Eng Int Syst (2025) 2: 155–167

Engineering
Intelligent Systems

Design of Function Approximation Algorithm Based on RBF Neural Network

Lanbao Hou*

School of Mathematics and Physics, Jingchu University of Technology, Jingmen 448000, Hubei, China

Function approximation is an important part of function theory, and its role in numerical computation is crucial. The application of a neural network in function approximation is an innovative means of developing function approximation. However, most of the current researches compare the function approximation of various networks in depth. The purpose of this study is to investigate ways to examine and analyze function approximation based on a neural network, and describe a radial neural network. This study addresses the problem of function approximation algorithm design. Because this is based on an artificial neural network, a typical neural network is taken as an example, and the related concepts and algorithms are described in detail. The radial basis function neural network (RBFNN) for function approximation is designed and analyzed using experimental simulation. According to the results of RBFNN and BPNN simulation experiments, the errors of RBFNN are close to 0, while the errors of BPNN have large oscillations around 0. From the results and analysis, it can be concluded that the approximation effect of RBFNN is better than that of BPNN, providing an important reference value for research on function approximation.

Keywords: function approximation, artificial neural network, radial neural network, BP neural network

1. INTRODUCTION

RBFNN has made progress in numerous application fields because of its straightforward structure, execution speed and capacity. In RBFNN, the selection of the center position of the hidden layer unit and the corresponding width value is the key to the performance of the entire network, directly affecting the network's accessibility. RBFNN is a new and viable brain criticism organization, which has the best guess execution and worldwide ideal execution that other supporting organizations don't have, with basic construction and quick preparation speed. In the RBF neural network, the number and position of the hidden layer center is the key to the performance of the entire network, which directly affects the overall performance of the network. It can also choose the number of centers, that is, the number of hidden level nodes, which can easily lead to over-adaptation and reduce the promotion ability. If too few centers are chosen, the learned network will not fully learn the information contained in the samples, resulting in generalization. In practical applications, the advantage of RBF networks is that linear learning algorithms can be used to do what nonlinear learning algorithms used to do. At the same time, neural networks have the high precision of nonlinear algorithms. However, when solving large-scale data problems, RBFNNs identified by traditional methods have obvious shortcomings in generalization ability.

Function approximation in high dimensions is a challenge for neural networks due to the issue of dimensionality. After studying curves on variants, Chen proposed a geometric method to solve the strong approximation problem on complex curve function domains. He proved that the strong approximation works for smooth, low-degree affine complete intersections whose boundaries are smooth at infinity [1]. Andras focused on the estimation error of brain networks prepared on projection spaces. He showed that such brain organizations ought to have preferred estimation execution over brain networks prepared on high-layered information in any event, while projecting moderately meager examples in light of the

*Corresponding author's Email: houbao79@163.com

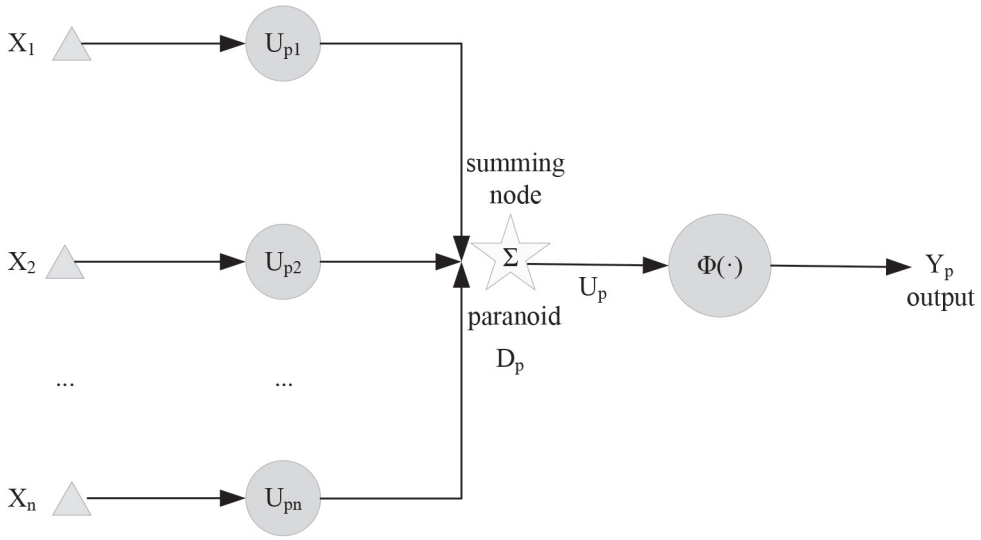


Figure 1 Nonlinear model of a neuron.

complex information [2]. Ziemke utilized direct capability estimation. MATSim, the traffic simulation software, carries out SARSA (λ) with Fourier premise elements to control traffic lights. Ziemke compared the outcomes of having normal regulators like fixed time, and investigated other cutting-edge, rule-based versatile techniques [3]. Sonker presented some results for the absolute almost generalized Nrlund summability of orthogonal series, and he also derived the most important corollary of the main results [4]. Nguyen proposed a piecewise linear (PWL) sigmoid function approximation based on the probability of a statistical distribution of neuron values in each layer. This approach is used to improve the accuracy of network identification using only adding circuits [5]. However, the comparison of various neural networks needs to be further strengthened.

At the same time, with the continuous expansion of the field of artificial neural network, the research on it is also gradually increasing. Ravichandran gave an overview of different brain organizations, focusing on networks in light of arising memristive gadgets. Memristor brain networks perform better than unadulterated corresponding metal-oxide-semiconductor (CMOS) executions [6]. Alanis presented the consequences of utilizing the repetitive brain network preparing calculation in view of broadened Kalman channel and its application in power cost gauging. He showed the materialness of the proposed gauging plan by one-stride ahead and n-stride ahead determining utilizing European power framework information [7]. Isik conducted forecasts of meteorological data used in the design of thermal systems for fifty cities representing the whole of Turkey. The data he obtained from the General Meteorological Administration (MGM) was modeled using ANN and an adaptive network-based fuzzy inference system [8]. Li examined how to utilize brain organizations to control network tied rectifiers/inverters to mitigate such restrictions. The brain network executes a unique programming calculation. Also, it is prepared by worldly backpropagation [9]. Cascardi proposed a simulated brain network-based logical model to foresee the capacity of FRP to restrict the strength of cement. The outcomes show that the model is reasonable for the plan of FRP-bound cement and ensures further developed precision comparative with existing contenders [10]. These algorithms provide a solution to the proposed problem to a certain extent, but are overly complex.

This paper uses the Matlab neural network toolbox to design the RBFNN and BPNN simulation experiments of the unit function. This paper also compares its approximation performance, generalization ability, convergence speed, error and so on. RBFNN is not only convenient in design, fast in training, but also can achieve better approximation accuracy. The innovation presented in this paper is that the function approximation is combined with ANN. It explains the theory and related methods of RBFNN in detail, and makes a comparative analysis of BPNN.

2. METHOD BASED ON NEURAL NETWORK METHOD

2.1 Artificial Neural Network

With the improvement of intelligent technology, brain network hypothesis has been generally utilized, in which feedforward and criticism network are two run-of-the mill network models. Feedforward network (including BPNN, RBFNN, etc.) is a powerful learning system with complex nonlinear processing capabilities. The application of mapping and approximation functions is a common feature of neural networks.

Neuron is the basic unit of an artificial neural network system, and it is also the basic unit of information processing of the whole system [11]. The nonlinear model of the artificial neural network is shown in Figure 1 below.

Artificial neurons are the basis of neural networks [12]. Usually, neuron models have three basic components, namely:

1. A synapse (or connection). Different neurons have different synaptic connection coefficients. Unlike the synapses of the human brain, the synaptic connection

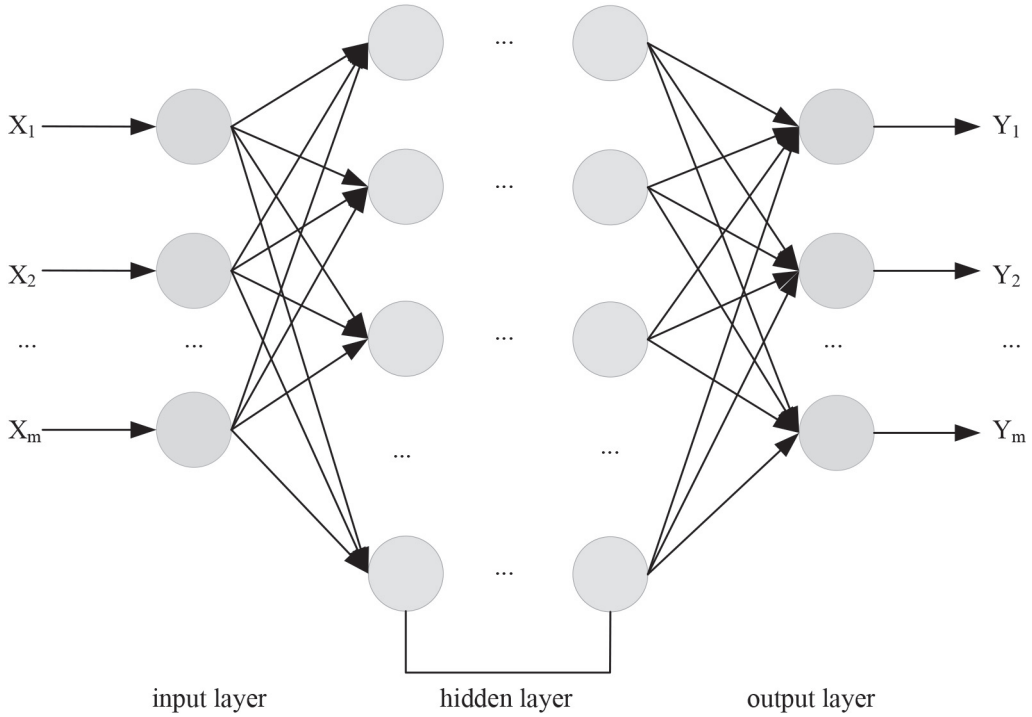


Figure 2 General model of feedforward neural network.

coefficient of artificial neurons has a certain range, which can be positive or negative.

2. Adder. It is used to calculate the sum of the synaptic connections corresponding to the neuron's input signal. This function constitutes a linear combinator as shown in expression (1).

3. Transfer function. This limits the range of values that the neuron outputs. The transfer function is also called the suppression function. The transfer function controls the output signal within a desired range.

The neuron model in Figure 1 also has an external bias. The effect of the bias is to increase or decrease the network input to the activation function based on the result of the summing node.

For any neuron, we can describe it with a set of equations like (1) and (2):

$$v_p = \sum_{b=1}^{n} u_{pb} x_p \tag{1}$$

$$y_p = \phi(v_p + d_p) \tag{2}$$

In equations (1) and (2), $x_1, x_2, ..., x_m$ is the input signal of the nervous system. In the equation, $v_{p1}, v_{p2}, ..., v_{pm}$ is the synaptic connection coefficient of neuron p, and v_p is the output after the input signal passes through the adder. The transfer function is $\phi(\cdot)$, and the function of the bias d_p is to transform the output d_p of the adder in Figure 1, as shown in the following equation (3):

$$u_p = v_p + d_p \tag{3}$$

The dozens of neural network models currently available can be divided into two types according to the transmission direction of the internal information of the neural network:

The neurons of a feedforward neural network are arranged in a plane. An input layer that receives the input signal, an output layer that outputs the signal, and a hidden layer between the input layer and the output layer are respectively formed. The main feature of neurofeedback networks is that only the neurons between two adjacent layers are connected to each other, and there is no feedback between neurons. In addition to the output layer, each neuron can receive multiple inputs from the previous layer, and only one output is sent to the neuron of the next layer as the input of the next layer. However, they do not accept neurons that leave hidden layers [13–14].

According to this feature, a general model of neural supply network can be obtained, as shown in Figure 2 below.

The structure of the neurofeedback network is shown in Figure 3. Stimulated by a signal input, the situation will continuously change, and finally, the output will be received and fed back at the input to create a new output, thus continuing this feedback process [15].

2.2 RBF Neural Network

(1) RBF algorithm

The RBFNN algorithm was proposed in 1989 by researchers who explained in depth the error backpropagation algorithm. The emergence of BP algorithm successfully solves the weight training and learning problems of multi-level network connections. It has completely changed people's understanding of the difficulty of training, that is, multi-layer perceptron networks (MLPNN), function approximation, image recognition and other fields have been widely used.

RBFNN is divided into an input layer, intermediate layer and output layer. The middle layer stores pattern vectors

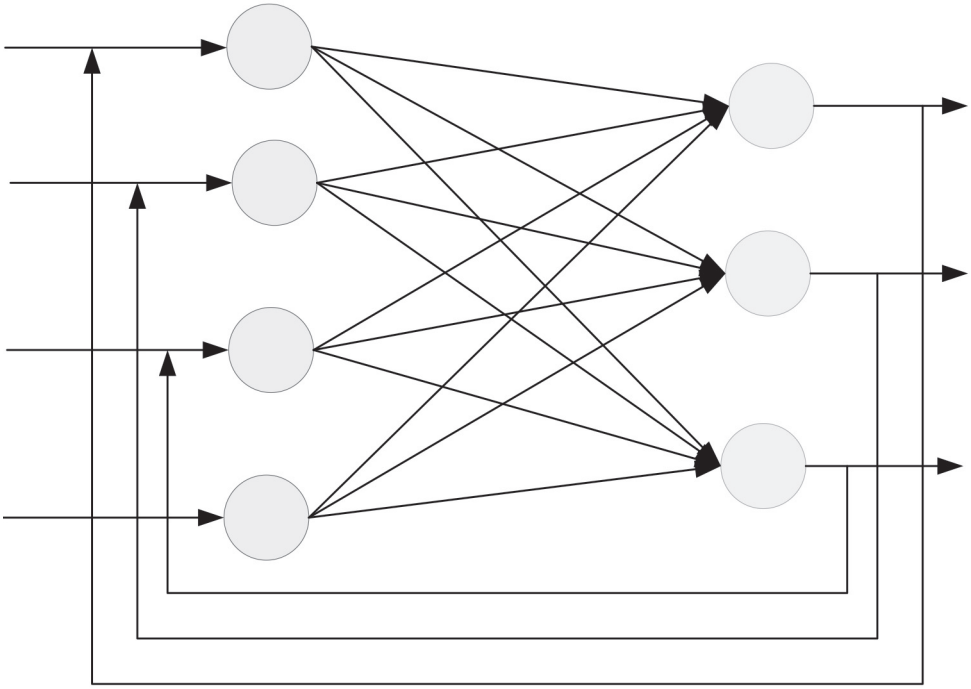


Figure 3 Feedback neural network with hidden layers.

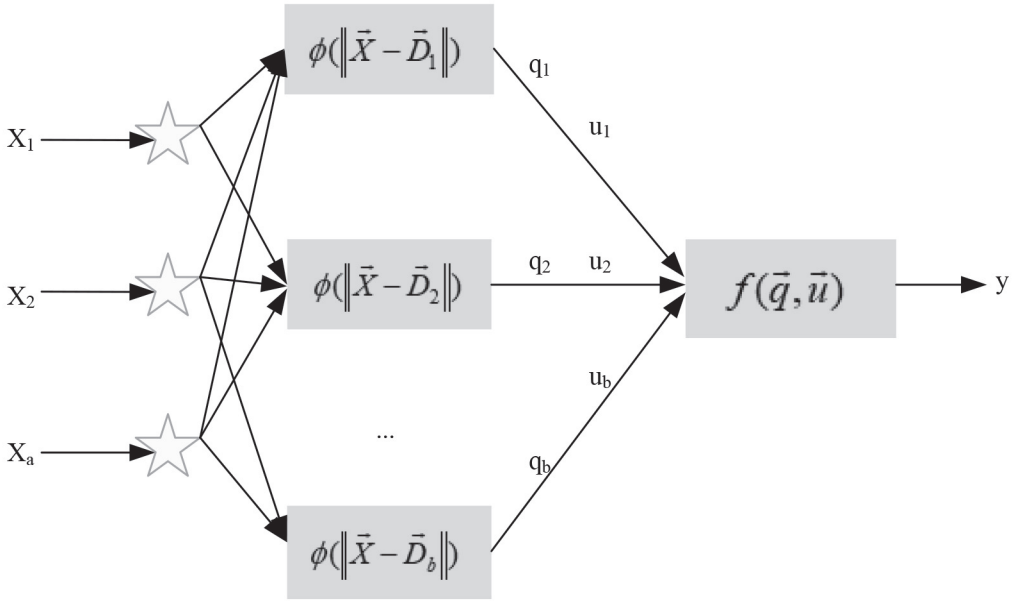


Figure 4 RBFNN structure schematic.

reflecting external things, performs nonlinear transformation on the external vectors, and then performs linear segmentation on the output plane to identify target vectors from the input vector set [16–17]. The structure of RBFNN is shown in Figure 4.

RBFNN has been widely used in nonlinear system modeling, analysis, data mining and other fields. It is also used to apply Bayesian rules and model the mapping of any continuous pair of input-output data. The wide applications of RBFNN leverage its excellent nonlinear approximation performance.

The RBF input-output function relationship is shown in equation (4).

$$y(p) = F(\vec{X}(p)) = f\left(\phi\vec{X}(p), \vec{D}\right), \vec{U}) \tag{4}$$

In equation (4): $\vec{X}(p)$: pth external input vector;

$\vec{D}$: the template vector vector stored in the neurons in the middle layer;

$\phi(\cdot)$: the activation function of the middle layer;

$\vec{U}$: connection weight matrix from the intermediate layer to the output layer;

$f(\cdot)$: output layer activation function.

RBFNN is a feedforward network. The greatest distinction between it and a BP network is that the latter is a model of worldwide brain organization. The greatest inconvenience of worldwide guess of brain network models is that any boundary

that can be adjusted to the whole organization model will influence the result results. One input and one output to the network will have an association weight to modify them, which builds the aggregate. The model advances gradually and can undoubtedly drop to a neighborhood local minimum, which influences exactness and requires some investment. The distinction of the RBFNN model is that it is a local approximation model. It just makes association loads for neighborhood regions of the information and influences the result. For every information info and result, RBFNN has a couple of association loads that should be changed. Unlike BP, which is utilized for each item of information and result information, it will change the association loads of the whole brain organization. This feature of the RBF function enables it to learn fast and avoid the local minima problem. Because of this feature, the RBFNN model has been broadly utilized in the fields of picture handling, blunder determination, capability estimate, and foreseeing information [18–19]. At present, there are two main types of RBFNN, namely permutation RBFNN and generalized RBFNN.

(2) Regularized RBFNN model

1) Function approximation and interpolation based on RBF function technology

Current neural network models go from an M-dimensional input space to a one-dimensional output space[20]. This method is also used to map the vectors of these two spaces to the one-dimensional space. The input vector in the M-dimensional space corresponds to the one-dimensional space, and becomes the output vector. We assume that there are M entry vectors [21–22] in the M-dimensional space. The corresponding values $X_m, m = 1, 2, 3, ..., m$ of these input vectors in a one-dimensional output space. In this case, m inputs and outputs constitute the basic set of artificial neural network training samples $Y_m, m = 1, 2, 3, ..., m$. The main purpose of the so-called interpolation is to determine the $F(X)$ matching function that satisfies the condition of equation (5):

$$F(x^m) = y^m, m = 1, 2, 3, ..., m \tag{5}$$

In the above equation, the F function represents the addition surface. The interjected surface should pass all preparing tests. In spiral premise works, the answer to this addition issue is to supplant the obstruction surface in the above equation with the premise capability m. All data in the training samples will have a basis for a function. The structure is displayed in equation (6):

$$\phi\left(\left\|X - X^m\right\|\right), m = 1, 2, 3, ..., m \tag{6}$$

In the above formula, the basis function ϕ is a nonlinear function, and the training data point X^m is its ϕ center. The independent variable of the basis function is the X^m distance between the entry area X point and the center. Because the distance is radial, the function ϕ is called the radial basis function. The interference function based on radial basis function technology is defined as a linear combination of basis functions, as shown in equation (7):

$$F(X) = \sum_{m=1}^{m} \eta_m \left(\left\|X - X^m\right\|\right) \tag{7}$$

Then using the interpolation conditions in equation (5) into (7), we will obtain m linear equations for η^m, as shown in equation (8):

$$\begin{cases} \sum_{m=1}^{m} \eta_m \varphi\left(\left\|X^1 - X^m\right\|\right) = y^1 \\ \sum_{m=1}^{m} \eta_m \varphi\left(\left\|X^2 - X^m\right\|\right) = y^2 \\ \cdots \\ \cdots \\ \cdots \\ \sum_{m=1}^{m} \eta_m \varphi\left(\|X^m - X^m\|\right) = y^m \end{cases} \tag{8}$$

We let $\varphi_{in} = \varphi\left(\|X^a - X^m\|\right)$, $a = 1, 2, 3, ..., m$,$m = 1, 2, 3, ..., m$, then, the above linear equation system becomes an $M \times M$ matrix, as shown in equation (9):

$$\begin{bmatrix} \phi_{11} & \phi_{12} & \cdots & \phi_{1m} \\ \phi_{21} & \phi_{22} & \cdots & \phi_{2m} \\ \cdots & \cdots & \cdots & \cdots \\ \phi_{m1} & \phi_{m2} & \cdots & \phi_{mm} \end{bmatrix} \begin{bmatrix} \eta_1 \\ \eta_2 \\ \cdots \\ \eta_m \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \cdots \\ y_m \end{bmatrix} \tag{9}$$

We denote the $M \times M$-order matrix whose element is φ_{in} in the above equation by X, and denote the following two vectors by V and Y respectively, then the above equation can be written in the form of a vector, as shown in equation (10):

$$XV = Y \tag{10}$$

In the above equation, X is known as an interpolation matrix. According to the rules of linear algebra, if X is an invertible matrix, then we can obtain the coefficient vector w from the above equation.

Michelli's theorem: For a class of functions, if $X^1, X^2, X^3, \ldots, X^M$ are not the same, then a matrix of order $M \times M$ is invertible. In radial basis functions, a large number of functions satisfy this theorem. Common radial basis functions include the following three categories:

Gauss function:

$$\phi(r) = \exp\left(-\frac{r^2}{2\delta^2}\right) \tag{11}$$

Invert the sigmoid function:

$$\phi(r) = \frac{1}{1 + \exp\left(\frac{r^2}{\delta^2}\right)} \tag{12}$$

Inverse multiple quadratic function:

$$\phi(r) = \frac{1}{\left(r^2 + \delta^2\right)^2} \tag{13}$$

where δ is the expansion constant or width of the basis function.

2) The principle and characteristics of regularized RBFNN

The structure of the regularized RBFNN is shown in Figure 5:

The regularized RBFNN model has the following characteristics:

The method of approximating permutation neural networks is general, since there are several hidden nodes, and arbitrary multivariate functions can be approximated with arbitrary precision required.

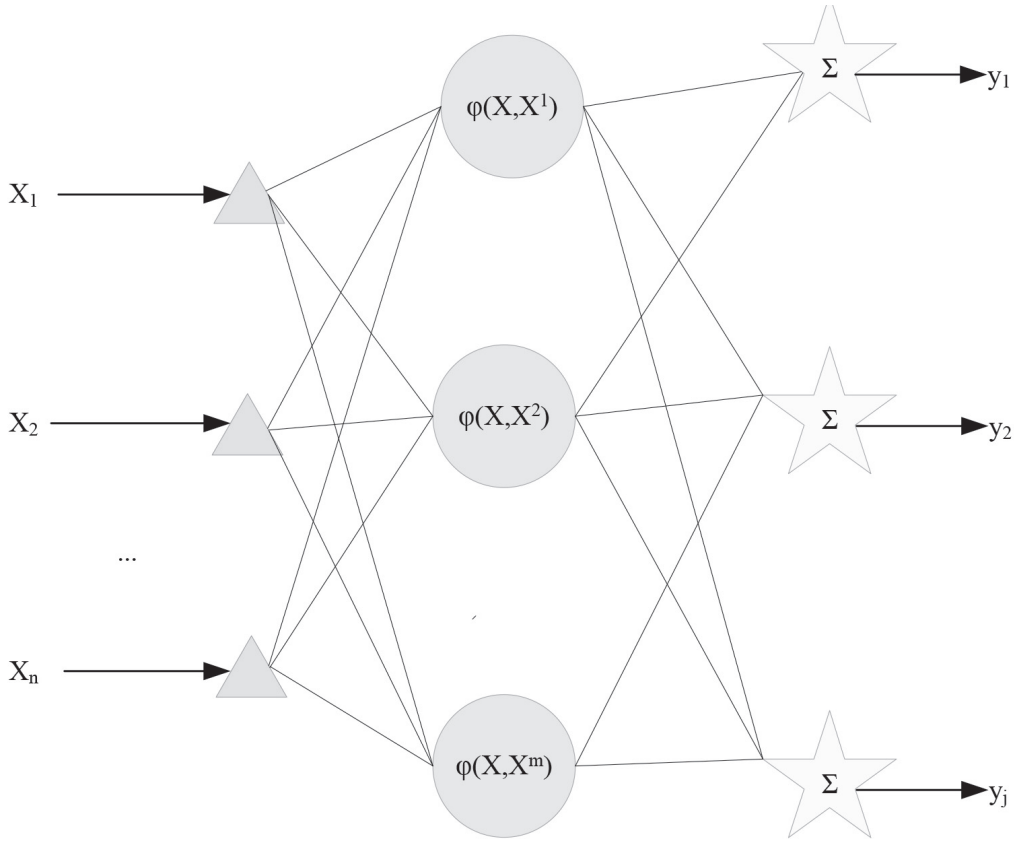


Figure 5 Regularized neural network structure.

The ability to approach is very good. For any unknown nonlinear function, a regularized neural network can always determine a set of weights to approximate the function.

The solution obtained by the regularized neural network model can meet both the approximation accuracy of the sample and the smoothness of the approximation curve.

3) Learning algorithm of regularized neural network

The most important feature of a regularized neural network is that the number of samples is equal to the number of hidden nodes. The sample itself is the center of the basis function, so the regularized neural network only needs to consider the expansion constant (or width) and the weight of the output joint when outputting.

In theory, the expansion constant (or width) of the radial basis function is determined from the distribution of the sample itself. To avoid the radial basis function being too sharp or too flat, we usually choose a method such as Equation (14) to set all the expansion constants (or widths) of the uniform adjustment:

$$\delta = \frac{c_{\max}}{\sqrt{2q}} \tag{14}$$

Among them, $c_{\max}$ represents the maximum distance between samples and samples, and q is the total number of samples.

The weight of the output layer is generally determined by LMS (Least Mean Square algorithm), the equation for which is:

$$\Delta V_p = \mu(c_p - V_p^T\theta)\theta \tag{15}$$

The components of ΔV_p are expressed as equation:

$$\Delta\varepsilon_{bp} = \mu(c_p - V_p^T\theta)\phi_b, b = 0, 1, 2, ..., m;$$
$$p = 1, 2, ..., i \tag{16}$$

Weights can be initialized to any value.

2.3 Concept of Function Approximation

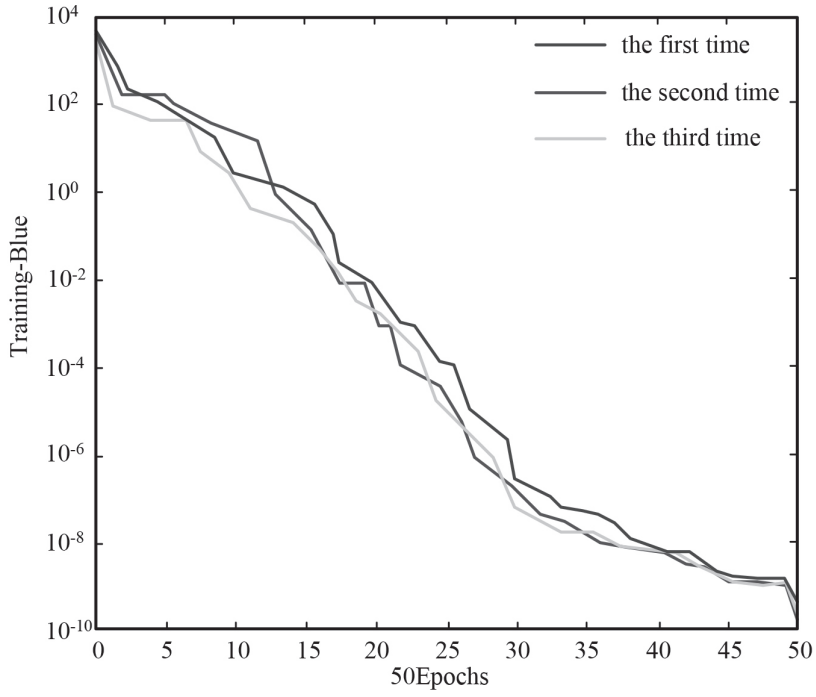
Approximate replacement of $f(z)$ with a simple $g(z)$ function is one of the most fundamental concepts and methods in computational mathematics. Approximate substitution is also known as approximation. The $f(z)$ function is called an approximation function, and $g(z)$ is called an approximation function. The difference between the two is expressed as equation (17):

$$P(z) = f(z) - g(z) \tag{17}$$

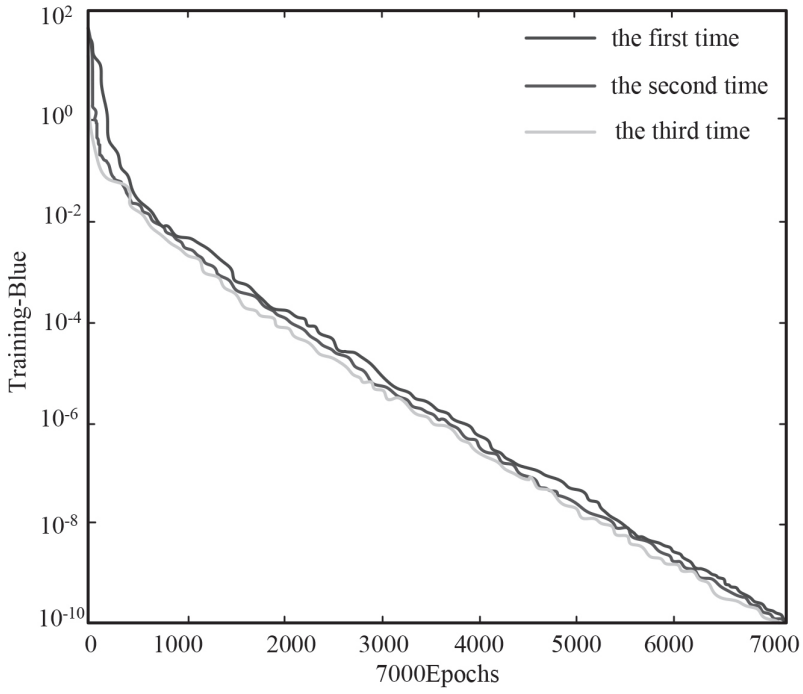
Equation (17) is called the error or remainder of the approximation.

The problem to be solved with function approximation is how to find an approximate relation with a relatively small amount of computation below a given precision to find another type of function $g(z)$ function class B, $J(\subset I)$, which is easy to compute and simpler. Therefore, the difference between $g(z)$ and $f(z)$ somehow reaches a minimum.

In general, the most common function I can be a continuous function in the space $[I, j]$. The most common classes of J-functions can be rational fractions, trigonometric polynomials, and algebraic polynomial functions.



(a) RBFNN training graph



(b) BPNN training graph

Figure 6 Two network training diagrams.

3. RBFNN FUNCTION APPROXIMATION SIMULATION EXPERIMENT

3.1 Approximation of Unary Functions by Different Networks

(1) Comparison of the convergence speed of the two networks

Figure 6(a), (b) are the training process of RBFNN and BPNN, respectively.

As can be seen from Figure 6, both networks achieve relatively good accuracy, but RBFNN achieves slightly higher accuracy. RBFNN can perform only 50 steps of training, which is much faster than BPNN.

(2) Comparison of the approximation ability of the training samples

The approximation errors of the two networks to the training samples are shown in Figure 7.

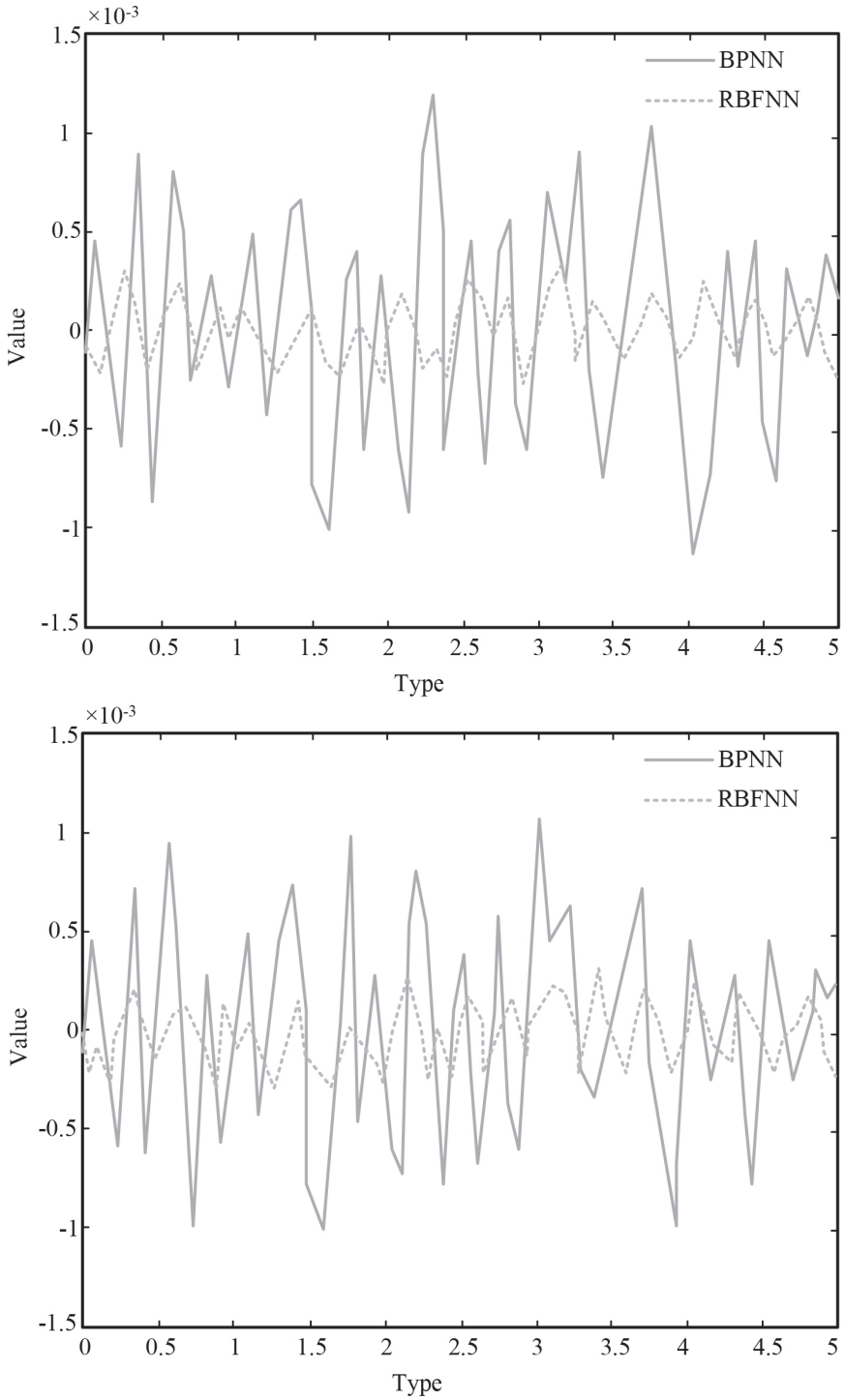


Figure 7 Comparison of simulation errors of the two networks for training samples.

As can be seen from the two comparisons in Figure 7, the approximation accuracy of RBFNN and BPNN is relatively high (order of magnitude 10–3). Therefore, it can be considered that the two almost reach a complete approximation of the training samples. However, the simulation errors of RBFNN are almost all around 0, while the errors of BPNN have large oscillations around 0. Therefore, we believe that the accuracy of the RBFNN method in the training samples is slightly higher than that of the BPNN method.

(3) Comparison of approximation capabilities for non-training samples

The approximation errors of the two networks for non-training samples are compared in Tables 1 and 2.

From Table 1 and Table 2, it can be seen that the ability to approximate RBFNN in non-training samples is significantly higher than that of BPNN. For non-training samples, RBFNN almost achieves a complete approximation, and the larger error does not exceed 0.1. Compared with RBFNN, BPNN has a larger error, and the error fluctuates greatly, and its maximum error is almost 0.5 (0.4653).

(4) Comparison of overall approximation capabilities

The overall approximation situation of the RBFNN and BPNN to the target functions is shown in Figure 8.

Table 1 Partial simulation numerical results of RBFNN.

sample serial number	exact value	RBFNN	
		Simulation value	Simulation error
1	4.4915	4.4903	0.0012
2	4.5124	4.5116	0.0008
3	4.5661	4.5741	−0.008
4	4.7015	4.7021	−0.0006
5	4.9541	4.9535	0.0006
6	5.3226	5.3215	0.0011
7	5.8304	5.8314	−0.001
8	6.4751	6.4724	0.0027
9	7.2163	7.2135	0.0028
10	8.0335	8.0353	−0.0018
11	8.8714	8.8721	−0.0007
12	9.7042	9.7028	0.0014
13	10.4652	10.4637	0.0015
14	11.1013	11.1030	−0.0017
15	11.6043	11.6036	0.0007

Table 2 Partial simulation numerical results of BPNN.

sample serial number	exact value	BPNN	
		simulation value	simulation error
1	4.4915	4.3142	0.1773
2	4.5124	4.5201	−0.0177
3	4.5661	4.5542	0.0119
4	4.7015	4.6893	0.0122
5	4.9541	5.0242	−0.0701
6	5.3226	5.3125	0.1101
7	5.8304	5.3651	0.4653
8	6.4751	6.5043	−0.0292
9	7.2163	7.0241	0.1922
10	8.0335	8.3250	−0.2915
11	8.8714	8.6260	0.2454
12	9.7042	9.6316	0.0726
13	10.4652	10.2940	0.1712
14	11.1013	11.2914	−0.1901
15	11.6043	11.4310	0.1733

A close examination of Figure 8 shows that RBFNN basically matches the function to be fully approximated, while BPNN does not match many function segments. Therefore, overall, RBFNN also significantly outperforms BPNN.

(5) Comparison of network structures

When using Matlab for simulation, the RBFNN generated by newrb will automatically adjust the network structure. The BPNN created by newff needs to adjust the network structure by itself until the network reaches the best state. However, the results of running the program show that the number of hidden nodes of BPNN does not vary greatly with the number of training samples. The number of hidden nodes in RBFNN is the same as the number of training samples. Table 3 shows the number of hidden layer neurons required for RBFNN and BPNN with different numbers of training samples.

It can be seen from Table 3 that under the same number of samples, the number of neurons in the hidden layer of RBFNN is much larger than that of BPNN. Therefore, when the number of training samples is too large, the structure of RBFNN will be more complicated than that of BPNN.

(6) The conclusion of the simulation approximation of the unary function

In summary, we can draw the following conclusions:

1) Both RBFNN and BPNN have high approximation accuracy for unary function training samples, which can achieve almost complete approximation. However, the generalization ability of RBFNN is better than BPNN in many aspects;

2) The design of RBFNN is very convenient. The network can automatically add neurons until the accuracy requirements are met, and the approximation accuracy is significantly higher than that of BPNN.

3) When there are too many training samples, the structure of RBFNN will be too large, while BPNN will not have this problem.

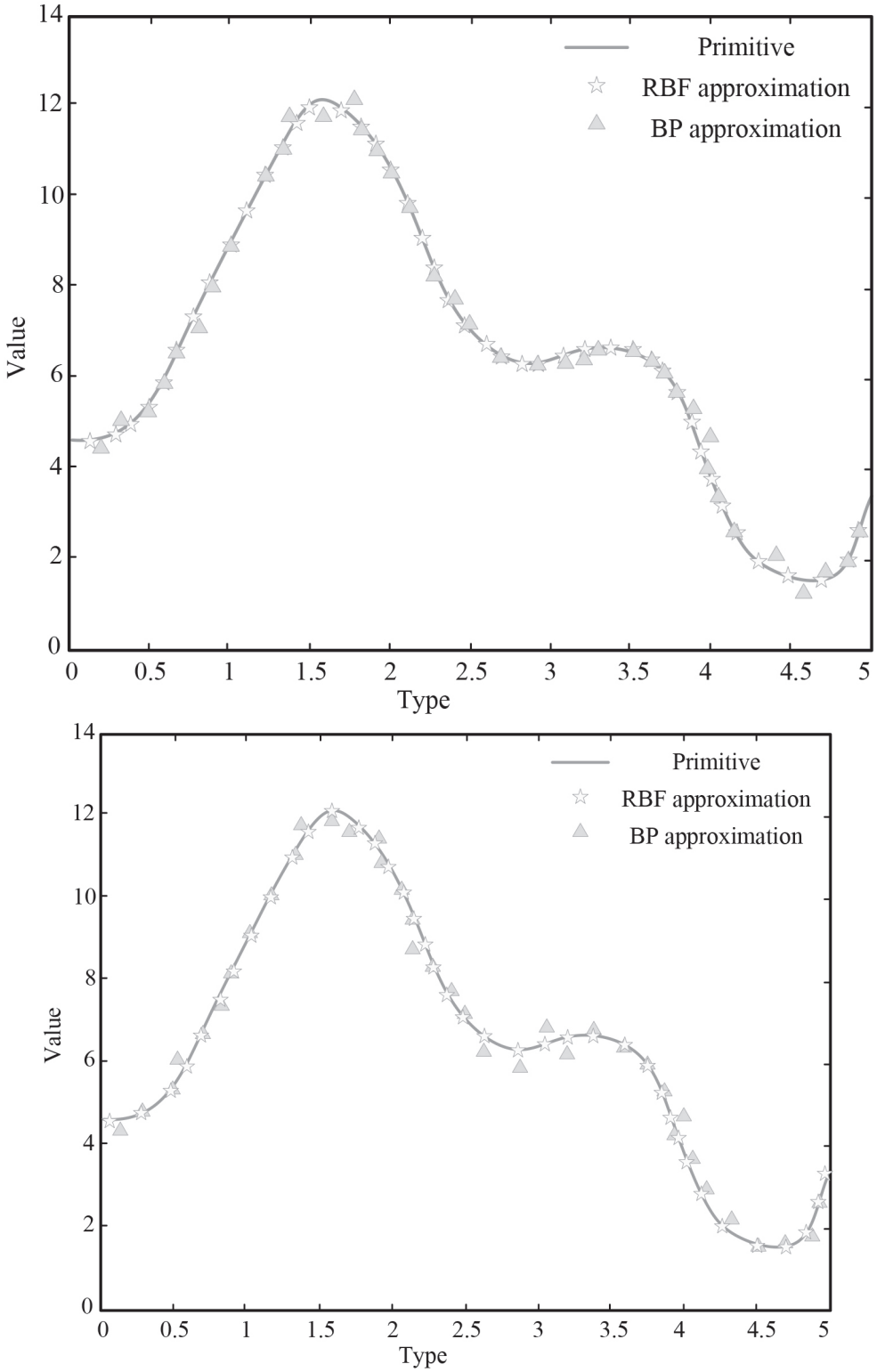


Figure 8 Overall approximation rendering of the two networks.

3.2 Approximation of the Sine Function

(1) Approximation of a sine function with specified accuracy

When the error target of the specified function approximation is 0.001, the approximation effect of each network sine function is shown in Figure 9 below.

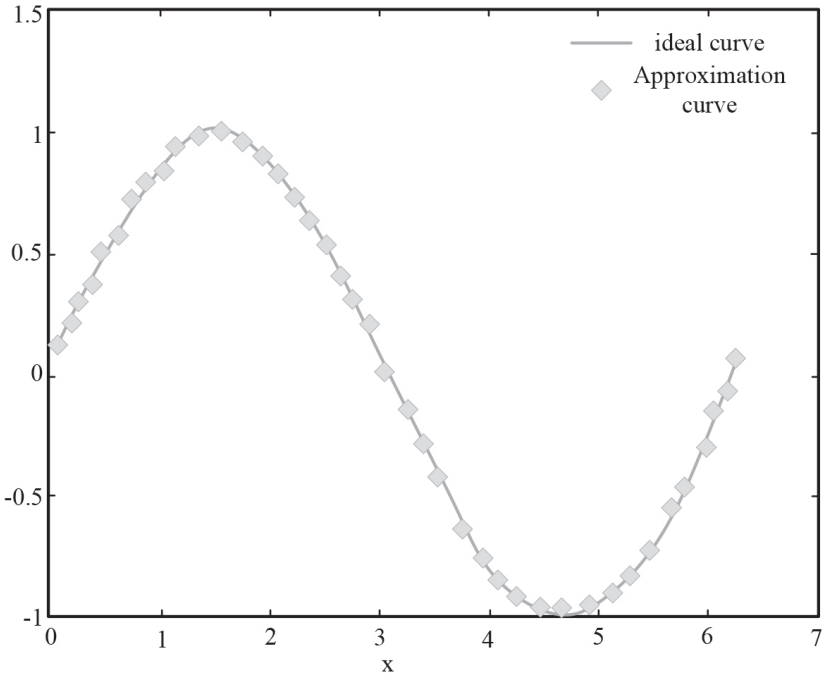
It can be seen from Figure 9 that BPNN and RBFNN require less learning time and fewer hidden neurons when the sine function obtains good approximation results. It can quickly achieve the required approximation accuracy.

(2) Approximation of the sine function of increasing frequency

When the error target of the function approximation is defined as 0.001, the effect of the sine function method on the network frequency approximation is as shown in Figure 10.

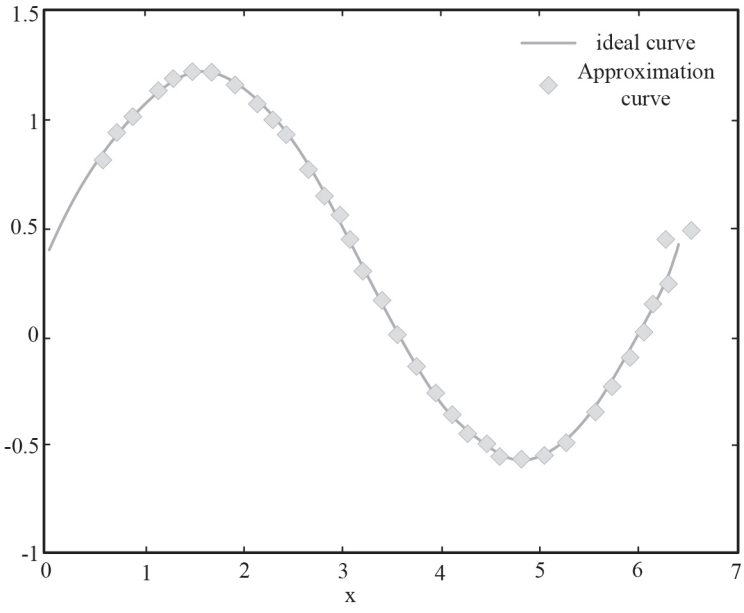
Table 3 Comparison of the number of neurons in the hidden layer of the two networks.

Number of samples	Number of RBF neurons	Number of BP neurons
22	22	14
36	36	25
42	42	21
68	68	23
89	89	36
100	100	42



（a）BPNN

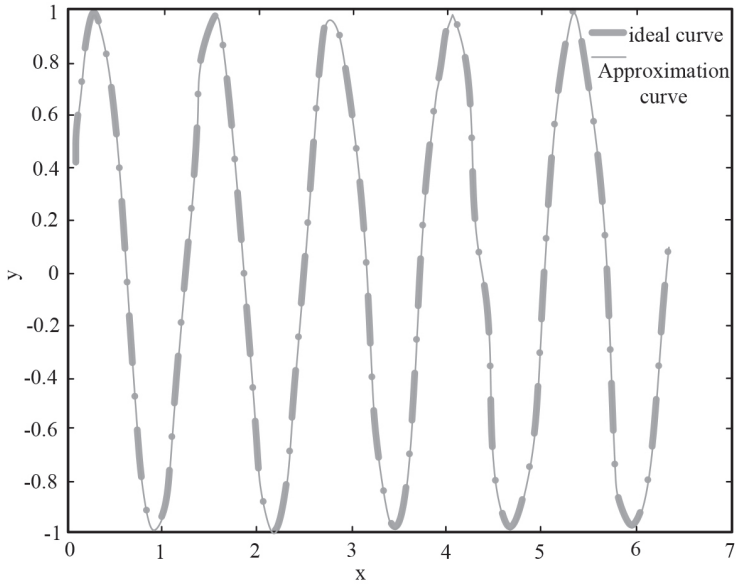
(A)



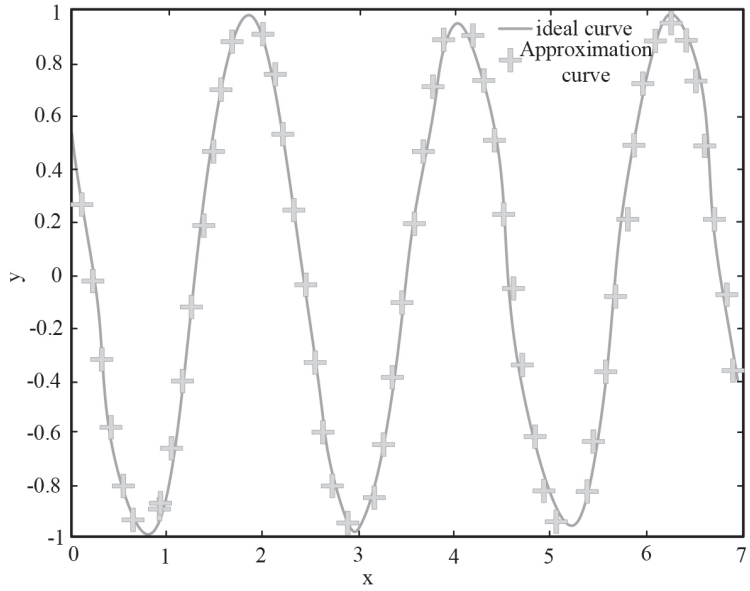
（b）RBFNN

(B)

Figure 9 Sine function approximation effect.



(a) BPNN effect



(b) RBFNN effect

Figure 10 The approximation effect of a sine function with increasing frequency.

Figure 10 shows that BPNN and RBFNN require less learning time and fewer hidden neurons when higher frequency sine functions can obtain better approximation results. It can quickly achieve the required approximation accuracy. However, when approaching BPNN, there is some distortion in the peak and valley positions. RBFNN seems to be more suitable when approximating continuous functions with higher frequency.

4. CONCLUSION

In this paper, in-depth research was conducted on RBFNN and RBFNN was applied to function approximation. Results show that RBFNN can consistently approximate any continuous function operation approximating a closed space with arbitrary precision without training. Then, the RBFNN and BPNN functions were approximated with the Matlab simulation tool, and the approximation results of the two methods were compared. The results show that RBFNN has a faster training speed and can achieve better approximation results. Some of the neural networks used in this paper are traditional and limited in number, and newer and more diverse neural networks should be studied. In conclusion, RBFNN is a huge improvement compared to BPNN. The RBFNN has made significant contributions to fields such as image processing, signal processing, and predictive modeling, and is expected to further advance these areas of science.

ACKNOWLEDGEMENTS

This work was supported by.
Project 1: Provincial teaching and research project for universities in Hubei province (2022450);

Project 2: School-level Research Platform of Jingchu University of Technology: Data Analysis Science Laboratory;
Project 3: B202215 Key Scientific Research Project of Jingchu University of Technology;
Project 4: Research project of Jingchu University of Technology (YB202301,JX2022-007,QN202419);
Project 5: Educational science planning project of Jingmen City (JMG2022004);
Project 6: China Association for Educational Technology Project (XJJ202205022).

REFERENCES

1. Chen Q, Yi Z. Strong approximation over function fields[J]. Journal of Algebraic Geometry, 2018, 27(4):703–725.
2. Andras P. High-Dimensional Function Approximation with Neural Networks for Large Volumes of Data [J]. IEEE Transactions on Neural Networks & Learning Systems, 2018, 29(2):500–508.
3. Ziemke T, Alegre L N, Bazzan A. Reinforcement learning vs. rule-based adaptive traffic signal control: A Fourier basis linear function approximation for traffic signal control [J]. Ai Communications, 2021, 34(2):1–15.
4. Sonker S, Munjal A. Approximation of the function $f \in Lip(\alpha, p)$ using infinite matrices of Cesáro submethod [J]. Nonlinear Studies, 2017, 24(1):113–125.
5. Nguyen V, Cai J, Wei L, Chu J. Neural Networks Probability-Based PWL Sigmoid Function Approximation [J]. IEICE Transactions on Information and Systems, 2020, E103.D(9): 2023–2026.
6. Ravichandran V, Li C, Banagozar A, Yang J J, Xia Q. Artificial neural networks based on memristive devices [J]. Science China Information Sciences, 2018, 61(6):1–14.
7. Alanis, Alma Y. Electricity Prices Forecasting using Artificial Neural Networks [J]. IEEE Latin America Transactions, 2018, 16(1):105–111.
8. Isik E, Inalli M. Artificial neural networks and adaptive neuro-fuzzy inference systems approaches to forecast the meteorological data for HVAC: The case of cities for Turkey [J]. Energy, 2018, 154(JUL.1):7–16.
9. Li S, Fairbank M, Johnson C, Wunsch D C, Alonso E, Proao J L. Artificial Neural Networks for Control of a Grid-Connected Rectifier/Inverter Under Disturbance, Dynamic and Power Converter Switching Conditions [J]. IEEE Transactions on Neural Networks and Learning Systems, 2017, 25(4): 738–750.
10. Cascardi A, Micelli F, Aiello M A. An artificial neural networks model for the prediction of the compressive strength of FRP-confined concrete circular columns [J]. Engineering Structures, 2017, 140(JUN.1):199–208.
11. Canziani G, Ferrati R, Marinelli C, Dukatz F. Artificial neural networks and remote sensing in the analysis of the highly variable Pampean shallow lakes [J]. Mathematical Biosciences & Engineering Mbe, 2017, 5(4):691–711.
12. B. Xiao, "Reactive Power Optimization Control of Power Grid utilizing the Improved RBF Artificial Neural Network", Engineering Intelligent Systems, vol. 27 no. 4, pp. 193–200, 2019.
13. Giovanis D G, Papaioannou I, Straub D, Papadopoulos V. Bayesian updating with subset simulation using artificial neural networks [J]. Computer Methods in Applied Mechanics & Engineering, 2017, 319(JUN.1):124–145.
14. Xu, Z., Jain, D.K., Neelakandan, S. et al. Hunger games search optimization with deep learning model for sustainable supply chain management. Discov Internet Things 3, 10 (2023).
15. Tarawneh B. Predicting standard penetration test N-value from cone penetration test data using artificial neural networks [J]. Geoence Frontiers, 2017, 8(1):199–204.
16. Dimitrijevic M , Andrejevic-Stosovic M, Milojkovic J, Litovski V. Implementation of artificial neural networks based AI concepts to the smart grid [J]. Facta universitatis - series: Electronics and Energetics, 2017, 27(27):411–424.
17. Safa M, Samarasinghe S, Nejat M. Prediction of Wheat Production Using Artificial Neural Networks and Investigating Indirect Factors Affecting It: Case Study in Canterbury Province, New Zealand [J]. Journal of Agricultural Science & Technology, 2018, 17(4):791–803.
18. Arora, A. S., Saboia, L., Arora, A., & McIntyre, J. R. (2025). Human-Centric Versus State-Driven: A Comparative Analysis of the European Union's and China's Artificial Intelligence Governance Using Risk Management. International Journal of Intelligent Information Technologies (IJIIT), 21(1), 1–13.
19. Mhaskar H N Poggio T. Function approximation by deep networks [J]. Communications on Pure & Applied Analysis, 2020, 19(8):4085–4095.
20. Q. Wang, "Plant Configuration Method of Landscape Architecture Based on Neural Network", Engineering Intelligent Systems, vol. 31 no. 2, pp. 81–91, 2023.
21. Tin P T, Tran M, Le A V, Dung N Q, Trang T T. Real interpolation method for transfer function approximation of distributed parameter system [J]. TELKOMNIKA Indonesian Journal of Electrical Engineering, 2019, 17(4): 1941–1947.
22. Sun Z. A meshless symplectic method for two-dimensional nonlinear Schrodinger equations based on radial basis function approximation [J]. Engineering analysis with boundary elements, 2019, 104(JUL.):1–7.