ENEE434 Spring 2004
Homework1 Solution: Y.J. 02/25/04

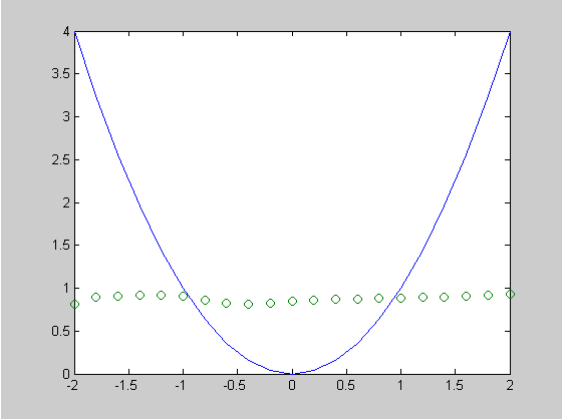
Problem#1:
a)

```
% enee434 hw1# problem1%

   P21=[-2:0.2:2];
   T21=P21.^2;


   P201=[-2:0.02:2];
   T201=P201.^2;

   P401=[-2:0.01:2];
   T401=P401.^2;

%set up the neural network%
   netpara = newff([-2 2],[5 3 1],{'tansig' 'tansig' 'purelin'});

   %net = newff creates a new network with a dialog box.
   %newff(PR,[S1 S2...SNl],{TF1 TF2...TFNl},BTF,BLF,PF) takes,
   %PR - R x 2 matrix of min and max values for R input elements.
   %Si - Size of ith layer, for Nl layers.
   %TFi - Transfer function of ith layer, default = 'tansig'.
   %BTF - Backpropagation network training function, default = 'traingdx'.
   %BLF - Backpropagation weight/bias learning function, default = 'learngdm'.
   %PF - Performance function, default = 'mse'.
   %and returns an N layer feed-forward backprop network.


   Y = sim(netpara,P21);
   plot(P21,T21,P21,Y,'o')

%network after 20 inputs training.%
   netpara.trainParam.epochs = 50;
   netpara = train(netpara,P21,T21);

%Newset of 200 inputs%
   Y = sim(netpara,P201);
   plot(P201,T201,P201,Y,'x');

   %rms erro%
   rms1=sqrt(mse(Y-T201))

%network after 200 inputs training.%
```

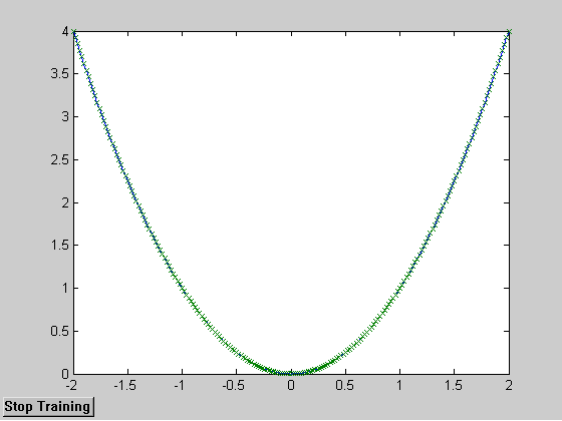
```
    netpara.trainParam.epochs = 50;
    netpara = train(netpara,P201,T201);

%Newset of 200 inputs%

    Y = sim(netpara,P401);
   % plot(P401,T401,P401,Y,'o');

    %rms erro%
    rms2=sqrt(mse(Y-T401))
    rms1
    rms2
```

b)

Before training:



After training:



c) rms1 =

    0.0023

d) rms2 =

  5.7711e-004

rms error is much smaller with more training examplers.

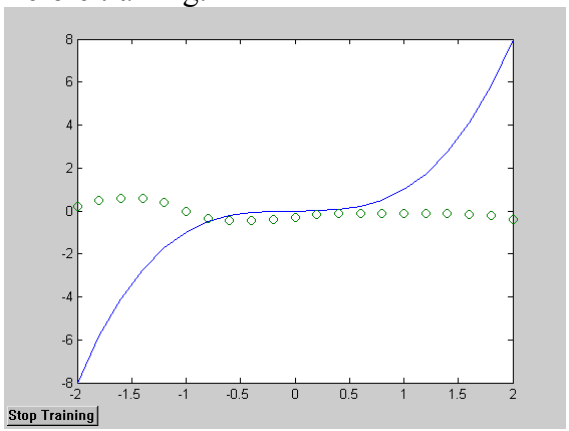
e) Hardlim is not chosen for the output layer for two reasons: 1) For this particular problem, the output is not just 0 and 1. 2) More important reason is that hardlim is not differentiable in the normal sense at 0 (the derivative is an impulse which is not representable in matlab).

The reason for not choosing both input and hidden layer to be purelin is that they would only give linear response yet y=x^2 is a nonlinear function.

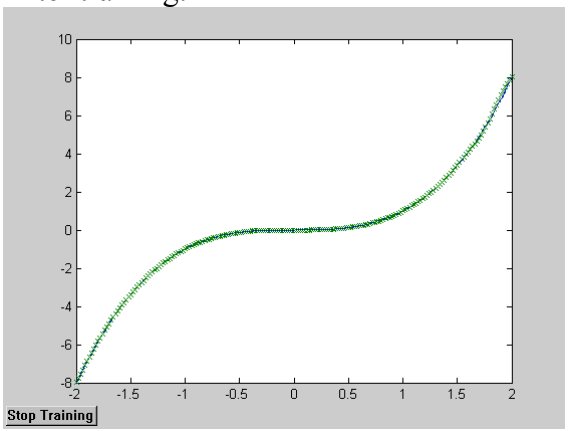
Problem#2:
a) replacing the training examplers by:
% enee434 hw1# problem1%

```
P21=[-2:0.2:2];
T21=P21.^3;

P201=[-2:0.02:2];
T201=P201.^3;
```

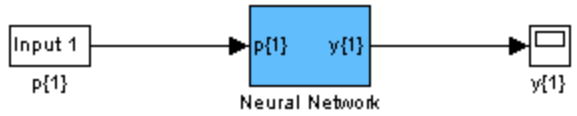
Before training:

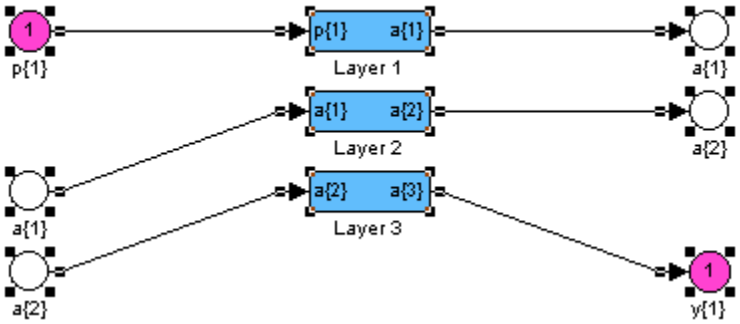


After training:

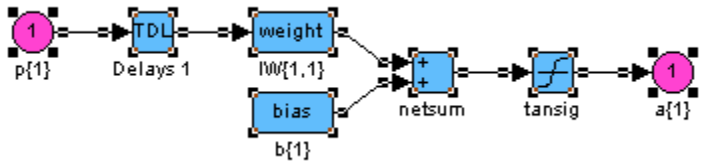


b) gensim(netcub); neural network:

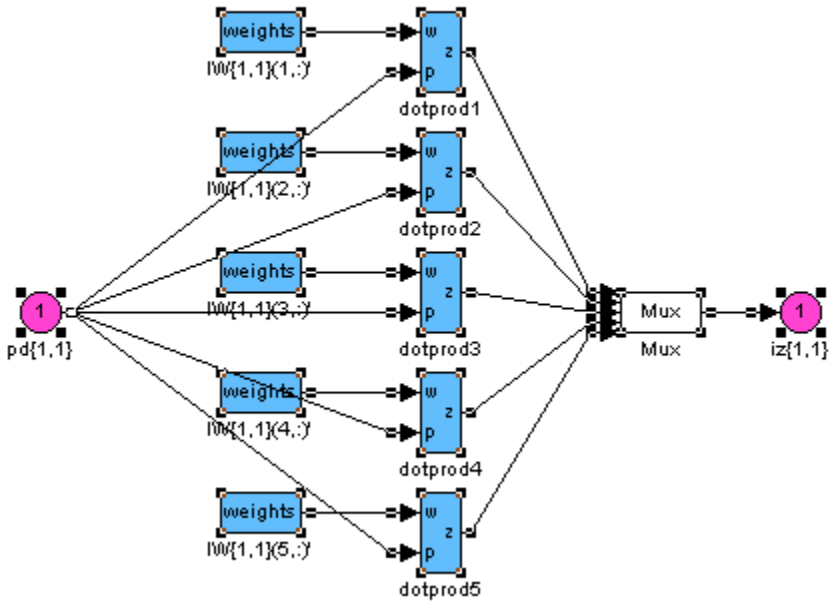




First layer:

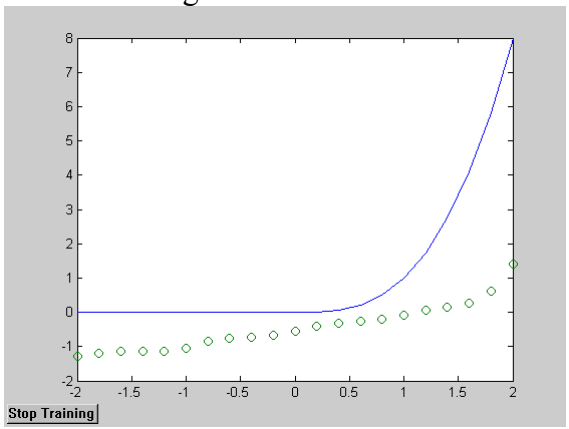


First layer: IW{1, 1}:

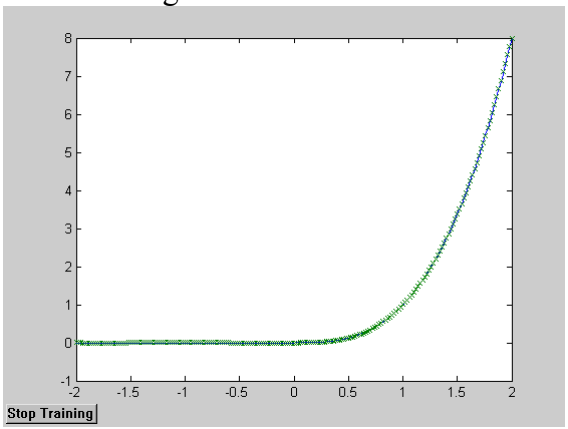


c)

Before training:



After training:



Problem#3:

Training examplers:

```
P =

  -10   -8   -6   -4   -2    0    2    4    6    8   10
   10    8    6    4    2    0   -2   -4   -6   -8  -10

T =

    0    0    0    0    0    0    0    0    1    1    1
```

```
% enee434 hw1# problem3%

%set up the neural network%

   netvec = newff([-10 10;-10 10],[4 4 1],{'tansig' 'logsig' 'logsig'});


   %net = newff creates a new network with a dialog box.

   %newff(PR,[S1 S2...SNl],{TF1 TF2...TFNl},BTF,BLF,PF) takes,

   %PR - R x 2 matrix of min and max values for R input elements.

   %Si - Size of ith layer, for Nl layers.

   %TFi - Transfer function of ith layer, default = 'tansig'.

   %BTF - Backpropagation network training function, default = 'traingdx'.

   %BLF - Backpropagation weight/bias learning function, default = 'learngdm'.

   %PF - Performance function, default = 'mse'.

   %and returns an N layer feed-forward backprop network.



   Y = sim(netvec,P);

   plot(PP,T,PP,Y,'o')
```

```
%network after training.%
   netvec.trainParam.epochs =50;
%    netvec.trainParam.goal=0.01;
   netvec= train(netvec,P,T);
   netvec.IW{1,1}

   %solution 1:
   %Surgested by Jeffrey Krotosky
   netvec.layers{3}.transferFcn='hardlim';
   netvec.IW{1,1}
   Y = sim(netvec,P);
   plot(PP,T,PP,Y,'x');

   %rms erro%
   rms=sqrt(mse(Y-T))

   %solution 2:
   %surgested by RWN:

   netvecn=newff([-10 10;-10 10],[4 4 1],{'tansig' 'logsig' 'hardlim'});
   %define the input weight and layer weight of the newnetwork:
   %input weight for first layer of nerons {1, and first input 1},
   netvecn.IW{1,1}=netvec.IW{1,1};
   %layer weight for second layer neurons {2, from first layer of neurons 1}
   netvecn.LW{2,1}=netvec.LW{2,1};
```

```
%layer weight for third layer neurons {3, from second layer of neurons 2}
netvecn.LW{3,2}=netvec.LW{3,2};
%define the bias:
%weight for first layer of neurons:
netvecn.b{1}=netvec.b{1};
netvecn.b{2}=netvec.b{2};
netvecn.b{3}=netvec.b{3};
Y = sim(netvec,P);
plot(PP,T,PP,Y,'*');
gensim(netvec)
```

After Training:

Target:

T =

0 0 0 0 0 0 0 0 1 1 1

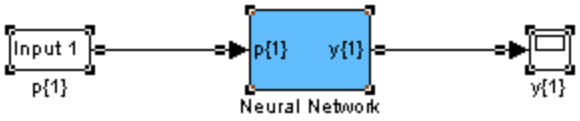
Output:

>> Y

Y =

0 0 0 0 0 0 0 0 1 1 1

b) gensim(netvet):



Input	[9, -7]	[7, 3]	[5.01, -2.99]	[4, -4]
Output	1	0	0 error	0