Junhao Du

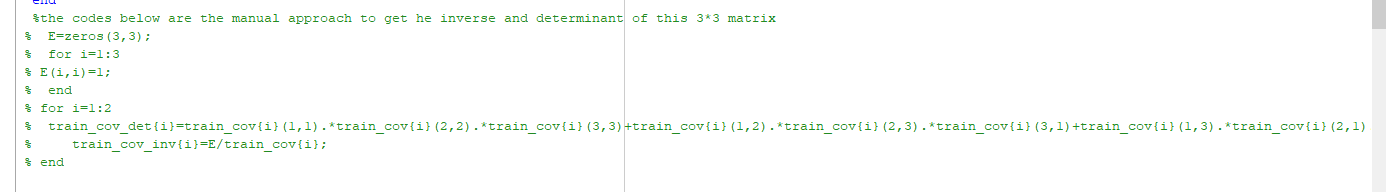
**1.**

f(x)=θ\*e^(- θ x)  
L(θ)=Пf(x)  
LnL(θ)=∑(Ln θ - θ xk)=nLn θ -∑ θ xk  
dLnL(θ)/d θ =0🡺n\*(1/ θ)-∑xk=0-🡺 θ =(1/n)∑xk

**2.**

This part requires us to implement the inverse and determinant of the matrix.

I did it but it’s not beautiful so I add them as comment like the picture below.



It contains 2 files.

1)

function results=IterationMLE

accuracy=zeros(10,1);

for i=1:10

accuracy(i)=MLE();

end

meanOfAccuracy=mean(accuracy);

standardDerivation=std(accuracy);

results={meanOfAccuracy,standardDerivation};

2)

function accuracy=MLE

%get the data

data=csvread('pima-indians-diabetes.csv');

data\_all=data(:,2:4);

%get the training dataset and testing dataset

y=length(data\_all);

c = cvpartition(y,'HoldOut',0.5);

% Now you can find the indices of your training and test sets

trainingIdx = training(c);

testIdx = test(c);

% Now you can find your training and test data

train\_class\_all = data\_all(trainingIdx,:);

testData = data\_all(testIdx,:);

%labels for training and test

train\_label = data(trainingIdx,9);

text\_label=data(testIdx,9);

[train\_row,train\_col]=size(train\_class\_all);

[m,n]=size(data\_all);

%?????????????

for i=1:2

train\_classification{i}=train\_class\_all(find(train\_label==i-1),:);%every data among feature2 to feature4

NUM{i}=length(train\_class\_all(find(train\_label==i-1),1));%number of every class

end

%priori probabilities

for i=1:2

PW{i}=NUM{i}/length(train\_class\_all);%the ratio

end

%the mean of MLE

for i=1:2

train\_mean{i}=(sum(train\_classification{i}))/NUM{i};

%train\_mean{i}=mean(train\_classification{i}); both are correct

end

% covariance matrix in the MLE

for i=1:2

for x=1:n

for y=1:n

train\_cov{i}(x,y)=(sum((train\_classification{i}(:,x)-train\_mean{i}(1,x)).\*(train\_classification{i}(:,y)-train\_mean{i}(1,y))))/(NUM{i}-1);

end

end

end

%train\_cov{1}=cov(train\_classification{1});

%train\_cov{2}=cov(train\_classification{2}); both are correct

for i=1:2

train\_cov\_inv{i}=inv(train\_cov{i});%inverse of the covariance matrix

train\_cov\_det{i}=det(train\_cov{i});%determinant of the covariance matrix

end

%the codes below are the manual approach to get he inverse and determinant of this 3\*3 matrix

% E=zeros(3,3);

% for i=1:3

% E(i,i)=1;

% end

% for i=1:2

% train\_cov\_det{i}=train\_cov{i}(1,1).\*train\_cov{i}(2,2).\*train\_cov{i}(3,3)+train\_cov{i}(1,2).\*train\_cov{i}(2,3).\*train\_cov{i}(3,1)+train\_cov{i}(1,3).\*train\_cov{i}(2,1).\*train\_cov{i}(3,2)-train\_cov{i}(3,1).\*train\_cov{i}(2,2).\*train\_cov{i}(1,3)-train\_cov{i}(3,2).\*train\_cov{i}(2,3).\*train\_cov{i}(1,1)-train\_cov{i}(3,3).\*train\_cov{i}(2,1).\*train\_cov{i}(1,2);

% train\_cov\_inv{i}=E/train\_cov{i};

% end

%minimum erroe with bayes

%get the minimum error for every data in every classifier

for i=1:2

for j=1:length(testData)

text\_data\_one=testData(j,:);

g{j,i}=(-0.5)\*(text\_data\_one-train\_mean{i})\*train\_cov\_inv{i}\*(text\_data\_one'-train\_mean{i}')-0.5\*log(abs(train\_cov\_det{i}))+log(PW{i});

end

end

%remove the infinity

for j=1:length(testData)

for i=1:2

if abs(g{j,i})<=10000

p{j,i}=g{j,i};

end

end

end

%get the maximum

nummax=zeros(length(testData),1);

for j=1:length(testData)

PRow=[p{j,:}];

nummax(j,1)=max(PRow);

end

%mark the label

label=zeros(length(testData),1);

for j=1:length(testData)

switch nummax(j,1)

case g{j,1}

label(j,1)=0;

case g{j,2}

label(j,1)=1;

end

end

bo=zeros(length(testData),1);

for j=1:length(testData)

if label(j,1)==text\_label(j,1)

bo(j)=1;

end

end

%get accuracy

correct=0;

for j=1:length(bo)

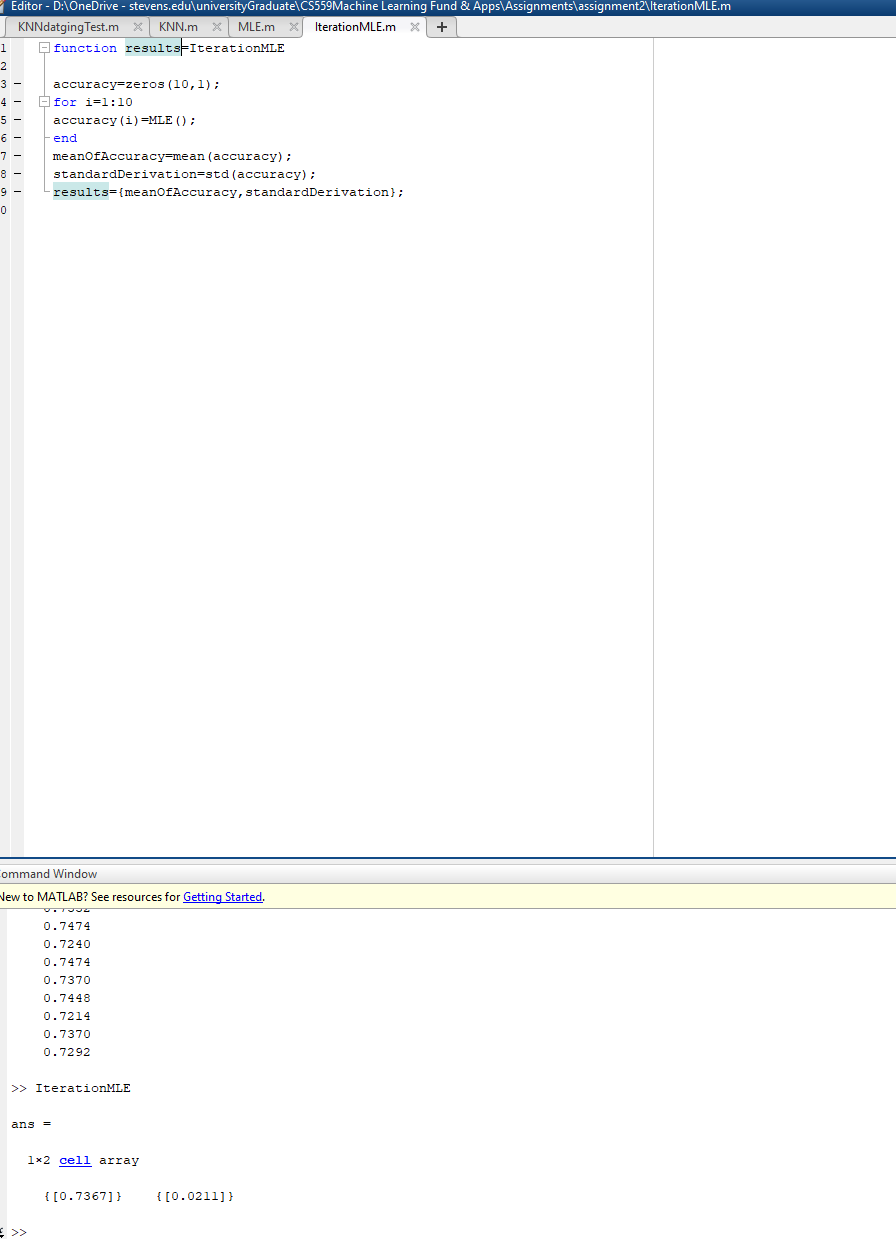
if bo(j)==1

correct=correct+1;

end

end

accuracy=correct/length(testData);

Screenshots: 

**3.**

It contains 2 files.

1)

%function result=KNNdatgingTest

data=csvread('pima-indians-diabetes.csv');

dataMat = data(:,2:4);

len = size(dataMat,1);

k = 11;

% ratio of datas

Ratio = 0.5;

numTest = Ratio \* len;

%normalization

maxV = max(dataMat);

minV = min(dataMat);

range = maxV-minV;

newdataMat = (dataMat-repmat(minV,[len,1]))./(repmat(range,[len,1]));

accuracy=zeros(10,1);

% test

for index=1:10

% First you make crossvalidation partitioning on your data

% y is a vector which contains the categories of your observations

% 'HoldOut' an optional property to make training and test set

% Fraction of data to form test set

y=length(newdataMat);

c = cvpartition(y,'HoldOut',0.5);

% Now you can find the indices of your training and test sets

trainingIdx = training(c);

testIdx = test(c);

% Now you can find your training and test data

trainingData = newdataMat(trainingIdx,:);

testData = newdataMat(testIdx,:);

error=0;

labels = data(trainingIdx,9);

for i = 1:numTest

classifyresult = KNN(testData(i,:),trainingData,labels,k);

fprintf('results:%d real results are:%d\n',[classifyresult labels(i)])

if(classifyresult~=labels(i))

error = error+1;

end

end

accuracy(index,1)=1-error/numTest;

fprintf('accuracy:%f\n',1-error/numTest);

end

meanOfAccuracy=mean(accuracy);

standardDerivation=std(accuracy);

result={meanOfAccuracy,standardDerivation};

2)

function relustLabel = KNN(test,sample,labels,k)

%

% test is from test dataset, sample is from the training dataset, labels

% are classification from training data.

%

[row , col] = size(sample);

differenceMatrix = repmat(test,[row,1]) - sample ;

distanceMatrix = sqrt(sum(differenceMatrix.^2,2));

[B , IX] = sort(distanceMatrix,'ascend');

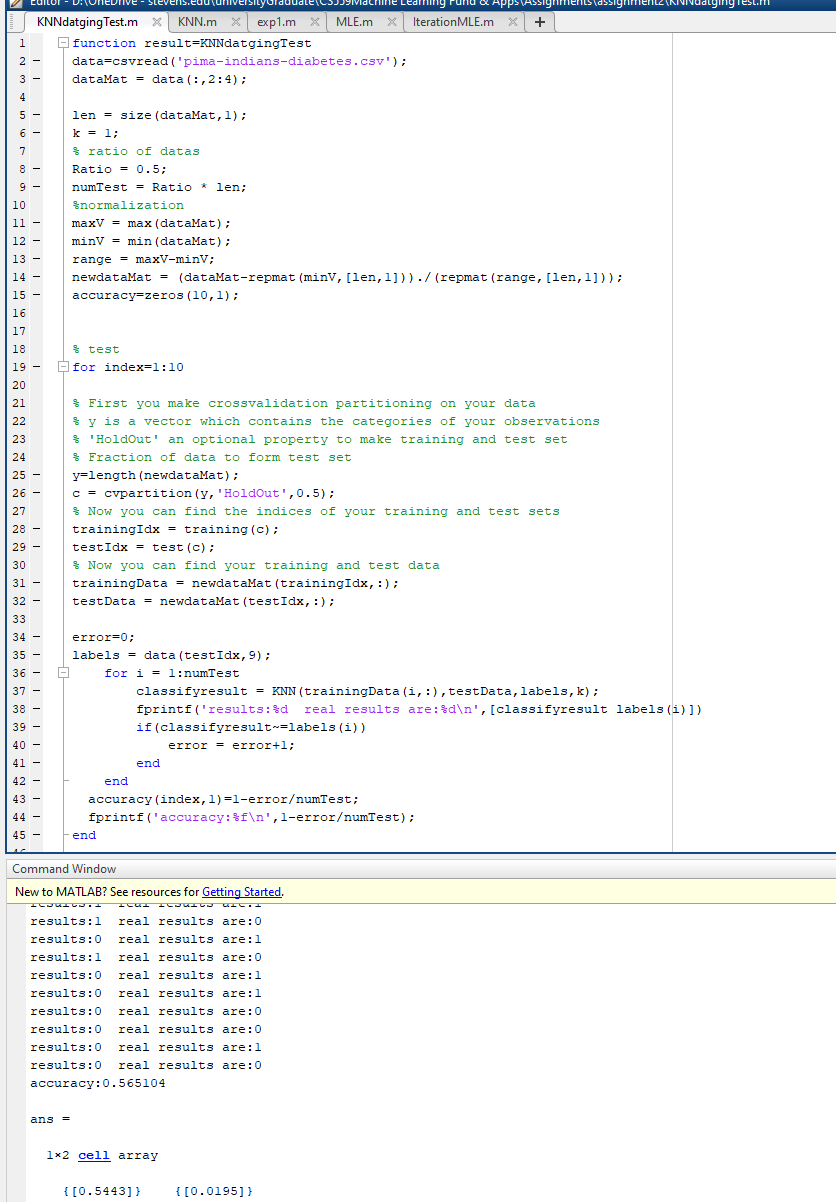
len = min(k,length(B));

relustLabel = mode(labels(IX(1:len)));

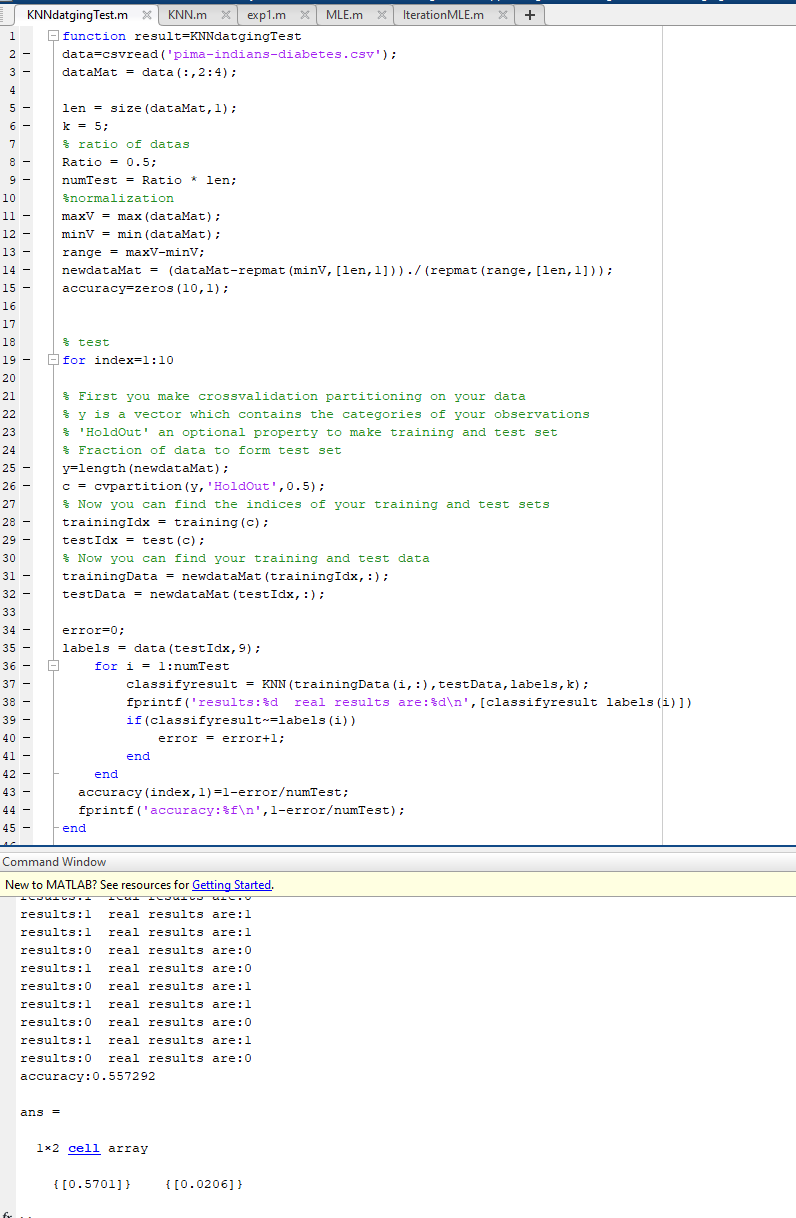
end

Screenshots:

K=1:



K=5:



K=11:

