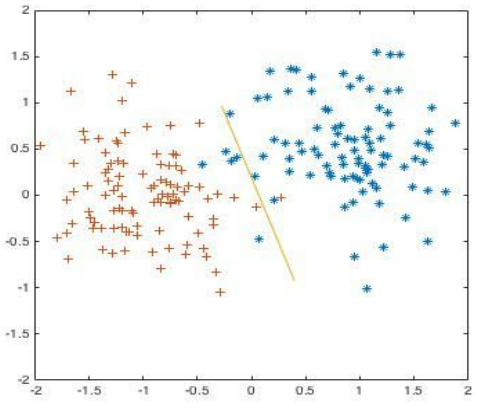
# Lab 1 results & discussion

### 4.2 delta rule w/separable data

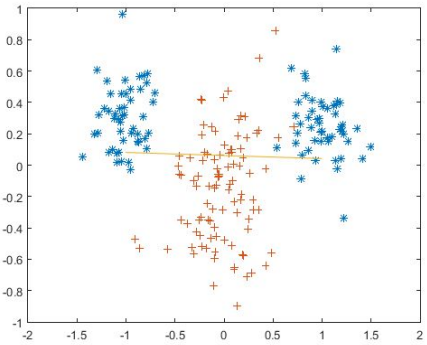


The effect of *epochs:*
the accuracy increases with epochs increases (computation time increases too), until error converges.

The effect of $\eta$:
$\eta$ is a scalar factor that controls each step when iterating to find a solution in order to make the network more stable. With a too small $\eta$, it leads to a slow convergence of error. On the contrary, a too large $\eta$ makes the weight changes too fast and end up with no solution.

Here we choose $\eta$ = 0.001, *epochs* = 40.

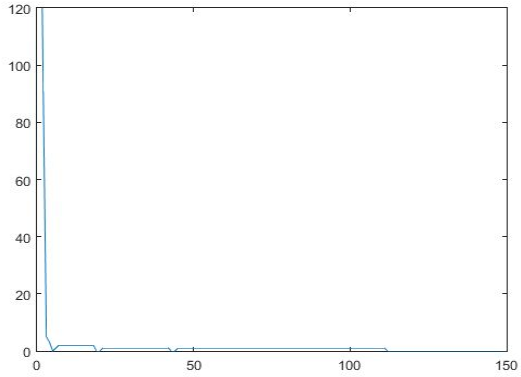
### 4.3 delta rule w/non-separable data



Apparently single-layer delta rule does not apply for non-separable data.

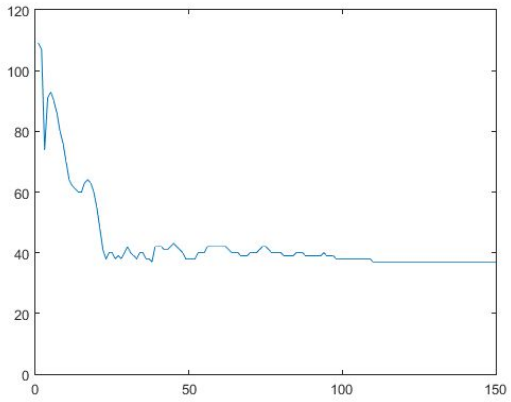
### 5.2 generalized delta-rule for non-separable data

Datasets are separated efficiently using two-layer perceptron. The following figure shows the classification of linear separable data, using hidden layer with only 1 node:

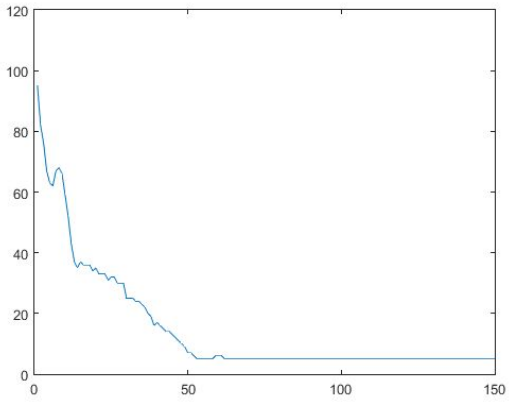


For non-linear separable data, muti-node hidden layer performs better than hidden layer with only 1 node. the error shows as below using different hidden:

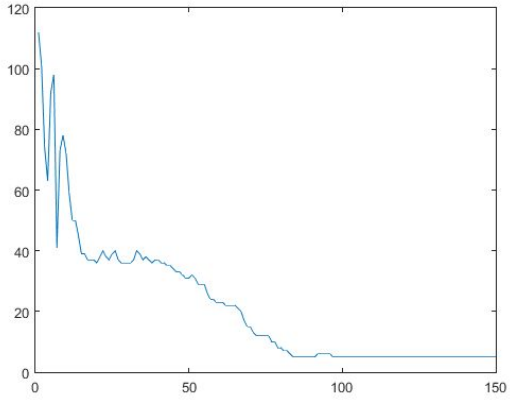
hidden = 1



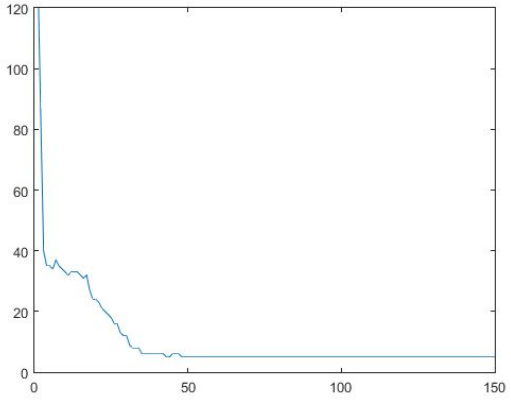

hidden = 2




hidden = 3




hidden = 20




From the figures we can find that, 2 nodes works well enough to separate the dataset, so using 20 nodes though have a better perform in smaller iteration, it may cause overfitting.

On the other hand, the iteration time (epochs) is sensitive to initial weight.

### 5.3 the encoder problem

```
47 -        error = [error; e];
48 -    end
49 -    disp(itr)
50
51
```

Command Window

New to MATLAB? See resources for Getting Started.

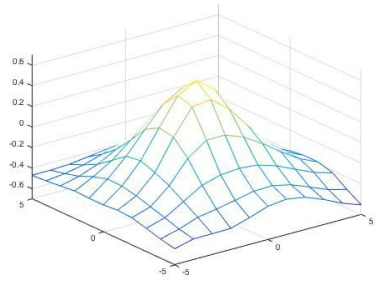
```
     1

     0

        9584
```

after 450~10000 iterations the program always shows correct network.
the sign of weight and hout shows as below:

```
>> sign(w)

ans =

     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

>> sign(hout)

ans =

     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
```

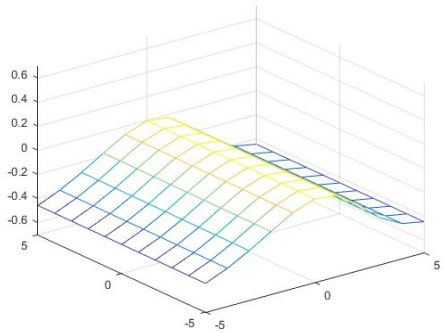
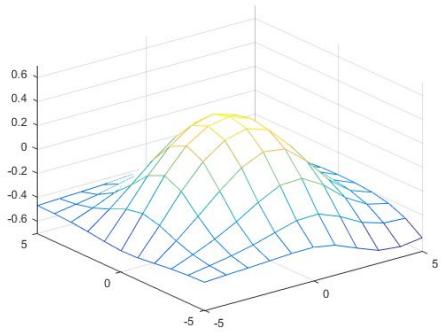
## 6 function generation

the final learning result shown below

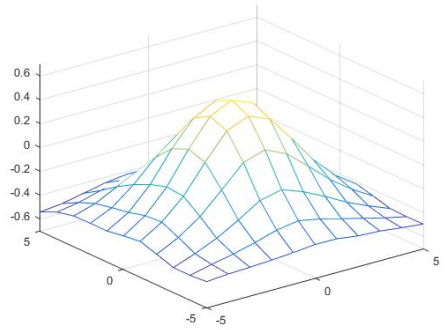


the accuracy increases with #iteration increases (and also computation time) with a small $\eta$ ($\eta$ = 0.001) as it converges slowly. When #hidden node increases, the average #iterations needed to get a promising result also increases (usually greater than 1000). besides, the final result has a small “variation” w.r.t. the correct figure as the iteration steps become very large.
why: since the number of hidden node increases, the cost for obtaining reasonable value of one node in the matrix also increases significantly.

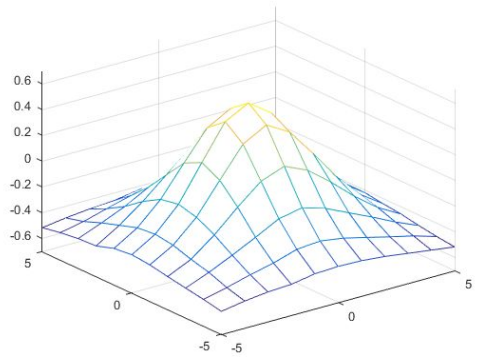
The influence of number of hidden node is shown as below (here we adopted a larger $\eta$ = 0.1 ~ 0.2):

| hidden = 2 | hidden = 5 |
| --- | --- |
|  |  |

hidden = 25



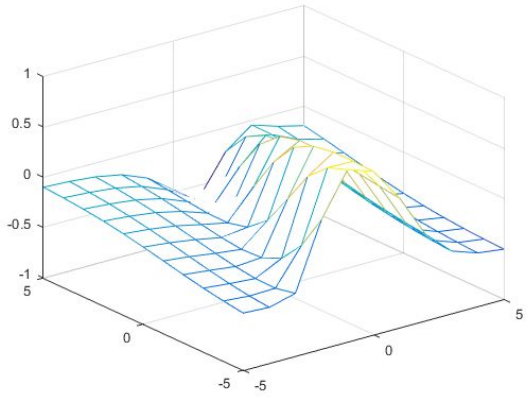
hidden = 65



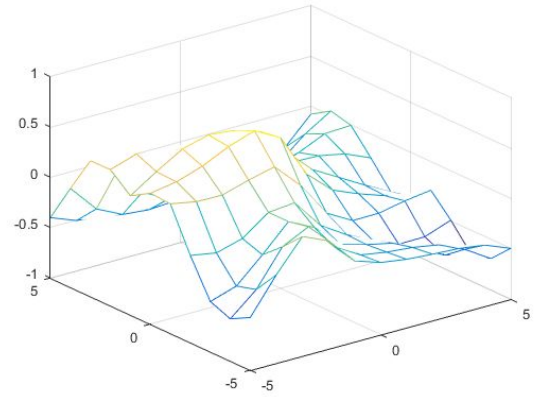
## 7 generalization

Since the convergence is slow in function approximation, here we adopt $\eta = 0.2$. The results using different **n** and **hidden** are shown as below:

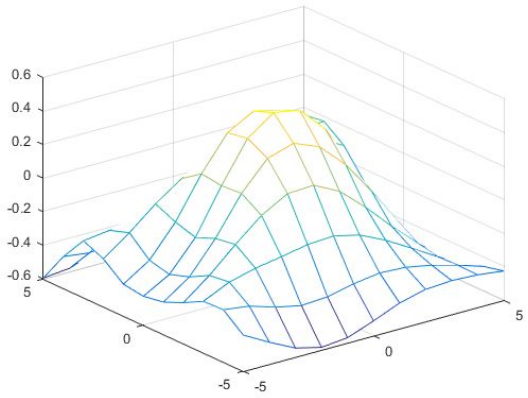
n = 10, hidden = 5



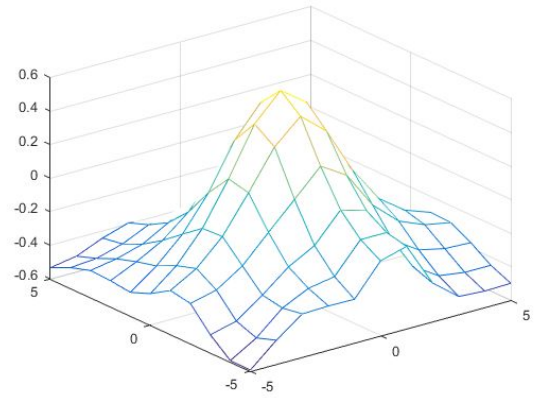
n = 10, hidden = 20



n = 25, hidden = 5



n = 25, hidden = 20



As the results show, a larger n (n = 25) performs better than a smaller n (n = 10), and a hidden node with 25 nodes performs better than 10 nodes. Thus n = 25, hidden = 20 is a suitable size for this dataset. However, a further increasement of the number of hidden nodes doesn't ensure a better result because of the problem of overfitting in the process of generalization. The figure below shows the result using hidden = 50: