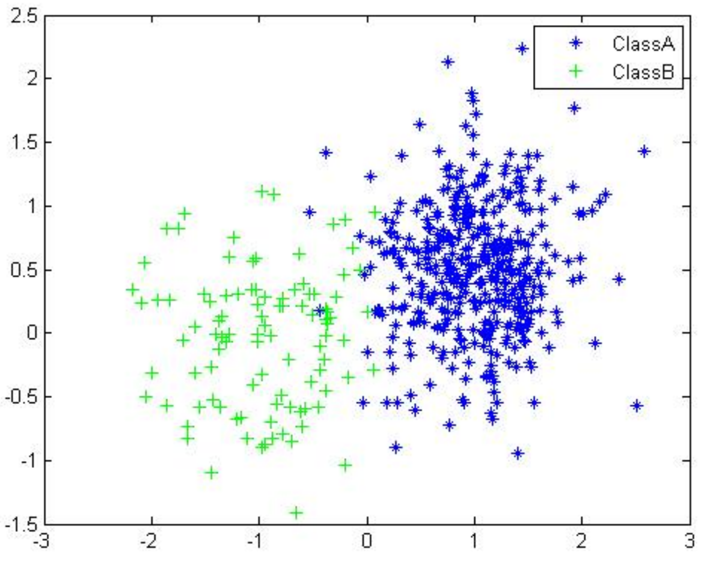
Artificial Neural Networks and other Learning Systems DD2432

Exercise 1: Feed forward networks — The delta rule and back propagation

1. Classification

1.1 Classification with one layer perceptron

(1) Separable data points

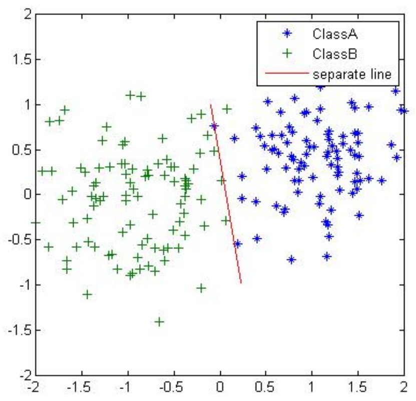


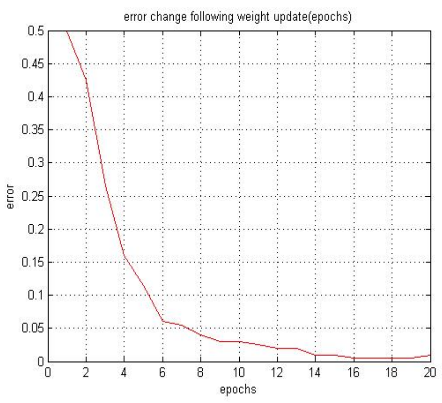
Class A has its midpoint at (1.0, 0.5) and class B with (−1.0, 0.0), both derivate are 0.5.

(2) Delta rule

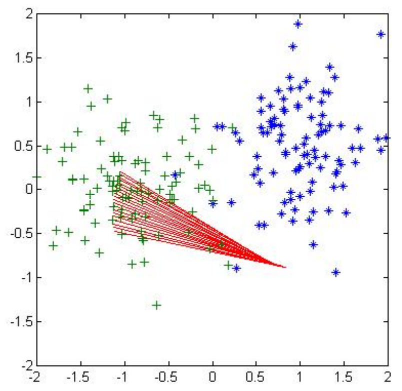
$$\Delta W = -\eta(WX - T)X^T$$

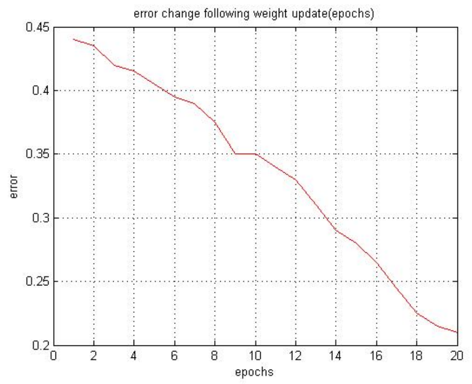
① $\eta = 0.001, \text{epochs} = 20$





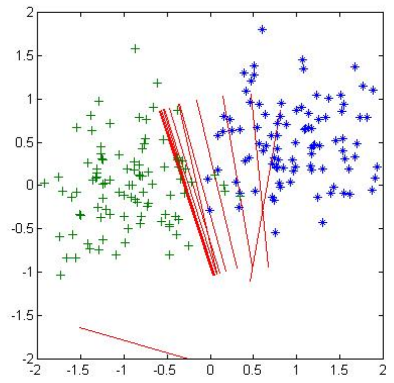
② $\eta = 0.0001, \text{epochs} = 20$

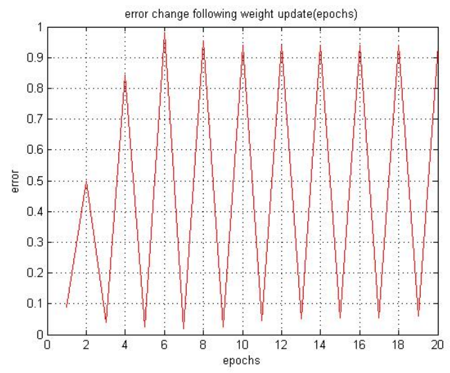




η influences the speed of weight changing. After 20 times iteration, the separate line is not the suitable one, we can see that $\eta = 0.0001$ the decrease speed of error is slower than $\eta = 0.001$

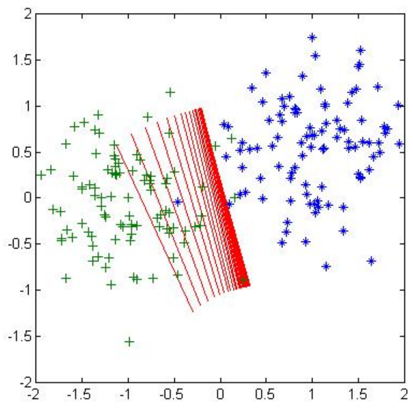
③ $\eta = 0.009, \text{epochs} = 20$

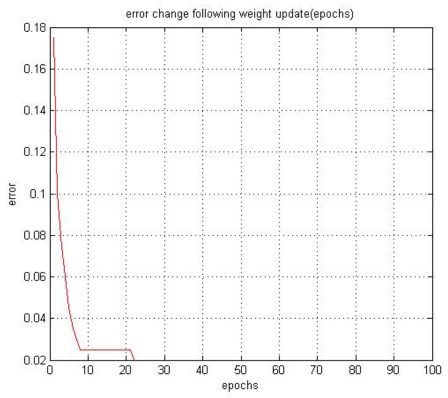




$\eta = 0.009$ can make the speed of updating weight faster than $\eta = 0.001$, but it generates oscillation (震荡), the error will diverge(发散), not converge.

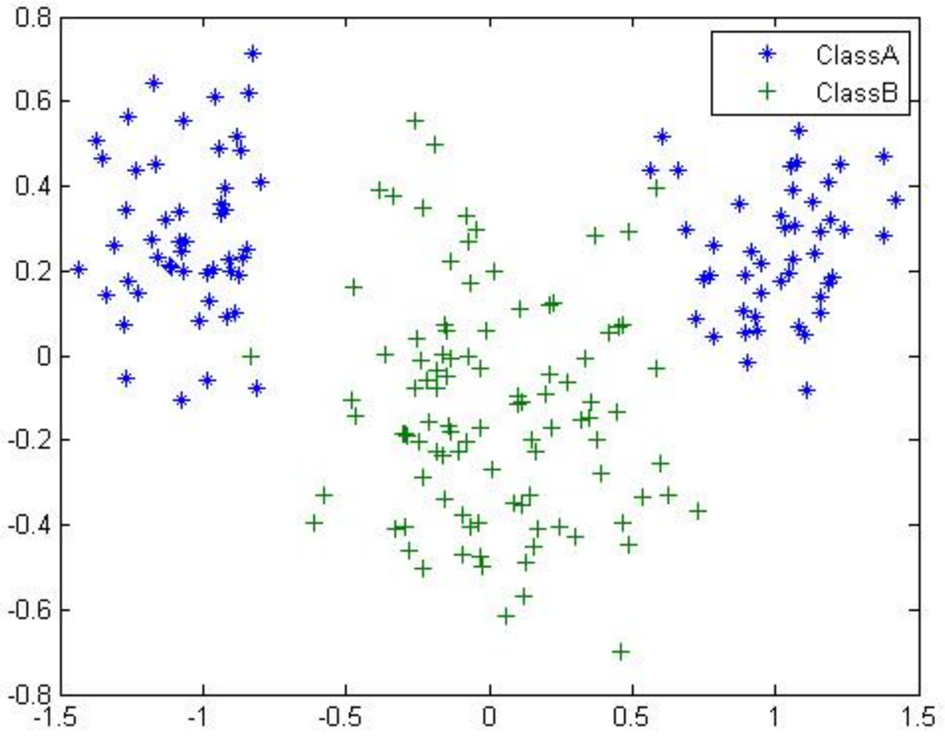
④ $\eta = 0.001, \text{epochs} = 100$



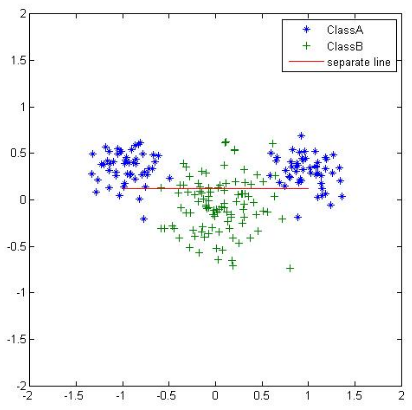


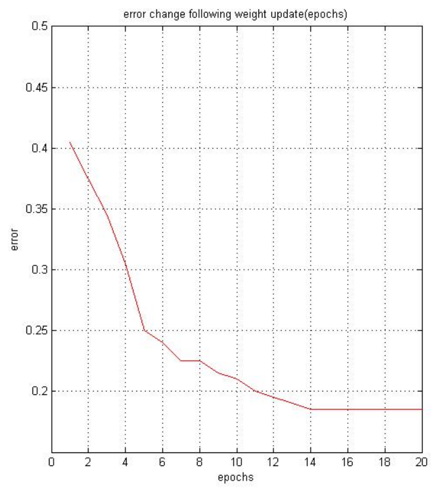
Epochs influence the accuracy of the classification when choosing a suitable $\eta = 0.001$. But after a certain iteration time, the error will not decrease obviously. In other word, the error will not change.

(3) Non-separable data



The $\mathrm{delta.m}$ will not work. Obviously, one separate line cannot separate above data set.





1.2 Classification with two layer perceptron

(1) Non-linear separable data (unsolved in one layer perception)

(2) Back-propagation

① forward pass

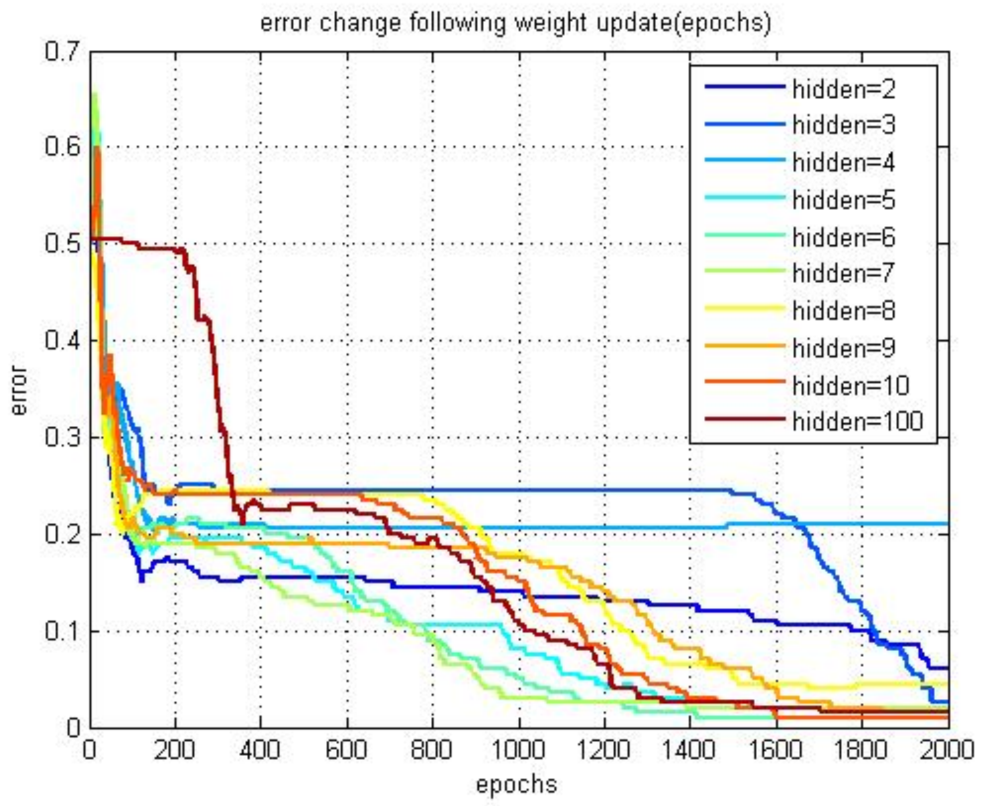
$$H = \phi(W X), \qquad O = \phi(V H)$$

② backward pass

$$\delta^{(o)} = (\mathrm{O} - \mathrm{T}) \odot \varphi'(O^*), \qquad \delta^{(h)} = (V^T \delta^{(o)}) \odot \varphi'(H^*)$$

③ weight update

$$\Theta = \alpha\Theta - (1-\alpha)\delta^{(h)}X^T, \qquad \Psi = \alpha\Psi - (1-\alpha)\,\delta^{(o)}H^T$$
$$\Delta W = \eta\Theta, \Delta V = \eta\Psi$$



Change the hidden nodes to see the effect. The number of hidden nodes represent the complexity of the two layer perceptron. We can indicate that at the beginning, the increase of the hidden nodes increase the accuracy of the classification. Considering the original data set can be separated by two straight lines, so the hidden nodes equal to 2, performs not bad, even good than hidden nodes=4 and 3, at epochs equal to 1800. But, when the epochs increase, the more nodes means more accuracy. Finally, according to the figure, we can draw a conclusion that hidden nodes equal to 6 can keep both low complexity and high accuracy.

To be honest, the lines draw in the figure, i.e. the error, are effected by the initial weights, w and v.

2. Data compression

The number of hidden nodes are much fewer than the number in the input and output layers. Classical 8-3-8 problem.

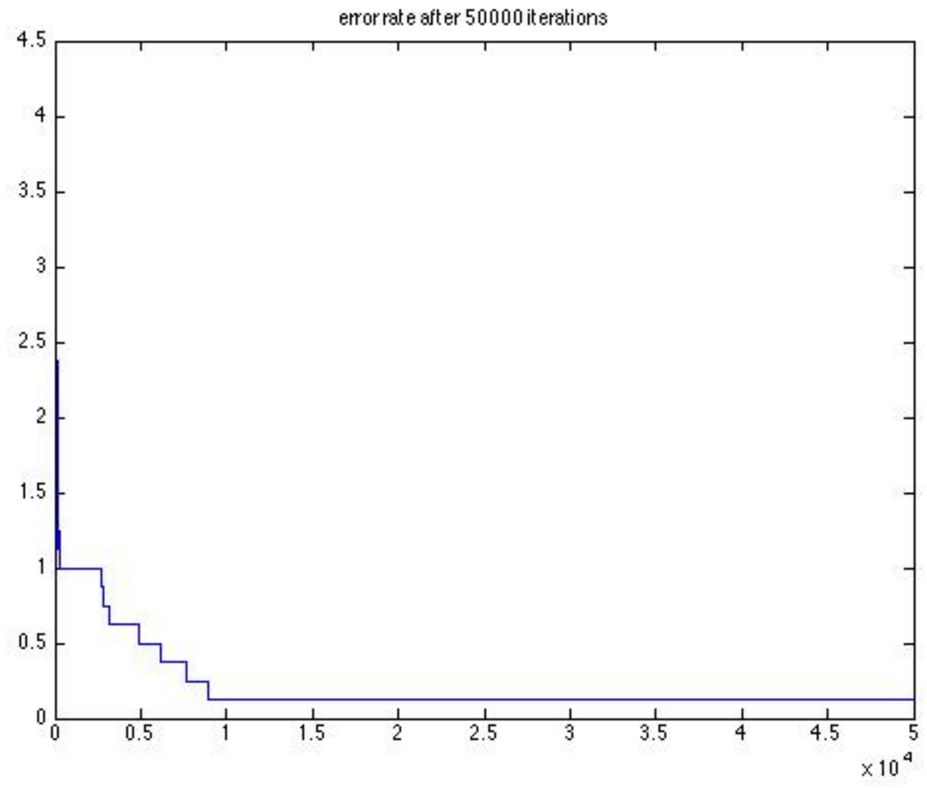
```
patterns =                                   targets =


     1    -1    -1    -1    -1    -1    -1    -1        1    -1    -1    -1    -1    -1    -1    -1
    -1     1    -1    -1    -1    -1    -1    -1       -1     1    -1    -1    -1    -1    -1    -1
    -1    -1     1    -1    -1    -1    -1    -1       -1    -1     1    -1    -1    -1    -1    -1
    -1    -1    -1     1    -1    -1    -1    -1       -1    -1    -1     1    -1    -1    -1    -1
    -1    -1    -1    -1     1    -1    -1    -1       -1    -1    -1    -1     1    -1    -1    -1
    -1    -1    -1    -1    -1     1    -1    -1       -1    -1    -1    -1    -1     1    -1    -1
    -1    -1    -1    -1    -1    -1     1    -1       -1    -1    -1    -1    -1    -1     1    -1
    -1    -1    -1    -1    -1    -1    -1     1       -1    -1    -1    -1    -1    -1    -1     1
```

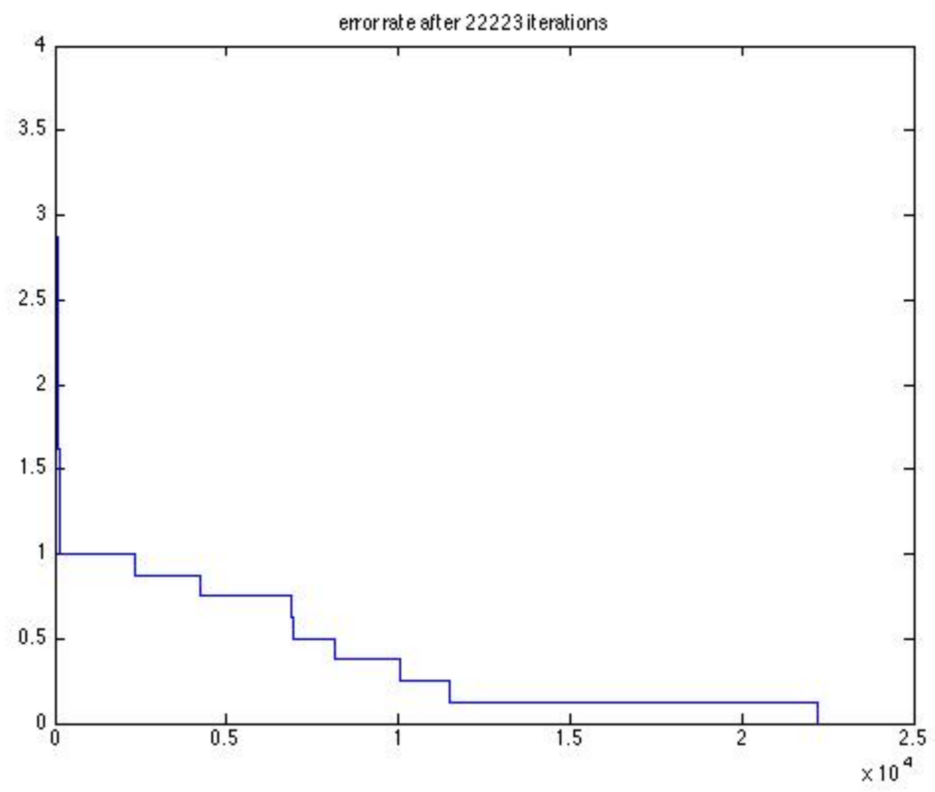
Hidden nodes equal to 3. Choose the maximum iteration is 50000. Sometimes, the classification error can reach zero during the iteraion steps, then break the loop. For example, after 22223 times iteration, the error becomes zero. Sometimes, the error state hold at a certain small value and will not decrease during the 50000 times iteration. That means the code does not always succeed.

Furthermore the weights are also different given correct models.

```
weight1 =
 -1     1     1     1     1    -1    -1    -1     1
 1    -1     1     1    -1    -1    -1     1     1
 1    -1    -1     1     1     1    -1    -1     1
weight2=
        1     1    -1    -1    -1     1     1     1    -1
        1     1     1    -1    -1     1    -1    -1    -1
        1    -1     1    -1     1     1    -1     1    -1
weight3 =
   1     1     1    -1     1    -1    -1    -1    -1
   1    -1     1     1    -1     1    -1    -1     1
   1    -1    -1    -1     1     1     1    -1     1
```

It seems there is no particular patter in the sign of the weights when the network can correctly classify the data.





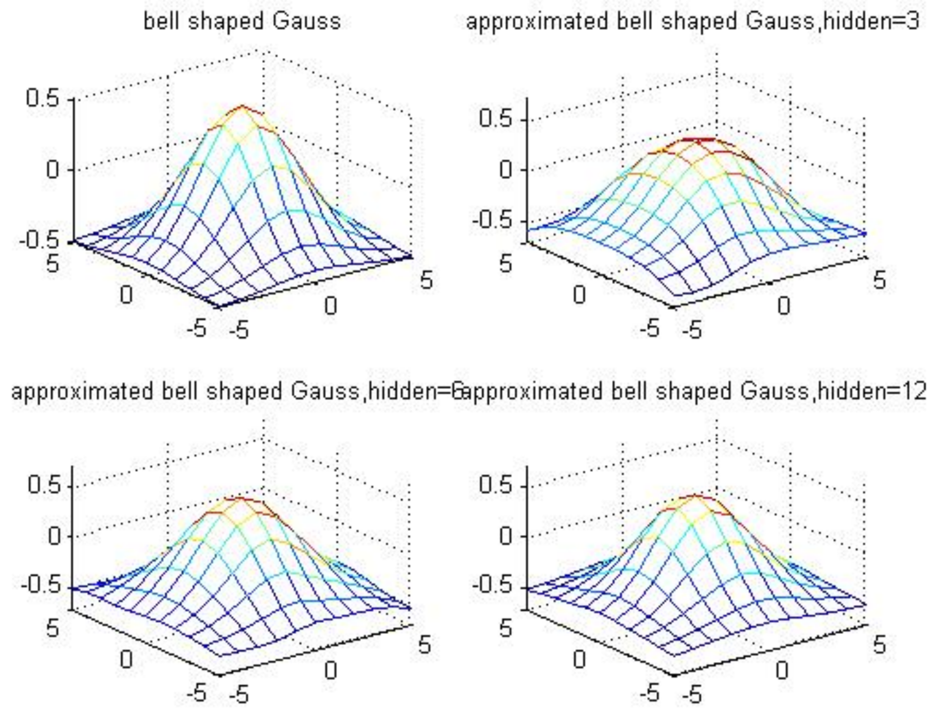
3. Function approximation(two layer network)

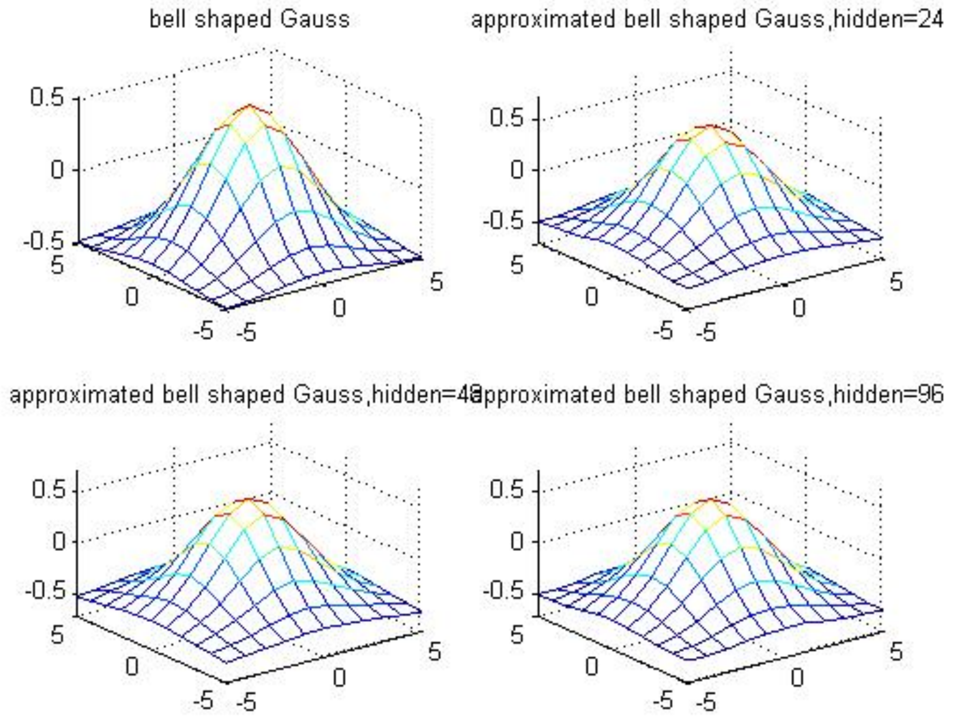
3.1 Generate function data

$$f(x, y) = e^{-(x^2+y^2)/10} - 0.5$$

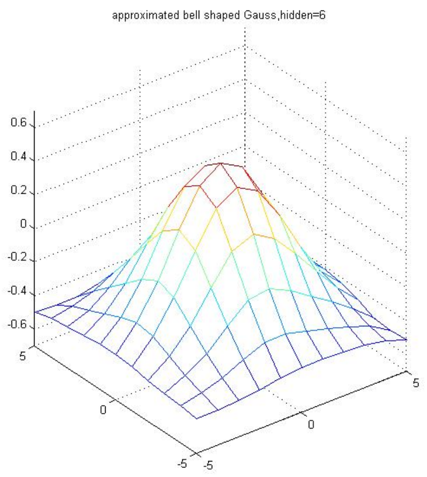
3.2 Reshape data in order to suit network

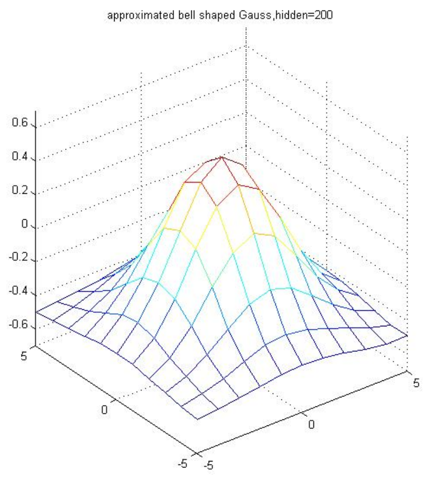
3.3 Train two layer network(change the number of hidden nodes)





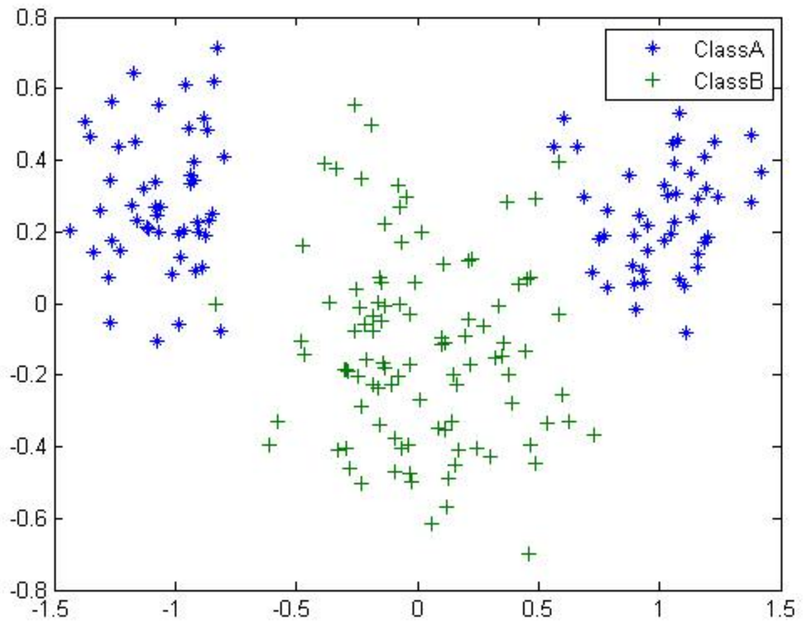
At the beginning, increasing the number of hidden nodes will approximate the original function well. After a certain value of hidden nodes, the approximate will not increase the accuracy intuitively. For instance, hidden nodes=6 and hidden nodes=200.





4. Generalization

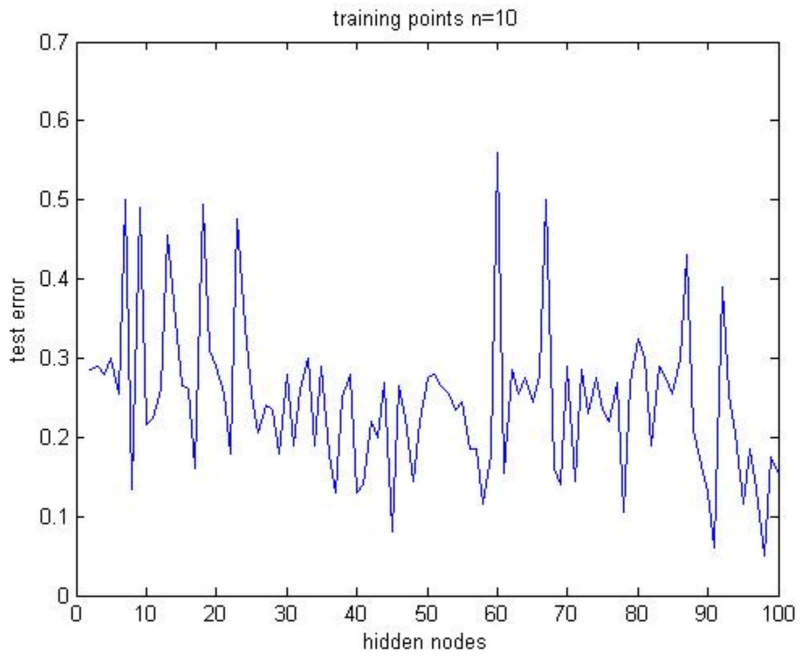
4.1 Data set (non-linear separable points)



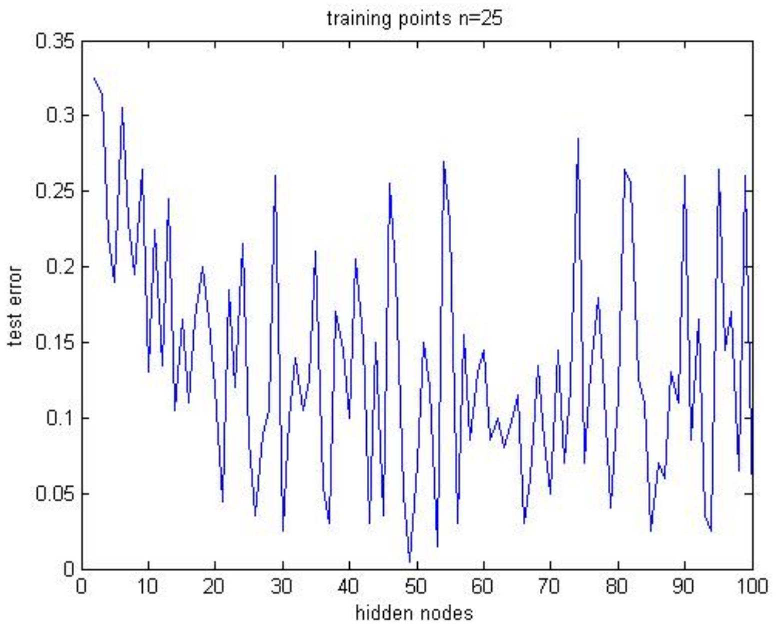
The total number of points is 200, we choose n=10 and n=25as training data points. In order to see that how the complexity of the model influence the test error, we should change the hidden nodes in the two layer network.

4.2 Test generation

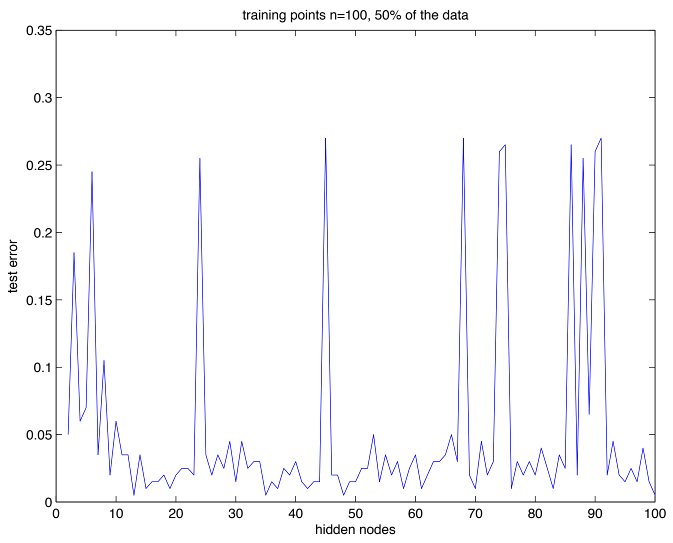
(1) n=20,change the hidden nodes from 2 to 100

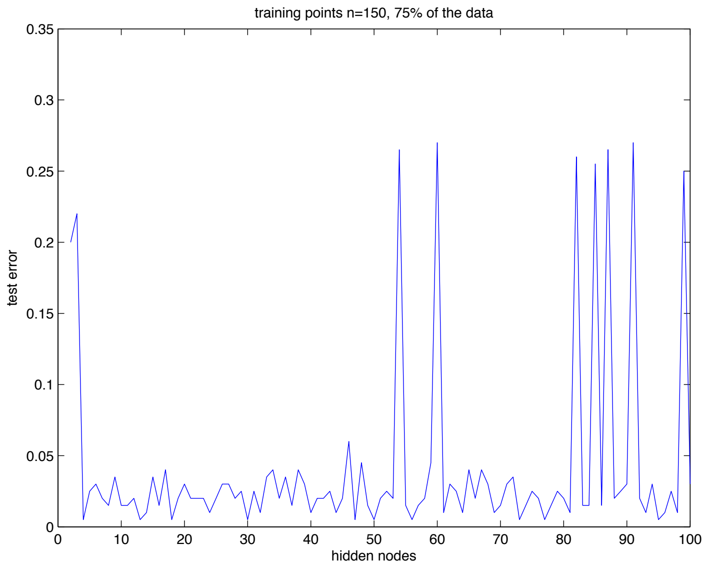


(2) n=25, change the hidden nodes from 2 to 100



(3) n=100 and n=150, change the hidden nodes from 2 to 100





In my opinion, the training data is too small that we cannot observe the real internal tend of the test error. While using 50% and 75% of the data for training, we see fewer fluctuations on test error.

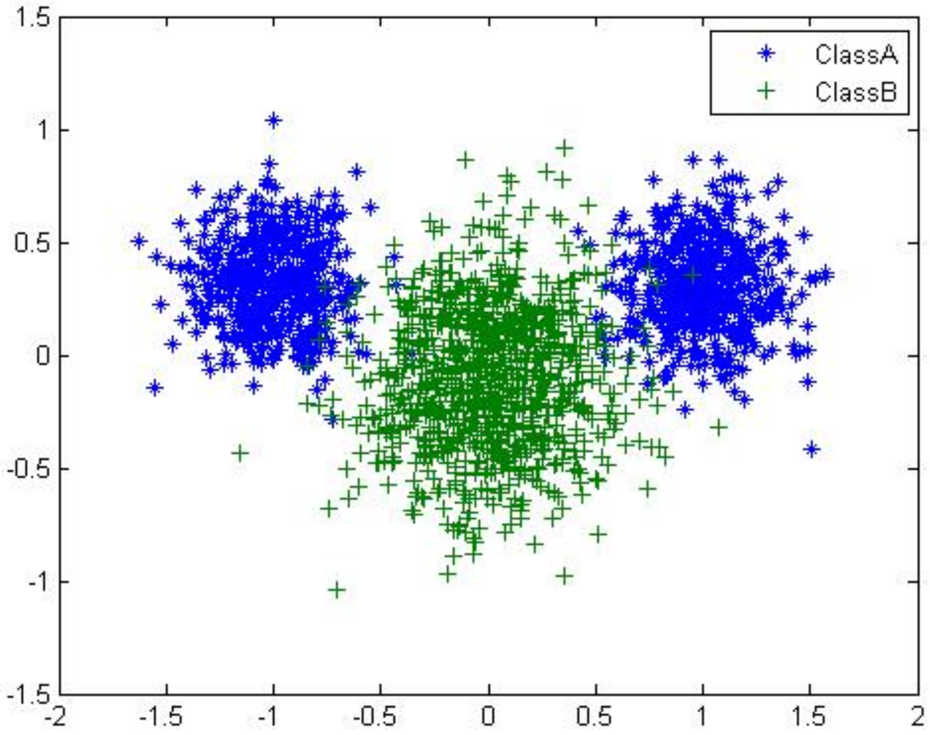
We can observe that:

1) When using small amount of data (n=10, 5%) for training, generalization performance is bad. The test error maintains relatively high.
2) When using larger amount of data (n=150,75%) for training, generalization performance is better. The test error is relatively high when hidden units n<=5 and it fluctuates a lot when hidden units n>=50.

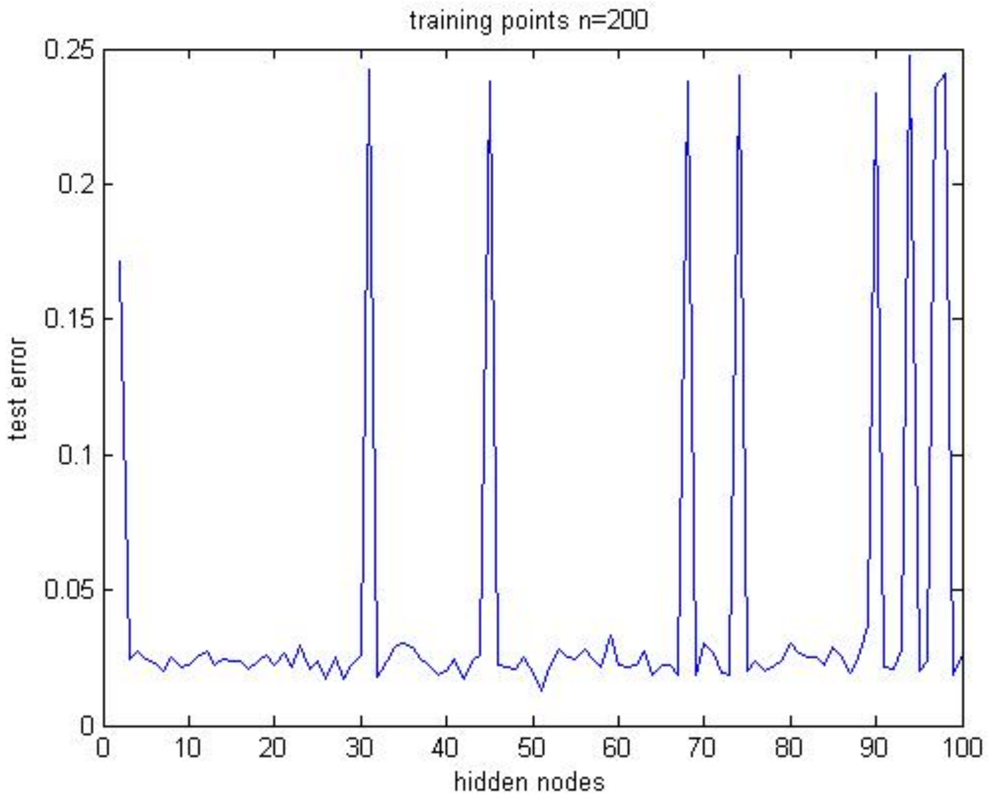
Our conclusion is that using only one small dataset for training is not a good way to deduce a good model. We can resample from the dataset to form training set iteratively and compute test error for each training set. Or we can use n-fold cross validation method by partitioning the dataset into different independent test sets.

The final good model with reasonable amount of hidden units is the one with both small average on test error and small variance on test error.

We generate a new data set with 2000 points, in following figure.



So, we use 200 data points (10%) to train the network.
n=200, change the hidden nodes from 2 to 100


training points n=200
test error
hidden nodes