Computer Exercise: MLP

Yuexin Bian

***Abstract*—In this paper, we contrive to understand the BP algorithm that masters the network structure of MLP and network correction through learning and testing the fixed sample set. Using Python language, the author extracts training samples and test samples from given functions, draws error curves, and simultaneously examines the effects of different implicit units and learning rates on the consequence of training and testing. In this process, we learn the algorithm principle of BP network, and change the effect of final training and test by changing and adjusting the corresponding parameters, as a result master the influence of its parameters on the network structure. The article aims to restore the entire learning process in order to thoroughly understand the underlying mechanism.**

***Index Terms*—BP algorithm, MLP, train, test, parameter, influence**

I. INTRODUCTION

BP (back propagation) neural network is a concept proposed by scientists led by Rumelhart and McClelland in 1986. It is a multi-layer feedforward neural network trained according to the error back propagation algorithm. It is the most widely used neural network. In the history of artificial neural networks, the Multilayer Perceptron (MLP) network has played an important role in the development of artificial neural networks. It is also considered to be a truly usable artificial neural network model. There has been a rush to study artificial neural networks. The single layer sensing network (M-P model) is the initial neural network, which has the advantages of clear model, simple structure and small calculation. However, with the deepening of the research work, people find that it still has some shortcomings, such as the inability to deal with nonlinear problems. Even if the function of the calculation unit does not use the valve function and other more complicated nonlinear functions, it can only solve the linear separate problem. Some basic functions are not implemented, which limits its application. The only way to enhance the classification and recognition capabilities of the network and solve the nonlinear problem is to use a multi-layer feed-forward network, which adds an implicit layer between the input layer and the output layer. Form a multi-layer feed-forward sensor network.

In the mid-1980s, David Runelhart. Geoffrey Hinton and Ronald W-llians, David Parker and others independently discovered Error Back Propagation Training (BP), which solves the problem of multi-layer neural network implicit layer connection weight learning and gives mathematics a complete derivation. A multi-layer feed-forward network using this algorithm for error correction is called a BP network.

BP neural network has arbitrarily complex pattern classification ability and excellent multi-dimensional function mapping ability, which solves the exclusive OR (XOR) and some other problems that cannot be solved by simple perceptron. Structurally speaking, the BP network has an input layer, a hidden layer and an output layer; in essence, the BP algorithm uses the square of the network error as the objective function and uses the gradient descent method to calculate the minimum value of the objective function.

The author completes the writing of the BP network by using the Python language, and trains and tests the fixed sample set, and masters many tasks such as parameter adjustment, BP network understanding, and mechanism familiarity.

II. PRINCIPLE AND METHOD

Artificial neural network does not need to determine the mathematical equation of the mapping relationship between input and output in advance, and only learns some rules through its own training, and obtains the result closest to the expected output value when given the input value.

The following paragraph focuses on the principles and the author's specific practices.

A. Principle and Structure

The core of the artificial neural network to achieve its function is the algorithm. BP neural network is a multi-layer feedforward network trained by error back propagation (referred to as error back propagation). Its algorithm is called BP algorithm. Its basic idea is gradient descent method, which uses gradient search technology to make the network The error mean square error between the actual output value

and the desired output value is minimal.

The basic BP algorithm includes two processes of forward propagation of signals and back propagation of errors. That is, the calculation of the error output is performed in the direction from the input to the output, and the adjustment weight and the threshold are performed from the output to the input. In the case of forward propagation, the input signal acts on the output node through the hidden layer and undergoes nonlinear transformation to generate an output signal. If the actual output does not match the expected output, the error propagates back into the error propagation process. Error back propagation is to pass the output error back to the input layer through the hidden layer, and distribute the error to all the units in each layer, so as to adjust the error value of each unit as the basis for adjusting the weight of each unit. By adjusting the connection strength between the input node and the hidden layer node and the connection strength and threshold of the hidden layer node and the output node, the error is decreased along the gradient direction, and after repeated learning training, the network parameters (weights and thresholds corresponding to the minimum error) are determined.), the training will stop. At this time, the trained neural network can process the non-linear conversion information with the smallest output error for the input information of similar samples.

BP network adds several layers (one or more layers) of neurons between the input layer and the output layer. These neurons are called hidden cells. They are not directly related to the outside world, but their state changes can affect the input. The relationship between the output and the output can have several nodes per layer.

B. Method

A multilayer perceptron using a back propagation algorithm is also known as a BP neural network. The BP algorithm is an iterative algorithm. Its basic idea is:

(1) Calculate the state and activation value of each layer until the last layer (that is, the signal is forward-propagating);

(2) Calculate the error of each layer. The error calculation process is advanced from the last layer (this is the origin of the name of the back-propagation algorithm);

(3) the parameters are updated (the goal is that the error becomes smaller). Iterate through the first two steps until the stopping criterion is met (for example, the difference between the errors of two adjacent iterations is small).

The calculation process of the BP neural network consists of a forward calculation process and a reverse calculation process. In the forward propagation process, the input mode is processed layer by layer from the input layer through the hidden cell layer, and turns to the output layer. The state of each layer of neurons only affects the state of the next layer of neurons. If the desired output cannot be obtained at the output layer, then the backward propagation is performed, and the error signal is returned along the original connection path, and the error signal is minimized by modifying the weight of each neuron.

The author's completion ideas are also composed of the above steps:

(1) Initialization parameters

(2) Information forward propagation

(3) Error back propagation

(4) After a certain number of trainings, the output is obtained.

C. Initialization parameters

Given the training samples, you can know the dimensions of the input and output. For example, the author uses a three-layer perceptron (ie, a multi-layer perceptron with only one hidden layer) as an example to initialize the weight vector from the input layer to the hidden layer, the weight vector from the hidden layer to the output layer, and the corresponding bias.

D. Information forward propagation

Taking the second function as an example, set a hidden layer. The number of hidden units in the hidden layer is 3, and the input quantity is 2, and the output is 1.

$$f_2(x_1, x_2) = \frac{\sin(x_1)}{x_1}\frac{\sin(x_2)}{x_2}$$

The second layer (hidden layer) neuron state and activation value of the neural network can be calculated by the following:

$$\begin{aligned}
z_1^{(2)} &= w_{11}^{(2)}x_1 + w_{12}^{(2)}x_2 + b_1^{(2)} \\
z_2^{(2)} &= w_{21}^{(2)}x_1 + w_{22}^{(2)}x_2 + b_2^{(2)} \\
z_3^{(2)} &= w_{31}^{(2)}x_1 + w_{32}^{(2)}x_2 + b_3^{(2)} \\
a_1^{(2)} &= f(z_1^{(2)}) \\
a_2^{(2)} &= f(z_2^{(2)}) \\
a_3^{(2)} &= f(z_3^{(2)})
\end{aligned}$$

Similarly, the state and activation values of the output layer neurons can be calculated by the following calculations:

$$\begin{aligned}
z_1^{(3)} &= w_{11}^{(3)}a_1^{(2)} + w_{12}^{(3)}a_2^{(2)} + w_{13}^{(3)}a_3^{(2)} + b_1^{(3)} \\
a_1^{(3)} &= f(z_1^{(3)})
\end{aligned}$$

Therefore, the forward transfer process of feed-forward neural network information is:

$$x = a^{(1)} \to z^{(2)} \to \ldots \to z^{(L)} \to a^{(L)} = y$$

E. Error back propagation

We use the root mean square error as the training error. After the gradient is derived, the weight and bias values of the hidden layer and the output layer are corrected after each forward propagation. After many corrections, the experimental error will decrease to the lowest.

After a rigorous calculation of the derivative, the error back propagation algorithm can be summarized into the

following four formulas. Since the author uses the weight value of each layer mentioned above as the array in the algorithm, the attached core formula is attached in matrix vector form.

$$\boldsymbol{\delta}^{(L)} = -(\boldsymbol{y} - \boldsymbol{a}^{(L)}) \odot f'(\boldsymbol{z}^{(L)}) \quad \text{(BP-1)}$$

$$\boldsymbol{\delta}^{(l)} = \left((W^{(l+1)})^{\mathsf{T}}\boldsymbol{\delta}^{(l+1)}\right) \odot f'(\boldsymbol{z}^{(l)}) \quad \text{(BP-2)}$$

$$\nabla_{W^{(l)}} E = \boldsymbol{\delta}^{(l)}(\boldsymbol{a}^{(l-1)})^{\mathsf{T}} \quad \text{(BP-3)}$$

$$\nabla_{b^{(l)}} E = \boldsymbol{\delta}^{l} \quad \text{(BP-4)}$$

F. Iteration

After several rounds of forward propagation reverse update parameter training, the weight values and offset values of each layer have been corrected to some extent, and the training error at this time is reduced to a small extent. At this time, the test sample can be tested to observe the effect of the network obtained by the training and the test error.

This is not an easy task, because the network effects of the training vary with the initial parameters. Be patient and modify the parameters to see how these parameters affect the results.

III. Consequence For Function1

According to the above method, we get the trained network which has good effects.

A. Extraction of training samples and test samples

```
train_x
```

```
array([[ 0.        ],
       [ 0.78539816],
       [ 1.57079633],
       [ 2.35619449],
       [ 3.14159265],
       [ 3.92699082],
       [ 4.71238898],
       [ 5.49778714],
       [ 6.28318531]])
```

```
train_y
```

```
array([[  0.00000000e+00],
       [  7.07106781e-01],
       [  1.00000000e+00],
       [  7.07106781e-01],
       [  1.22464680e-16],
       [ -7.07106781e-01],
       [ -1.00000000e+00],
       [ -7.07106781e-01],
       [ -2.44929360e-16]])
```

Fig.1. The figure is a training sample extracted at equal intervals, and the reference value of the function value is used to calculate the error. Since there are many test samples, the same method is used, so it will not be described here.

B. Error performance indicator curve

The basic parameters are provided, and the curve of the error performance index with the training algebra is shown in the figure. It can be seen that as the number of training increases, the error gradually decreases.

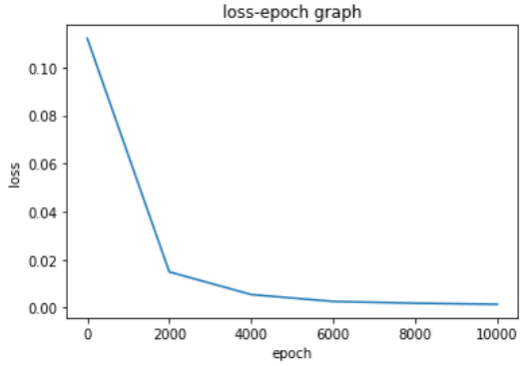


Fig.2.

After 10,000 iterations, the initial error is 0.123677, and the final error is 0.001369. It can be seen that the error is constantly decreasing.

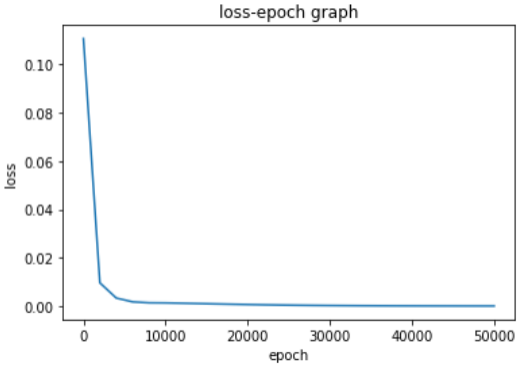


Fig.3.

After 50,000 iterations, the initial error is 0.110675, and the final error is 0.000107. It can be seen that the error is smaller than after 10,000 iterations.

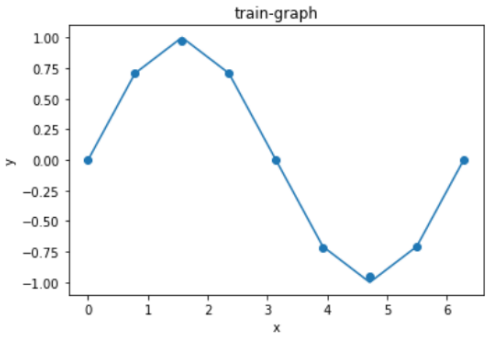


Fig.4.graph of train samples

Taking 10,000 training times as an example, after repeated training, the nine points of the training sample curve are well fitted, and the final training error is 0.001369.

The figure below shows the fitting curve of the test sample. It can be seen that the fitting effect is good, and the final test error is 0.00296.

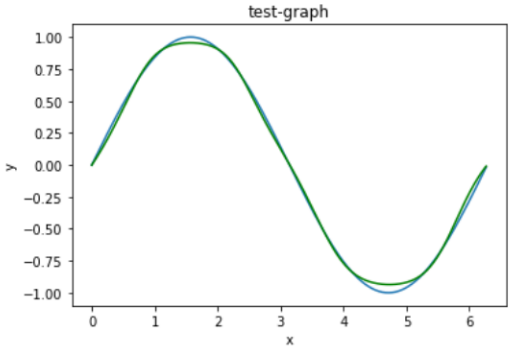


Fig.5.graph of test samples

C. Parameter influence

TABLE I
THE IMPACT OF LEARNING RATE ON TRAINING AND TESTING

Learning-rate	Train-error	Final training error as a percentage of initial training error	Test-error
0.01	0.051999	46.680%	0.12053
0.1	0.001296	1.165%	0.00625
0.2	0.000767	0.667%	0.00421
0.3	0.000955	0.855%	0.00555
0.4	0.001102	0.979%	0.00578
0.5	0.000275	0.253%	0.00157
0.6	0.000216	0.197%	0.00093
0.7	0.000149	0.132%	0.00282
0.8	0.000885	0.793%	0.00511
0.9	0.001549	1.366%	0.00957
0.95	0.001989	1.786%	0.00894

Control the following parameters unchanged:
(1) The number of iterations is 20,000 times
(2) The number of hidden units is 5
(3) The initial value of the offset value is 0, and the weight matrix is randomly generated by the computer.

TABLE II
THE IMPACT OF NUMBER OF HIDDEN UNITS ON TRAINING AND TESTING

Number of hidden units	Train-error	Final training error as a percentage of initial training error	Test-error
1	0.030355	27.335%	0.07362
2	0.016368	14.685%	0.04187
3	0.001303	1.175%	0.00883
5	0.000180	0.161%	0.00096
6	0.000158	0.137%	0.00338
8	0.016269	14.660%	0.04083
10	0.045620	39.984%	0.09538

Control the following parameters unchanged:
(1) The number of iterations is 20,000 times
(2) Leaning rate equals 0.7
(3) The initial value of the offset value is 0, and the weight matrix is randomly generated by the computer.

IV. CONSEQUENCE FOR FUNCTION2

The difference between the second function and the first function is that the number of feature quantities input is 2, and the expression is in three dimensions to see the specific fitting effect.

A. Extraction of training samples and test samples

```
train_x
```

```
array([[-10.        , -10.        ],
       [-10.        ,  -8.18181818],
       [-10.        ,  -6.36363636],
       [-10.        ,  -4.54545455],
       [-10.        ,  -2.72727273],
       [-10.        ,  -0.90909091],
       [-10.        ,   0.90909091],
       [-10.        ,   2.72727273],
       [-10.        ,   4.54545455],
       [-10.        ,   6.36363636],
       [-10.        ,   8.18181818],
       [ -8.18181818, -10.        ],
       [ -8.18181818,  -8.18181818],
       [ -8.18181818,  -6.36363636],
       [ -8.18181818,  -4.54545455],
       [ -8.18181818,  -2.72727273],
       [ -8.18181818,  -0.90909091],
       [ -8.18181818,   0.90909091],
       [ -8.18181818,   2.72727273],
       [ -8.18181818,   4.54545455],
```

```
train_z
```

```
array([[  2.95958969e-03],
       [ -6.29502120e-03],
       [ -6.87026616e-04],
       [  1.18020881e-02],
       [ -8.03019116e-03],
       [ -4.72123286e-02],
       [ -4.72123286e-02],
       [ -8.03019116e-03],
       [  1.18020881e-02],
```

Fig.6. The figure is a training sample extracted at equal intervals, and the reference value of the function value is used to calculate the error. Because there are more samples, so only the part is shown in the figure.

B. Error performance indicator curve

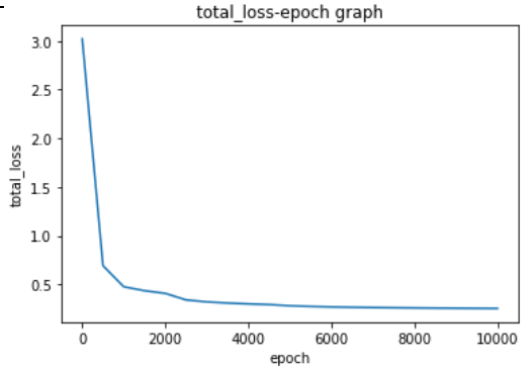


Fig.7

It can be seen that as the number of training increases, the loss curve of this function is also monotonically

decreasing. As the number of times increases, the decreasing curve is slower..

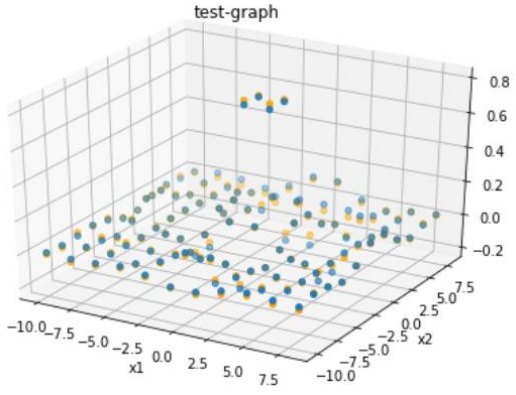


Fig.8.graph of train samples

Taking 10,000 training times as an example, after repeated training, the 141 points of the training sample curve are well fitted, and the final training error is 0.00028.

The figure below shows the fitting curve of the test sample. It can be seen that the fitting effect is good, and the final test error is 0.00212.

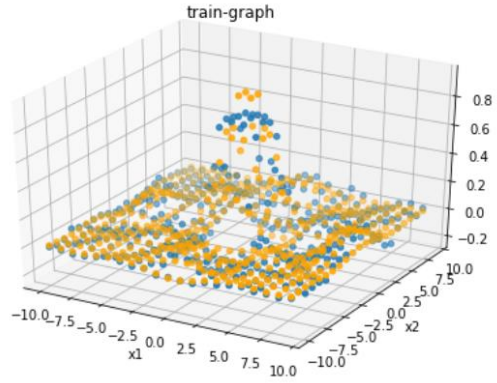


Fig.9.graph of test samples

C. Parameter influence

TABLE I
THE IMPACT OF LEARNING RATE ON TRAINING AND TESTING

Learning-rate	Train-error	Test-error
0.01	0.01312	0.01323
0.1	0.00414	0.00491
0.2	0.00429	0.00610
0.3	0.00307	0.00429
0.4	0.00391	0.00703
0.5	0.00494	0.00804
0.6	0.00450	0.00832
0.7	0.00338	0.00653
0.8	0.00436	0.00667
0.9	0.00503	0.00853
0.95	0.00429	0.00670

Control the following parameters unchanged:
(1) The number of iterations is 10000 times
(2) The number of hidden units is 10
(3) The initial value of the offset value is 0, and the weight matrix is randomly generated by the computer.

TABLE II
THE IMPACT OF NUMBER OF HIDDEN UNITS ON TRAINING AND TESTING

Number of hidden units	Train-error	Test-error
3	0.01216	0.01453
5	0.00790	0.00907
8	0.00426	0.00769
10	0.00307	0.00429
12	0.00304	0.00362
15	0.00148	0.00237
30	0.00073	0.00165
50	0.00040	0.00192

Control the following parameters unchanged:
(1) The number of iterations is 10,000 times
(2) Leaning rate equals 0.3
(3) The initial value of the offset value is 0, and the weight matrix is randomly generated by the computer.

V. ANALYSIS

A. The impact of learning rate on training and testing

From the curve of the error performance index with algebra, the speed of the error decline can be seen. With different learning rates, the speed also varies. As can be seen from the table, different learning rates also have a certain degree of influence on the final training error and test error.

1) Error decline speed

The smaller the learning rate, the slower the rate of loss gradient and the longer the convergence time. Conversely, the greater the learning rate, the greater the rate at which the loss gradient decreases.

2) Training and test error

When the learning rate is too large or too small, it is possible to cross the optimal solution. In particular, when the learning rate is too large, the loss may rise rather than fall. It can be seen from the tables of two given functions that the learning rate has a better training effect in one interval, the learning rate rises and falls, and the training test error becomes larger and the effect is affected.

The optimal learning rate of the two functions is also different, and you need to slowly explore in the process of trying.

The first function learning rate has a greater influence on its training test error. In the same interval the second function have less influence.

B. The influence of the number of hidden units on training and testing

As with the learning rate, the choice of the number of cells in the hidden layer is also a very complicated problem. It does not have an ideal analytical expression.
When the number is too small, the information that the network can obtain to solve the problem is too small; if the

number is too much, the training time will increase, the learning time is too long, and the error is not necessarily optimal.

For the first given function, when the number of hidden units is 6, the training error and the test error are the lowest, the number of hidden units increases, and the error increases, but the increase is slower and the number of hidden units decreases. The error increases quickly and the training effect is very poor.

For example, when the number of hidden units is 3, training and test fit is poor indeed.

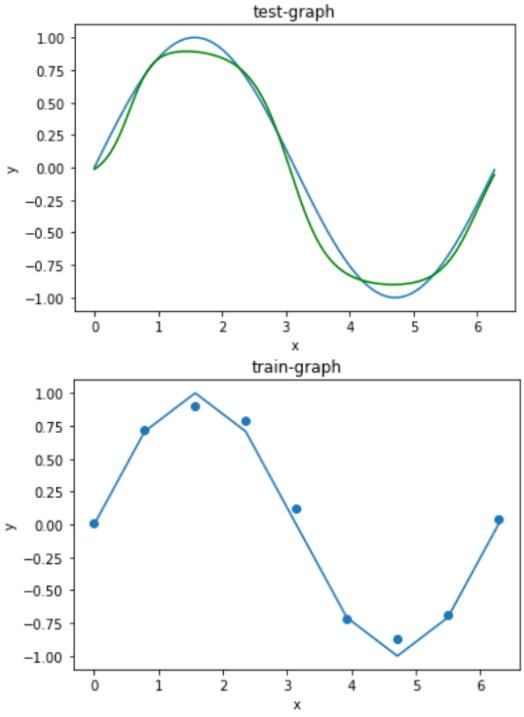
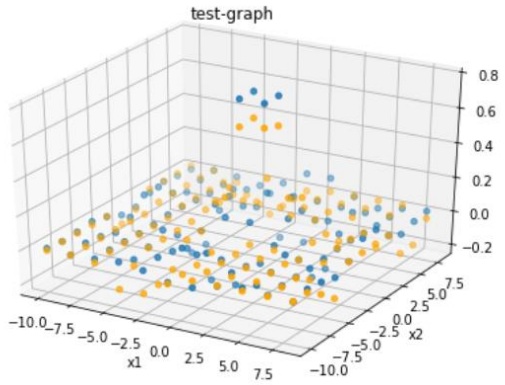


Fig.10.graph of test&train samples

For the second given function, in the several data that have been tested, as the number of hidden units increases, the error is decreasing.

Also, using the same number of hidden layer units as the first function, the effect is very bad. The reason is that the number of inputs to the second function is 2, which is more complicated than the first one.



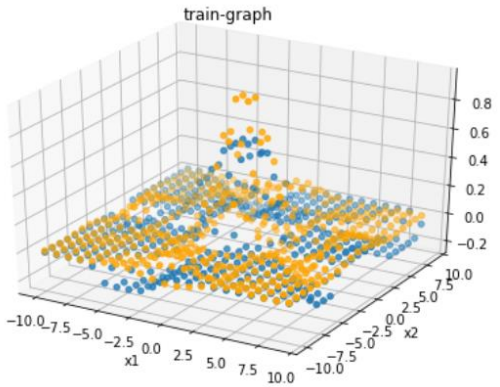


Fig.11.graph of test&train samples

VI. Conclusion

In order to complete the assignment, I reviewed the corresponding papers and materials of the BP network algorithm to understand the structure of the BP network, the definitions of forward propagation, and the derivation of the back propagation gradient to reduce loss. After the code is officially written, many problems are encountered in the process of writing code, such as dot multiplication and dot between matrix vectors. Adjustment parameters are also essential. Only after constant adjustment, training, and review of the results can we have a certain degree of understanding of the impact of the parameters on the final result.

In conclusion, we understand that there are many definitions of loss, and the author chose the root mean square error in this assignment. BP network as the simplest network, in the first step of the entry neural network, writing the process of back-propagation instead of calling the function is very helpful for our study.

BP neural network is a "universal model + error correction function". Each time, based on the results obtained from the training, the error analysis is performed with the expected result, and then the weight and threshold are modified, and the model with the output and the expected result is obtained step by step. Setting different learning-rates and training times and the number of hidden units have different effects on the final result. If the number of training increases, the error decreases. The number of hidden units increases, and the error decreases, but it is necessary to pay attention to the fact that the hidden unit is too high to reduce the speed and miss the optimal solution. If the learning rate is too large or too small, it will have a negative impact on the results.