ENEE434 Spring 2005
Homework2 Solution: Y.J. 02/15/04

Problem1: (E4.7 page 4-37 from text book)
i) Matlabcode: (use newp)

```
% enee434 spring2005 hw2# problem#1%

% Input vector p%
   p=[[1;4] [1;5] [2;4] [2;5] [3;1] [3;2] [4;1] [4;2]];

% Target output t%
   t=[0 0 0 0 1 1 1 1];

% setup the neural network netFWAF%

   netFWAF=newp([0 5;0 5],1);
      %NET = NEWP(PR,S,TF,LF) takes these inputs,
            %PR - Rx2 matrix of min and max values for R input elements.
            %S  - Number of neurons.
            %TF - Transfer function, default = 'hardlim'.
            %LF - Learning function, default = 'learnp'.%
   net.trainParam.epochs=50;

% output y before training%
   y = sim(netFWAF,p);

% Training the newwork netFWAF with p and t%
   netFWAF=train(netFWAF, p, t);

% Weight and bias value%
   netFWAF.iw{1,1}
   netFWAF.b{1,1}

% output y after training%
   y = sim(netFWAF,p);

% enee434 spring2005 hw2# problem#1 end%
```

ii) The weight and bias after training:

Weight:	Bias:
netFWAF.iw{1,1} ans = 2 -3	netFWAF.b{1,1} ans = 0

iii) Checking if the output y after training agree with target:

```
p =
   1   1   2   2   3   3   4   4
   4   5   4   5   1   2   1   2
t =
   0   0   0   0   1   1   1   1
```

before training the neural network output:
y =
    1   1   1   1   1   1   1   1

after training the neural network output:
y =
    0   0   0   0   1   1   1   1
clearly y=t.

Problem2: (E5.9 page 5.29 from text book)

$x= [1\ 2\ 2]^T$

$v_1= [-1\ 1\ 0]^T$, $v_2= [1\ 1\ 2]^T$, $v_3= [1\ 1\ 0]^T$,

B=[-1 1 1, 1 1 1, 0 -2 0],

B^{-1}=[-0.5 0.5 0, 0 0 -0.5, 0.5 0.5 0.5],

$r_1= [-0.5\ 0.5\ 0]^T$, $r_2= [0\ 0\ -0.5]^T$, $r_3= [0.5\ 0.5\ 0.5]^T$,

$x_1^v=r_1^T*x= [-0.5\ 0.5\ 0]*[1\ 2\ 2]^T =0.5$,
$x_2^v=r_2^T*x= [0\ 0\ -0.5]*[1\ 2\ 2]^T =-1$,
$x_3^v=r_3^T*x= [0.5\ 0.5\ 0.5]*[1\ 2\ 2]^T =2.5$

$x=[1\ 2\ 2]^T =0.5*[-1\ 1\ 0]^T -1*[1\ 1\ 2]^T +2.5*[1\ 1\ 0]^T$.

Problem3:
a)give the training pairs p and t: (p is 2*39 and t is 1*39)
$p=[n, IT]^T$,

p =
  Columns 1 through 7
   -3.0000   -2.5000   -2.0000   -1.5000   -1.0000   -0.5000   0
    0.5000   0.5000   0.5000   0.5000   0.5000   0.5000   0.5000
  Columns 8 through 14
    0.5000   1.0000   1.5000   2.0000   2.5000   3.0000   -3.0000
    0.5000   0.5000   0.5000   0.5000   0.5000   0.5000   1.5000
  Columns 15 through 21
   -2.5000   -2.0000   -1.5000   -1.0000   -0.5000   0   0.5000
    1.5000   1.5000   1.5000   1.5000   1.5000   1.5000   1.5000
  Columns 22 through 28
    1.0000   1.5000   2.0000   2.5000   3.0000   -3.0000   -2.5000
    1.5000   1.5000   1.5000   1.5000   1.5000   2.5000   2.5000
  Columns 29 through 35
   -2.0000   -1.5000   -1.0000   -0.5000   0   0.5000   1.0000
    2.5000   2.5000   2.5000   2.5000   2.5000   2.5000   2.5000
  Columns 36 through 39
    1.5000   2.0000   2.5000   3.0000

```
   2.5000    2.5000    2.5000    2.5000
```

t=a(n) from hw#1, problem3. (**Notice t(1, 28) and t(1, 38) are not real by using a(n)**).

```
t =
  Columns 1 through 4
  -0.5000         -0.5000         -0.5000         -0.5000
  Columns 5 through 8
  -0.5000         -0.4330               0          0.4330
  Columns 9 through 12
   0.5000          0.5000          0.5000          0.5000
  Columns 13 through 16
   0.5000         -1.5000         -1.5000         -1.5000
  Columns 17 through 20
  -1.2990         -1.4142         -0.8292               0
  Columns 21 through 24
   0.8292          1.4142          1.2990          1.5000
  Columns 25 through 28
   1.5000          1.5000         -2.5000               0 - 2.7951i
  Columns 29 through 32
  -2.0000         -2.4875         -2.0000         -1.0897
  Columns 33 through 36
        0          1.0897          2.0000          2.4875
  Columns 37 through 39
   2.0000               0 + 2.7951i   2.5000
```

b) Set up the network: (matlab code is attached in the end of this problem).
Important command lines:

```
netan = newff([-3 3; 0 2.5],[5 3 1],{'tansig' 'tansig' 'purelin'});
netan.trainParam.epochs = 50;
netan=train(netan, p, t);
```

c) test vector Ptest(pt), desired output for Ptest Ttest(tt): Ptest(pt)=$[n, IT]^T$, (2*26):

```
pt =
  Columns 1 through 7
   -3.0000   -2.5000   -2.0000   -1.5000   -1.0000   -0.5000         0
    0.2000    0.2000    0.2000    0.2000    0.2000    0.2000    0.2000
  Columns 8 through 14
    0.5000    1.0000    1.5000    2.0000    2.5000    3.0000   -3.0000
    0.2000    0.2000    0.2000    0.2000    0.2000    0.2000    1.2000
  Columns 15 through 21
   -2.5000   -2.0000   -1.5000   -1.0000   -0.5000         0    0.5000
    1.2000    1.2000    1.2000    1.2000    1.2000    1.2000    1.2000
  Columns 22 through 26
    1.0000    1.5000    2.0000    2.5000    3.0000
    1.2000    1.2000    1.2000    1.2000    1.2000
```

Desired output Ttest(tt):

```
tt =
  Columns 1 through 7
   -0.2000   -0.2000   -0.2000   -0.2000   -0.2000   -0.2000         0
```

```
  Columns 8 through 14
    0.2000    0.2000    0.2000    0.2000    0.2000    0.2000   -1.2000
  Columns 15 through 21
   -1.2000   -1.2000   -1.2000   -1.1832   -0.7331         0    0.7331
  Columns 22 through 26
    1.1832    1.2000    1.2000    1.2000    1.2000
```

Command to calculate mean square error between neural net output y and desired output tt:

```
rms=sqrt(mse(y-tt))
```

```
rms =
   0.7275 - 0.6461i
```

Since there is imaginary number involved, you might get error message.

The matlab code:

```
% enee434 spring2005 hw2# problem#3%

% Input vector p and output t%
   for i=1:13;
      p(1,i)=-3.5+0.5*i;
      p(2,i)=0.5;
      if p(1,i)<-0.5
         t(i)=-0.5;
      elseif p(1,i)<=0.5
         t(i)=p(1,i)*sqrt(2*0.5-p(1, i)^2);
      else
         t(i)=0.5;
      end
   end

   for i=14:26;
      p(1, i)=-3.5+0.5*i-6.5;
      p(2, i)=1.5;
      if p(1,i)<-1.5
        t(i)=-1.5;
      elseif p(1,i)<=1.5
         t(i)=p(1,i)*sqrt(2*1.5-p(1, i)^2);
      else
           t(i)=1.5;
      end
   end

   for i=27:39;
      p(1, i)=-3.5+0.5*i-13;
      p(2, i)=2.5;
      if p(1,i)<-2.5
         t(i)=-2.5;
      elseif p(1,+i)<=2.5
         t(i)=p(1,i)*sqrt(2*2.5-p(1, i)^2);
```

```
      else
         t(i)=2.5;
      end
   end

% Input vector testing vector pt and testing output tt%
   for i=1:13;
      pt(1,i)=-3.5+0.5*i;
      pt(2,i)=0.2;
      if pt(1,i)<-0.2
         tt(i)=-0.2;
      elseif pt(1,i)<=0.2
         tt(i)=pt(1,i)*sqrt(2*0.2-pt(1, i)^2);
      else
         tt(i)=0.2;
      end
   end

   for i=14:26;
      pt(1, i)=-3.5+0.5*i-6.5;
      pt(2, i)=1.2;
      if pt(1,i)<-1.2
        tt(i)=-1.2;
      elseif pt(1,i)<=1.2
         tt(i)=pt(1,i)*sqrt(2*1.2-pt(1, i)^2);
      else
         tt(i)=1.2;
      end
   end

% setup the neural network netan%

   netan = newff([-3 3; 0 2.5],[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.
   netan.trainParam.epochs = 50;

% Training the newwork netan with p and t%
   netan=train(netan, p, t);

% output y after training with in put pt%
   y = sim(netan,pt);
```

```
%rms error between the neural network output y and desired target tt%
   rms=sqrt(mse(y-tt))

% enee434 spring2005 hw2# problem#3 end%
```