**Investigating single layer and multilayer perceptron networks and kohonen networks**

**Introduction**

This lab was used to investigate three different types of neural network, single layer and multilayer perceptron networks were used to demonstrate supervised learning where a Kohonen network was used to demonstrate unsupervised learning.

**Method**

First, I investigated single layer perceptron network with two voice recordings. I had the network determine that the two were different. Second, I Investigated multi-layer perceptron network with multiple voice recordings had the network determine between different and distinct voices from a group I then Investigated Korhonen network with multiple trian recordings and had that distinguish between each of the train sounds. The results and outcomes are discussed below.

**Outcomes**

A single layer perceptron network is a neural network made up of a single layer of perceptrons this is a supervised learning network meaning that desired outcomes and data inputs are provided by the user. A single perceptron is built up of and number of inputs and their weights passed through a summing function the result of that is then passed through an activation function this is then compared to the desired outputs the error is calculated and an error is calculated this error is passed through a learning algorithm that adjusts the weights and retries the calculation see Fig.1

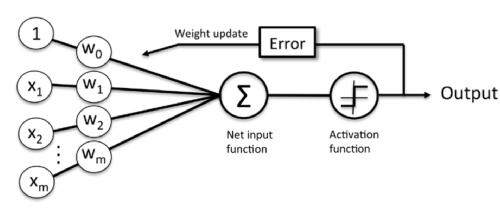


Fig.1 figure shows the basic structure of a neuron in a SLP Hong K (2016)

These single layer perceptron networks are capable of forming a linear decision boundary through classes. However, the flaw with SLP networks is that they are unable to separate non-linearly separable classes.

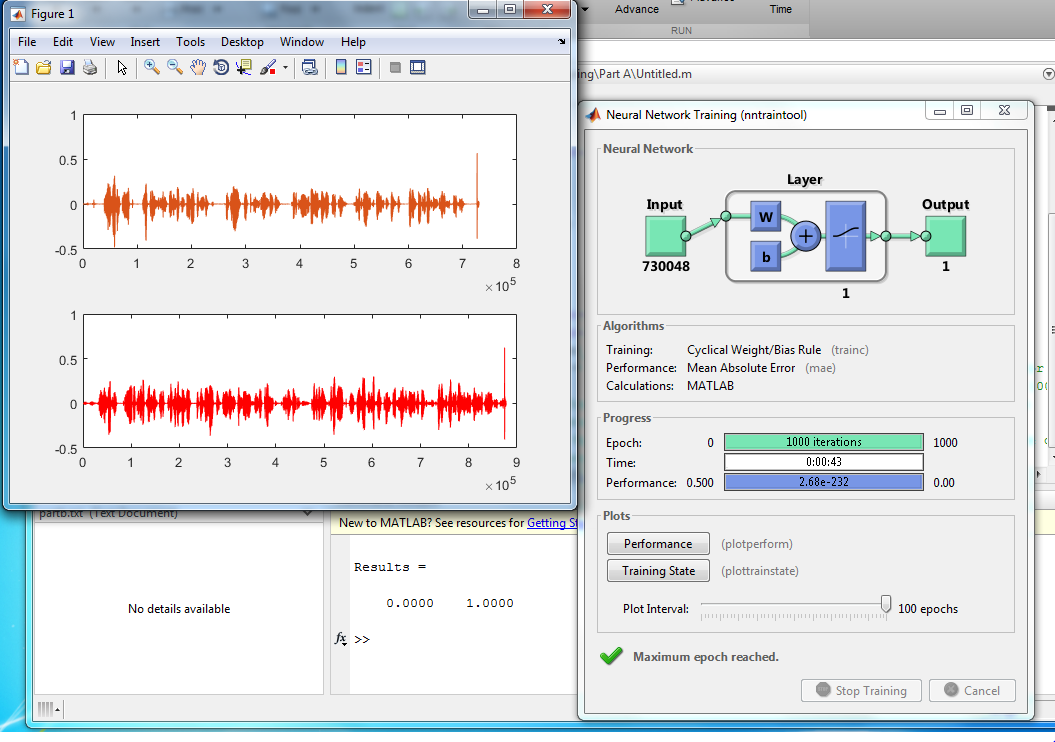


Fig.2 figure shows the output windows for the single layer perceptron network

The results in Fig.2 show a working SLP having completed 1000 iterations this network has used two sound files of two voices and has learned to differentiate between the two of them and we see this from the bottom centre where the lines Results = 0.0000 1.0000 is present the 0 and 1 represent two distinct classed separated linearly.

A multi-layer perceptron network is a neural network made up of a multiple layers of perceptrons this is a supervised learning network meaning that desired outcomes and data inputs are provided by the user. The layers of perceptrons are named the input layer, this is the layer where the data first gets input into the network and is the first layer of perceptrons, the hidden layer, this is the layer where multiple layers of perceptrons can be hidden it is called the hidden layer because it is unseen from outside the network, and the output layer this is the last layer of perceptrons and any output data, see Fig.3.

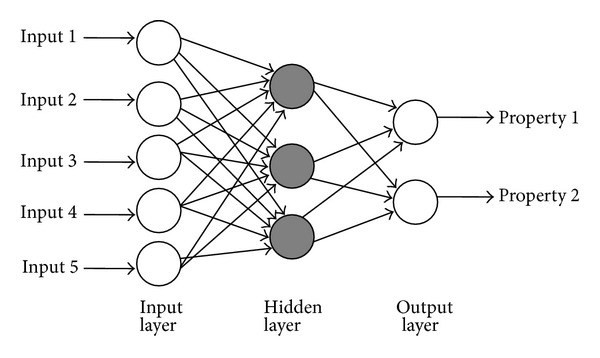
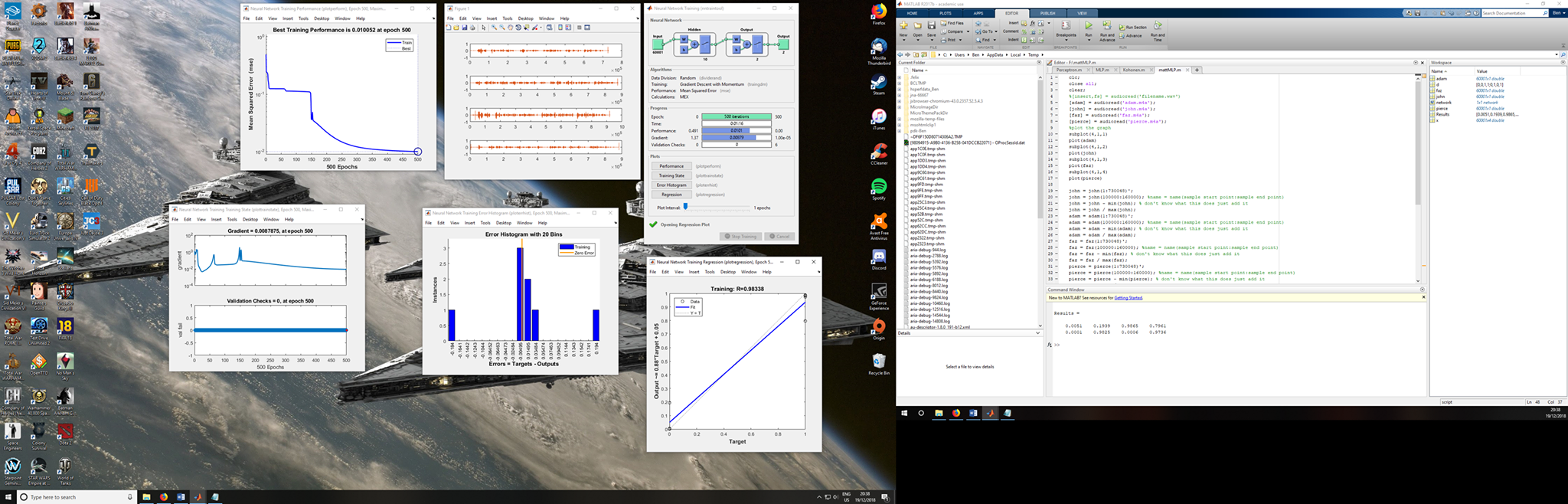


Fig.3 this figure shows the structure of a multi-layer perceptron network Mathur P (2016)

The advantages of using a multi-layer perceptron network over a single layer network is the ability to use exclusive or functions, this means that unlike single layer the multi-layer network can use non-linear methods of separating classes.



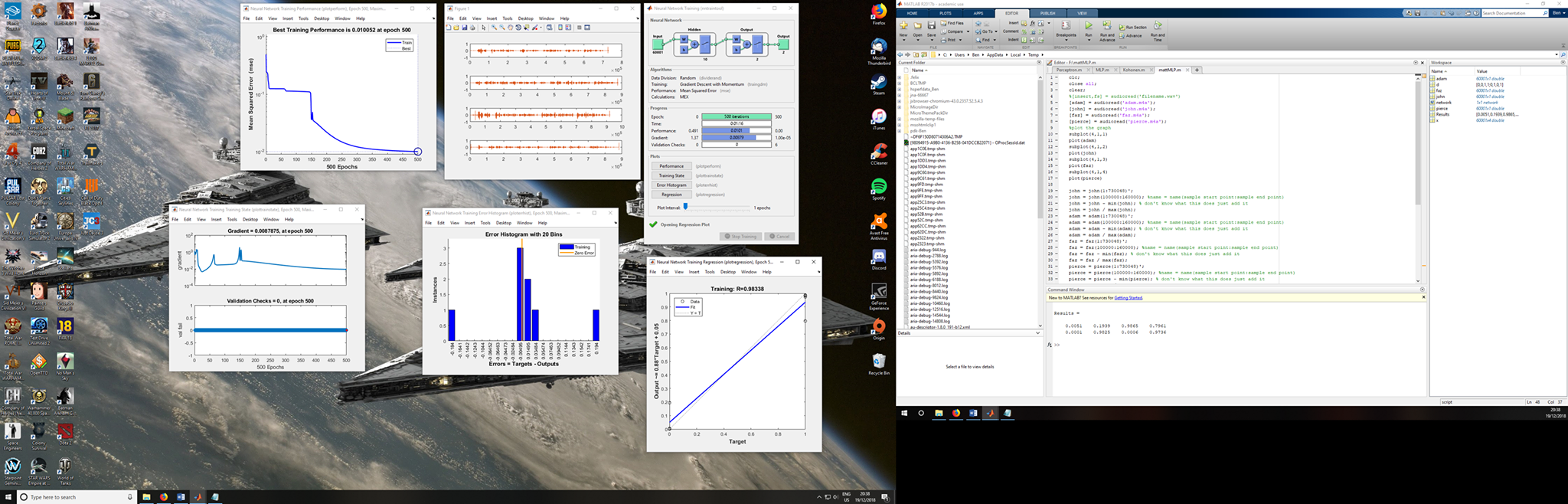


Fig.4 & 5 these figures show the results of the MLP network

The results in Fig.2 show a working MLP having completed 500 iterations this network has used four sound files of four voices and has learned to differentiate between all four of them and we see this from Fig.5 that shows each of the classes very close to the original results that we wanted.

Kohonen Networks are not like the other two networks that were investigated this network is an unsupervised learning neural network and is like the brain in the way that it learns these networks are also called self-organising maps due their layout as can be seen in both figures six and 7.

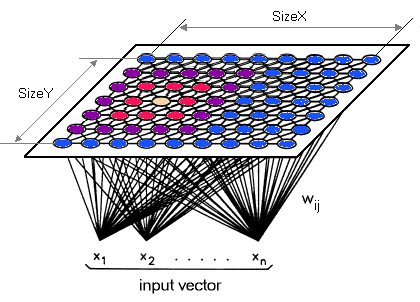


Fig.7 this shows a Kohonen network Ahn J & Yeon Syn S

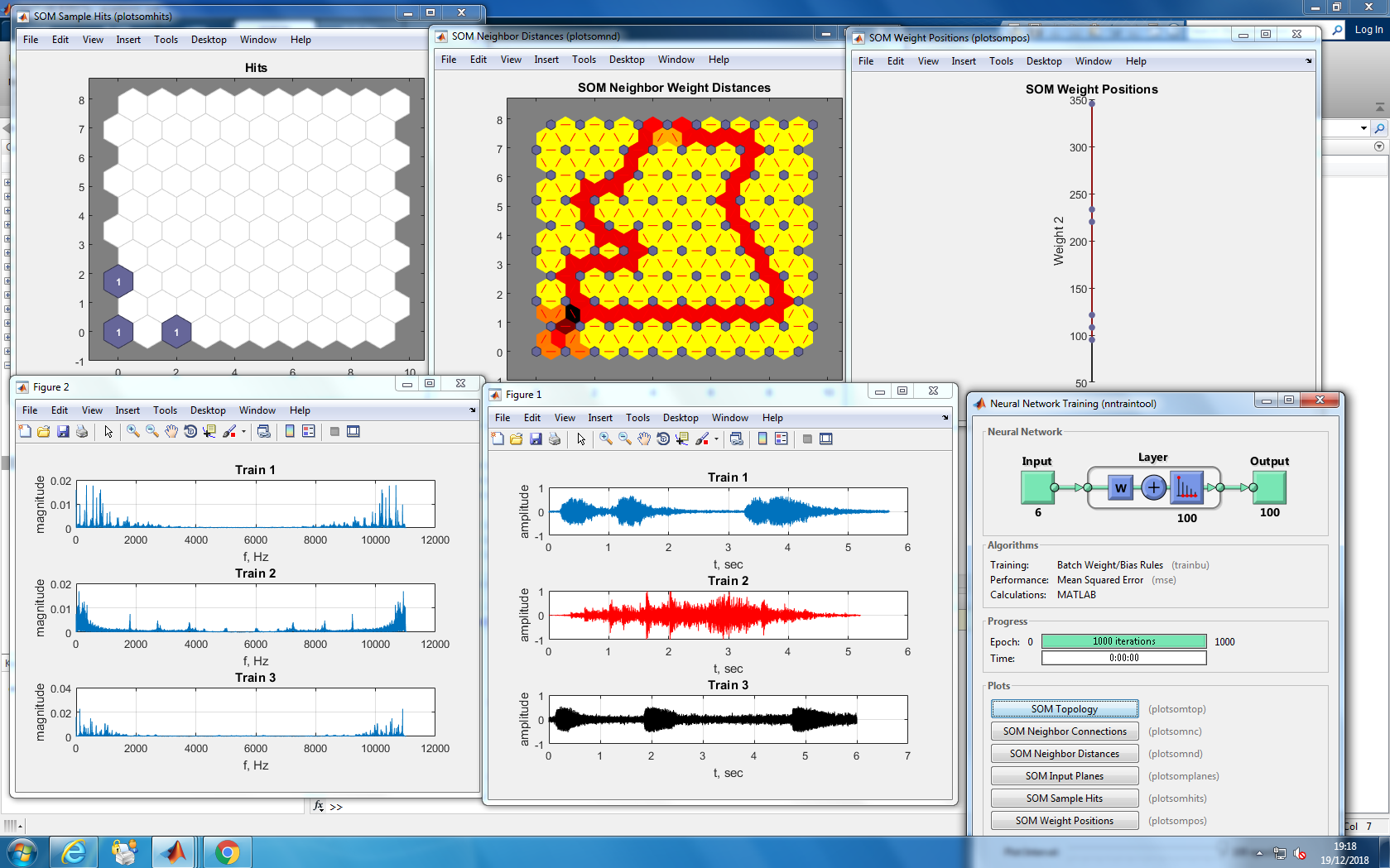


Fig.7 this figure shows the results from the kohonen network.

For each of the three trains that I used in this network there is a colour associated to it on the map in the top centre of Fig.7, other than yellow which is a background colour, as we can see the network distinguished the three different train sounds.

**Conclusion**

Each of these types of network are suited to different tasks the SLP is best suited to distinguishing between distinct classes where theses classes can have a linear boundary. An MLP is best suited to tackling more complex issues that use primarily non linear separable classes that require a non linear boundary finally a Kohonen network is best left to find correlations in data and patterns in data as it is left to find patterns after it is trained.

**References**

Hong K. (2016). Single Layer Neural Network - Perceptron model on the Iris dataset using Heaviside step activation function. <https://www.bogotobogo.com/python/scikit-learn/Perceptron_Model_with_Iris_DataSet.php>

Mathur P. (2016). A Simple Multi-Layer Perceptron with TensorFlow. <https://medium.com/pankajmathur/a-simple-multilayer-perceptron-with-tensorflow-3effe7bf3466>

Ahn J & Yeon Syn S. (2005). Self-Organising Maps. <http://www.pitt.edu/~is2470pb/Spring05/FinalProjects/Group1a/tutorial/som.html>

**Appendix**

**Code listing 1 single layer perceptron network**

clc;

close all;

[matt] = audioread('matt.m4a');

subplot(211)

plot(matt)

[john] = audioread('john.m4a');

subplot(212)

plot(john,'r')

john = john(1:730048);

john = john';

x = [john matt(:,1)];

d = [0 1];

learning\_function='learnp';

transfer\_function='logsig';

net = newp(x,d,transfer\_function,learning\_function);

net.trainParam.epochs = 1000;

net=train(net,x,d);

Results = sim(net, x)

**Code listing 2 Multilayer perceptron network**

clc;

close all;

clear;

[matt] = audioread('matt.m4a');

[john] = audioread('john.m4a');

[faz] = audioread('faz.m4a');

[pierce] = audioread('pierce.m4a');

subplot(4,1,1)

plot(matt)

subplot(4,1,2)

plot(john)

subplot(4,1,3)

plot(faz)

subplot(4,1,4)

plot(pierce)

john = john(1:730048)';

john = john(100000:160000);

john = john - min(john);

john = john / max(john);

matt = matt(1:730048)';

matt = matt(100000:160000);

matt = matt - min(matt);

matt = matt / max(matt);

faz = faz(1:730048)';

faz = faz(100000:160000);

faz = faz - min(faz);

faz = faz / max(faz);

pierce = pierce(1:730048)';

pierce = pierce(100000:160000);

pierce = pierce - min(pierce);

pierce = pierce / max(pierce);

x = [john matt faz pierce];

d = [0 0; 0 1; 1 0; 1 1]';

network = feedforwardnet();

network.layers{1}.size = 10;

network.layers{2}.size = 2;

network = init(network);

network.trainFcn = 'traingdm';

network.trainParam.epochs = 500;

network.trainParam.lr = 0.05;

network.layers{1}.transferFcn = 'logsig';

network.layers{2}.transferFcn = 'logsig';

network.divideParam.trainRatio = 1;

network.divideParam.valRatio = 0;

network.divideParam.testRatio = 0;

network=train(network,x,d);

Results = sim(network, x)

**Code listing 3 Kohonen network**

clear;

close all

[train1, fs1]=audioread('train1.wav');

train1 = downsample(train1,2);

[train2, fs2]=audioread('train2.wav');

[train3, fs3]=audioread('train3.wav');

train3 = downsample(train3,2);

fs=fs2;

Ts= 1/fs;

len\_train1= length(train1);

len\_train2= length(train2);

len\_train3= length(train3);

t1=(0:len\_train1-1)\*Ts;

t2=(0:len\_train2-1)\*Ts;

t3=(0:len\_train3-1)\*Ts;

figure(1)

subplot(311)

plot(t1,train1)

xlabel('t, sec')

ylabel('amplitude')

grid

title('Train 1')

subplot(312)

plot(t2,train2, 'r')

xlabel('t, sec')

ylabel('amplitude')

grid

title('Train 2')

subplot(313)

plot(t3,train3, 'k')

xlabel('t, sec')

ylabel('amplitude')

grid

title('Train 3')

train1\_spec= (2/len\_train1)\* abs(fft(train1));

train2\_spec= (2/len\_train2)\* abs(fft(train2));

train3\_spec= (2/len\_train3)\* abs(fft(train3));

f1=(0:len\_train1 -1 ) /(len\_train1\*Ts);

f2=(0:len\_train2 -1 ) /(len\_train2\*Ts);

f3=(0:len\_train3 -1 ) /(len\_train3\*Ts);

figure(2)

subplot(311)

plot(f1,train1\_spec)

xlabel('f, Hz')

ylabel('magnitude')

grid

title('Train 1')

subplot(312)

plot(f2,train2\_spec)

xlabel('f, Hz')

ylabel('magnitude')

grid

title('Train 2')

subplot(313)

plot(f3,train3\_spec)

xlabel('f, Hz')

ylabel('magnitude')

grid

title('Train 3')

fea1 = [0.01786,345.2, 0.01764,565.1,0.01599, 815.8]

fea2 = [0.01677, 94.98, 0.01255, 270.7, 0.01213, 208.4]

fea3 = [0.0226, 121.3, 0.01484, 480.6, 0.01229, 528.6]

p = [fea1;fea2;fea3]';

dimension = [10,10];

coverStep = 500;

initNeighbor = 4;

topologyFcn = 'hextop';

distanceFcn = 'linkdist';

net=selforgmap(dimension,coverStep,initNeighbor,topologyFcn,distanceFcn);

[net,y]=train(net,p);

figure(3)

plotsompos(net,p)

figure(4)

plotsomnd(net)

figure(5)

plotsomhits(net,p)