Course Number: EEL 5840

UF ID: 0699 6263

Name: Fang Zhu

**Homework 5:**

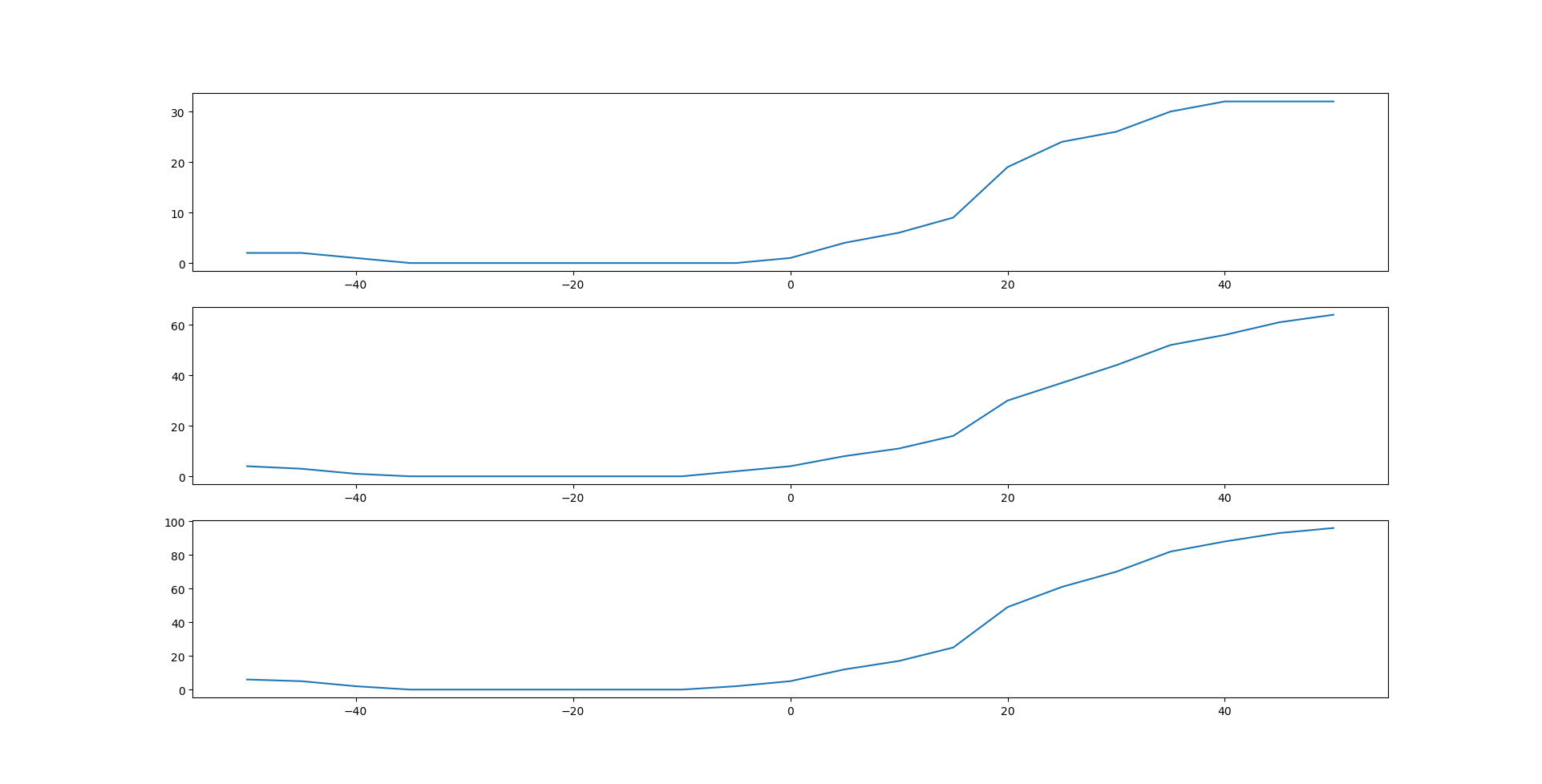
Q1.

Implement the Bayes classifier, under the assumption that your likelihood model p(x | j) is unimodal Gaussian and that the prior probabilities p(j) are dictated by the number of samples nj belong to R that you have for each class. This classifier is given by the following discriminant function for each class j. (the order of the matrix for the species from up to down and from left to right is 0 and 1)

1. Matrix for test data:
2. Matrix for training data:
3. Matrix for all the data:
4. Processing timer: 0.3445889949798584 seconds

Q2.

Implement a classifier based on linear discriminant analysis. This is a particular case of the Bayes classifier. Here, T belongs to R is a potentially non-zero threshold value that you should choose, in some fashion, on your own. The variables µ0; µ1 2 R7 are the class means for the two classes and Σ0, Σ1 belongs to R7×7 are the class covariances for the two classes. (the order of the matrix for the species from up to down and from left to right is 0 and 1) To make less error I plot a picture of T and sum of error from -50 to 50 as follow and found that the threshold around -20 can cause no error.



The picture of the sum of error and threshold T

1. Matrix for test data:
2. Matrix for training data:
3. Matrix for all the data:
4. Processing time: 0.1093592643737793 seconds

While I did this question, I found that the right of the g(x) is too big, when T is closed to zero. Afterwards, I check out the data and found that the last two dimensions are ‘male’ and ‘female’. These two-opposite phenomena cause the covariance of the two class to be very big. So I add the regularized term lambda to adjust the problem (If I drop one of the last two dimension, the problem can also be solved). I choose the lambda equals to 0.01. To get the threshold, I set the different threshold T, and I got the answers above.

Code 1:

import numpy as np

import matplotlib.pyplot as plt

import math

import time

start\_time = time.time()

#initial parameters and load data

data = np.loadtxt("dataset.txt")

alldata = data[:,1:8]

training = data[0:140,1:8]

test = data[140:200,1:8]

validate\_training = data[0:140,0]

validate\_test = data[140:200,0]

validate\_data = data[:,0]

count = 0

for i in range(140):

if validate\_training[i] == 1:

count = count + 1

p\_prior1 = count/140

p\_prior0 = 1 - (count/140)

tr\_1 = np.zeros([count,7])

tr\_0 = np.zeros([140-count,7])

ind1 = 0

ind0 = 0

for i in range(140):

if validate\_training[i] == 1:

tr\_1[ind1] = training[i,:]

ind1 = ind1 + 1

if validate\_training[i] == 0:

tr\_0[ind0] = training[i,:]

ind0 = ind0 + 1

mean\_1 = np.zeros(7)

mean\_0 = np.zeros(7)

for i in range(7):

mean\_1[i] = sum(tr\_1[:,i])/count

M1 = mean\_1[i]\*np.ones([count,1])

mean\_0[i] = sum(tr\_0[:,i])/(140-count)

M0 = mean\_0[i]\*np.ones([(140-count),1])

sig\_1 = 0

sig\_0 = 0

for i in range(count):

sig\_1 = sig\_1 + (tr\_1[i,:]-np.asmatrix(mean\_1)).T@(tr\_1[i,:]-np.asmatrix(mean\_1))/count

for i in range(140-count):

sig\_0 = sig\_0 + (tr\_0[i,:]-np.asmatrix(mean\_0)).T@(tr\_0[i,:]-np.asmatrix(mean\_0))/(140-count)

#sig\_1 = np.cov(tr\_1.T)

#sig\_0 = np.cov(tr\_0.T)

count1 = 0

count0 = 0

for i in range(60):

g1 = -0.5\*((np.asmatrix(test[i,:]) - np.asmatrix(mean\_1))@np.linalg.inv(sig\_1)@(np.asmatrix(test[i,:]) - np.asmatrix(mean\_1)).T) - 3.5\*math.log1p(2\*math.pi) - 0.5\*math.log1p(np.linalg.det(sig\_1)) + math.log1p(p\_prior1)

g0 = -0.5\*((np.asmatrix(test[i,:]) - np.asmatrix(mean\_0))@np.linalg.inv(sig\_0)@(np.asmatrix(test[i,:]) - np.asmatrix(mean\_0)).T) - 3.5\*math.log1p(2\*math.pi) - 0.5\*math.log1p(np.linalg.det(sig\_0)) + math.log1p(p\_prior0)

if g1 >= g0:

if validate\_test[i] == 1:

count1 = count1 + 1

else:

if g0 > g1:

if validate\_test[i] == 0:

count0 = count0 + 1

c = 0

for i in range(60):

if validate\_test[i] == 1:

c = c + 1

error1 = c - count1

error0 = 60 - c - count0

Matrix1 = np.ones([2,2])

Matrix1[0,0] = count0

Matrix1[1,0] = error0

Matrix1[0,1] = error1

Matrix1[1,1] = count1

count1 = 0

count0 = 0

for i in range(140):

g1 = -0.5\*((np.asmatrix(training[i,:]) - np.asmatrix(mean\_1))@np.linalg.inv(sig\_1)@(np.asmatrix(training[i,:]) - np.asmatrix(mean\_1)).T) - 3.5\*math.log1p(2\*math.pi) - 0.5\*math.log1p(np.linalg.det(sig\_1)) + math.log1p(p\_prior1)

g0 = -0.5\*((np.asmatrix(training[i,:]) - np.asmatrix(mean\_0))@np.linalg.inv(sig\_0)@(np.asmatrix(training[i,:]) - np.asmatrix(mean\_0)).T) - 3.5\*math.log1p(2\*math.pi) - 0.5\*math.log1p(np.linalg.det(sig\_0)) + math.log1p(p\_prior0)

if g1 >= g0:

if validate\_training[i] == 1:

count1 = count1 + 1

else:

if g0 > g1:

if validate\_training[i] == 0:

count0 = count0 + 1

c = 0

for i in range(140):

if validate\_training[i] == 1:

c = c + 1

error1 = c - count1

error0 = 140 - c - count0

Matrix2 = np.ones([2,2])

Matrix2[0,0] = count0

Matrix2[1,0] = error0

Matrix2[0,1] = error1

Matrix2[1,1] = count1

count1 = 0

count0 = 0

for i in range(200):

g1 = -0.5\*((np.asmatrix(alldata[i,:]) - np.asmatrix(mean\_1))@np.linalg.inv(sig\_1)@(np.asmatrix(alldata[i,:]) - np.asmatrix(mean\_1)).T) - 3.5\*math.log1p(2\*math.pi) - 0.5\*math.log1p(np.linalg.det(sig\_1)) + math.log1p(p\_prior1)

g0 = -0.5\*((np.asmatrix(alldata[i,:]) - np.asmatrix(mean\_0))@np.linalg.inv(sig\_0)@(np.asmatrix(alldata[i,:]) - np.asmatrix(mean\_0)).T) - 3.5\*math.log1p(2\*math.pi) - 0.5\*math.log1p(np.linalg.det(sig\_0)) + math.log1p(p\_prior0)

if g1 >= g0:

if validate\_data[i] == 1:

count1 = count1 + 1

else:

if g0 > g1:

if validate\_data[i] == 0:

count0 = count0 + 1

c = 0

for i in range(200):

if validate\_data[i] == 1:

c = c + 1

error1 = c - count1

error0 = 200 - c - count0

Matrix3 = np.ones([2,2])

Matrix3[0,0] = count0

Matrix3[1,0] = error0

Matrix3[0,1] = error1

Matrix3[1,1] = count1

print(Matrix1)#test

print(Matrix2)#training

print(Matrix3)#all

print("--- %s seconds ---" % (time.time() - start\_time))#processing time

Code 2:

import numpy as np

import matplotlib.pyplot as plt

import math

import time

start\_time = time.time()

#initial parameters and load data

data = np.loadtxt("dataset.txt")

alldata = data[:,1:8]

training = data[0:140,1:8]

test = data[140:200,1:8]

validate\_training = data[0:140,0]

validate\_test = data[140:200,0]

validate\_data = data[:,0]

count = 0

for i in range(140):

if validate\_training[i] == 1:

count = count + 1

p\_prior1 = count/140

p\_prior0 = 1 - (count/140)

tr\_1 = np.zeros([count,7])

tr\_0 = np.zeros([140-count,7])

ind1 = 0

ind0 = 0

for i in range(140):

if validate\_training[i] == 1:

tr\_1[ind1] = training[i,:]

ind1 = ind1 + 1

if validate\_training[i] == 0:

tr\_0[ind0] = training[i,:]

ind0 = ind0 + 1

mean\_1 = np.zeros(7)

mean\_0 = np.zeros(7)

for i in range(7):

mean\_1[i] = sum(tr\_1[:,i])/count

M1 = mean\_1[i]\*np.ones([count,1])

mean\_0[i] = sum(tr\_0[:,i])/(140-count)

M0 = mean\_0[i]\*np.ones([(140-count),1])

sig\_1 = 0

sig\_0 = 0

for i in range(count):

sig\_1 = sig\_1 + (tr\_1[i,:]-np.asmatrix(mean\_1)).T@(tr\_1[i,:]-np.asmatrix(mean\_1))/count

for i in range(140-count):

sig\_0 = sig\_0 + (tr\_0[i,:]-np.asmatrix(mean\_0)).T@(tr\_0[i,:]-np.asmatrix(mean\_0))/(140-count)

#sig\_1 = np.cov(tr\_1.T)

#sig\_0 = np.cov(tr\_0.T)

lambda\_0 = 0.01

lambda\_1 = 0.01

sig\_0 = sig\_0 + lambda\_0\*np.eye(7)

sig\_1 = sig\_1 + lambda\_1\*np.eye(7)

count1 = 0

count0 = 0

T = -5

for i in range(60):

g1 = np.asmatrix(test[i,:])@np.linalg.inv(0.5\*(sig\_1+sig\_0))@(np.asmatrix(mean\_1)-np.asmatrix(mean\_0)).T

g0 = 0.25\*(np.asmatrix(mean\_0)@np.linalg.inv(sig\_0)@np.asmatrix(mean\_0).T - np.asmatrix(mean\_1)@np.linalg.inv(sig\_1)@np.asmatrix(mean\_1).T + T)

if g1 >= g0:

if validate\_test[i] == 1:

count1 = count1 + 1

else:

if g1 < g0:

if validate\_test[i] == 0:

count0 = count0 + 1

print(g0)

c = 0

for i in range(60):

if validate\_test[i] == 1:

c = c + 1

error1 = c - count1

error0 = 60 - c - count0

Matrix1 = np.ones([2,2])

Matrix1[0,0] = count0

Matrix1[1,0] = error0

Matrix1[0,1] = error1

Matrix1[1,1] = count1

count1 = 0

count0 = 0

T = -10

for i in range(140):

g1 = np.asmatrix(training[i,:])@np.linalg.inv(0.5\*(sig\_1+sig\_0))@(np.asmatrix(mean\_1)-np.asmatrix(mean\_0)).T

g0 = 0.25\*(np.asmatrix(mean\_0)@np.linalg.inv(sig\_0)@np.asmatrix(mean\_0).T - np.asmatrix(mean\_1)@np.linalg.inv(sig\_1)@np.asmatrix(mean\_1).T + T)

if g1 >= g0:

if validate\_training[i] == 1:

count1 = count1 + 1

else:

if g1 < g0:

if validate\_training[i] == 0:

count0 = count0 + 1

c = 0

for i in range(140):

if validate\_training[i] == 1:

c = c + 1

error1 = c - count1

error0 = 140 - c - count0

Matrix2 = np.ones([2,2])

Matrix2[0,0] = count0

Matrix2[1,0] = error0

Matrix2[0,1] = error1

Matrix2[1,1] = count1

count1 = 0

count0 = 0

T = -10

for i in range(200):

g1 = np.asmatrix(alldata[i,:])@np.linalg.inv(0.5\*(sig\_1+sig\_0))@(np.asmatrix(mean\_1)-np.asmatrix(mean\_0)).T

g0 = 0.25\*(np.asmatrix(mean\_0)@np.linalg.inv(sig\_0)@np.asmatrix(mean\_0).T - np.asmatrix(mean\_1)@np.linalg.inv(sig\_1)@np.asmatrix(mean\_1).T + T)

if g1 >= g0:

if validate\_data[i] == 1:

count1 = count1 + 1

else:

if g1 < g0:

if validate\_data[i] == 0:

count0 = count0 + 1

print(g0)

c = 0

for i in range(200):

if validate\_data[i] == 1:

c = c + 1

error1 = c - count1

error0 = 200 - c - count0

Matrix3 = np.ones([2,2])

Matrix3[0,0] = count0

Matrix3[1,0] = error0

Matrix3[0,1] = error1

Matrix3[1,1] = count1

print(Matrix1)#test

print(Matrix2)#training

print(Matrix3)#all

print("--- %s seconds ---" % (time.time() - start\_time))#processing time

Code 3:

import numpy as np

import matplotlib.pyplot as plt

import math

import time

start\_time = time.time()

fig = plt.figure(figsize=(30,10))

#initial parameters and load data

data = np.loadtxt("dataset.txt")

alldata = data[:,1:8]

training = data[0:140,1:8]

test = data[140:200,1:8]

validate\_training = data[0:140,0]

validate\_test = data[140:200,0]

validate\_data = data[:,0]

count = 0

for i in range(140):

if validate\_training[i] == 1:

count = count + 1

p\_prior1 = count/140

p\_prior0 = 1 - (count/140)

tr\_1 = np.zeros([count,7])

tr\_0 = np.zeros([140-count,7])

ind1 = 0

ind0 = 0

for i in range(140):

if validate\_training[i] == 1:

tr\_1[ind1] = training[i,:]

ind1 = ind1 + 1

if validate\_training[i] == 0:

tr\_0[ind0] = training[i,:]

ind0 = ind0 + 1

mean\_1 = np.zeros(7)

mean\_0 = np.zeros(7)

for i in range(7):

mean\_1[i] = sum(tr\_1[:,i])/count

M1 = mean\_1[i]\*np.ones([count,1])

mean\_0[i] = sum(tr\_0[:,i])/(140-count)

M0 = mean\_0[i]\*np.ones([(140-count),1])

sig\_1 = 0

sig\_0 = 0

for i in range(count):

sig\_1 = sig\_1 + (tr\_1[i,:]-np.asmatrix(mean\_1)).T@(tr\_1[i,:]-np.asmatrix(mean\_1))/count

for i in range(140-count):

sig\_0 = sig\_0 + (tr\_0[i,:]-np.asmatrix(mean\_0)).T@(tr\_0[i,:]-np.asmatrix(mean\_0))/(140-count)

#sig\_1 = np.cov(tr\_1.T)

#sig\_0 = np.cov(tr\_0.T)

lambda\_0 = 0.01

lambda\_1 = 0.01

sig\_0 = sig\_0 + lambda\_0\*np.eye(7)

sig\_1 = sig\_1 + lambda\_1\*np.eye(7)

count1 = 0

count0 = 0

error = np.zeros(21)

d = np.arange(-50,55,5)

for t in range(0,21,1):

T = t\*5 - 50

for i in range(60):

g1 = np.asmatrix(test[i,:])@np.linalg.inv(0.5\*(sig\_1+sig\_0))@(np.asmatrix(mean\_1)-np.asmatrix(mean\_0)).T

g0 = 0.25\*(np.asmatrix(mean\_0)@np.linalg.inv(sig\_0)@np.asmatrix(mean\_0).T - np.asmatrix(mean\_1)@np.linalg.inv(sig\_1)@np.asmatrix(mean\_1).T + T)

if g1 >= g0:

if validate\_test[i] == 1:

count1 = count1 + 1

else:

if g1 < g0:

if validate\_test[i] == 0:

count0 = count0 + 1

c = 0

for i in range(60):

if validate\_test[i] == 1:

c = c + 1

error1 = c - count1

error0 = 60 - c - count0

error[t] = error1 + error0

count1 = 0

count0 = 0

p1 = fig.add\_subplot(\*[3,1,1])

p1.plot(d,error)

for t in range(0,21,1):

T = t\*5 - 50

for i in range(140):

g1 = np.asmatrix(training[i,:])@np.linalg.inv(0.5\*(sig\_1+sig\_0))@(np.asmatrix(mean\_1)-np.asmatrix(mean\_0)).T

g0 = 0.25\*(np.asmatrix(mean\_0)@np.linalg.inv(sig\_0)@np.asmatrix(mean\_0).T - np.asmatrix(mean\_1)@np.linalg.inv(sig\_1)@np.asmatrix(mean\_1).T + T)

if g1 >= g0:

if validate\_training[i] == 1:

count1 = count1 + 1

else:

if g1 < g0:

if validate\_training[i] == 0:

count0 = count0 + 1

c = 0

for i in range(140):

if validate\_training[i] == 1:

c = c + 1

error1 = c - count1

error0 = 140 - c - count0

error[t] = error1 + error0

count1 = 0

count0 = 0

p2 = fig.add\_subplot(\*[3,1,2])

p2.plot(d,error)

for t in range(0,21,1):

T = t\*5 - 50

for i in range(200):

g1 = np.asmatrix(alldata[i,:])@np.linalg.inv(0.5\*(sig\_1+sig\_0))@(np.asmatrix(mean\_1)-np.asmatrix(mean\_0)).T

g0 = 0.25\*(np.asmatrix(mean\_0)@np.linalg.inv(sig\_0)@np.asmatrix(mean\_0).T - np.asmatrix(mean\_1)@np.linalg.inv(sig\_1)@np.asmatrix(mean\_1).T + T)

if g1 >= g0:

if validate\_data[i] == 1:

count1 = count1 + 1

else:

if g1 < g0:

if validate\_data[i] == 0:

count0 = count0 + 1

c = 0

for i in range(200):

if validate\_data[i] == 1:

c = c + 1

error1 = c - count1

error0 = 200 - c - count0

error[t] = error1 + error0

count1 = 0

count0 = 0

p3 = fig.add\_subplot(\*[3,1,3])

p3.plot(d,error)