: (1) sensitivity measurement (SM) method and (2) structural self-organization optimization (SSO) strategy.

3.1. Sensitivity Measurement (SM) Method

Sensitivity analysis provides an effective method to evaluate the impact of different inputs on output results, and it can accurately express the causal response between input changes and corresponding outputs [27,28]. Different from existing studies that focus on improving modeling accuracy or looking for indicator variables, in this study, SM is introduced to quantify the impact of network input changes on output changes, and to intuitively represent the sensitivity of network output to input changes. Thus, the network structure can be adjusted accordingly. Suppose that the inputs are independent and not identically distributed; then, from this, each input feature has its own expectation μ_{x_i} and variance, $\delta_{x_i}^2$:

$$\phi_j(t) = w_j^2(t) e^{\delta_{d_j}(t)/2\sigma_j^4(t) - \mu_{d_j}(t)/\sigma_j^2(t)}, \tag{9}$$

with:

$$\begin{cases} d_j(t) = \left\|\mathbf{x}(t) - \mathbf{c}_j(t)\right\|^2 \\ \delta_{d_j}(t) = \sum_{i=1}^{n} \mu\left[(x_i(t) - \mu_{x_i}(t))^4\right] - \left(\delta_{x_i}^2(t)\right)^2 + 4\delta_{x_i}^2(t)\left(\mu_{x_i}(t) - c_{ji}(t)\right)^2 + \\ 4\mu\left[(x_i(t) - \mu_{x_i}(t))^3\right]\left(\mu_{x_i}(t) - c_{ji}(t)\right) \end{cases} \tag{10}$$

and:

$$\begin{cases} \mu_{d_j}(t) = \sum_{i=1}^{n}\left[\delta_{x_i}^2(t) + \left(\mu_{x_i}(t) - c_{ji}(t)\right)^2\right] \\ \sigma_j(t) = \phi_j(t)\sum_{i=1}^{n}\left(\delta_{x_i}^2(t) + \left(\mu_{x_i}(t) - c_{ji}(t)\right)^2/\sigma_j^4(t)\right) \end{cases}. \tag{11}$$

In theory, as long as the input variation is finite, we do not strictly limit its data distribution. In this instance, we assume that the unseen samples **S** -neighborhood of the training samples obey the uniform distribution, and thus we obtain $\delta_{\Delta x_i}^2(t) = S^2/3$. By the law of large numbers, the sensitivity of RBFNN is:

$$\begin{aligned} &E_{\mathbf{X}_S,\Delta y^2}(t) \\ &= \frac{1}{N}\left\{\sum_{a=1}^{N}\int_{\mathbf{X}_{S,x_a}}\left[f(\mathbf{x}_a(t) + \Delta\mathbf{x}(t)) - f(\mathbf{x}_a(t))\right]^2 p(\Delta\mathbf{x}(t))d\Delta\mathbf{x}(t)\right\} \\ &\approx \sum_{j=1}^{m}\phi_j(t)\left\{\sum_{i=1}^{n}\left[\delta_{\Delta x_i}^2(t)\left(\delta_{x_i}^2(t) + \left(\mu_{x_i}(t) - c_{ji}(t)\right)^2 + 0.2\delta_{\Delta x_i}^2(t)\right)\right]/\sigma_j^4(t)\right\} \end{aligned} \tag{12}$$

with:

$$\phi_j(t) = \sigma_j^4(t)\xi_j(t). \tag{13}$$

therefore, we can obtain:

$$E_{\mathbf{X}_S,\Delta y^2}(t) \approx \frac{1}{45}S^4 n\sum_{j=1}^{m}\xi_j(t) + \frac{1}{3}S^2\sum_{j=1}^{m}\sigma_j(t). \tag{14}$$

Remark 2. *Limiting cases of $E_{gen,S}(t)$. Clearly, when $S \to \infty, \mathbf{X}_S(t) \to I$, that is $S \to 0, \mathbf{X}_S(t) \to D$. If $0 \le S_1 \le \cdots \le S_k \le S_\infty$, the relationship $D \subseteq \mathbf{X}_{S_1}(t) \subseteq \cdots \subseteq \mathbf{X}_{S_k}(t) \subseteq I$ holds. Therefore, in the case of $S \to \infty$, $E_{gen}(t) < E_{gen,S}(t)$.*

Remark 3. *Statistical performance. Compared with the regression error bound, which only uses the effective parameters and the number of training samples, the proposed RBFNN-GP algorithm has clear advantages, because it considers statistical characteristics, such as the mean and variance of the training data set.*

3.2. Structural Self-Organization Optimization (SSO) Strategy

In order to construct a RBFNN with a high generalization performance, an SSO strategy is designed based on the sensitivity measurement. This SSO strategy can adjust the structure and parameters (including center, width and weight) of RBFNN at the same time. The self-organization strategies are shown as follows:

$$\begin{cases} \mathbf{c}(t+1) = \mathbf{c}(t) - \eta\Delta\mathbf{c}(t) \\ \boldsymbol{\sigma}(t+1) = \boldsymbol{\sigma}(t) - \eta\Delta\boldsymbol{\sigma}(t) \\ \mathbf{w}(t+1) = \mathbf{w}(t) - \eta\Delta\mathbf{w}(t) \\ m(t+1) = \begin{cases} m(t)+1, \sigma_j < \lambda_1 \\ m(t), \lambda_1 < \sigma_j < \lambda_2 \\ m(t)-1, \sigma_j > \lambda_2 \end{cases} \end{cases}, \tag{15}$$

where $\sigma_j \geq 1$, $\mathbf{c}(t+1) = [\mathbf{c}_1(t+1), \mathbf{c}_2(t+1), \ldots, \mathbf{c}_m(t+1)]$ and $\boldsymbol{\sigma}(t+1) = [\sigma_1(t+1), \sigma_2(t+1), \ldots, \sigma_m(t+1)]$ are the center and width vectors of the hidden neuron at time $t+1$, $\mathbf{w}(t+1)$ is the weight of output layer at time $t+1$, $m(t)$ represents the number of hidden layer neurons at time t, and $m(t) \geq 1$, η is the learning rate, $\lambda_1 \leq -2/3S^2 n\xi_m$, ξ_m is the ratio of statistical output and width of the mth neuron (λ_1 is a dynamic threshold and the statistical residual is negative), and $\lambda_2 \geq 0$ are used to ensure the convergence performance of the network (here the value is twice that of the input dimension). It should be noted that, only when λ_1 and λ_2 acquire the equals sign at the same time, will the number of neurons remain unchanged, that is, the structure of the neural network holds. The variables of $\Delta\mathbf{c}(t)$, $\Delta\boldsymbol{\sigma}(t)$, $\Delta\mathbf{w}(t)$ present the changes of the centers, widths and weights at time t, respectively. We obtain:

$$\begin{cases} \Delta\mathbf{c}(t) = [\Delta\mathbf{c}_1(t), \Delta\mathbf{c}_2(t), \cdots, \Delta\mathbf{c}_m(t)] \\ \Delta\boldsymbol{\sigma}(t) = [\Delta\sigma_1(t), \Delta\sigma_2(t), \cdots, \Delta\sigma_m(t)] \\ \Delta\mathbf{w}(t) = [\Delta w_1(t), \Delta w_2(t), \cdots, \Delta w_m(t)] \end{cases}, \tag{16}$$

where $\Delta\mathbf{c}_m(t) = [\Delta c_{m,1}(t), \cdots, \Delta c_{mn}(t)]$ is the change of the center of the mth neuron at time t, and $\Delta\sigma_m(t)$ and $\Delta w_m(t)$ are the changes of width and weight of the mth neuron at time t, respectively. Moreover, the updates details of the parameters are:

$$\begin{cases} \Delta\mathbf{c}_j(t) = \partial E_{emp}(t)/\partial\mathbf{c}_j(t) = \left(\mathbf{x}(t) - \mathbf{c}_j(t)/\sigma_j^2\right) w_j\theta_j(1-\theta_j)e(t) \\ \Delta\sigma_j(t) = \partial E_{emp}(t)/\partial\sigma_j(t) = \left[\left(\mathbf{x}(t) - \mathbf{c}_j(t)\right)^2/\sigma_j^3\right] w_j\theta_j(1-\theta_j)e(t) \\ \Delta w_j(t) = \partial E_{emp}(t)/\partial w_j(t) = \theta_j\mathbf{e}(t) \end{cases}. \tag{17}$$

Based on the above analysis, the detailed steps of neuron growth and pruning are given below.

3.2.1. Growth Stage

If $\sigma_j < \lambda_1$, new neurons are added to the neural network to reduce the approximation error and improve the generalization performance. At this time, the number of neurons becomes, and the parameter of new neurons is:

$$\begin{cases} \mathbf{c}_{new} = \mathbf{x}(t) \\ \sigma_{new} = \frac{1}{m}\sum_{j=1}^{m}\sigma_i(t) \\ w_{new} = (y(t) - \hat{y}(t))e^{\sum_{i=1}^{n}\frac{(x_i(t) - c_{i,new}(t))^2}{2\sigma_{i,new}^2(t)}} \end{cases}, \tag{18}$$

where $\mathbf{c}_{new}$, σ_{new} and w_{new} represent the center, width and weight of the new neuron,$x_i(t)$represents the ith element in the input vector at time t, $c_{i,new}(t)$ and $\sigma_{i,new}(t)$ are the ith elements of the center and width of the new neuron at time t, respectively. After structural adjustment, the parameters are updated as:

$$\begin{cases} \mathbf{c}(t+1) = [\mathbf{c}(t); \mathbf{c}_{new}] \\ \boldsymbol{\sigma}(t+1) = [\boldsymbol{\sigma}(t); \sigma_{new}] \\ \mathbf{w}(t+1) = [\mathbf{w}(t); w_{new}] \end{cases}, \tag{19}$$

where $\mathbf{c}(t+1)$, $\boldsymbol{\sigma}(t+1)$ and $\mathbf{w}(t+1)$ are the center, width and weight of RBFNN at time $t+1$, respectively.

3.2.2. Prune Stage

If $\sigma_j > \lambda_2$, the jth neuron with the least information in the hidden layer is deleted. In this way, the network complexity is reduced under the premise of ensuring generalization ability. At this time, the number of neurons becomes $m(t-1)$, and the parameters of the new neurons are:

$$\begin{cases} \mathbf{c}_j(t+1) = 0 \\ \boldsymbol{\sigma}_j(t+1) = 0 \\ w_j(t+1) = 0 \end{cases}, \tag{20}$$

where $\mathbf{c}_j(t+1)$, $\boldsymbol{\sigma}_j(t+1)$ and $w_j(t+1)$ represent the center, width and weight of the jth neuron at time $t+1$, respectively. After structural adjustment, the parameters of the ith neuron were as follows:

$$\begin{cases} \mathbf{c}_i(t+1) = \mathbf{c}_i(t) \\ \boldsymbol{\sigma}_i(t+1) = \boldsymbol{\sigma}_i(t) \\ w_i(t+1) = w_i(t) + \frac{w_j \theta_j(t)}{\theta_i(t)} \end{cases}. \tag{21}$$

Among them, the ith neuron is the neuron closest to the Euclidean distance from the jth neuron, $\mathbf{c}_i(t+1)$, $\boldsymbol{\sigma}_i(t+1)$ and $w_i(t+1)$ are the center, width and weight of the ith neuron at time $t+1$.

Remark 4. *Because there is only one hidden layer in the model, the network structure is simple, which greatly reduces the error accumulation in the process of back propagation. Furthermore, no extra parameters are added during network training, which reduces the amount of computation. Therefore, the stability of the network is well guaranteed.*

4. Convergence Analysis

Another important problem of neural network structure is convergence analysis, which not only affects the application in practical engineering, but also reflects the generalization ability of neural network. If the neural network cannot guarantee convergence or meet convergence conditions, it is difficult to realize the successful application of the neural network. In addition, for RBFNN-GP, its convergence is not only related to the parameter optimization algorithm, but also related to structural changes. Therefore, this paper analyzes the convergence from three aspects: convergence in the stable stage, growth stage and deletion stage.

Hypothesis 1 (H1). *The center $\boldsymbol{c}$ of the hidden layer, the width σ of the hidden layer and the input–output weight $\boldsymbol{w}$ satisfy the boundedness, that is $\|\mathbf{c}\| \leq \mu_c$, $\|\sigma\| \leq \mu_\sigma$, $\|\mathbf{w}\| \leq \mu_w$, where μ_c,μ_σ,μ_w are positive real numbers.*

Hypothesis 2 (H2). *There is a set of "optimal" network parameters, $\mathbf{c}^*$,σ^* and $\mathbf{w}^*$, that is, the optimal center, width and weight.*

4.1. Convergence Analysis of RBFNN with Fixed Structure

For the convenience of discussing its convergence, the differential equation of e is expressed as follows [29].

$$\dot{e}(t) = -e(t) + \mathbf{w}^{*T}(t)\boldsymbol{\theta}^{*}(t) - \mathbf{w}^{T}(t)\boldsymbol{\theta}(t), \tag{22}$$

where $\Delta\boldsymbol{\theta} = \boldsymbol{\theta}^{*} - \boldsymbol{\theta}, \Delta\mathbf{w} = \mathbf{w}^{*} - \mathbf{w}$, $\Delta\boldsymbol{\theta}$ is the change in the hidden layer neuron output, $\Delta\mathbf{w}$ is the change of weight. Equation (22) is reformulated as:

$$\dot{e}(t) = -e(t) + \mathbf{w}^{*T}(t)\Delta\boldsymbol{\theta}(t) + \Delta\mathbf{w}^{T}(t)\boldsymbol{\theta}(t). \tag{23}$$

In order to transform the nonlinear output of the network into a partially linear form, the Taylor expansion of $\Delta\boldsymbol{\theta}$ is:

$$\Delta\boldsymbol{\theta} = \boldsymbol{\varphi}_{\mathbf{c}}^{T}(\mathbf{c}^{*} - \mathbf{c}) + \boldsymbol{\varphi}_{\boldsymbol{\sigma}}^{T}(\boldsymbol{\sigma}^{*} - \boldsymbol{\sigma}) + \Omega, \tag{24}$$

where Ω is the higher order infinitesimal of Taylor expansion.

Theorem 1. *Suppose the number of hidden-layer neurons of RBFNN is fixed, and the network parameters are updated according to Equtions (14)–(16); when $t \to \infty, e(t) \to 0$, the convergence of the network is guaranteed.*

Proof of Theorem 1. The Lyapunov function is defined as:

$$V(e, \mathbf{c}, \boldsymbol{\sigma}, \mathbf{w}) = \frac{1}{2}\left(e^{2} + \Delta\mathbf{c}^{T}\Delta\mathbf{c} + \Delta\boldsymbol{\sigma}^{T}\Delta\boldsymbol{\sigma} + \Delta\mathbf{w}^{T}\Delta\mathbf{w} + \Delta\mathbf{v}^{T}\Delta\mathbf{v}\right) \tag{25}$$

with:

$$\begin{cases} \Delta\mathbf{c} = \mathbf{c}^{*} - \mathbf{c} \\ \Delta\boldsymbol{\sigma} = \boldsymbol{\sigma}^{*} - \boldsymbol{\sigma} \\ \Delta\mathbf{w} = \mathbf{w}^{*} - \mathbf{w} \\ \dot{\mathbf{v}} = e \\ \Delta\mathbf{v} = \mathbf{v}^{*} - \mathbf{v} = \mathbf{w}^{*T}\left(\boldsymbol{\varphi}_{\mathbf{c}}^{T}\mathbf{c}^{*} + \boldsymbol{\varphi}_{\boldsymbol{\sigma}}^{T}\boldsymbol{\sigma}^{*} + \Omega\right) - \mathbf{w}^{T}\left(\boldsymbol{\varphi}_{\mathbf{c}}^{T}\mathbf{c}^{*} + \boldsymbol{\varphi}_{\boldsymbol{\sigma}}^{T}\boldsymbol{\sigma}^{*}\right) \end{cases} \tag{26}$$

where $\mathbf{v}$ is the compensator and $\mathbf{v}^{*}$ is the optimal compensator. The partial derivative of the Lyapunov function is:

$$V'(e, \mathbf{c}, \boldsymbol{\sigma}, \mathbf{w}) = ee' + \Delta\mathbf{c}^{'T}\Delta\mathbf{c} + \Delta\boldsymbol{\sigma}^{'T}\Delta\boldsymbol{\sigma} + \Delta\mathbf{w}^{'T}\Delta\mathbf{w} + \Delta\mathbf{v}^{'T}\Delta\mathbf{v}. \tag{27}$$

According to Equations (22)–(26), we can obtain:

$$\begin{aligned} & V'(e, \mathbf{c}, \boldsymbol{\sigma}, \mathbf{w}) \\ &= ee' + (\mathbf{c}^{*} - \mathbf{c})'^{T}\Delta\mathbf{c} + (\boldsymbol{\sigma}^{*} - \boldsymbol{\sigma})'^{T}\Delta\boldsymbol{\sigma} + (\mathbf{w}^{*} - \mathbf{w})'^{T}\Delta\mathbf{w} + (\mathbf{v}^{*} - \mathbf{v})'^{T}\Delta\mathbf{v} \\ &= -e^{2} + e\left\{ \begin{array}{c} \mathbf{w}^{*T}\left[\boldsymbol{\varphi}_{\mathbf{c}}^{T}(\mathbf{c}^{*} - \mathbf{c}) + \boldsymbol{\varphi}_{\boldsymbol{\sigma}}^{T}(\boldsymbol{\sigma}^{*} - \boldsymbol{\sigma}) + \Omega\right] + \Delta\mathbf{w}^{T}\boldsymbol{\theta} - \\ \mathbf{w}^{T}\left[\boldsymbol{\varphi}_{\mathbf{c}}^{T}(\mathbf{c}^{*} - \mathbf{c}) + \boldsymbol{\varphi}_{\boldsymbol{\sigma}}^{T}(\boldsymbol{\sigma}^{*} - \boldsymbol{\sigma})\right] - \boldsymbol{\varphi}_{\mathbf{w}}^{T}\Delta\mathbf{w} \end{array} \right\} - e(\mathbf{v}^{*} - \mathbf{v}) \\ &= -e^{2} + e\left[\mathbf{v}^{*} - \mathbf{v} + \Delta\mathbf{w}^{T}\left(-\boldsymbol{\varphi}_{\mathbf{c}}^{T}\mathbf{c} - \boldsymbol{\varphi}_{\boldsymbol{\sigma}}^{T}\boldsymbol{\sigma} + \boldsymbol{\theta}\right) - \boldsymbol{\varphi}_{\mathbf{w}}^{T}\Delta\mathbf{w}\right] - e(\mathbf{v}^{*} - \mathbf{v}) \\ &= -e^{2} \end{aligned} \tag{28}$$

and:

$$V'(e, \mathbf{c}, \boldsymbol{\sigma}, \mathbf{w}) \leq 0. \tag{29}$$

Thus,V' is the seminegative definite in the above space. In light of the Lyapunov theorem, we can obtain:

$$\lim_{t \to \infty} e(t) = 0. \tag{30}$$

So far, the convergence of a fixed-structure RBFNN is proved. □

4.2. Convergence Analysis of RBFNN with Changeable Structure

Theorem 2. *If the network structure is self-organizing in the learning process, then the parameters are adjusted according to Equations (14)–(21). When $t \to \infty, e(t) \to 0$, the convergence of RBFNN based on sensitivity can be guaranteed.*

Proof Theorem 2. The structure self-organization process of RBFNN is divided into two parts: structure growth and structure pruning stages. □

4.2.1. Growth Stage

At time t, there are m neurons in the hidden layer of self-organizing RBFNN based on sensitivity, and the network error is $e_m(t)$. When the growth condition is satisfied, the number of hidden layer neurons is increased by 1. Then, the number of hidden layer neurons is $m + 1$, and the output error of the RBFNN is:

$$e_{m+1}(t) = \frac{1}{2}(\hat{y}(t) - \hat{y}_{m+1}(t))^2, \tag{31}$$

where $e_{m+1}(t)$ is the network error of $m + 1$ hidden neurons, $\hat{y}_m(t)$ and $\hat{y}_{m+1}(t)$ represent the network output before and after the addition of hidden layer neurons, respectively. According to Equations (17)–(19), we can obtain:

$$e_{m+1}(t) = \frac{1}{2}\left[\hat{y}(t) - \left(\sum_{j=1}^{m} w_j(t)\theta_j(t) + w_{m+1}(t)\theta_{m+1}(t)\right)\right]^2 = 0. \tag{32}$$

so:

$$e_{m+1}(t) = 0. \tag{33}$$

It can be seen that when a new neuron is added to the hidden layer, the convergence speed of the network is accelerated based on the parameter setting of the newly added neurons.

4.2.2. Prune Stage

When the pruning condition is satisfied, the jth neuron in the hidden layer is deleted, and the error of the network is $e_{m-1}(t)$. Thus, the error of the network can be rewritten as:

$$e_{m-1}(t) = \hat{y}_m(t) - \left(\sum_{l=1}^{m} w_l\theta_l(t) - w_j\theta_j(t)\right), \tag{34}$$

where $\hat{y}_m(t)$represents the network output when the number of hidden layer neurons is m:

$$\begin{aligned} e_{m-1}(t) &= \hat{y}_m(t) - \left(\sum_{l=1,l\neq i}^{m} w_l\theta_l(t) - w_j\theta_j(t) + \left(w_i + w_j\frac{\theta_j(t)}{\theta_i(t)}\right)\theta_i(t)\right) \\ &= \hat{y}_m(t) - \left(\sum_{l=1,l\neq i}^{m} w_l\theta_l(t) + w_i\theta_i(t)\right) \\ &= 0 \end{aligned} \tag{35}$$

In summary, the error of the neural network remains unchanged before and after the jth neuron is deleted, that is, the process of deleting neurons does not destroy the convergence of the original neural network.

5. Experimental Studies

This section describes the experiments conducted to assess the effectiveness of the generalization performance of the RBFNN-GP algorithm. The experiment includes one benchmark and two practical problems, namely, the approximation of the Mexican straw hat function and the prediction of key water quality parameters, ammonia nitrogen and

membrane permeability in the wastewater treatment process. In addition, the good generalization performance of the proposed RBFNN-GP is further illustrated by comparisons with the existing six algorithms.

5.1. Benchmark Example A

In this case, the RBFNN-GP algorithm is applied to approximate the Mexican straw hat function, which is a benchmark problem used in [30,31] to checkout many prevalent algorithms. The Mexican straw hat function is:

$$y = \sin(\sqrt{x_1^2 + x_2^2}) / \sqrt{x_1^2 + x_2^2}. \tag{36}$$

In the training phase, the training samples are $\mathbf{x} = \{x_1, x_2; y\}^N$, where x_1, x_2 are stochastic and generated within the scope of $(-2\pi, 2\pi)$, $N = 2100$ is the number of training samples; the testing samples are $\{x_1, x_2, y\}^M$, where $M = 700$ is the number of testing samples, and the learning rate is set to $\eta = 0.003$.

The experimental results are shown in Figure 2, where Figure 2a shows the number of hidden neurons in the training process. It can be seen that the proposed RBFNN-GP can adjust the network structure by pruning or increasing the number of neurons in the learning process. Figure 2b shows the change trajectory of the root-mean-square error (RMSE) in the learning process. Meanwhile, the prediction error results and output error surface are depicted in Figure 2c,d. It is clear that the proposed RBFNN-GP can approximate the Mexican straw hat function with small predicting errors. In order to further prove the excellent generalization ability of the proposed method, the prediction results of the RBFNN-GP are compared with those of the other dynamic neural networks based on structural adjustment, such as SASOA-FNN [12], SOFNN-HPS [17], SAS-RBFNN [18], AGMOPSO [21], ASOL-SORBFNN [23] and the RBFNN with a fixed structure (fixed-RBFNN). In order to make the comparison more meaningful, all algorithms in this experiment use the same data set, including training samples and test samples, and ensure that the initial number of neurons is the same. In addition, all of the algorithms run 10 times, and then take the average value to make the results more convincing. The results are shown in Table 1, where Max. is the maximum and Dev. is the deviation.

As can be seen from Table 1, the proposed RBFNN-GP requires fewer hidden nodes and output errors and has a better generalization ability than the self-organizing network based on information minimization and structural risk minimization. This mainly depends on the fact that the method considers not only the number of effective parameters in the network, but also the mean and variance of input data.

Table 1. Comparison results with other self-organizing methods.

Methods	No. of NNs	CPU Time(s)		Testing RMSE			Training RMSE
		Mean	Dev.	Mean	Dev.	Max.	Mean
Fixed-RBFNN	8	100.10	0.096	0.031	0.0037	0.040	0.042
SASOA-FNN [12]	8	108.29	0.031	0.029	0.0043	0.033	0.041
SOFNN-HPS [17]	12	131.02	0.076	0.047	0.0053	0.063	0.057
SAS-RBFNN [18]	9	119.35	0.024	0.039	0.0051	0.059	0.052
AGMOPSO [21]	8	106.29	0.028	0.024	0.0037	0.039	0.041
ASOL-SORBFNN [23]	8	135.46	0.031	0.035	0.0046	0.041	0.040
RBFNN-GP	7	100.36	0.012	0.027	0.0033	0.029	0.039

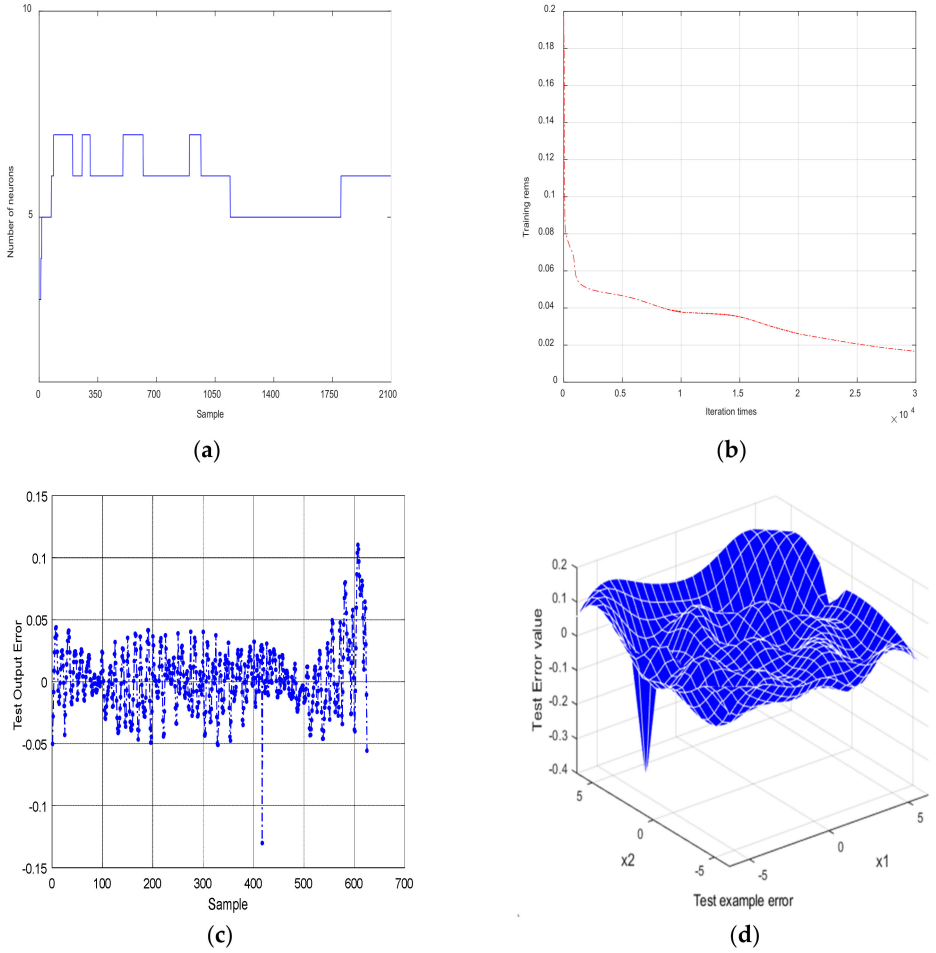


Figure 2. Approximation results of the RBFNN-GP in example A. (**a**) Number of neurons; (**b**) The training RMSE value in example A; (**c**) Test examples output error in example A; (**d**) Test examples output error surface.

5.2. Benchmark Example B

In this example, the effectiveness of the RBFNN-GP is applied for the Mackey–Glass chaotic time series prediction problem, which is a famous benchmark [32,33]. The discrete expression of the time series is given by:

$$\begin{aligned} x(t+1) &= (1-a_1)x(t) + a_2 x(t-\tau)/\left[1+x^{10}(t-\tau)\right] \\ x(t+1) &= f[x(t), x(t-6), x(t-12), x(t-18)] \end{aligned} \tag{37}$$

where a_1, a_2, τ are constants, $x(0)$ represents the initial value, and $a_1 = 0.1$, $a_2 = 0.2$, $\tau = 17$, $x(0) = 1.2$. In the training phase, 1000 data samples are extracted from $t = 21$ to 1021. Additionally, 500 samples are used as training samples and 500 samples as test samples. The preset training error is 0.001, the initial learning rate $\eta = 0.02$, and the number of neurons is 5. It should be noted that the other parameters in all comparison methods are the same, and the running results of the experiment are shown in Figure 3.

From Figure 3a, it can be seen that there are several increasing and decreasing stages in the learning process of RBFNN-GP. In the early period of training, the number of neurons changes frequently, and the network structure is unstable. Figure 3b shows the comparison between the predicted output of RBFNN-GP and the real value. Figures 3c,d show the prediction error value of the network and the RSME value in the training stage, respectively. It is worth mentioning that the reason why the RSME curve is so smooth is closely related to the sensitivity of the network. Since the mean and variance of the training samples are fully considered, the stability of the network is improved.

Similarly, Table 2 shows the comparative experimental results with the other four methods: SASOA-FNN [12], SOFNN-HPS [17], SAS-RBFNN [18], AGMOPSO [21], ASOL-SORBFNN [23] and fixed-RBFNN. As shown in Table 2, the optimal number of hidden nodes obtained by using the RBFNN-GP is only six, and the mean and deviation of the test error are minimal in the comparison method. Since the parameters are updated at the same time in the process of structural adjustment to avoid repeated calculations, the computational complexity is greatly reduced compared with the dynamic structural adjustment method based on information strength. Nevertheless, compared with other self-organizing networks based on structural risk, the proposed RBFNN-GP takes a little longer time to calculate the mean and variance information of input samples. However, we still have reason to believe that RBFNN-GP demonstrates a good compromise between generalization ability and training accuracy.

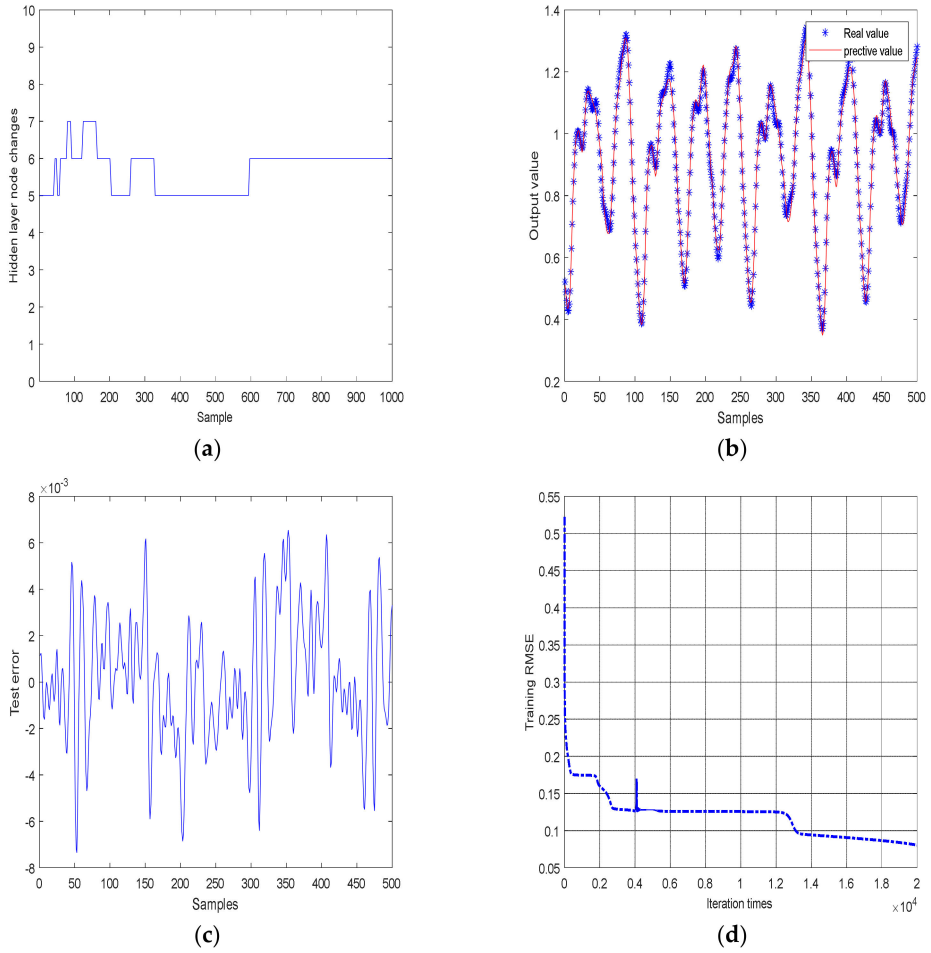


Figure 3. Prediction results of the RBFNN-GP in example B: (**a**) Number of neurons; (**b**) Fitting results of the RBFNN-GP in example B; (**c**) Test output error value; (**d**) The training RMSE value in example B.

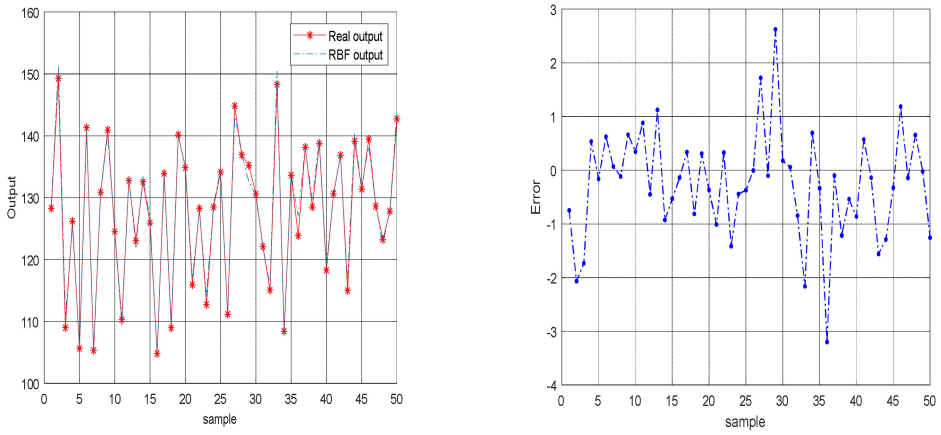
Table 2. Comparison results with other self-organizing methods.

Methods	No. of NNs	CPU Time(s)		Testing RMSE			Training RMSE
		Mean	Dev.	Mean	Dev.	Max.	Mean
Fixed-RBFNN	8	104.10	0.069	0.036	0.0039	0.035	0.312
SASOA-FNN [12]	7	126.29	0.030	0.029	0.0041	0.028	0.124
SOFNN-HPS [17]	11	116.3	0.076	0.054	0.0066	0.068	0.261
SAS-RBFNN [18]	10	123.97	0.026	0.034	0.0045	0.031	0.219
AGMOPSO [21]	8	119.89	0.029	0.027	0.0039	0.031	0.217
ASOL-SORBFNN [23]	8	133.91	0.065	0.043	0.0044	0.040	0.251
RBFNN-GP	6	105.36	0.022	0.026	0.0032	0.019	0.215

5.3. Permeability Prediction of Membrane Bio-Reactor

Membrane bio-reactor (MBR) is a new wastewater treatment technology combining membrane separation technology and biotechnology, which is widely used in wastewater treatment process (WWTP). However, in the process of MBR wastewater treatment, membrane pollution will shorten the service life of the membrane and cause an unnecessary loss of process energy consumption. It is of great practical significance to correctly predict the permeability of the membrane and increase its working efficiency [34–36]. Therefore, the proposed RBFNN-GP is applied to predict the permeability of MBR in WWTP. The real data of the experiment come from a wastewater treatment plant in Beijing, China. After disposing of the abnormal data, 500 samples were obtained and normalized.

In this experiment, 50 training samples and 50 test samples are selected to test the performance of RBFNN-GP. In order to remove the correlation between variables, partial least squares method was used to select nine variables from twenty-two variables as the input of RBFNN-GP. Due to the wide range of membrane bioreactor permeability, the number of iterations is $T = 200$, the learning rate is $\eta = 0.003$, and the time length is $h = 10$. The intuitive prediction results are shown in Figure 4. Figure 4a shows the comparison results between the actual and predicted values of membrane bioreactor permeability, and the prediction error is shown in Figure 4b. It can be seen from Figure 4 that the proposed RBFNN-GP has a good prediction performance for the permeability of the MBR, and the prediction error within the range $[-4, 3]$.



(**a**) (**b**)

Figure 4. The permeability prediction results for MBR in WWTP: (**a**) Fitting results of RBFNN-GP for MBR; (**b**) Test output error for MBR.

Moreover, Table 3 shows the comparison results of SASOA-FNN [12], SOFNN-HPS [17], SAS-RBFNN [18], AGMOPSO [21], ASOL-SORBFNN [23] and Fixed-RBFNN in predicting the permeability of MBR. It can be seen from Table 3 that, under the same iteration times, the learning time of SASOA-FNN [12] and RBFNN-GP is almost the same. However, the

accuracy of the latter is better than that of the former. Thus, we have sufficient reasons to believe that the RBFNN-GP method shows a great improvement in training accuracy, calculation speed and generalization ability compared with the above methods.

Table 3. Comparison results with other self-organizing methods.

Methods	No. of NNs	CPU Time(s)		Testing RMSE			Training RMSE
		Mean	Dev.	Mean	Dev.	Max.	Mean
Fixed-RBFNN	8	105.65	0.037	0.033	0.0034	0.037	0.031
SASOA-FNN [12]	8	109.21	0.035	0.028	0.0033	0.021	0.024
SOFNN-HPS [17]	11	126.34	0.032	0.042	0.0049	0.025	0.032
SAS-RBFNN [18]	9	121.62	0.042	0.025	0.0038	0.029	0.035
AGMOPSO [21]	8	114.25	0.031	0.030	0.0042	0.031	0.114
ASOL-SORBFNN [23]	11	125.31	0.040	0.028	0.0041	0.040	0.127
RBFNN-GP	6	108.24	0.032	0.025	0.0040	0.022	0.022

6. Discussion

6.1. Computational Complexity

Computational complexity is an important indicator for evaluating the model. For the proposed RBFNN-GP, the calculation involved is closely related to the training process of and N. Suppose $[\mathbf{x}(t), y(t); t = 1, \cdots, N]$ is a set of training samples. When the structure of RBFNN-GP is *k-m-l* (k represents the input variable, m represents the number of hidden layer neurons, and l represents the output variable), the computational complexity of RBFNN-GP is O[m(t)]. It can be seen that the computational burden of RBFNN-GP is not heavy.

6.2. Future Trends

In this paper, which aims to realize the online prediction of key water quality parameter membrane pollution in wastewater treatment process, a prediction model based on self-organizing RBFNN is established that meets the needs of accurate prediction of key water quality parameters in wastewater treatment process. At the same time, the self-organizing network modeling method is developed. Due to its good generalization performance and theoretical support, the proposed method can also be extended to other types of networks, such as multi-input, single-output fuzzy neural networks and multi-input, multiple-output RBF neural network.

7. Conclusions

In this paper, an RBFNN-GP algorithm is proposed to improve the model generalization ability. Firstly, the upper bound of the local generalization error is found, and the network structure and parameters are adjusted within the allowable error range. Then, a generalization error formula based on network sensitivity and approximate error is introduced to improve the generalization performance, while ensuring its accuracy. Finally, experimental validation is carried out on two different benchmark data sets and a real application of wastewater treatment process. The results show that the RBFNN-GP algorithm can learn robust networks with good generalization performance and compact scale. In summary, RBFNN-GP has the following advantages:

1. With the help of sensitivity measurements and locally generalized error bounds, the network has a statistical performance and can reasonably achieve structure self-organization without a high dependence on sample numbers.
2. The convergence of RBFNN-GP for fixed and variable structures is guaranteed by the thresholds λ_1 and λ_2. Therefore, the proposed RBFNN-GP can not only reduce the number of additional parameters, but also decreases the computational burden.
3. Compared with existing algorithms, the proposed RBFNN-GP shows a good generalization ability in the prediction of key water quality parameters in wastewater

treatment processes. Furthermore, this approach can be extended to other types of networks and industrial domains.

Author Contributions: Conceptualization, X.G. and Y.Y.; methodology, P.W.; software, P.W.; validation, Y.Y.; formal analysis, Y.Y.; investigation, Y.Y.; resources, X.G.; data curation, P.W.; writing—original draft preparation, Y.Y.; writing—review and editing, Y.Y.; visualization, Y.Y.; supervision, X.G.; project administration, P.W.; funding acquisition, P.W. All authors have read and agreed to the published version of the manuscript.

Funding: This study was funded by the National Natural Science Foundation of China, grant number is 61640312; the Natural Science Foundation of Beijing Municipality, grant numbers are 4172007 and 4192011.

Institutional Review Board Statement: Not applicable.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

1. Xu, L.; Qian, F.; Li, Y.; Li, Q.; Yang, Y.-W.; Xu, J. Resource allocation based on quantum particle swarm optimization and RBF neural network for overlay cognitive OFDM System. *Neurocomputing* **2016**, *173*, 1250–1256. [CrossRef]
2. Xiong, T.; Bao, Y.; Hu, Z.; Chiong, R. Forecasting interval time series using a fully complex-valued RBF neural network with DPSO and PSO algorithms. *Inf. Sci.* **2015**, *305*, 77–92. [CrossRef]
3. Han, H.G.; Zhang, L.; Hou, Y.; Qiao, J.F. Nonlinear Model Predictive Control Based on a Self-Organizing Recurrent Neural Network. *IEEE Trans. Neural Netw. Learn. Syst.* **2015**, *27*, 402–415. [CrossRef] [PubMed]
4. Yang, Y.-K.; Sun, T.-Y.; Huo, C.-L.; Yu, Y.-H.; Liu, C.-C.; Tsai, C.-H. A novel self-constructing Radial Basis Function Neural-Fuzzy System. *Appl. Soft Comput.* **2013**, *13*, 2390–2404. [CrossRef]
5. Niyogi, P.; Girosi, F. On the Relationship between Generalization Error, Hypothesis Complexity, and Sample Complexity for Radial Basis Functions. *Neural Comput.* **1996**, *8*, 819–842. [CrossRef]
6. Qiao, J.-F.; Han, H.-G. Optimal Structure Design for RBFNN Structure. *Acta Autom. Sin.* **2010**, *36*, 865–872. [CrossRef]
7. Vapnik, V.N. *Estimation of Dependences Based on Empirical Data*; Springer: Berlin, Germany, 1982.
8. Pollard, D. *Convergence of Stochastic Processes*; Springer: Berlin, Germany, 1984.
9. Haussler, D. Decision-theoretic generalizations of the PAC model for neural net and other learning applications. *Inf. Comput.* **1992**, *100*, 78–150. [CrossRef]
10. Barron, A.R. Approximation and estimation bounds for artificial neural networks. *Mach. Learn.* **1994**, *14*, 115–133. [CrossRef]
11. Yeung, D.S.; Ng, W.W.Y.; Wang, D.; Tsang, E.C.C.; Wang, X.-Z. Localized Generalization Error Model and Its Application to Architecture Selection for Radial Basis Function Neural Network. *IEEE Trans. Neural Netw.* **2007**, *18*, 1294–1305. [CrossRef]
12. Sarraf, A. A tight upper bound on the generalization error of feedforward neural networks. *Neural Netw.* **2020**, *127*, 1–6. [CrossRef]
13. Han, H.; Wu, X.; Liu, H.; Qiao, J. An Efficient Optimization Method for Improving Generalization Performance of Fuzzy Neural Networks. *IEEE Trans. Fuzzy Syst.* **2019**, *27*, 1347–1361. [CrossRef]
14. Terada, Y.; Hirose, R. Fast generalization error bound of deep learning without scale invariance of activation functions. *Neural Netw.* **2020**, *129*, 344–358. [CrossRef]
15. Suzuki, T. Fast generalization error bound of deep learning from a kernel perspective. In Proceedings of the International Conference on Artificial Intelligence and Statistics, Playa Blanca, Lanzarote, 9–11 April 2018; pp. 1397–1406.
16. Ng, W.; Yeung, D.; Wang, X.-Z.; Cloete, I. A study of the difference between partial derivative and stochastic neural network sensitivity analysis for applications in supervised pattern classification problems. In Proceedings of the 2004 International Conference on Machine Learning and Cybernetics (IEEE Cat. No. 04EX826), Shanghai, China, 26–29 August 2004.
17. Zhou, H.; Zhang, Y.; Duan, W.; Zhao, H. Nonlinear systems modelling based on self-organizing fuzzy neural network with hierarchical pruning scheme. *Appl. Soft Comput.* **2020**, *95*, 106516. [CrossRef]
18. Xie, Y.; Yu, J.; Xie, S.; Huang, T.; Gui, W. On-line prediction of ferrous ion concentration in goethite process based on self-adjusting structure RBF neural network. *Neural Netw.* **2019**, *116*, 1–10. [CrossRef] [PubMed]
19. Huang, G.-B.; Saratchandran, P.; Sundararajan, N. An Efficient Sequential Learning Algorithm for Growing and Pruning RBF (GAP-RBF). *IEEE Trans. Syst. Man Cybern. Part B-Cybern.* **2004**, *34*, 2284–2292. [CrossRef]
20. Huang, G.-B.; Saratchandran, P.; Sundararajan, N. A Generalized Growing and Pruning RBF (GGAP-RBF) Neural Network for Function Approximation. *IEEE Trans. Neural Netw.* **2005**, *16*, 57–67. [CrossRef]
21. Han, H.G.; Wu, X.L.; Zhang, L.; Tian, Y.; Qiao, J.F. Self-Organizing RBF neural network using an adaptive gradient multi-objective particle swarm optimization. *IEEE Trans. Cybern.* **2019**, *49*, 69–82. [CrossRef] [PubMed]
22. Han, H.-G.; Chen, Q.-L.; Qiao, J.-F. An efficient self-organizing RBF neural network for water quality prediction. *Neural Netw. Off. J. Int. Neural Netw. Soc.* **2011**, *24*, 717–725. [CrossRef]

23. Han, H.-G.; Ma, M.-L.; Yang, H.-Y.; Qiao, J.-F. Self-organizing radial basis function neural network using accelerated second-order learning algorithm. *Neurocomputing* **2021**, *469*, 1–12. [CrossRef]
24. Li, Z.-Q.; Zhao, Y.-P.; Cai, Z.-Y.; Xi, P.-P.; Pan, Y.-T.; Huang, G.; Zhang, T.-H. A proposed self-organizing radial basis function network for aero-engine thrust estimation. *Aerosp. Sci. Technol.* **2019**, *87*, 167–177. [CrossRef]
25. Shi, J.R.; Wang, D.; Shang, F.H.; Zhang, H.Y. Research advances on stochastic gradient descent algorithms. *Acta Autom. Sin.* **2021**, *47*, 2103–2119. [CrossRef]
26. Hoeffding, W. Probability Inequalities for Sums of Bounded Random Variables. *J. Am. Stat. Assoc.* **1963**, *58*, 13–30. [CrossRef]
27. Jiang, Q.; Gao, D.-C.; Zhong, L.; Guo, S.; Xiao, A. Quantitative sensitivity and reliability analysis of sensor networks for well kick detection based on dynamic Bayesian networks and Markov chain. *J. Loss Prev. Process. Ind.* **2020**, *66*, 104180. [CrossRef]
28. Belouz, K.; Nourani, A.; Zereg, S.; Bencheikh, A. Prediction of greenhouse tomato yield using artificial neural networks combined with sensitivity analysis. *Sci. Hortic.* **2021**, *293*, 110666. [CrossRef]
29. Han, H.-G.; Qiao, J.-F. Adaptive computation algorithm for RBF neural network. *IEEE Trans. Neural Netw. Learn. Syst.* **2011**, *23*, 342–347. [CrossRef]
30. Xie, T.; Yu, H.; Hewlett, J.; Rózycki, P.; Wilamowski, B. Fast and efficient second-order method for training radial basis function networks. *IEEE Trans. Neural Networks Learn. Syst.* **2012**, *23*, 609–619. [CrossRef]
31. Han, H.-G.; Ma, M.-L.; Qiao, J.-F. Accelerated gradient algorithm for RBF neural network. *Neurocomputing* **2021**, *441*, 237–247. [CrossRef]
32. Zemouri, R.; Racoceanu, D.; Zerhouni, N. Recurrent radial basis function network for time-series prediction. *Eng. Appl. Artif. Intell.* **2003**, *16*, 453–463. [CrossRef]
33. Han, H.-G.; Lu, W.; Hou, Y.; Qiao, J.-F. An Adaptive-PSO-Based Self-Organizing RBF Neural Network. *IEEE Trans. Neural Netw. Learn. Syst.* **2016**, *29*, 104–117. [CrossRef]
34. Teychene, B.; Guigui, C.; Cabassud, C. Engineering of an MBR supernatant fouling layer by fine particles addition: A possible way to control cake compressibility. *Water Res.* **2011**, *45*, 2060–2072. [CrossRef]
35. Huyskens, C.; Brauns, E.; Vanhoof, E.; Dewever, H. A new method for the evaluation of the reversible and irreversible fouling propensity of MBR mixed liquor. *J. Membr. Sci.* **2008**, *323*, 185–192. [CrossRef]
36. Feng, H.-M. Self-generation RBFNs using evolutional PSO learning. *Neurocomputing* **2006**, *70*, 241–251. [CrossRef]