## Question 1:

Actual

Predict

Class 0

Class 1

Class 0

Class 1

100

0

0

100

Entire Dataset

Actual

Predict

Class 0

Class 1

Class 0

Class 1

72

0

0

68

Training Dataset

Actual

Predict

Class 0

Class 1

Class 0

Class 1

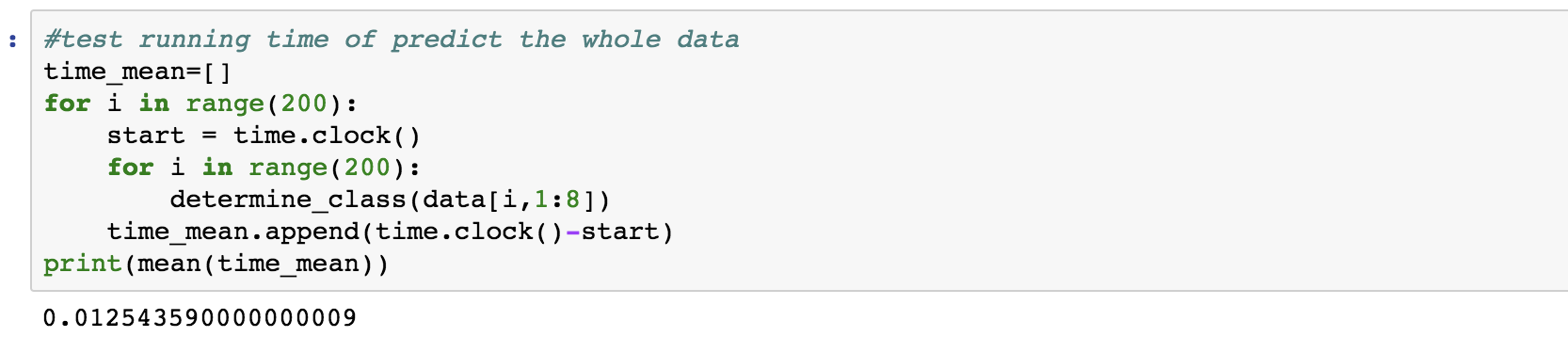
28

0

0

32

Testing Dataset



I run the entire data set to predict for 200 times. The average time of predict the whole dataset is 0.012(s)

## Question 2:

Actual

Predict

Class 0

Class 1

Class 0

Class 1

100

0

0

100

Entire Dataset

Actual

Predict

Class 0

Class 1

Class 0

Class 1

72

0

0

68

Training Dataset

Actual

Predict

Class 0

Class 1

Class 0

Class 1

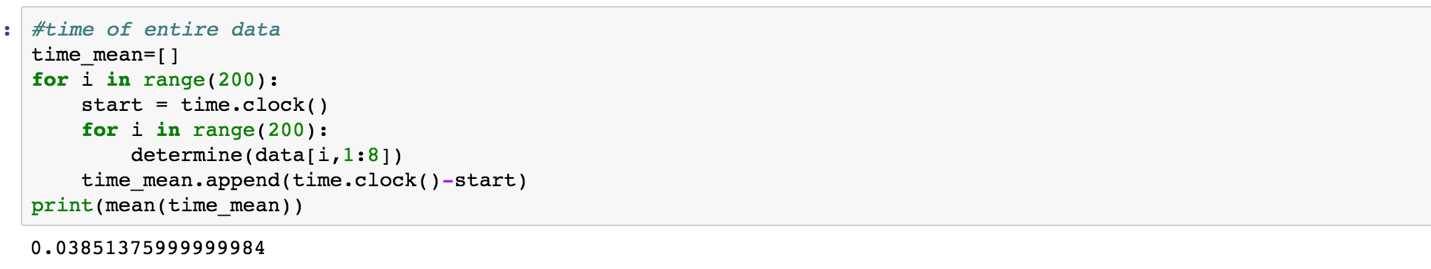
28

0

0

32

Testing Dataset



The range of T in this question is (-10.66,4.12). when I choose T=-9 the accuracy is 100%.

I run the entire data set to predict for 200 times. The average time of predict the whole dataset is 0.0385(s)

In conclusion, in my experimental test, the method of Bayes classifier can provide faster computation than the LDA method. Both of them have the accuracy of 100%.

## CODE:

Code for Question 1:

import numpy as np

from numpy.linalg import inv

from numpy.linalg import det

import matplotlib

import matplotlib.pyplot as plt

import math

from math import pi

import time

%matplotlib inline

#init variable

data = np.zeros([200,8])

train = np.zeros([140,8])

test = np.zeros([60,8])

#read file and split data as training and testing

def read(filepath):

file = open(filepath)

line = file.readline()

n = 0

while(line):

line = file.readline()

row = line.split()

if(row!=[]):

if(n<=139):

for i in range(8):

train[n,i]=row[i]

data[n,i]=row[i]

if(n>139):

for i in range(8):

test[n-140,i]=row[i]

data[n,i]=row[i]

n=n+1

file.close()

return data,train,test

#calculate mean

def mean(number):

return sum(number)/len(number)

#calculate standard deviation for each feature of each class

def stdev(number):

st = mean(number)

sum = 0

for n in range(len(number)):

sum=sum + (number[n]-st)\*\*2

return math.sqrt(sum/len(number))

#seperate data to two different class for training

read("HW5\_dataset.txt")

n0 = 0 #number of class 0

n1 = 0 #number of class 1

for label in train[:,0]:

if label==0:

n0 = n0+1

if label==1:

n1 = n1+1

s0 = np.zeros([n0,7])

s1 = np.zeros([n1,7])

s00 = 0

s11 = 0

for i in range(140):

if(train[i,0]==0):

s0[s00,:]=train[i,1:8]

s00=s00+1

if(train[i,0]==1):

s1[s11,:]=train[i,1:8]

s11=s11+1

#training data species0 s0, species1 s1

#possiblity of each class in sample

s0poss = s00/(s00+s11)

s1poss = s11/(s00+s11)

print(s0poss,s1poss)

#calculate mu0 and mu1

mu0=np.zeros([7,1])

mu1=np.zeros([7,1])

for n in range(7):

mu0[n,0]=mean(s0[:,n])

mu1[n,0]=mean(s1[:,n])

#calculate sigma for class 0 and 1

#@parameter: sigma0, sigma1

sigma0 = np.zeros([7,7])

sigma1 = np.zeros([7,7])

for num in range(len(s0)):

sigma0 = sigma0 + (s0[num:num+1,:].T-mu0)@(s0[num:num+1,:].T-mu0).T

for num in range(len(s1)):

sigma1 = sigma1 + (s1[num:num+1,:].T-mu1)@(s1[num:num+1,:].T-mu1).T

sigma0=sigma0/len(s0)

sigma1=sigma1/len(s1)

def determine\_class(x):

x.shape=(7,1)

poss0=0

poss1=0

g0 = -1/2\*(x-mu0).T@inv(sigma0)@(x-mu0)-1/2\*math.log(det(-sigma0))-7/2\*math.log(2\*pi)+math.log(s0poss)

g1 = -1/2\*(x-mu1).T@inv(sigma1)@(x-mu1)-1/2\*math.log(det(sigma1))-7/2\*math.log(2\*pi)+math.log(s1poss)

if g0>g1:

return 0

else:

return 1

#test accuracy on testing data

right=0

for i in range(60):

label = test[i,0]

answer = determine\_class(test[i,1:8])

if label==answer:

right=right+1

print("Accuracy to predicte testing data:",right/60)

#time needed for the entire dataset

start = time.clock()

right = 0

wrong\_predict=[]

setnum=[]

for i in range(200):

label = data[i,0]

answer = determine\_class(data[i,1:8])

if label==answer:

right=right+1

if label!=answer:

wrong\_predict.append(data[i,0:8])

setnum.append(i)

print("Time to classify the entire dataset",time.clock()-start,"s")

print("Number of correct prediction:",right,"Accuracy:",right/200)

#print(wrong\_predict)

print("Line of wrong predicted data:",setnum)

#calculate time for entire dataset

time\_mean=[]

for i in range(200):

start=time.clock()

for n in range(200):

determine\_class(data[i,1:8])

time\_mean.append(time.clock()-start)

print(mean(time\_mean))

Code for question 2:

import numpy as np

from numpy.linalg import pinv

from numpy.linalg import det

import math

from math import pi

import time

#init variable

data = np.zeros([200,8])

train = np.zeros([140,8])

test = np.zeros([60,8])

def read(filepath):

file = open(filepath)

line = file.readline()

n = 0

while(line):

line = file.readline()

row = line.split()

if(row!=[]):

if(n<=139):

for i in range(8):

train[n,i]=row[i]

data[n,i]=row[i]

if(n>139):

for i in range(8):

test[n-140,i]=row[i]

data[n,i]=row[i]

n=n+1

file.close()

return data,train,test

#calculate mean

def mean(number):

return sum(number)/len(number)

#calculate standard deviation for each feature of each class

def stdev(number):

st = mean(number)

sum = 0

for n in range(len(number)):

sum=sum + (number[n]-st)\*\*2

return math.sqrt(sum/len(number))

#seperate data to two different class for training

read("HW5\_dataset.txt")

n0 = 0 #number of class 0

n1 = 0 #number of class 1

for label in train[:,0]:

if label==0:

n0 = n0+1

if label==1:

n1 = n1+1

s0 = np.zeros([n0,7])

s1 = np.zeros([n1,7])

s00 = 0

s11 = 0

for i in range(140):

if(train[i,0]==0):

s0[s00,:]=train[i,1:8]

s00=s00+1

if(train[i,0]==1):

s1[s11,:]=train[i,1:8]

s11=s11+1

#training data species0 s0, species1 s1

#possiblity of each class in sample

s0poss = s00/(s00+s11)

s1poss = s11/(s00+s11)

#calculate mu0 and mu1

mu0=np.zeros([7,1])

mu1=np.zeros([7,1])

for n in range(7):

mu0[n,0]=mean(s0[:,n])

mu1[n,0]=mean(s1[:,n])

#calculate sigma for class 0 and 1

#@parameter: sigma0, sigma1

sigma0 = np.zeros([7,7])

sigma1 = np.zeros([7,7])

for num in range(len(s0)):

sigma0 = sigma0 + (s0[num:num+1,:].T-mu0)@(s0[num:num+1,:].T-mu0).T

for num in range(len(s1)):

sigma1 = sigma1 + (s1[num:num+1,:].T-mu1)@(s1[num:num+1,:].T-mu1).T

sigma0=sigma0/len(s0)

sigma1=sigma1/len(s1)

#determine class

#@structure determine((7,1),T)

def determine(x):

x.reshape(7,1)

left = 4\*((pinv(sigma0+sigma1)@(mu1-mu0)).T@x)

right = mu0.T@pinv(sigma0)@mu0-mu1.T@pinv(sigma1)@mu1

if left-right<-9:

return 0

else:

return 1

#accuracy of test data

correct = 0

for i in range(60):

x = determine(test[i,1:8])

if x == test[i,0]:

correct = correct+1

if x != test[i,0]:

print(i+141,test[i])

print(correct/60)

#accuracy of training data

correct = 0

for i in range(140):

x = determine(train[i,1:8])

if x == train[i,0]:

correct = correct + 1

if x != train[i,0]:

print(i,train[i,:])

print(correct/140)

#accuracy of entire data

correct = 0

wrong = 0

start=time.clock()

for i in range(200):

x = determine(data[i,1:8])

if x == data[i,0]:

correct=correct+1

if x!=data[i,0]:

print(i,data[i,:])

print("run time: ",time.clock()-start,"s")

print("accuracy:",correct/200)

#time of entire data

time\_mean=[]

for i in range(200):

start = time.clock()

for i in range(200):

determine(data[i,1:8])

time\_mean.append(time.clock()-start)

print(mean(time\_mean))