**Materials Informatics – Fall 2017**

**Computer Project 1 – Solutions**

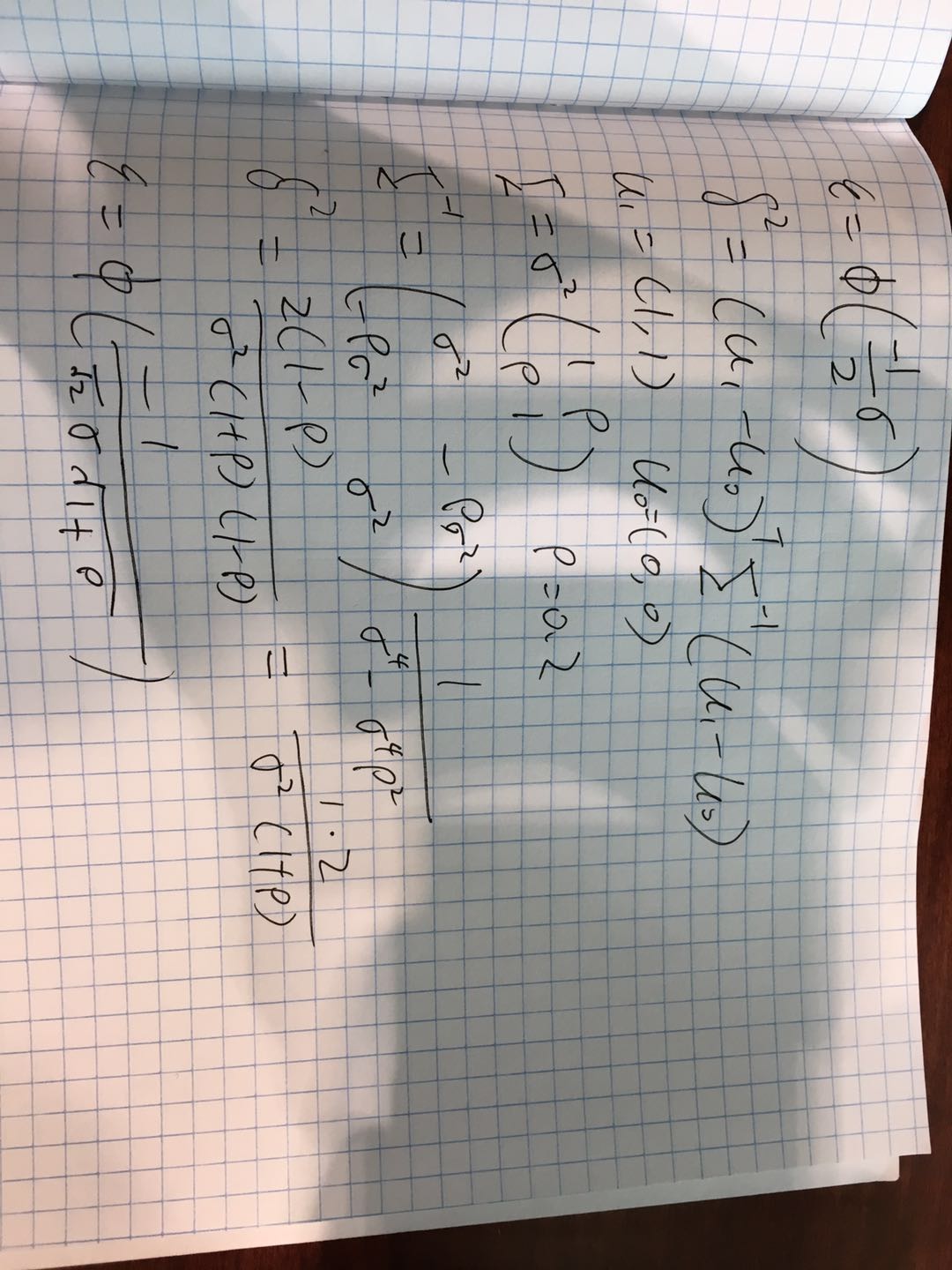
**Due on: Oct 10 2017 11:59pm**

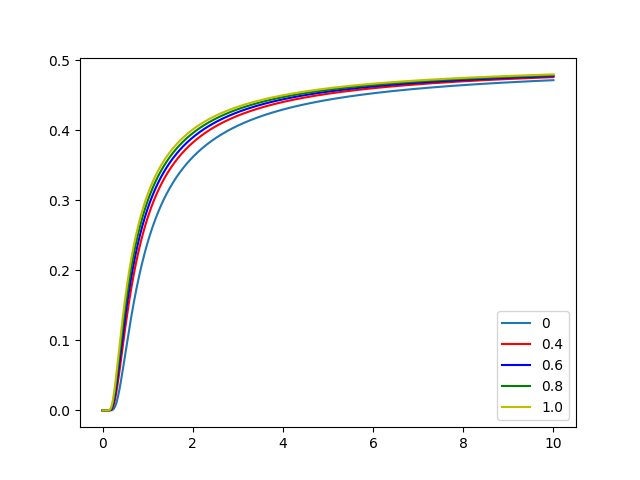
Jianfeng Song

UIN:426009910

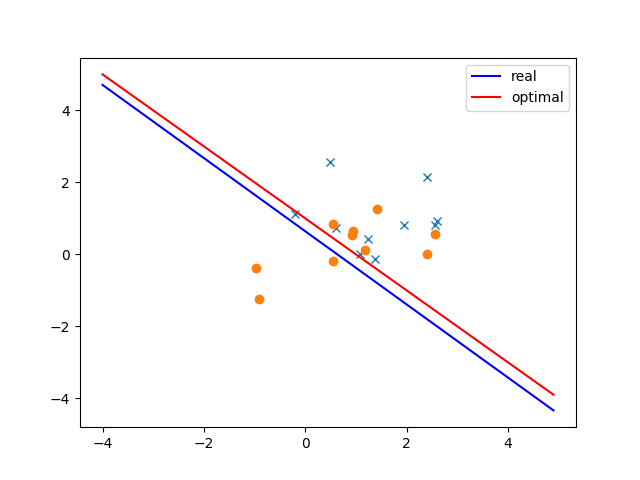
[Jsong26@tamu.edu](mailto:Jsong26@tamu.edu)

Assignment 1:

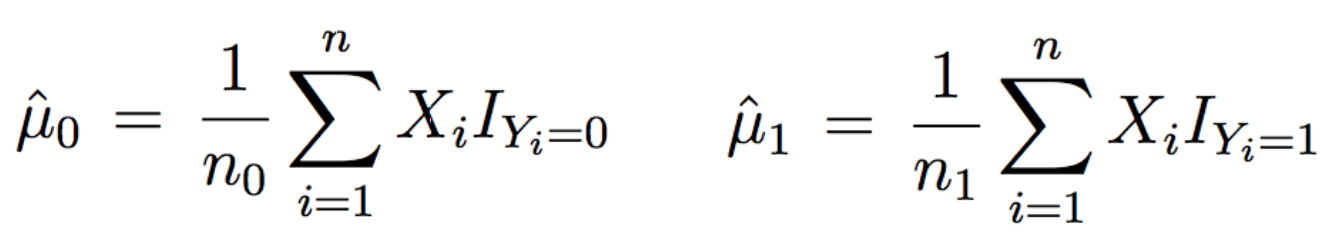


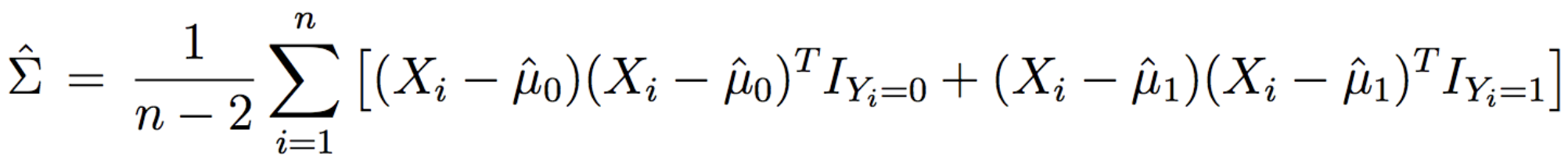


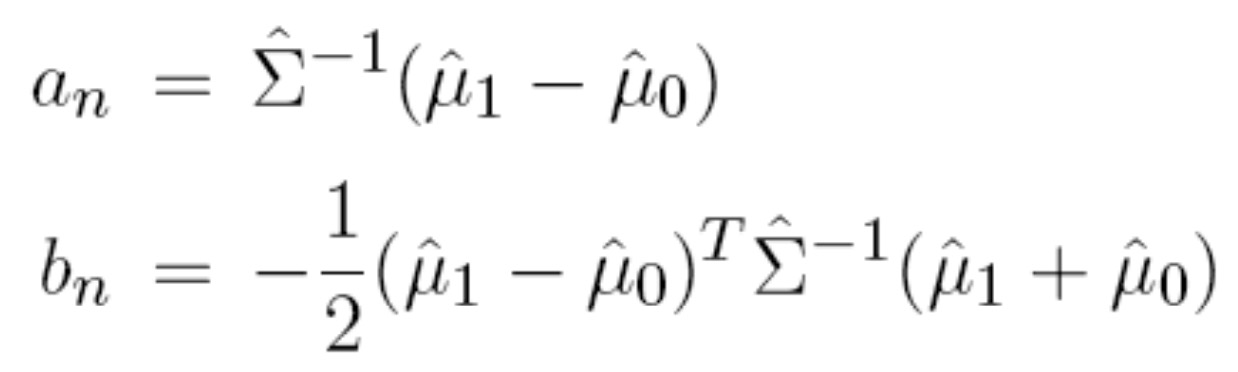
When we fixed p, as we increase variance the optimal error is also increased, when we fixed variance, the optimal error will increase as we increase p.

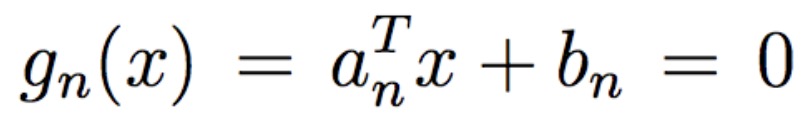


LDA uses the Gaussian assumption to estimate the optimal classifier, using the sample means and sample covariance matrices.

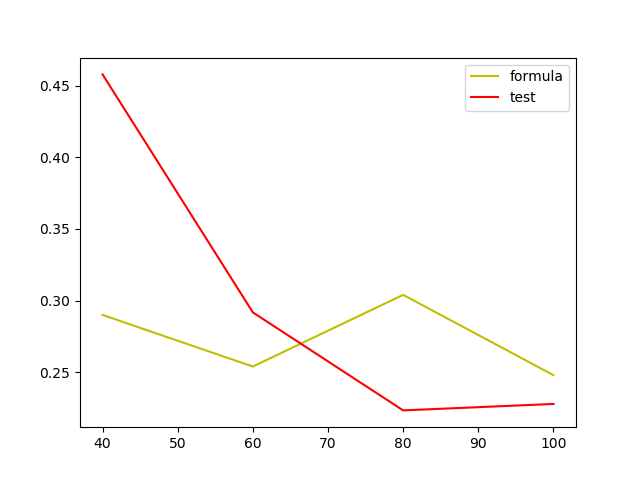








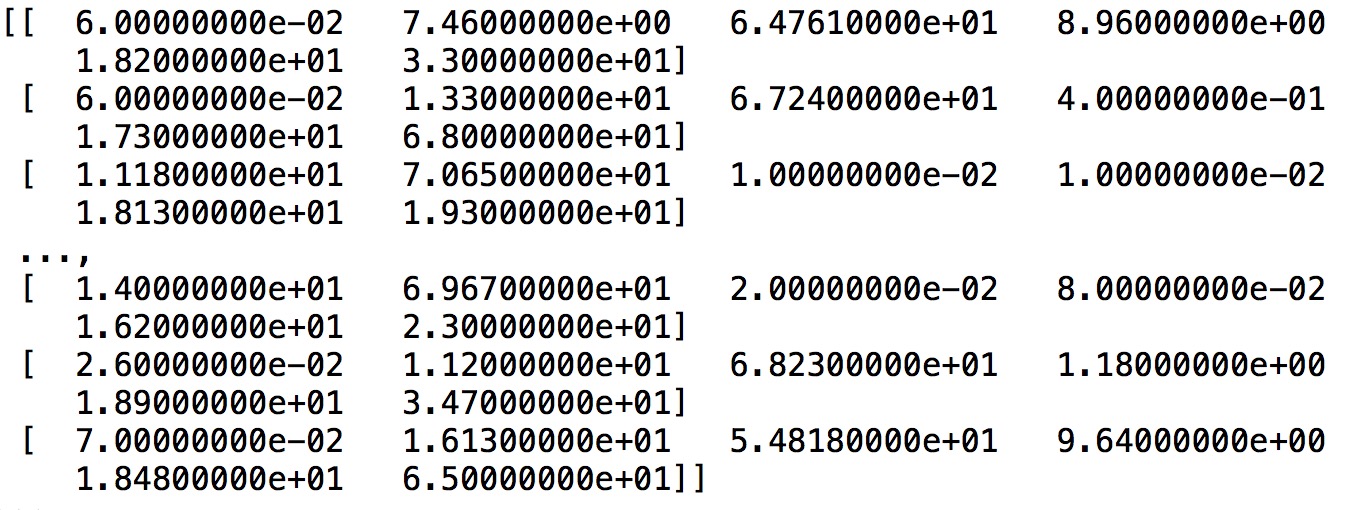
By develop the optimal classifier, we find that the optimal classifier has better describe about the data. Our designed LDA classifier should be better as we increase the size of our data, so the mean and covariance will be close to optimal classifier.



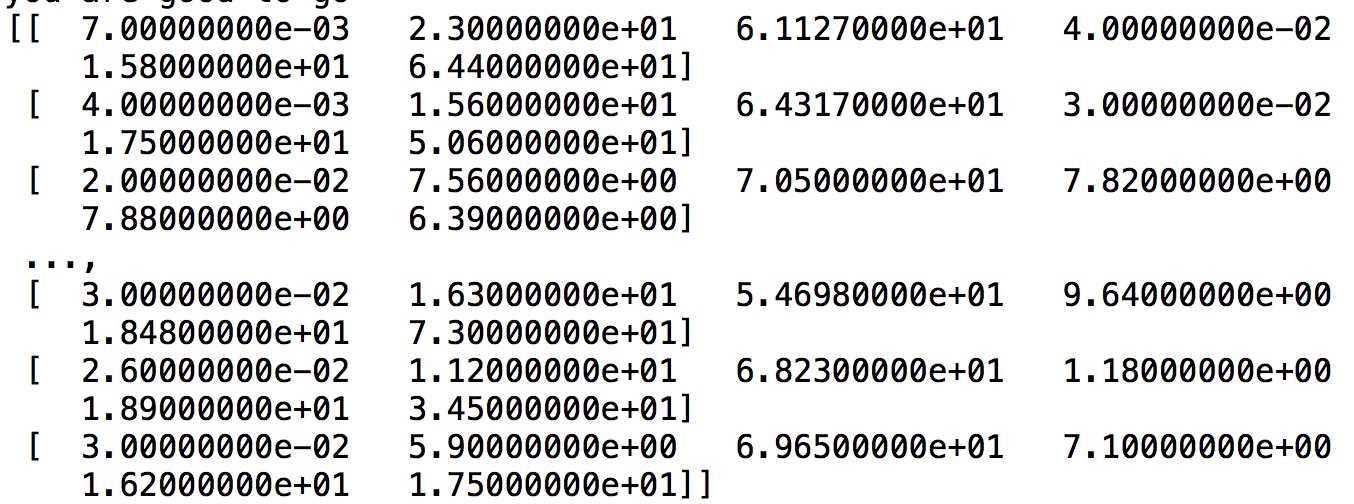
As we increase size of sample data, my test set error and formula error both became smaller. When the size of sample data is small, the test set error is bigger than formula error. In the figure above, when size of sample data is more than 65,then the test set error is better than formula error.

Assignment 2:

1. This is my training data after pre-processing



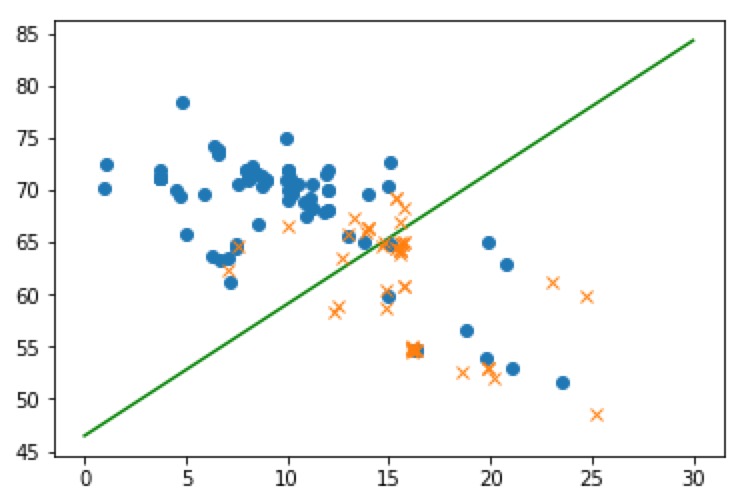
This is my test data after pre-processing



1. A

{'Ni': Ttest\_indResult(statistic=2.1832777249025574, pvalue=0.044256486360899815), 'Fe': Ttest\_indResult(statistic=1.4593669470171782, pvalue=0.16371947301978981), 'C': Ttest\_indResult(statistic=-1.4370077964097883, pvalue=0.16872848917102379), 'Mn': Ttest\_indResult(statistic=-0.555911128003719, pvalue=0.58332167105335986), 'Cr': Ttest\_indResult(statistic=-0.36725489505964665, pvalue=0.7164182125651759)}

1. Classification error with top 2: [0.07575757575757576]



1. A

Classification error with top 3:[0.06944444444444445]

Classification error with top 4:[0.06818181818181818]

Classification error with top 5:[0.06781414141414141]

As we increase number of predictors, the classification error will decrease, because we will have a better classifier.

Code:

Assignment 1

#!/usr/bin/env python3

# -\*- coding: utf-8 -\*-

"""

Created on Fri Oct 6 21:32:11 2017

@author: jianfengsong

"""

import numpy as np

import matplotlib.pyplot as plt

import scipy.integrate as integrate

import scipy.stats as ns

import math

infn=np.inf

def var(x1,y1):

h=-np.sqrt(2)\*x1\*np.sqrt(1+y1)

return 1/h

# return -math.pow(h,-1)

#def snrv(x):

# return [(2\*np.pi)\*\*0.5]\*{integrate.quad(np.exp(-(x\*\*2))/2,lambda x :-infn,lambda x:x}

x=np.arange(0.001,10.0,0.001)

y=np.arange(0.4,1.0,0.2)

plt.figure(1)

#for a in y:

# for b in x:

plt.plot(x,ns.norm.cdf(var(x,0)),label='0')

plt.plot(x,ns.norm.cdf(var(x,0.4)),'r',label='0.4')

plt.plot(x,ns.norm.cdf(var(x,0.6)),'b',label='0.6')

plt.plot(x,ns.norm.cdf(var(x,0.8)),'g',label='0.8')

plt.plot(x,ns.norm.cdf(var(x,1.0)),'y',label='1.0')

# print(x,var(x,a))

plt.legend()

plt.show()

#print (snrv(1))

#!/usr/bin/env python3

# -\*- coding: utf-8 -\*-

"""

Created on Fri Oct 6 21:32:11 2017

@author: jianfengsong

"""

import numpy as np

import matplotlib.pyplot as plt

import scipy.integrate as integrate

import scipy.stats as ns

import math

import sympy as sym

#import scipy as sym

x=sym.Symbol('x')

p=np.array([[1,0.2],[0.2,1]])

pt=p\*\*-1

u1=np.array([[1,1]])

u1t=np.matrix.transpose(u1)

u0=np.array([[0,0]])

u0t=np.matrix.transpose(u0)

x1ur=0

x2ur=0

y1ur=0

y2ur=0

plt.figure(1)

x1l=list()

x2l=list()

sample\_size=10

x1= np.random.multivariate\_normal([1,1], p,sample\_size)

x2= np.random.multivariate\_normal([0,0], p,sample\_size)

an=np.dot(pt,(u1t-u0t))

bn=(-1/2)\*np.dot(np.dot([np.matrix.transpose(u1t-u0t)],p\*\*-1),(u1t+u0t))

plt.plot(x1[:,0],x1[:,1],'x')

plt.plot(x2[:,0],x2[:,1],'o')

sumx=0

sumy=0

for a in x1:

sumx=a+sumx

sum1=sumx/sample\_size #sum1 is mean of u1(1,1)

for a in x2:

sumy=a+sumy

sum0=sumy/sample\_size#sum0 is mean of u0(0,0)

pn1=(np.dot((x1-sum0).T,(x1-sum0))+np.dot((x2-sum1).T,(x2-sum1)))/(2\*sample\_size-2)

an1=np.dot(pn1\*\*-1,(sum1-sum0))

bn1=(-1/2)\*np.dot(np.dot((sum1-sum0).T,pn1\*\*-1),(sum1-sum0))

print(an1)

print(bn1)

x1=np.arange(-4,5,0.1)

x2p=-bn1/an1[1]-an1[0]\*x1/an1[1]

plt.plot(x1,x2p,'b',label='real')

x12=np.arange(-4,5,0.1)

bn2=(-1/2)\*np.dot(np.dot((u1t-u0t).T,p\*\*-1),(u1t-u0t))

x22p=-bn2/an[1]-an[0]\*x12/an[1]

plt.plot(x12,x22p.T,'r',label='optimal')

plt.legend()

plt.show()

#optimal

#pn2=(np.dot((x1-u0.T).T,(x1-u0.T))+np.dot((x2-u1.T).T,(x2-u1.T)))/(2\*sample\_size-2)

#an2=np.dot(pn2\*\*-1,(u1-u0))

#bn2=(-1/2)\*np.dot(np.dot((u1-u0).T,pn2\*\*-1),(u1-u0))

##print(an1)

##print(bn1)

#x12=np.arange(-4,5,0.1)

#x22p=-bn2/an2[1]-an2[0]\*x1/an2[1]

#plt.plot(x12,x22p,'r')

#!/usr/bin/env python3

# -\*- coding: utf-8 -\*-

"""

Created on Sun Oct 8 15:56:58 2017

@author: jianfengsong

"""

import numpy as np

import matplotlib.pyplot as plt

import scipy.stats as ns

import math

from scipy.linalg import det

p=np.array([[1,0.2],[0.2,1]])

u0=np.array([[0,0]])

u1=np.array([[1,1]])

sample\_size=np.array([20,30,40,50])

LDA\_error\_set=list()

error\_set=list()

for a in sample\_size:

x1= np.random.multivariate\_normal([1,1], p,a)

x2= np.random.multivariate\_normal([0,0], p,a)

x3= np.random.multivariate\_normal([1,1],p,250)

x4= np.random.multivariate\_normal([0,0],p,250)

sumx1=0

sumx2=0

for b in x1:

sumx1=b+sumx1

mean\_x1=sumx1/a

for c in x2:

sumx2=c+sumx2

mean\_x0=sumx2/a

cov=(np.dot((x1-mean\_x0).T,(x1-mean\_x0))+np.dot((x2-mean\_x1).T,(x2-mean\_x1)))/(2\*a-2)

an=np.dot(cov\*\*-1,(mean\_x1-mean\_x0))

bn=(-1/2)\*np.dot(np.dot((mean\_x1-mean\_x0).T,cov\*\*(-1)),(mean\_x1-mean\_x0))

var\_x0=(np.dot(an,mean\_x0.T)+bn)/math.sqrt(np.dot(np.dot(an,p),an.T))

var\_x1=(np.dot(an,mean\_x1.T)+bn)/math.sqrt(np.dot(np.dot(an,p),an.T))

LDA\_error=1/2\*(ns.norm.cdf(var\_x0)+ns.norm.cdf(-var\_x1))

LDA\_error\_set.append(LDA\_error)

clas\_x3\_y=-bn/an[1]-an[0]\*x3[:,0]/an[1]

clas\_x4\_y=-bn/an[1]-an[0]\*x4[:,0]/an[1]

error\_time=0

for t in range(250):

if x3[t,1] < clas\_x3\_y[t]:

error\_time=error\_time+1

if x4[t,1] > clas\_x4\_y[t]:

error\_time=error\_time+1

error\_set.append(error\_time/(500))

plt.plot(sample\_size\*2,error\_set,'y',label='formula')

plt.plot(sample\_size\*2,LDA\_error\_set,'r',label='test')

plt.legend()

plt.show()

Assignment 2

1. A

#!/usr/bin/env python3

# -\*- coding: utf-8 -\*-

"""

Created on Sun Oct 8 20:04:49 2017

@author: jianfengsong

"""

import xlrd as xl

import numpy as np

import matplotlib.pyplot as plt

import scipy.integrate as integrate

import scipy.stats as ns

import math

import random

import xlwt as xlw

rows\_value=list()

cols\_value=list()

excel=xl.open\_workbook('SFE\_Dataset.xlsx')

data\_table=excel.sheet\_by\_index(0)

rows=data\_table.nrows

cols=data\_table.ncols

for a in range(1,rows,1):

if data\_table.row\_values(a,cols-1)[0]>=45 or data\_table.row\_values(a,cols-1)[0]<=35 :

rows\_value.append(data\_table.row\_values(a))

#for a in range(1,cols,1):

# if data\_table.col\_values(cols-1):

# cols\_value.append(data\_table.col\_values(a))

#for a in range(rows):

ele\_0=list()

num\_0=0

for a in range(0,cols,1):

for b in range(0,len(rows\_value),1):

if rows\_value[b][a]<=0.00000001:

num\_0=num\_0+1

if num\_0/len(rows\_value)>=0.4:

# print(len(rows\_value))

num\_0=0

ele\_0.append(a)

for row in rows\_value:

num\_del=0

for col in ele\_0:

col=col-num\_del

row.remove(row[col])

num\_del=num\_del+1

num\_del\_row=list()

num\_0\_row=0

#delete row value that are 0

for row in rows\_value:

# print (row)

for a in row:

# print(a)

if a==0:

num\_0\_row=num\_0\_row+1

if num\_0\_row>0:

num\_0\_row=0

num\_del\_row.append(row)

for a in range(0,len(num\_del\_row),1):

num=0

a=a-num

rows\_value.remove(num\_del\_row[a])

num=num+1

#random choose value for test set

numtrain=list()

numtest=list()

test\_set=list()

train\_set=list()

status=True

go=0

#num=0

while (status):

num=0

go=go+1

print(go)

random\_set=random.sample(range(len(rows\_value)),len(rows\_value))

for a in range(int(len(rows\_value)\*0.2)):

numtrain.append(random\_set[a])

for a in range(len(numtrain)):

train\_set.append(rows\_value[numtrain[a]])

for a in range(len(rows\_value)-int(len(rows\_value)\*0.2)):

numtest.append(random\_set[a+int(len(rows\_value)\*0.2)])

for a in range(len(numtest)):

test\_set.append(rows\_value[numtest[a]])

for row in train\_set:

if row[len(row)-1]<=35:

num=num+1

if num/len(train\_set)>=0.55 or num/len(train\_set)<=0.45:

numtrain=list()

numtest=list()

test\_set=list()

train\_set=list()

print("not yet")

# if num/len(train\_set)<0.55 and num/len(train\_set)<0.45:

else:

status=False

print("you are good to go")

data\_final=xlw.Workbook()

sheet1=data\_final.add\_sheet('sheet1',cell\_overwrite\_ok=True)

for i in range(len(train\_set)):

for j in range(len(train\_set[i])):

sheet1.write(i,j,train\_set[i][j])

data\_final.save('data\_final.xls')

test\_set1=np.asarray(test\_set)

train\_set0=np.asarray(train\_set)

print(test\_set1)

print(train\_set0)

1. A

import xlrd as xl

import numpy as np

import matplotlib.pyplot as plt

import scipy.integrate as integrate

import scipy.stats as ns

import math

import random

rows\_value=list()

cols\_value=list()

excel=xl.open\_workbook('SFE\_Dataset.xlsx')

data\_table=excel.sheet\_by\_index(0)

rows=data\_table.nrows

cols=data\_table.ncols

for a in range(1,rows,1):

if data\_table.row\_values(a,cols-1)[0]>=45 or data\_table.row\_values(a,cols-1)[0]<=35 :

rows\_value.append(data\_table.row\_values(a))

ele\_0=list()

num\_0=0

for a in range(0,cols,1):

for b in range(0,len(rows\_value),1):

if rows\_value[b][a]<=0.00000001:

num\_0=num\_0+1

if num\_0/len(rows\_value)>=0.4:

num\_0=0

ele\_0.append(a)

for row in rows\_value:

num\_del=0

for col in ele\_0:

col=col-num\_del

row.remove(row[col])

num\_del=num\_del+1

num\_del\_row=list()

num\_0\_row=0

for row in rows\_value:

for a in row:

if a==0:

num\_0\_row=num\_0\_row+1

if num\_0\_row>0:

num\_0\_row=0

num\_del\_row.append(row)

for a in range(0,len(num\_del\_row),1):

num=0

a=a-num

rows\_value.remove(num\_del\_row[a])

num=num+1

#random choose value for test set

numtrain=list()

numtest=list()

test\_set=list()

train\_set=list()

status=True

go=0

while (status):

num=0

go=go+1

# print(go)

random\_set=random.sample(range(len(rows\_value)),len(rows\_value))

for a in range(int(len(rows\_value)\*0.2)):

numtrain.append(random\_set[a])

for a in range(len(numtrain)):

train\_set.append(rows\_value[numtrain[a]])

for a in range(len(rows\_value)-int(len(rows\_value)\*0.2)):

numtest.append(random\_set[a+int(len(rows\_value)\*0.2)])

for a in range(len(numtest)):

test\_set.append(rows\_value[numtest[a]])

for row in train\_set:

if row[len(row)-1]<=35:

num=num+1

if num/len(train\_set)>=0.55 or num/len(train\_set)<=0.45:

numtrain=list()

numtest=list()

test\_set=list()

train\_set=list()

# print("not yet")

# if num/len(train\_set)<0.55 and num/len(train\_set)<0.45:

else:

status=False

# print("you are good to go")

#save excel doc

data\_final=xlw.Workbook()

sheet1=data\_final.add\_sheet('sheet1',cell\_overwrite\_ok=True)

for i in range(len(train\_set)):

for j in range(len(train\_set[i])):

sheet1.write(i,j,train\_set[i][j])

data\_final.save('data\_final.xls')

#Assignment2(b)

train\_set35=list()

train\_set45=list()

length\_train=len(train\_set)

for a in range(0,length\_train,1):

if train\_set[a][len(train\_set[a])-1]<=35:

train\_set35.append(a)

else:

train\_set45.append(a)

train35=list()

train45=list()

for a in train\_set35:

train35.append(train\_set[a])

for a in train\_set45:

train45.append(train\_set[a])

#train\_35=list()

#train\_45=list()

#for a in range(len(train35[1])):

# for b in range(len(train35)):

# train\_35.append(train35[b][a])

#for a in range(len(train45[1])):

# for b in range(len(train45)):

# train\_45.append(train45[b][a])

train\_35=[[] for i in range(len(train35[1]))]

train\_45=[[] for i in range(len(train45[1]))]

Tset=list()

for a in range(len(train35[1])):

for b in range(len(train35)):

train\_35[a].append(train35[b][a])

for a in range(len(train45[1])):

for b in range(len(train45)):

train\_45[a].append(train45[b][a])

for a in range(len(train\_35)-1):

h=ns.ttest\_ind(train\_35[a],train\_45[a],equal\_var=False)

Tset.append(h)

Tset\_sta=list()

for a in range(len(Tset)):

Tset\_sta.append(abs(Tset[a][0]))

#print (Tset\_sta)

Tset\_sta\_name={'C':Tset\_sta[0],'Ni':Tset\_sta[1],'Fe':Tset\_sta[2],'Mn':Tset\_sta[3],'Cr':Tset\_sta[4]}

Tset\_name={'C':Tset[0],'Ni':Tset[1],'Fe':Tset[2],'Mn':Tset[3],'Cr':Tset[4]}

Tset\_name0=np.asarray(Tset\_name)

print(Tset\_name0)

import operator

sorted\_tset = sorted(Tset\_sta\_name.items(), key=operator.itemgetter(1),reverse=True)

#print(sorted\_tset)

1. A

import xlrd as xl

import numpy as np

import matplotlib.pyplot as plt

import scipy.integrate as integrate

import scipy.stats as ns

import math

import random

import operator

rows\_value=list()

cols\_value=list()

excel=xl.open\_workbook('SFE\_Dataset.xlsx')

data\_table=excel.sheet\_by\_index(0)

rows=data\_table.nrows

cols=data\_table.ncols

for a in range(1,rows,1):

if data\_table.row\_values(a,cols-1)[0]>=45 or data\_table.row\_values(a,cols-1)[0]<=35 :

rows\_value.append(data\_table.row\_values(a))

ele\_0=list()

num\_0=0

for a in range(0,cols,1):

for b in range(0,len(rows\_value),1):

if rows\_value[b][a]<=0.00000001:

num\_0=num\_0+1

if num\_0/len(rows\_value)>=0.4:

num\_0=0

ele\_0.append(a)

for row in rows\_value:

num\_del=0

for col in ele\_0:

col=col-num\_del

row.remove(row[col])

num\_del=num\_del+1

num\_del\_row=list()

num\_0\_row=0

for row in rows\_value:

for a in row:

if a==0:

num\_0\_row=num\_0\_row+1

if num\_0\_row>0:

num\_0\_row=0

num\_del\_row.append(row)

for a in range(0,len(num\_del\_row),1):

num=0

a=a-num

rows\_value.remove(num\_del\_row[a])

num=num+1

#random choose value for test set

numtrain=list()

numtest=list()

test\_set=list()

train\_set=list()

status=True

go=0

while (status):

num=0

go=go+1

# print(go)

random\_set=random.sample(range(len(rows\_value)),len(rows\_value))

for a in range(int(len(rows\_value)\*0.2)):

numtrain.append(random\_set[a])

for a in range(len(numtrain)):

train\_set.append(rows\_value[numtrain[a]])

for a in range(len(rows\_value)-int(len(rows\_value)\*0.2)):

numtest.append(random\_set[a+int(len(rows\_value)\*0.2)])

for a in range(len(numtest)):

test\_set.append(rows\_value[numtest[a]])

for row in train\_set:

if row[len(row)-1]<=35:

num=num+1

if num/len(train\_set)>=0.55 or num/len(train\_set)<=0.45:

numtrain=list()

numtest=list()

test\_set=list()

train\_set=list()

else:

status=False

#save excel doc

#data\_final=xlw.Workbook()

#sheet1=data\_final.add\_sheet('sheet1',cell\_overwrite\_ok=True)

#for i in range(len(train\_set)):

# for j in range(len(train\_set[i])):

# sheet1.write(i,j,train\_set[i][j])

#data\_final.save('data\_final.xls')

#Assignment2(b)

train\_set35=list()

train\_set45=list()

length\_train=len(train\_set)

for a in range(0,length\_train,1):

if train\_set[a][len(train\_set[a])-1]<=35:

train\_set35.append(a)

else:

train\_set45.append(a)

train35=list()

train45=list()

for a in train\_set35:

train35.append(train\_set[a])

for a in train\_set45:

train45.append(train\_set[a])

train\_35=[[] for i in range(len(train35[1]))]

train\_45=[[] for i in range(len(train45[1]))]

Tset=list()

for a in range(len(train35[1])):

for b in range(len(train35)):

train\_35[a].append(train35[b][a])

for a in range(len(train45[1])):

for b in range(len(train45)):

train\_45[a].append(train45[b][a])

for a in range(len(train\_35)-1):

h=ns.ttest\_ind(train\_35[a],train\_45[a],equal\_var=False)

Tset.append(h)

Tset\_sta=list()

for a in range(len(Tset)):

Tset\_sta.append(abs(Tset[a][0]))

#print (Tset\_sta)

Tset\_sta\_name={'C':Tset\_sta[0],'Ni':Tset\_sta[1],'Fe':Tset\_sta[2],'Mn':Tset\_sta[3],'Cr':Tset\_sta[4]}

sorted\_tset = sorted(Tset\_sta\_name.items(), key=operator.itemgetter(1),reverse=True)

print(sorted\_tset)

#Assignment2(c)

top=list()

#for a in range(len(sorted\_tset)):

# top.append(sorted\_tset[a][1])

#x1=[[] for i in range(len(train35))]

#x2=[[] for i in range(len(train45))]

x11=list()

x21=list()

#for a in range(len(x1)):

#for a in range(len(train35)):

first=Tset\_sta.index(sorted\_tset[0][1])

second=Tset\_sta.index(sorted\_tset[1][1])

for b in range(len(train35)):

x11.append([train35[b][1],train35[b][2]])

for b in range(len(train45)):

x21.append([train45[b][1],train45[b][2]])

###different

#for b in range(len(train35)):

# x11.append([train35[b][first],train35[b][second]])

#for b in range(len(train45)):

# x21.append([train45[b][first],train45[b][second]])

sumx1=0

sumx2=[0,0]

x1=np.asarray(x11)

x2=np.asarray(x21)

for b in x1:

sumx1=b+sumx1

mean\_x1=sumx1/len(x1)

for c in x2:

sumx2=c+sumx2

mean\_x0=sumx2/len(x2)

cov=(np.dot((x1-mean\_x0).T,(x1-mean\_x0))+np.dot((x2-mean\_x1).T,(x2-mean\_x1)))/(min(len(train35),len(train45)-2))

an=np.dot(cov\*\*-1,(mean\_x1-mean\_x0))

bn=(-1/2)\*np.dot(np.dot((mean\_x1-mean\_x0).T,cov\*\*(-1)),(mean\_x1-mean\_x0))

plt.figure(1)

plt.plot(x1[:,0],x1[:,1],'x')

plt.plot(x2[:,0],x2[:,1],'o')

x1=np.arange(-10,100,1)

x2p=-bn/an[1]-an[0]\*x1/an[1]

plt.plot(x1,x2p.T,'r')

#test set

test\_set35=list()

test\_set45=list()

length\_test=len(test\_set)

for a in range(0,length\_test,1):

if test\_set[a][len(test\_set[a])-1]<=35:

test\_set35.append(a)

else:

test\_set45.append(a)

test35=list()

test45=list()

for a in test\_set35:

test35.append(test\_set[a])

for a in test\_set45:

test45.append(test\_set[a])

test\_35=[[] for i in range(len(test35[1]))]

test\_45=[[] for i in range(len(test45[1]))]

#Tset=list()

x112=list()

x212=list()

for a in range(len(test35[1])):

for b in range(len(test35)):

test\_35[a].append(test35[b][a])

for a in range(len(test45[1])):

for b in range(len(test45)):

test\_45[a].append(test45[b][a])

for b in range(len(test35)):

x112.append([test35[b][1],test35[b][2]])

for b in range(len(test45)):

x212.append([test45[b][1],test45[b][2]])

sumx12=0

sumx22=[0,0]

x12=np.asarray(x112)

x22=np.asarray(x212)

for b in x12:

sumx12=b+sumx12

mean\_x12=sumx12/len(x12)

for c in x22:

sumx22=c+sumx22

mean\_x02=sumx22/len(x22)

cov2=(np.dot((x12-mean\_x02).T,(x12-mean\_x02))+np.dot((x22-mean\_x12).T,(x22-mean\_x12)))/(min(len(test35),len(test45)-2))

an2=np.dot(cov2\*\*-1,(mean\_x12-mean\_x02))

bn2=(-1/2)\*np.dot(np.dot((mean\_x12-mean\_x02).T,cov2\*\*(-1)),(mean\_x12-mean\_x02))

var\_x0=(np.dot(an2,mean\_x02.T)+bn2)/math.sqrt(np.dot(np.dot(an2,cov2),an2.T))

var\_x1=(np.dot(an2,mean\_x12.T)+bn2)/math.sqrt(np.dot(np.dot(an2,cov2),an2.T))

LDA\_error=1/2\*(ns.norm.cdf(var\_x0)+ns.norm.cdf(-var\_x1))

clas\_x12\_y=-bn2/an2[1]-an2[0]\*x12[:,0]/an[1]

clas\_x22\_y=-bn/an[1]-an[0]\*x22[:,0]/an[1]

error\_time=0

error\_set=list()

for t in range(min(len(test45),len(test35))):

if x22[t,1] > clas\_x22\_y[t]:

error\_time=error\_time+1

if x12[t,1] < clas\_x12\_y[t]:

error\_time=error\_time+1

error\_set.append(error\_time/(len(test\_set)\*len(test\_set[1])))

print(error\_set)

#print(LDA\_error)

#LDA\_error\_set.append(LDA\_error)

1. For part d we just need to change two variable to calculate the result that we want.

With 3 predictors:

for b in range(len(test35)):

x112.append([test35[b][1],test35[b][2],test35[b][3]])

for b in range(len(test45)):

x212.append([test45[b][1],test45[b][2],test45[b][3]])

With 4 predictors:

for b in range(len(test35)):

x112.append([test35[b][1],test35[b][2],test35[b][3],test35[b][0]])

for b in range(len(test45)):

x212.append([test45[b][1],test45[b][2],test45[b][3],test45[b][0]])

With 5 predictors:

for b in range(len(test35)):

x112.append([test35[b][1],test35[b][2],test35[b][3],test35[b][0],test35[b][4]])

for b in range(len(test45)):

x212.append([test45[b][1],test45[b][2],test45[b][3],test45[b][0],test35[b][4]])