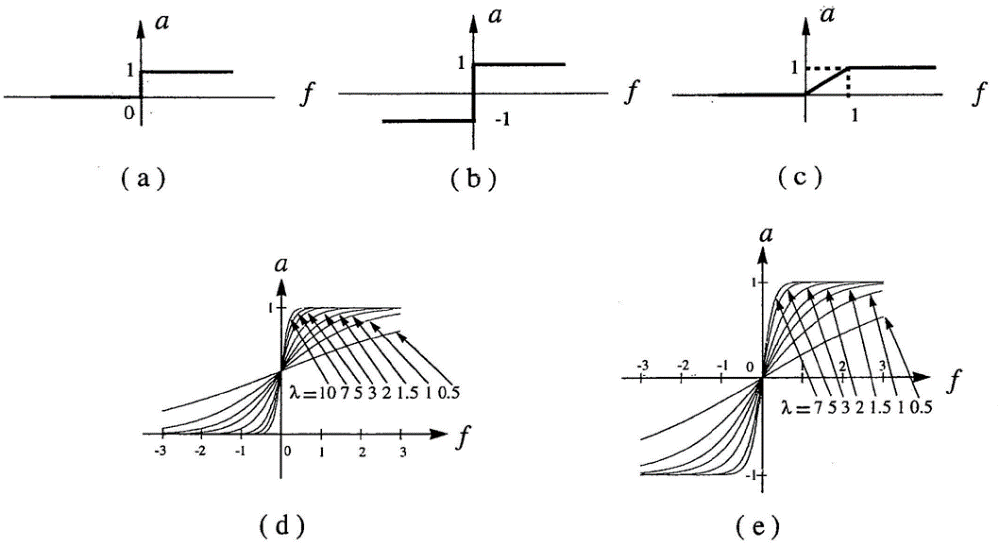
Backpropogtion

Research Background:

In recent years the security of operations and authentication provided over the computer network has become very important. It is necessary to protect these actions against attackers who misuse the system. Many cryptographic protocols and schemes were designed to solve problems of this type. Out of these techniques, secret sharing schemes provide an efficient solution. The secret sharing scheme is used for providing authentication to a group communication. These schemes make it possible to store secret information in a network, such that only authorized users can take its advantage and only they can reconstruct the secret. Fuzzy optimization BP neural network model proposed by Chen S Y combined of both fuzzy theory and neural network. Especially, fuzzy optimization model is as activation functions of neural network, so physical meaning of the model is comprehensible.

Research Method:

## Activation function or transfer function



a = step function

b = Hard limiter

c = Ramp function

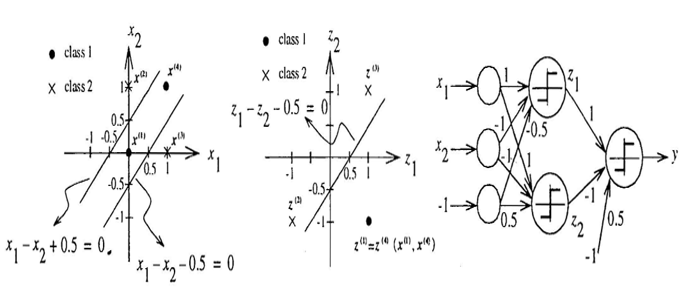
d = Unipolar sigmoid function

e = Bipolar sigmoid function

## Learning Rule

## 

## Backpropogation network



**Matlab implementation to solve the XOR problem with a multilayer perceptron**

Source code:

%Define 4 clusters of input data

close all, clear all, clc, format compact

% number of samples of each class

K = 100;

% define 4 clusters of input data

q = .6; % offset of classes

A = [rand(1,K)-q; rand(1,K)+q];

B = [rand(1,K)+q; rand(1,K)+q];

C = [rand(1,K)+q; rand(1,K)-q];

D = [rand(1,K)-q; rand(1,K)-q];

% plot clusters

figure(1)

plot(A(1,:),A(2,:),'k+')

hold on

grid on

plot(B(1,:),B(2,:),'bd')

plot(C(1,:),C(2,:),'k+')

plot(D(1,:),D(2,:),'bd')

% text labels for clusters

text(.5-q,.5+2\*q,'Class A')

text(.5+q,.5+2\*q,'Class B')

text(.5+q,.5-2\*q,'Class A')

text(.5-q,.5-2\*q,'Class B')

%Define output coding for XOR problem

% encode clusters a and c as one class, and b and d as another class

a = -1; % a | b

c = -1; % -------

b = 1; % d | c

d = 1; %

%Prepare inputs & outputs for network training

% define inputs (combine samples from all four classes)

P = [A B C D];

% define targets

T = [repmat(a,1,length(A)) repmat(b,1,length(B)) ...

repmat(c,1,length(C)) repmat(d,1,length(D)) ];

% view inputs |outputs

%[P' T']

%Create and train a multilayer perceptron

% create a neural network

net = feedforwardnet(2);

% train net

net.divideParam.trainRatio = 1; % training set [%]

net.divideParam.valRatio = 0; % validation set [%]

net.divideParam.testRatio = 0; % test set [%]

net.layers{1}.transferFcn = 'tansig';

% train a neural network

[net,tr,Y,E] = train(net,P,T);

% show network

view(net)

%plot targets and network response to see how good the network learns the data

figure(2)

plot(T','linewidth',2)

hold on

plot(Y','r--')

grid on

legend('Targets','Network response','location','best')

ylim([-1.25 1.25])

%Plot classification result for the complete input space

% generate a grid

span = -1:.005:2;

[P1,P2] = meshgrid(span,span);

pp = [P1(:) P2(:)]';

% simulate neural network on a grid

aa = net(pp);

% translate output into [-1,1]

%aa = -1 + 2\*(aa>0);

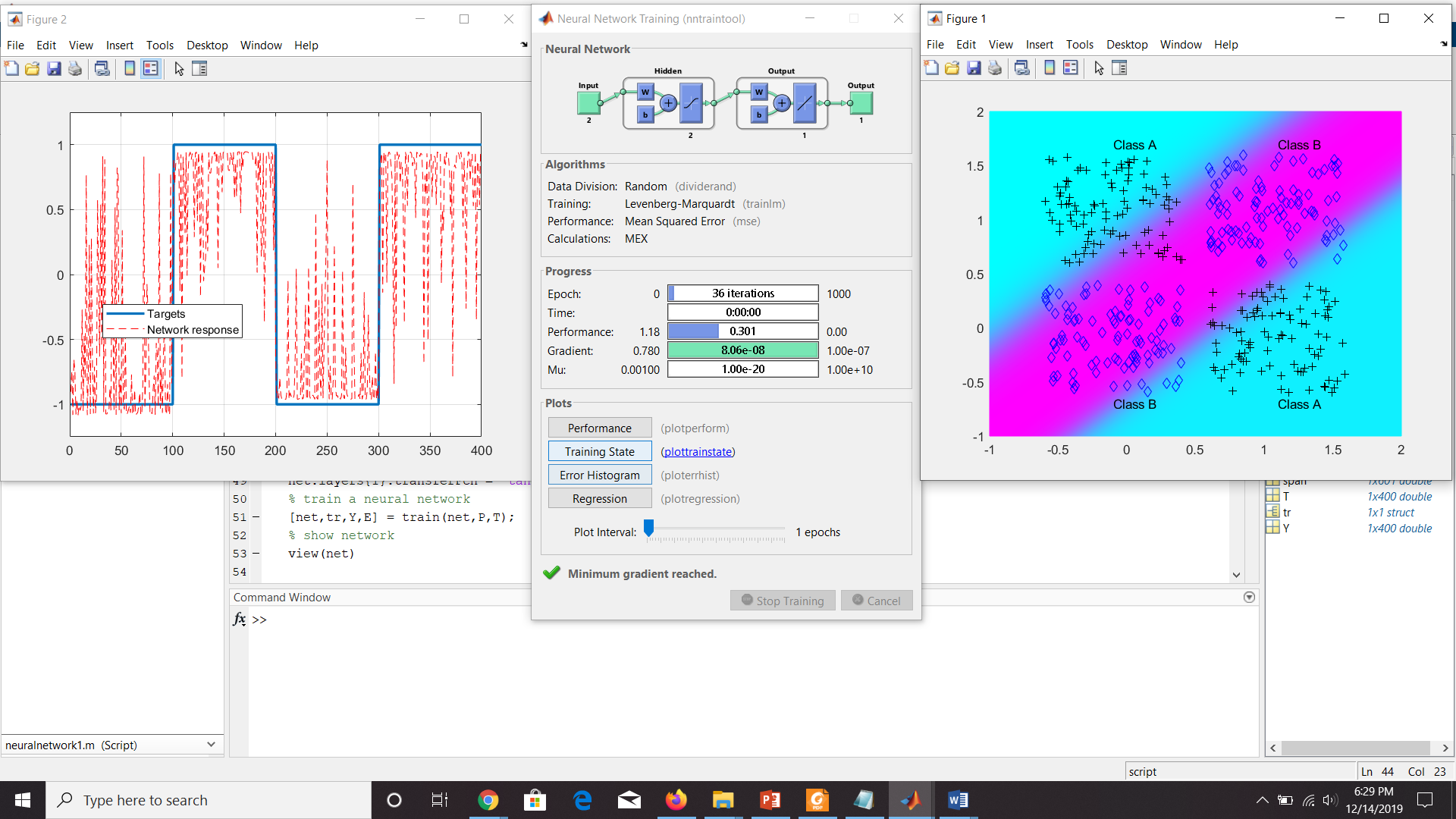
% plot classification regions

figure(1)

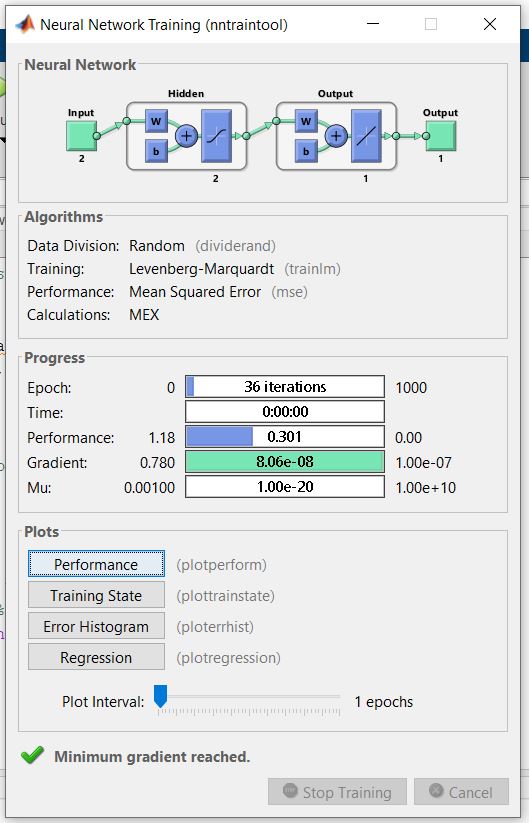
mesh(P1,P2,reshape(aa,length(span),length(span))-5);

colormap cool

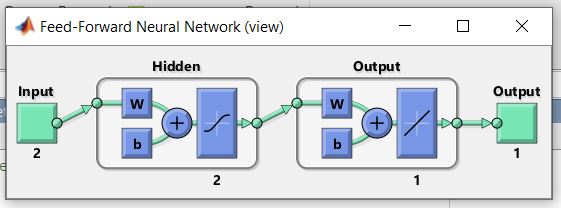
**Results**



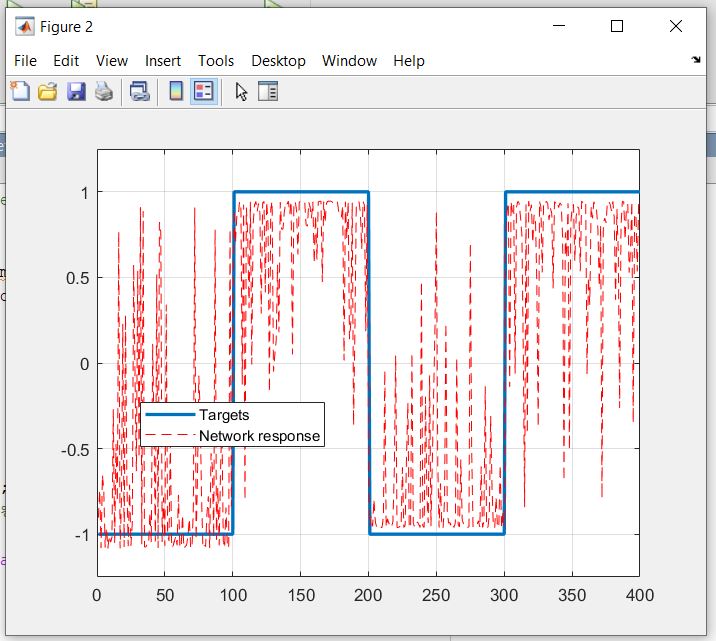
Training Summary:



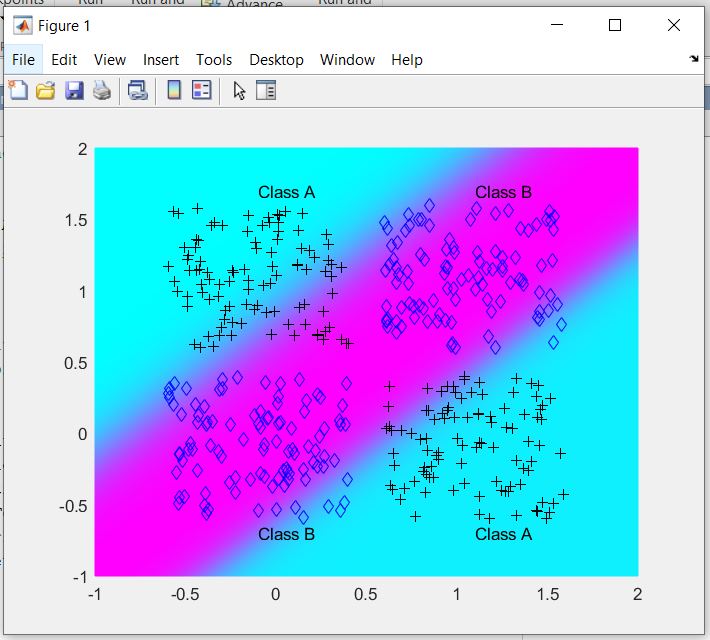
Neural Network:



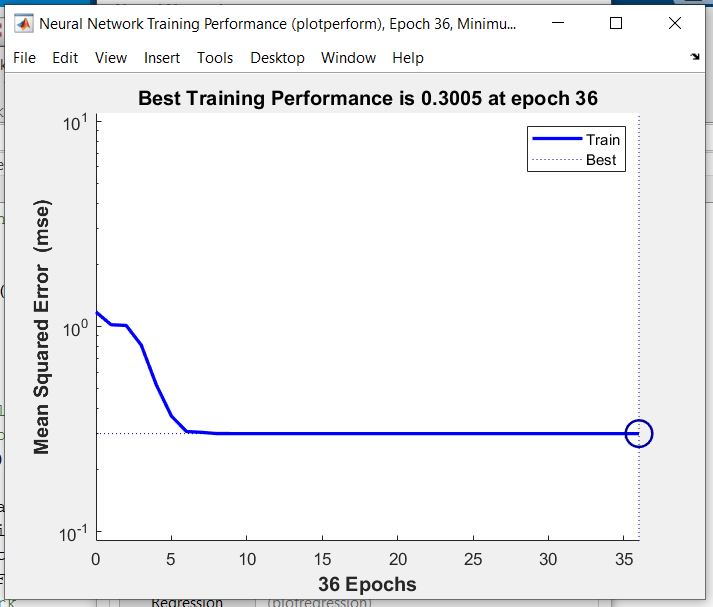
Plot targets and network response:



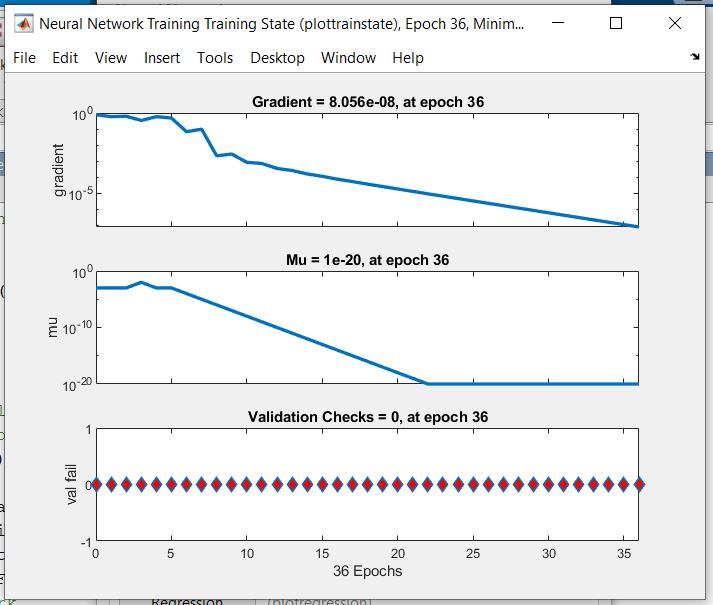
Plot classification result:



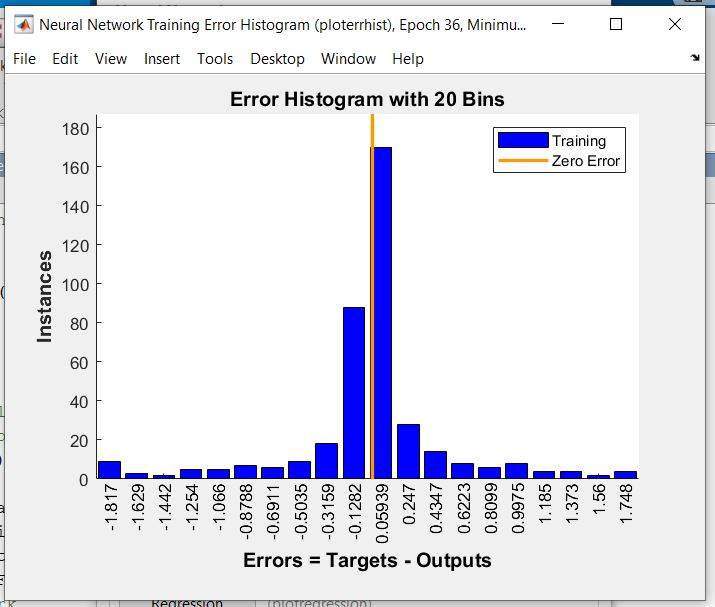
Performance:



Training state:



Error Histogram:



Regression:

