%%%

```matlab
function [QDAmodel]= QDA_train(X_train, Y_train, numofClass)
%
% Training QDA
%
% EC 503 Learning from Data
% Gaussian Discriminant Analysis
%
% Assuming D = dimension of data
% Inputs :
% X_train : training data matrix, each row is a training data point
% Y_train : training labels for rows of X_train
% numofClass : number of classes
%
% Assuming that the classes are labeled  from 1 to numofClass
% Output :
% QDAmodel : the parameters of QDA classifier which has the following
% fields
% QDAmodel.Mu : numofClass * D matrix, i-th row = mean vector of class i
% QDAmodel.Sigma : D * D * numofClass array, Sigma(:,:,i) = covariance
% matrix of class i
% QDAmodel.Pi : numofClass* 1 vector, Pi(i) = prior probability of class i
%
[N,D]=size(X_train);
%
% Initialize the outputs
%
QDAmodel.Mu = zeros(numofClass, D);
QDAmodel.Sigma = zeros(D,D,numofClass);
QDAmodel.Pi = zeros(numofClass, 1);
%
% Estimate parameters
%
for i=1:numofClass
    samples = X_train(Y_train == i,:);
    QDAmodel.Mu(i,:) = mean(samples);
    QDAmodel.Sigma(:,:,i) = cov(samples);
    QDAmodel.Pi(i) = size(samples,1);
end
%
% Normalize Pi
%
QDAmodel.Pi = QDAmodel.Pi/N;
end
```

%%%

%%%

```
function [Y_predict] = QDA_test(X_test, QDAmodel, numofClass)
%
% Testing for QDA
%
% EC 503 Learning from Data
% Gaussian Discriminant Analysis
%
% Assuming D = dimension of data
% Inputs :
% X_test : test data matrix, each row is a test data point
% numofClass : number of classes
% QDAmodel: the parameters of QDA classifier which has the following
% fields
% QDAmodel.Mu : numofClass * D matrix, i-th row = mean vector of class i
% QDAmodel.Sigma : D * D * numofClass array, Sigma(:,:,i) = covariance
% matrix of class i
% QDAmodel.Pi : numofClass* 1 vector, Pi(i) = prior probability of class i
%
% Assuming that the classes are labeled  from 1 to numofClass
% Output :
% Y_predict predicted labels for all the testing data points in X_test
%
Pi = QDAmodel.Pi;
Sigma =QDAmodel.Sigma;
Mu = QDAmodel.Mu;
[N, D]=size(X_test);
Pi = Pi/sum(Pi);
logvalues = zeros(numofClass,1);
for k = 1 : numofClass
    eigs = eig(Sigma(:,:,k));
    logvalues(k) = -0.5* sum(log(eigs)) + log(Pi(k));
end
probs = zeros(N,numofClass);
for i = 1 : N
    for k = 1 : numofClass
        probs(i,k) = logvalues(k) - 0.5*((X_test(i,:)-
Mu(k,:))/(Sigma(:,:,k)))*(X_test(i,:)-Mu(k,:))';
    end
end
[maxv, Y_predict]=max(probs,[],2);
end
```

%%%

%%

```
function [LDAmodel]= LDA_train(X_train, Y_train, numofClass)
%
% Training LDA
%
% EC 503 Learning from Data
% Gaussian Discriminant Analysis
%
% Assuming D = dimension of data
% Inputs :
% X_train : training data matrix, each row is a training data point
% Y_train : training labels for rows of X_train
% numofClass : number of classes
%
% Assuming that the classes are labeled  from 1 to numofClass
% Output :
% LDAmodel : the parameters of LDA classifier which has the following
% fields
% LDAmodel.Mu : numofClass * D matrix, i-th row = mean vector of class i
% LDAmodel.Sigmapooled : D * D  covariance matrix
% LDAmodel.Pi : numofClass* 1 vector, Pi(i) = prior probability of class i
%
% Get the size/dimension of data
[N,D]=size(X_train);
%
% Initialize the outputs
%
LDAmodel.Mu = zeros(numofClass, D);
LDAmodel.Sigmapooled = zeros(D,D);
LDAmodel.Pi = zeros(numofClass, 1);
%
% Estimate the parameters
%
for i=1:numofClass
    samples = X_train(Y_train == i,:);
    LDAmodel.Mu(i,:) = mean(samples);
    LDAmodel.Pi(i) = size(samples,1);
    LDAmodel.Sigmapooled = LDAmodel.Sigmapooled + ((LDAmodel.Pi(i)-1) / (N
- numofClass) ).*cov(samples);
end
%
% normalize LDAmodel.Pi
%
LDAmodel.Pi = LDAmodel.Pi/N;
end
```

%%

%%

```
function [Y_predict] = LDA_test(X_test, LDAmodel, numofClass)
%
% Testing for LDA
%
% EC 503 Learning from Data
% Gaussian Discriminant Analysis
%
% Assuming D = dimension of data
% Inputs :
% X_test : test data matrix, each row is a test data point
% numofClass : number of classes
% LDAmodel : the parameters of LDA classifier which has the follwoing
% fields
% LDAmodel.Mu : numofClass * D matrix, i-th row = mean vector of class i
% LDAmodel.Sigmapooled : D * D  covariance matrix
% LDAmodel.Pi : numofClass* 1 vector, Pi(i) = prior probability of class i
%
% Assuming that the classes are labeled  from 1 to numofClass
% Output :
% Y_predict predicted labels for all the testing data points in X_test
%
Pi = LDAmodel.Pi;
Sigma =LDAmodel.Sigmapooled;
Mu = LDAmodel.Mu;
[N, D]=size(X_test);
%
% Normalize Pi
%
Pi = Pi/sum(Pi);
logvalues = zeros(numofClass,1);
for k = 1 : numofClass
    logvalues(k) =  log(Pi(k));
end
probs = zeros(N,numofClass);
%
% Calculate the log probabilities
%
for i = 1 : N
    for k = 1 : numofClass
        probs(i,k) = logvalues(k) - 0.5*((X_test(i,:)-
Mu(k,:))/(Sigma))*(X_test(i,:)-Mu(k,:))';
    end
end
[maxv, Y_predict]=max(probs,[],2);
end
```

%%

%%

```
function [RDAmodel]= RDA_train(X_train, Y_train,lambda, numofClass)
%
% Training RDA
%
% EC 503 Learning from Data
% Gaussian Discriminant Analysis
%
% Assuming D = dimension of data
% Inputs :
% X_train : training data matrix, each row is a training data point
% Y_train : training labels for rows of X_train
% numofClass : number of classes
% lambda: the regularization parameter between 0 and 1 so that:
%-----------------------------------------------------------------
% Sigma_RDA = (1-lamba)*Sigma_LDA + lambda diag(Sigma_LDA);
%-----------------------------------------------------------------
%
% Assuming that the classes are labeled  from 1 to numofClass
% Output :
% RDAmodel : the parameters of RDA classifier which has the following
% fields
% RDAmodel.Mu : numofClass * D matrix, i-th row = mean vector of class i
% RDAmodel.Sigmapooled : D * D  covariance matrix
% RDAmodel.Pi : numofClass* 1 vector, Pi(i) = prior probability of class i
%
[N,D]=size(X_train);
%
% Initialize the outputs
%
RDAmodel.Mu = zeros(numofClass, D);
RDAmodel.Sigmapooled = zeros(D,D);
RDAmodel.Pi = zeros(numofClass, 1);
for i=1:numofClass
    samples = X_train(Y_train == i,:);
    RDAmodel.Mu(i,:) = mean(samples);
    RDAmodel.Pi(i) = size(samples,1);
    RDAmodel.Sigmapooled = RDAmodel.Sigmapooled + ((RDAmodel.Pi(i)-1) / (N
- numofClass) ).*cov(samples);
end
%
% Regularization
%
RDAmodel.Sigmapooled = (1-lambda)*RDAmodel.Sigmapooled +
lambda*diag(diag(RDAmodel.Sigmapooled));
% normalize Pi
RDAmodel.Pi = RDAmodel.Pi/N;
End
```

%%

%%%

```
function [Y_predict] = RDA_test(X_test, RDAmodel, numofClass)
%
% Testing for RDA
%
% EC 503 Learning from Data
% Gaussian Discriminant Analysis
%
% Assuming D = dimension of data
% Inputs :
% X_test : test data matrix, each row is a test data point
% numofClass : number of classes
% RDAmodel : the parameters of RDA classifier which has the following
% fields
% RDAmodel.Mu : numofClass * D matrix, i-th row = mean vector of class i
% RDAmodel.Sigmapooled : D * D  covariance matrix
% RDAmodel.Pi : numofClass* 1 vector, Pi(i) = prior probability of class i
%
% Assuming that the classes are labeled  from 1 to numofClass
% Output :
% Y_predict predicted labels for all the testing data points in X_test
%
Pi = RDAmodel.Pi;
Sigma =RDAmodel.Sigmapooled;
Mu = RDAmodel.Mu;
[N, D]=size(X_test);
Pi = Pi/sum(Pi);
logvalues = zeros(numofClass,1);
for k = 1 : numofClass
    logvalues(k) =  log(Pi(k)) - 0.5*(Mu(k,:)/Sigma)*Mu(k,:)';
    beta(k,:)=Mu(k,:)/Sigma;
end
probs = zeros(N,numofClass);
for i = 1 : N
    for k = 1 : numofClass
        probs(i,k) = logvalues(k) + beta(k,:)*X_test(i,:)';
    end
end
[maxv, Y_predict]=max(probs,[],2);
end
```

%%%

%%

```
%
% EC 503 Learning from Data
% Gaussian Discriminant Analysis
% Code for part (d)
% ---------------------------------------------------
load data_cancer.mat
[N,D]=size(X); % N = total number of data , D = dimension of observation
%
% Creating training/test splits
%
N_train = 150; % number of training data
index_train = randperm (N, N_train)';
index_test = setdiff([1:N]',index_train);
X_train = X(index_train,:);
Y_train = Y(index_train);
X_test = X(index_test,:);
Y_test = Y(index_test);
%
% Get results for different lambda values
%
lambdas = [0.1:0.05:1];
CCRs_test = zeros(size(lambdas));
CCRs_train = zeros(size(lambdas));
for i  = 1:length(lambdas)
    lambda = lambdas(i);
    [ model ] = RDA_train(X_train, Y_train, lambda,2);
    [Y_train_pred] = RDA_test( X_train,model,2);
    [ Y_predict] = RDA_test( X_test,model,2);
    CCRs_test(i)= 1 - sum(Y_test~=Y_predict)/length(Y_predict);
    CCRs_train(i)=1 - sum(Y_train~=Y_train_pred)/length(Y_train_pred);
end
%
% Create plots
%
figure
hold on
plot(lambdas,CCRs_train,'-*k');
plot(lambdas,CCRs_test,'-+r');
legend('training CCR','test CCR')
xlabel('{\lambda}')
ylim([0.5,1.2]), ylabel('CCR')
title('CCR on CANCER dataset using RDA for different {\lambda}')
hold off
```

%%

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% ---------------------------------------------------
% ENG EC 503
% Learning from Data
% Boston University
% Instructor: Prakash Ishwar
% Assignment 3
% Code for part (d), Nearest Neighbor Classification
% ---------------------------------------------------
%

load data_mnist_train.mat
load data_mnist_test.mat

X_train = sparse(X_train);
X_test = sparse(X_test);

%
% Precompute sum of squares term for speed
%
XtrainSOS = sum(X_train.^2,2);
% XtestSOS  = sum(X_test.^2,2);

ntest= length(Y_test);
nbatches = 20;
batches = mat2cell(1:ntest,1,(ntest/nbatches)*ones(1,nbatches));
X_temp = repmat(XtrainSOS', ntest/nbatches,1);

Y_pred = zeros(ntest,1);
%
% Classify test points
%
for i=1:nbatches
    i
    dst = -2*X_test(batches{i},:)*X_train' + X_temp;
    [junk,closest] = min(dst,[],2);
    Y_pred(batches{i}) = Y_train(closest);
end

%
% Report results
%
errorRate = mean(Y_pred ~= Y_test);
CCR = 1- errorRate
CFmtx = confusionmat(Y_pred, Y_test);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```