% cauculate output layer

o(i,:) = 1 ./ (1 + exp(-w2 \* z(i,:)')); % Binary Sigmoid function

%Backward phase

% calculate gragient of the output layer

del\_2(i,:) = (t(i,:) - o(i,:)) .\* o(i,:) .\* (1 - o(i,:));

% calculate gragient of the hidden layer

del\_1(i,:) = z(i,:) .\* (1 - z(i,:)) .\* (del\_2(i,:) \* w2(:,2:end)); w2\_new(:,:,i) = lr \* del\_2(i,:)' \* z(i,:);

w1\_new(:,:,i) = lr \* del\_1(i,:)' \* x(i,:); end

% sum of training pattern w2\_new = sum(w2\_new,3); w1\_new = sum(w1\_new,3);

%update weights

w2 = w2 + w2\_new; % update w2 w1 = w1 + w1\_new; % update w2

% mean square error calculation mse(1,epoch) = sum((o-t).^2)/(a\*numOut);

%stopping criteria

if mse(1,epoch) <=0.001 break

end

end

%Plot the learning curve figure

plot(1:epoch, mse, 'b'); grid on

title(['Learning curve at learning rate =',num2str(lr)]) xlabel('epoch');

ylabel('mse')

accuracy=(100-(sum(abs(o-t))/4)\*100) toc

for epoch = 1 : max\_iteration

for i = 1:a

% Forworad phase

% calculate hidden layer

z(i,:) = 1 ./ (1 + exp(-w1 \* x(i,:)')); % Binary Sigmoid function

% output layer's unit size

% learning rate

% Maximum iteration if the program does not break

% MSE initialization

% input-hidden layer weight range -1 to 1

% Hidden-output neuron weight range -1 to 1

% feature vector

% desired output value

% acquire number of training patterns and feature

clc; clear all; close all tic

x = [0 0;0 1;1 0;1 1];

t = [0;1;1;0];

[a, b] = size(x); vectors

numOut = size(t,2); lr = 0.3;

max\_iteration = 100000;

mse = 1;

w1 = 2 \* rand(2,2)- 1; w2 = 2 \* rand(1,2) - 1;