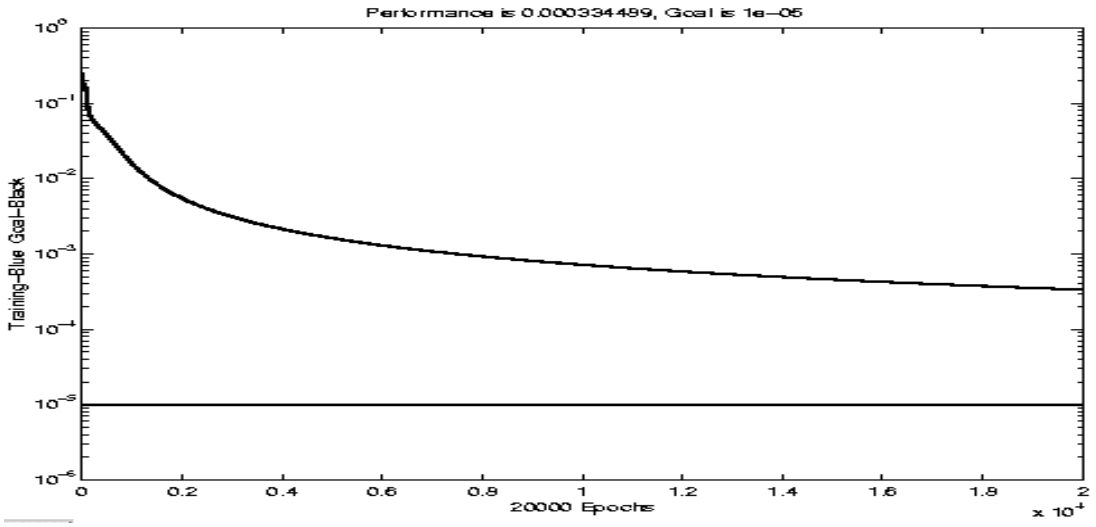
ENEE 434 HW#3 Solutions

1a. Here is the matlab code for the AND logic network. This code is used with small changes for other sections of the question 1. Learning rate is set to 0.5, which is just the middle value in the spectrum, [0, 1].

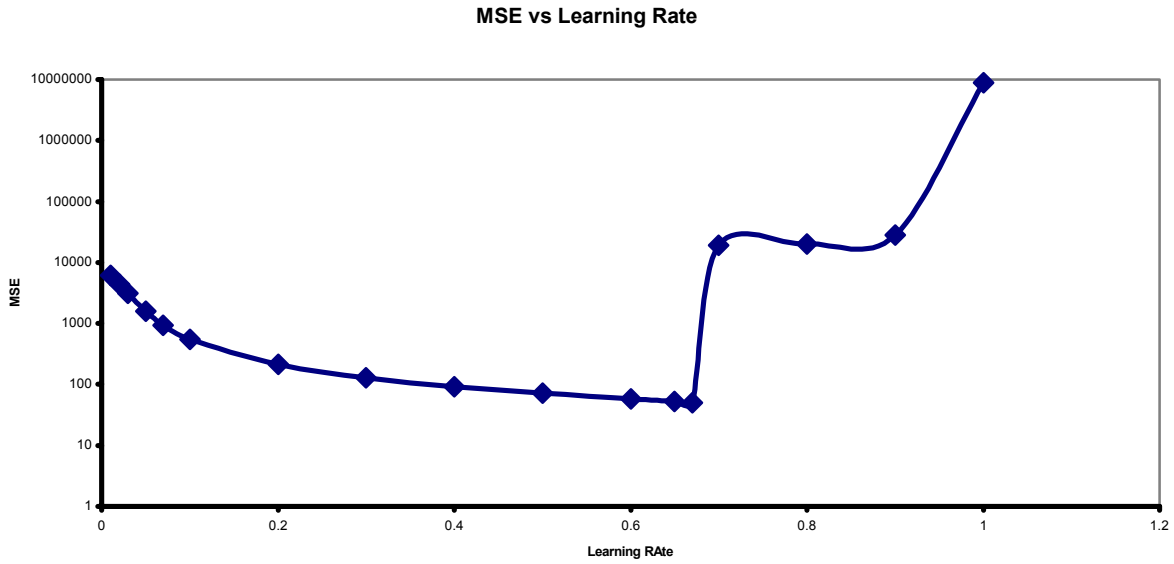
```
% Homework 3 Question 1
function [a, net, tr] = odev2

p = [0 0 1 1; 0 1 0 1];
t = [0 0 0 1];
net = newff(minmax(p), [2,1], {'logsig', 'purelin'}, 'traingd');
net.trainParam.show = 100;
net.trainParam.epochs = 20000;
net.trainParam.lr = 0.5;
net.trainParam.goal = 1e-5;
net.IW{1} = [0 0; 0 0];
net.LW{2} = [0 0];
net.b{1} = [0; 0];
net.b{2} = [0];
[net, tr] = train(net, p, t);
```

After this matlab code is run, the error rate drops to 10^{-3} around 10000 epochs and it saturates at 10^{-4} error rate after 20000 epochs. Then 10000 seem to be a good estimate for optimum number of epochs such that the error converges to a small value. The output figure for this run is shown below.

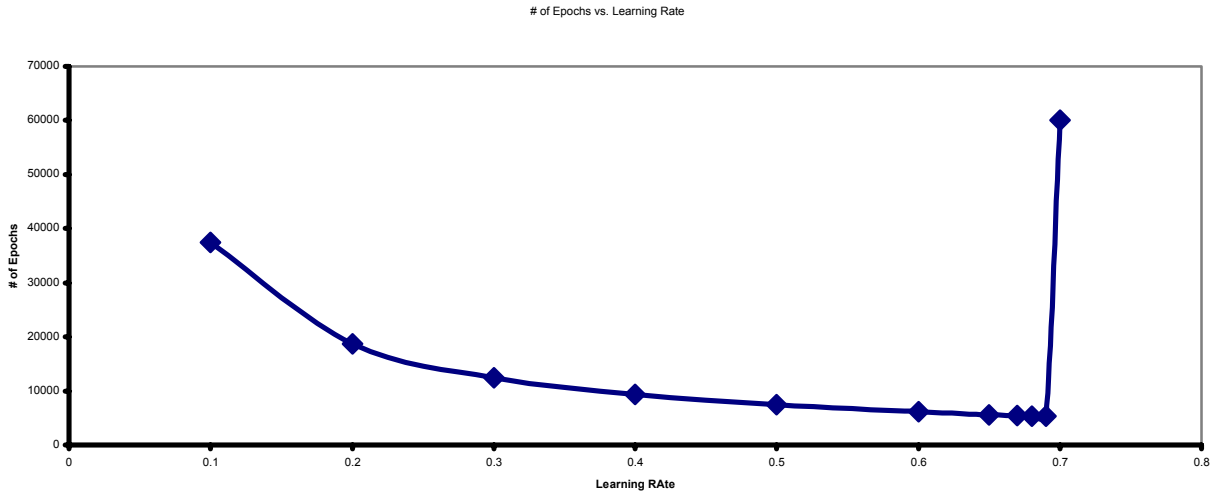


1b. Number of epochs is set to 10000. Running the matlab code for several times with different learning rates, observed error rates (in logarithmic scale) are as follows.



As observed the error rate decreases with increasing learning rate up to ~0.67, then it starts to increase sharp. Also oscillations in the error rate start once the learning rate exceeds 0.6. Then the training algorithm starts to behave unstable as expected.

1c. Since training converges to slow for error rate of 1e-5, and even it does not converge to this rate for small learning rate, error rate is set to 1e-3 for this part. Running the matlab code for different learning rates, the result is as follows:



2. Below is the matlab code that initializes the desired hopfield network, runs it with a random input until it converges to stability points and plots the resulting trajectory.

```
% Homework 3 Question 2
function [y] = odev2

  T = [1 1; -1 -1]';
  net = newhop(T);
  Ai = T;
  [Y, Pf, Af] = sim(net, 2, [], Ai);
  y = Y;

  plot(T(1,1), T(2,1), 'x');
  hold on
  plot(T(1,2), T(2,2), 'x');
  Ai = [-0.4 -0.8; 0.8 0.2]'
  for i = 1:5
     plot(Ai(1,1), Ai(2,1), 'x');
     plot(Ai(1,2), Ai(2,2), 'x');
     [Y, Pf, Af] = sim(net, 2, [], Ai);
     Ai = Y;
  end
```

Once you run this code you get the same state diagram with the one, on page 324 of matlab manual.