Rongrong Su

20751802

Q1

The first 5 components of the optimum value of the logistic beta parameter:

Beta0:-15.7669974993069,

Beta1:-0.0986760937308624,

Beta2:-0.0276040411901145,

Beta3:0.0376605690276412,

Beta4:0.000212393352265759

Error rate of training data: 0.0146150679291890

Error rate of test data: 0.480932203389831

Code:

```
load('faces')
training_data=[train_faces' train_nonfaces'];
% (This will be a 361 by 4858 matrix.)
test_data=[test_faces' test_nonfaces'];
% (This will be a 361 by 944 matrix.)
x=[ones(1,size(training_data,2));training_data];
x_test=[ones(1,size(test_data,2));test_data]';
size1 = size(x,2);
size2 = size(x,1);
size3 = size(train_faces,1);
size4 = size(train_nonfaces,1);
y=[ones(1,size3) zeros(1,size4)]';
y2=[ones(1,472) zeros(1,472)]';
w = zeros(size1,size1);
beta0 = zeros(size2,1);
for k=1:100
    pi = zeros(size1,1);
    for i = 1:size1
        pi(i,1)=exp(beta0'*x(:,i))/(1+exp(beta0'*x(:,i)));
    end
    for i=1:size1
        w(i,i)=pi(i,1)*(1-pi(i,1));
    end
    hmatrix=x*w*x';
    beta0=beta0+inv(hmatrix)*x*(y-pi);
```

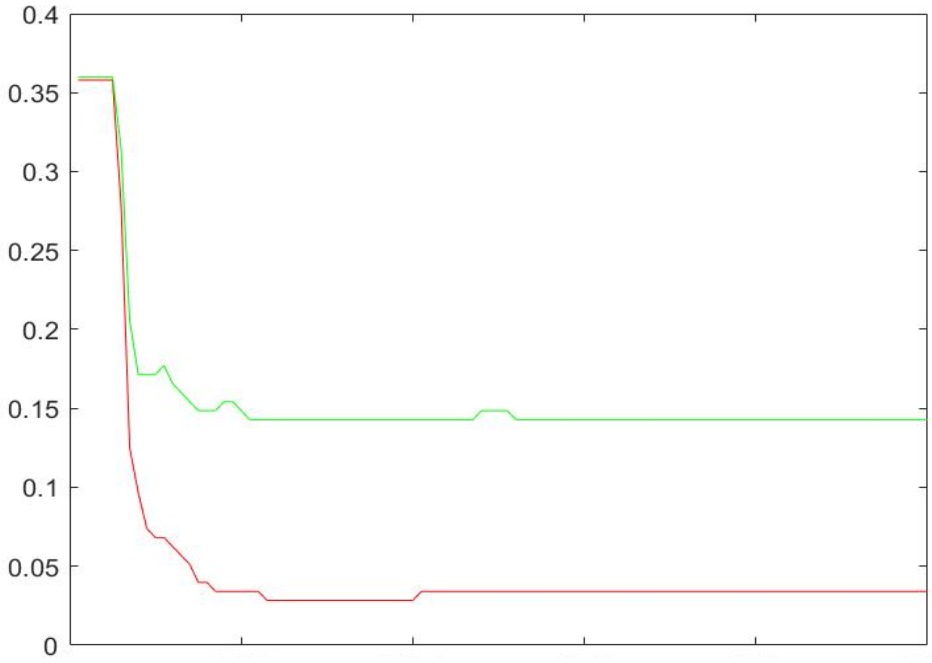
```
end
result=[pi>=0.5];
num=sum(y~=result);
errorrate1=num/size1;
pi2=zeros(size(x_test,2),1);
for i = 1:size(x_test,2);
      pi2(i,1)=exp(beta0'*x_test(:,i))/(1+exp(beta0'*x_test(:,i)));
end
result2=[pi2>=0.5];
num2=sum(y2~=result2);
errorrate2=num2/size(x_test,2);
```

Q2

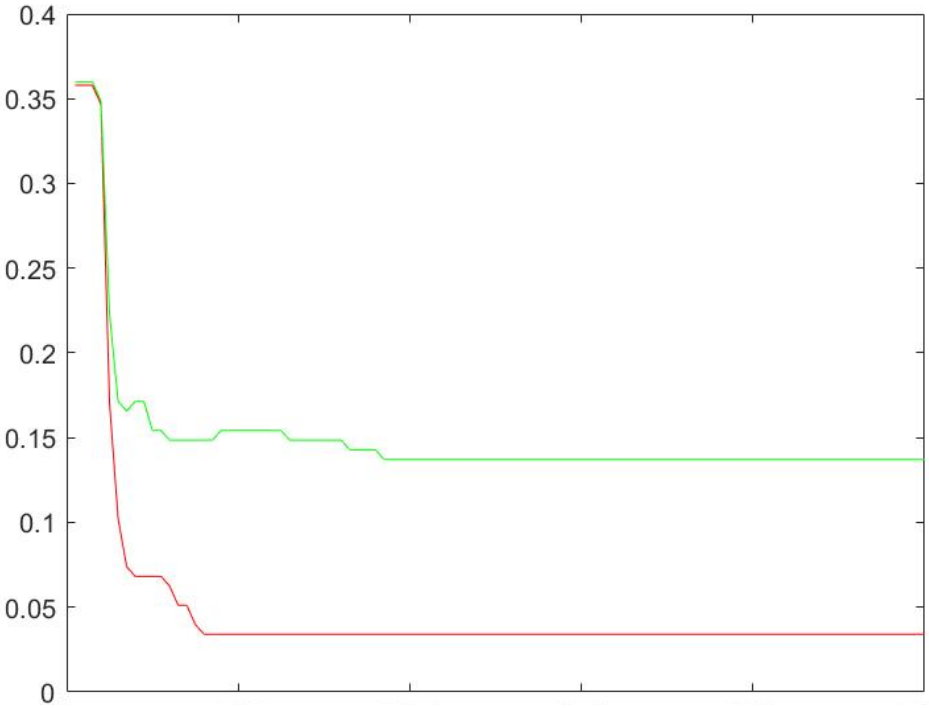
(b)

weight=0.05

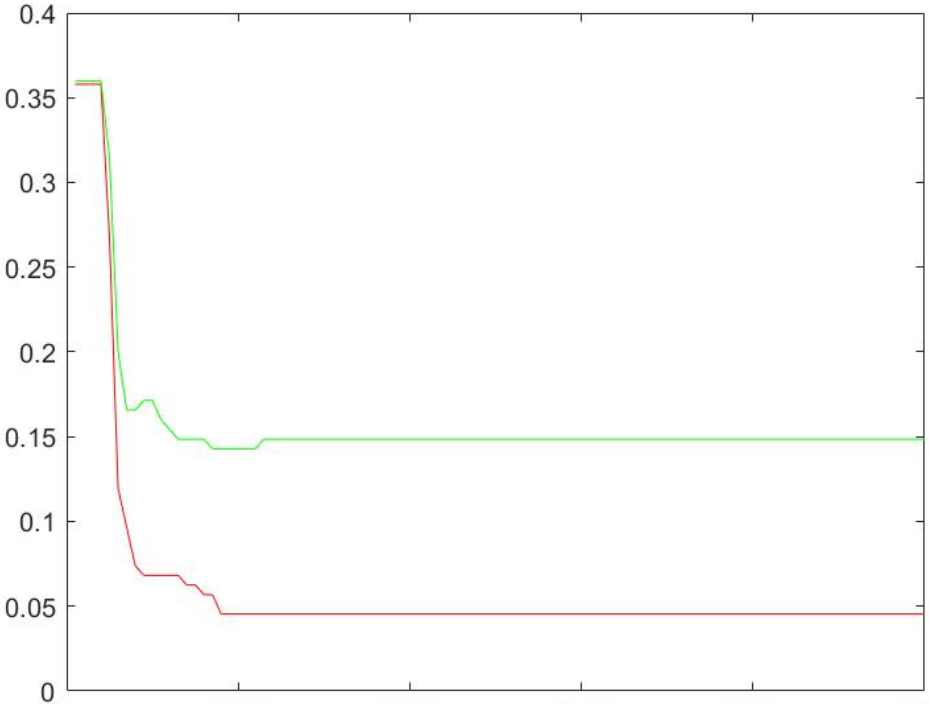
epoch=500



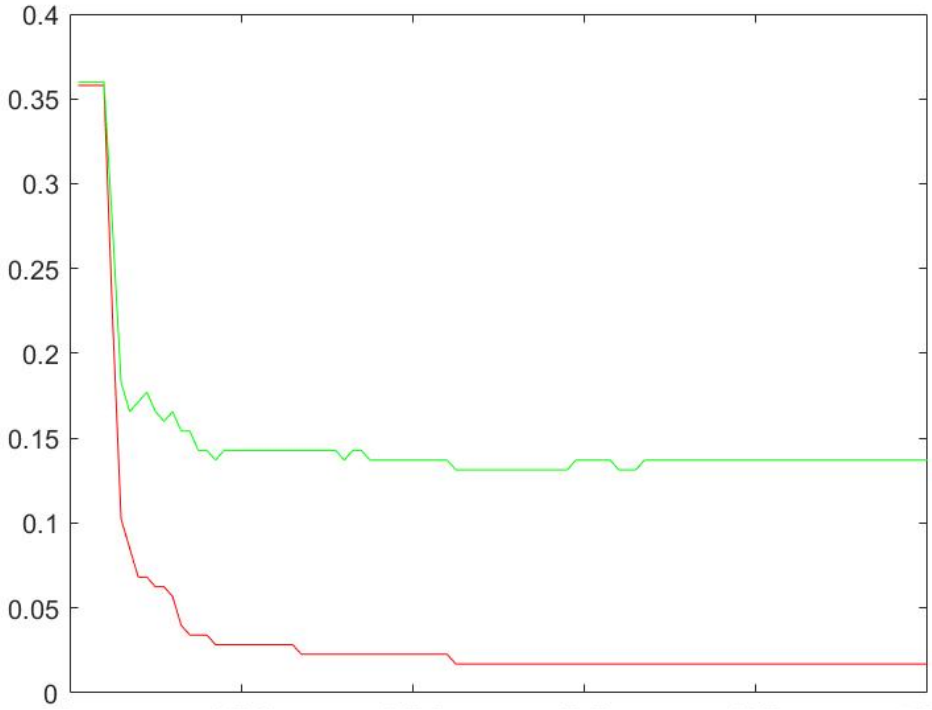
weight=0.01



weight=0.015



weight=0.001



I notice that the gap between training error and test error increases as epoch increases and the training error is smaller than test error(this is because overfitting)

(c)

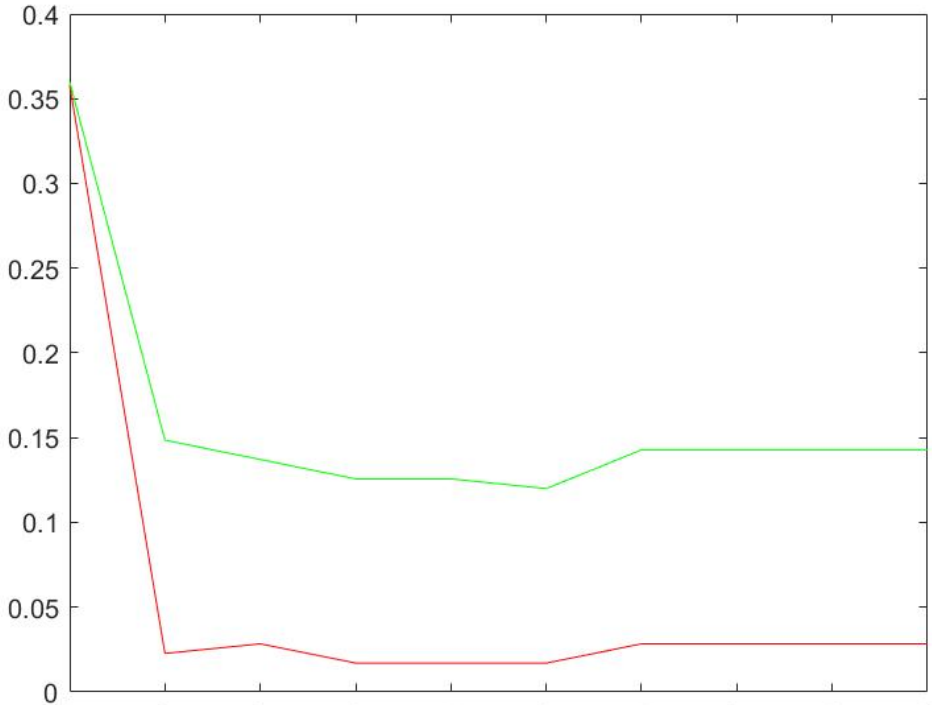
epoch = 100

learning rate= 0.01

decay = 0

maximum nodes=10

The number of hidden nodes I choose is 5 which allow model fits train data well and avoid overfitting.



(d)

with epoch=100 nodes=10 learning rare=0.01

	h(x)=0	h(x)=1
Y=0	109	3
Y=1	18	45

Rongrong Su

Q3 $f(x_i;\beta) = \left(\frac{e^{\beta^T x_i}}{1+e^{\beta^T x_i}}\right)^{y_i} \left(\frac{1}{1+e^{\beta^T x_i}}\right)^{1-y_i}$

$f(\beta) = -l(\beta) = -\sum_{i=1}^{n} \log f(x_i, \beta)$

$= -\sum_{i=1}^{n} \left(y_i \cdot \log\left(\frac{e^{\beta^T x_i}}{1+e^{\beta^T x_i}}\right) + (1-y_i) \cdot \log\left(\frac{1}{1+e^{\beta^T x_i}}\right) \right)$

$= -\sum_{i=1}^{n} \left[y_i \cdot (\beta^T x - \log(1+e^{\beta^T x_i}) - (1-y_i) \log(1+e^{\beta^T x}) \right]$

$= -\sum_{i=1}^{n} \left[y_i \beta^T x_i - \log(1+e^{\beta^T x_i}) \right]$

$\frac{\partial f(\beta)}{\partial \beta} = \sum_{i=1}^{n} \left(\frac{x_i e^{\beta^T x_i}}{1+e^{\beta^T x}} - y_i x_i^T \right)$.

$\frac{\partial f(\beta)}{\partial \beta \partial \beta^T} = \sum_{i=1}^{n} \left(\frac{e^{\beta^T x}}{1+e^{\beta^T x}} \cdot \frac{1}{e^{\beta^T x}+1} \cdot x_i \cdot x_i^T \right)$ (X is not a zero matrix so $X \cdot X^T > 0$)

$\therefore$ We notice $\frac{\partial f(\beta)}{\partial \beta \cdot \partial \beta^T} > 0$. $\therefore$ So it is a convex with repect to β

Suppose there exists 2 minimum value β_1 β_2 ($f(\beta_1) < f(\beta_2)$) then by the property of convexity of function 对于任意的 $f(\lambda\beta_1 + (1-\lambda)\beta_2) < \lambda f(\beta_1) + (1-\lambda) f(\beta_2) < f(\beta_2)$ Contradiction

So. $f(\beta)$ has a unique minimum value.

4 (a) $\mathrm{cov}(\tilde{X}) = [I - u_1 u_1^T) \cdot X \cdot [(I - u_1 u_1^T) X]^T$

$= \frac{1}{n} [(I - u_1 u_1^T) \cdot X \cdot X^T \cdot (I - u_1 u_1^T)^T]$

$= \frac{1}{n} \cdot [(I - u_1 u_1^T) \cdot XX^T (I - u_1 u_1^T)]$

$= \frac{1}{n} (XX^T - u_1 u_1^T XX^T - XX^T u_1 u_1^T + u_1 u_1^T XX^T u_1 u_1^T)$

$= \frac{1}{n} (XX^T - u_1 n\lambda u_1^T - n \cdot \lambda u_1 u_1^T + u_1 n\lambda u_1^T \cdot u_1 u_1^T)$

$= \frac{1}{n} (XX^T - 2n\lambda u_1 u_1^T + n\lambda u_1 u_1^T)$

$= \frac{1}{n} (XX^T + n\lambda u_1 u_1^T)$

$= \frac{1}{n} XX^T - \lambda u_1 u_1^T$.

(b) $\tilde{S} = \frac{1}{n} XX^T - \lambda \cdot u_1 u_1^T$

$= S - \frac{1}{n} XX^T u_1 u_1^T$

$= S - S u_1 u_1^T$

$\tilde{S} \cdot u_1 = S u_1 - S u_1 u_1^T \cdot u_1$

$= 0$.

for $j+1$

$\tilde{S} \cdot u_j = S u_j - S u_1 u_1^T u_j$

$= S u_j$

$= \lambda_j u_j$

So u_j be the j-th eigenvector with largest eigenvalues of $\tilde{S}$

So $V = u_2$.

5 $\beta : (d+1)(k-1)$ k classes d-dimentional

X: $d+1$-dimention add 1 to the first row like $\begin{pmatrix} 1 \\ x_1 \\ x_d \end{pmatrix}$ (for one column)

then $P(y=j|X) = \frac{e^{\beta_j X}}{1 + \sum_{j=1}^{k-1} e^{\beta_j X}}$ (for $j \in (1, k-1)$)

$$P(y=k|X) = \frac{1}{1 + \sum_{j=1}^{k-1} e^{\beta_j X}}$$

$$l(\beta) = \log \prod_{i=1}^{k} p_i^{y_i} = \sum_{i=1}^{k} y_i \beta_i^T X - \log\left[1 + \sum_{j=1}^{k} \exp(\beta_j^T X)\right]$$

$$\frac{\partial l(\beta)}{\partial \beta} = \sum_{j=1}^{k} y_i X^T - \frac{\exp(\beta_j X)}{1 + \sum_{j=1}^{k-1} \exp(\beta_j^T X)} X^T$$

$= (y - p_{i,j}) X^T$ (Since we want to find β s.t. $\frac{\partial l(\beta)}{\partial \beta} = 0$ then we use Newton-Raphson algorithm)

$$= \frac{\partial l(\beta)}{\partial \beta \partial \beta^T} = \sum \frac{-\exp(\beta_j X^T)\left(1 + \sum_{j=1}^{k-1} \exp(\beta_j^T X)\right) - \exp(\beta_j X^T)\exp(\beta_j^T X)}{\left[1 + \sum_{j=1}^{k-1} \exp(\beta_j^T X)\right]^2} XX^T$$

$$= \sum_{j=1}^{k} p_j(1 - p_j) XX^T$$

Then ① initialize β^0

② $\beta^{i+1} = \beta^i - \frac{l(\beta)}{l'(\beta)}$

③ repeat until convergence (i.e. $|\beta^{k+1} - \beta^k| < \varepsilon$)

6

(a) $P(X|Y_i) = P(X|\mu_i, \Sigma) = \frac{1}{(2\pi)^{\frac{1}{2}} |\Sigma|^{\frac{1}{2}}} \exp\left(-\frac{1}{2}(X-\mu_i)^T \Sigma^{-1} (X-\mu_i)\right)$

$$P(Y_i|X) = \frac{P(X|Y_i)\, P(Y_i)}{P(X)} = \frac{P(X|Y_i)\, \pi_i}{P(X)}$$

So $L(\pi_1) = \prod_{i=1}^{n_1} \frac{p(x|\gamma_1)\cdot\pi_1}{p(x)} \cdot \prod_{i=1}^{n_2} \frac{p(x|\gamma_2)(1-\pi_1)}{p(x)}$

$\ell(\pi_1) = \sum_{i=1}^{n_1} \log\pi_1 + \sum_{i=1}^{n_2} \log(1-\pi_1) + \sum_{i=1}^{n_1} \log p(x|\gamma_1) + \sum_{i=1}^{n_2} \log p(x|\gamma_2) - \sum_{i=1}^{n_1+n_2} p(x)$

$\frac{\partial \ell(\pi_1)}{\partial \pi_1} = n_1\pi_1 - \frac{n_2}{1-\pi_1}$

$\frac{\partial \ell(\pi_1)}{\partial \pi_1} = 0 \Rightarrow n_1\pi_1 - \frac{n_2}{1-\pi_1} = 0 \Rightarrow \hat{\pi}_1 = \frac{n_1}{n_1+n_2}$. So $\hat{\pi}_2 = \frac{n_2}{n_1+n_2}$.

(b) for class 1

$p(x|\gamma_1) = \frac{1}{(2\pi)^{\frac{1}{2}}|\Sigma|^{\frac{1}{2}}} \exp\left(-\frac{1}{2}(x-\mu_1)^T\Sigma^{-1}(x-\mu_1)\right)$

$L(\mu) = \prod_{i=1}^{n_1} \frac{1}{(2\pi)^{\frac{1}{2}}|\Sigma|^{\frac{1}{2}}} \exp\left(-\frac{1}{2}(x-\mu_1)^T\Sigma^{-1}(x-\mu_1)\right)$

$\ell(\mu) = \sum_{i=1}^{n_1} \left(-\frac{1}{2}\log(2\pi) - \frac{1}{2}\log|\Sigma| - \frac{1}{2}(x-\mu_1)^T\Sigma^{-1}(x-\mu_1)\right)$

$\frac{\partial \ell(\mu)}{\partial \mu} = \sum_{i=1}^{n_1} (x-\mu_1)\cdot\Sigma^{-1} = 0 \Rightarrow$ so $\hat{\mu}_1 = \frac{1}{n_1}\sum_{i=1}^{n_1} x_i$

So for the same reason $\hat{\mu}_2 = \frac{1}{n_2}\sum_{i=1}^{n_2} x_i$

(c) $L(\Sigma) = \prod_{i=1}^{n_1} \frac{p(x|\gamma_1)\,p(\gamma_1)}{p(x)} \cdot \prod_{i=1}^{n_2} \frac{p(x|\gamma_2)\cdot p(\gamma_2)}{p(x)}$

$\ell(\Sigma) = \sum_{i=1}^{n_1} \log\pi_1 + \sum_{i=1}^{n_1}\left(-\frac{1}{2}\log(2\pi) - \frac{1}{2}\log|\Sigma| - \frac{1}{2}(x-\mu_1)^T\Sigma^{-1}(x-\mu_1)\right) - \log p(x) +$
$\sum_{i=1}^{n_2} \log\pi_2 + \sum_{i=1}^{n_2}\left(-\frac{1}{2}\log 2\pi - \frac{1}{2}\log|\Sigma| - \frac{1}{2}(x-\mu_2)^T\Sigma^{-1}(x-\mu_2)\right) - \log p(x)$

$\frac{\partial \ell(\Sigma)}{\partial \Sigma} = \frac{1}{2}\left[\sum_{i=1}^{n_1}(x_i-\hat{\mu}_{1k})^T(x_i-\hat{\mu}_{1k}) + \sum_{i=1}^{n_2}(x_i-\hat{\mu}_{2k})^T(x_i-\hat{\mu}_{1k})\right](\Sigma^{-1})^2 - \frac{1}{2}(n_1+n_2)\Sigma^{-1} = 0$

$\hat{\Sigma} = \frac{\sum_{i=1}^{n_1}(x_i-\hat{\mu}_1)(x_i-\hat{\mu}_1)^T + \sum_{i=1}^{n_2}(x_i-\hat{\mu}_2)(x_i-\hat{\mu}_2)^T}{n_1+n_2}$

the unbiased estimator of $\hat{\Sigma}$ is $\frac{\sum_{i=1}^{n_1}(x_i-\hat{\mu}_1)(x_i-\hat{\mu}_1)^T + \sum_{i=1}^{n_2}(x_i-\hat{\mu}_2)(x_i-\hat{\mu}_2)^T}{n_1+n_2-2}$

Code for Q1

```
load('faces')
training_data=[train_faces' train_nonfaces'];
% (This will be a 361 by 4858 matrix.)
test_data=[test_faces' test_nonfaces'];
% (This will be a 361 by 944 matrix.)
x=[ones(1,size(training_data,2));training_data];
x_test=[ones(1,size(test_data,2));test_data]';
size1 = size(x,2);
size2 = size(x,1);
size3 = size(train_faces,1);
size4 = size(train_nonfaces,1);
y=[ones(1,size3) zeros(1,size4)]';
y2=[ones(1,472) zeros(1,472)]';
w = zeros(size1,size1);
beta0 = zeros(size2,1);
for k=1:100
    pi = zeros(size1,1);
    for i = 1:size1
        pi(i,1)=exp(beta0'*x(:,i))/(1+exp(beta0'*x(:,i)));
    end
    for i=1:size1
        w(i,i)=pi(i,1)*(1-pi(i,1));
    end
    hmatrix=x*w*x';
    beta0=beta0+inv(hmatrix)*x*(y-pi);

end
result=[pi>=0.5];
num=sum(y~=result);
errorrate1=num/size1;
pi2=zeros(size(x_test,2),1);
for i = 1:size(x_test,2);
    pi2(i,1)=exp(beta0'*x_test(:,i))/(1+exp(beta0'*x_test(:,i)));
end
result2=[pi2>=0.5];
num2=sum(y2~=result2);
errorrate2=num2/size(x_test,2);
```

Code for Q2

```
(b)
size1=size(Xtrain,1);
size2=size(Xtrain,2);
nodes = 5;
epoch = 500;
I_matrix = -1*ones(size1,nodes);
u1 = I_matrix+rand(size1,nodes)*2; %33*5
u2 = -1*ones(nodes,1)+rand(nodes,1)*2; %5*1
layer1_z = zeros(nodes,size2); %5*176
layer1_a = zeros(nodes,size2); %5*176
```

```
y_input = zeros(1,size2); %1*176
y_output = zeros(1,size2);
lr=0.02; %learning rate
decay=0; %weight decay
delta_u2=zeros(nodes,1);
p1=zeros(nodes,1);
p2=zeros(nodes,1);
delta_u1=zeros(size1,nodes);
train_error=zeros(epoch,1);
test_error=zeros(epoch,1);

for e=1:epoch

      for i = 1:size2;
          layer1_a(:,i)=u1'*Xtrain(:,i);%5*1
          layer1_z(:,i)=1/(1+exp((layer1_a(:,i))*(-1)));
          y_input(1,i)=u2'*layer1_z(:,i);


          %from output to layer1
          delta_u2=-2*(ytrain(i,1)-y_input(1,i));%1

          %from layer1 to input
           delta_u1=delta_u2*Xtrain(:,i)*(u2.*layer1_z(:,i).*(ones(nodes,1)-
layer1_z(:,i)))';%33*5


          u2=u2-(lr*delta_u2)-(u2*lr*decay);
          u1=u1-(lr*delta_u1)-(u1*lr*decay);
      end
     %test train matrix
     train_class=zeros(size(Xtrain,2),1);
     ytrain_output=zeros(1,size(Xtrain,2));
     for i=1:size(Xtrain,2)
          layer1_a(:,i)=u1'*Xtrain(:,i);
          layer1_z(:,i)=1/(1+exp((layer1_a(:,i))*(-1)));
          ytrain_output(1,i)=u2'*layer1_z(:,i);
          if ytrain_output(1,i)>=0.5
            train_class(i,1)=1;
          else
            train_class(i,1)=0;
          end
     end

     number1=sum(train_class==ytrain);
     train_error(e,1)=1-(number1/size(ytrain,1));


     %test test matrix
     test_class2=zeros(size(Xtest,2),1);
     ytest_input=zeros(1,size(Xtest,2));
     ytest_output=zeros(1,size(Xtest,2));
     for i=1:size(Xtest,2)
```

```
        layer1_a(:,i)=u1'*Xtest(:,i);
        layer1_z(:,i)=1/(1+exp((layer1_a(:,i))*(-1)));
        ytest_output(1,i)=u2'*layer1_z(:,i);
        if ytest_output(1,i)>=0.5;
          test_class2(i,1)=1;
        else
          test_class2(i,1)=0;
        end
    end

    number2=sum(test_class2==ytest);
    test_error(e,1)=1-(number1/size(ytest,1));


end

plot(1:epoch,train_error,'r');
hold on;
plot(1:epoch,test_error,'g');


(d)

size1=size(Xtrain,1);
size2=size(Xtrain,2);
nodes = 10;
epoch = 100;
I_matrix = -1*ones(size1,nodes);
u1 = I_matrix+rand(size1,nodes)*2; %33*5
u2 = -1*ones(nodes,1)+rand(nodes,1)*2; %5*1
layer1_z = zeros(nodes,size2); %5*176
layer1_a = zeros(nodes,size2); %5*176
y_input = zeros(1,size2); %1*176
y_output = zeros(1,size2);
lr=0.01; %learning rate
decay=0; %weight decay
delta_u2=zeros(nodes,1);
p1=zeros(nodes,1);
p2=zeros(nodes,1);
delta_u1=zeros(size1,nodes);
train_error1=zeros(nodes,1);
test_error1=zeros(nodes,1);

for n=1:nodes
    for e=1:epoch
        for i = 1:size2;
            layer1_a(:,i)=u1'*Xtrain(:,i);%5*1
            layer1_z(:,i)=1/(1+exp((layer1_a(:,i))*(-1)));
            y_input(1,i)=u2'*layer1_z(:,i);


            %from output to layer1
            delta_u2=-2*(ytrain(i,1)-y_input(1,i));%1
```

```
            %from layer1 to input
            delta_u1=delta_u2*Xtrain(:,i)*(u2.*layer1_z(:,i).*(ones(nodes
,1)-layer1_z(:,i)))';%33*5


            u2=u2-(lr*delta_u2)-(u2*lr*decay);
            u1=u1-(lr*delta_u1)-(u1*lr*decay);
        end
    end
    %test train matrix
    train_class=zeros(size(Xtrain,2),1);
    ytrain_output=zeros(1,size(Xtrain,2));
    for i=1:size(Xtrain,2)
        layer1_a(:,i)=u1'*Xtrain(:,i);
        layer1_z(:,i)=1/(1+exp((layer1_a(:,i))*(-1)));
        ytrain_output(1,i)=u2'*layer1_z(:,i);
        if ytrain_output(1,i)>=0.5
          train_class(i,1)=1;
        else
          train_class(i,1)=0;
        end
    end

    number1=sum(train_class==ytrain);
    train_error11(n,1)=1-(number1/size(ytrain,1));


    %test test matrix
    test_class2=zeros(size(Xtest,2),1);
    ytest_input=zeros(1,size(Xtest,2));
    ytest_output=zeros(1,size(Xtest,2));
    for i=1:size(Xtest,2)
        layer1_a(:,i)=u1'*Xtest(:,i);
        layer1_z(:,i)=1/(1+exp((layer1_a(:,i))*(-1)));
        ytest_output(1,i)=u2'*layer1_z(:,i);
        if ytest_output(1,i)>=0.5;
          test_class2(i,1)=1;
        else
          test_class2(i,1)=0;
        end
    end

    number2=sum(test_class2==ytest);
    test_error11(n,1)=1-(number1/size(ytest,1));


end

plot(1:nodes,train_error11,'r');
hold on;
plot(1:nodes,test_error11,'g');
```