Mohammad Saleh
GRA at PHY WSU
m.saleh@cern.ch
fa9520@wayne.edu
Jan 25, 2017

CSC 5825
Homework 1

Solutions:

Question 1:

a. Let start the comparison by looking at the advantages of OCR over scanning and faxing it: – The OCR will be able to save the document as a characters (A, a, b). This will make the documents editable. While scanning will give you an image only and you cannot change any text in the document. So if you looking to edit or copy the text the OCR will be more preferable in this case.
– The disadvantage of using OCR is the accuracy. As the document may include different fonts and hand writing, the OCR will not be accurate where the scan will show the accurate document. So if the details in the document is the important concern then the scan will be more preferable.
– May be one other advantage of the optical reader is the size of the machine as the OCR can be smaller than the scanner but it will actually depend on the model of the scanner and the OCR. In case your are constrained to small area, probably the optical reader will be more preferable.

b. The OCR will fail in many situation: As we mentioned before many fonts are available and some of them can be cursive which will be very hard for the OCR to e able to detect the characters. Also the OCR is matching the character with a template pixel by pixel. If we have our character with low resolution (small number of pixel), the OCR will fail as it will not have enough pixel information.
The barcode reader is still used because it very reliable as the barcode is a combination of line and the barcode reader will not fail to estimate the difference in the distance between whose line. Also the technology used in the barcode is simple and efficient and money wise it is cheaper than the OCR. So in case you need to tag a merchandise you don't need to assign characters to be read by OCR later on. The barcode will be enough to tag the merchandise.

Question 2:

According to the equation of the circle eq.1, we have the radius and the center coordinates parameters that define any circle. The parameters for the hypothesis circle will be in between the most specified and the most general hypothesis in this sense will be only talking about changing the radius of the circle. The ellipse will be more useful as in the circle case we are constrained with the size of the radius and the variance across the x-axis and the y-axis will be the same where in the ellipse you have two different sizes of the axes (major and minor) and you can see the correlation between the x-axis and the y-axis more clearly. In addition to the major and minor axes we can also use a rotation angle for ellipse to include desired point (as example figure1) rather than the circle which will not have difference if you do rotation (isotropic). To generalize for K¿2, we can use N binary hypothesis, with problems less than using triangles as using ellipse we will have more control on the K class using the two-axes and rotation angle.

$$(x - x_0)^2 + (y - y_0)^2 = R^2 \tag{1}$$

Question 3:

The answer for this question depends mainly on the application but we can discuss the different possible cases. S is the closest to the positive (doesn't make false positive) and G is close to the negative (doesn't make false negative). In the case where false positive and false negative have the same cost, we can choose h halfway between S and G. Whereas if false positive costs more than false negative, we choose h closer to S. Also, if false negative costs more than false positive, we choose h closer to G. As we said early it is application dependent.

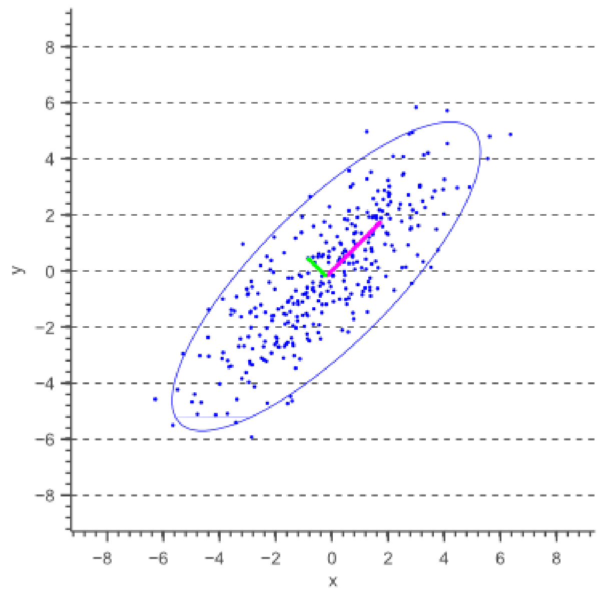

Figure 1: an example of particle correlation represented by an ellipse rotated by some angle.

Question 4:
For given training set, we set S, most specified hypothesis, and G, most generalized hypothesis. Then we choose our h between S and G but we are not sure if h closer to S or G. SO the region between S and G depend on our intuitive. So the best queries by supervisor will be for instances between S and Q. In this case the region between S and G will be smaller as more instances will be available and we will be able to set larger boundaries on S and G.

Question 5:
a. The likelihood ratio is defined as $p(\mathbf{x}|C_i)/p(\mathbf{x}|C_j)$, The discriminant function discriminates between two classes, defined as the ratio of class conditional density using Baye's rule: $g(\mathbf{x}) = P(C_i|x)/P(C_j|x)$, i≠j, where it will choose C_i if $g(\mathbf{x}) > 1$ and C_j otherwise. Also from Baye's rule: $P(C_k|x) = p(x|C_k) * P(C_k)/p(x)$. Implies that $g(x) = \frac{p(x|C_i)}{p(x|C_j)} * \frac{P(C_i)}{P(C_j)}$, which include the liklihood ratio.

b. The same analogy here where the log odds ratio is defined as $\log \frac{P(C_i|x)}{P(C_j|x)}$. Also the discriminating function will be log formulated, $g(x) = \log \frac{P(C_i|x)}{P(C_j|x)}$, which choose C_i if $g(\mathbf{x}) > 0$ (since log(1)=0) and C_j otherwise. Also from Baye's rule: $P(C_k|x) = p(x|C_k) * P(C_k)/p(x)$. Implies that $g(x) = \log \frac{p(x|C_i)}{p(x|C_j)} + \log \frac{P(C_i)}{P(C_j)}$

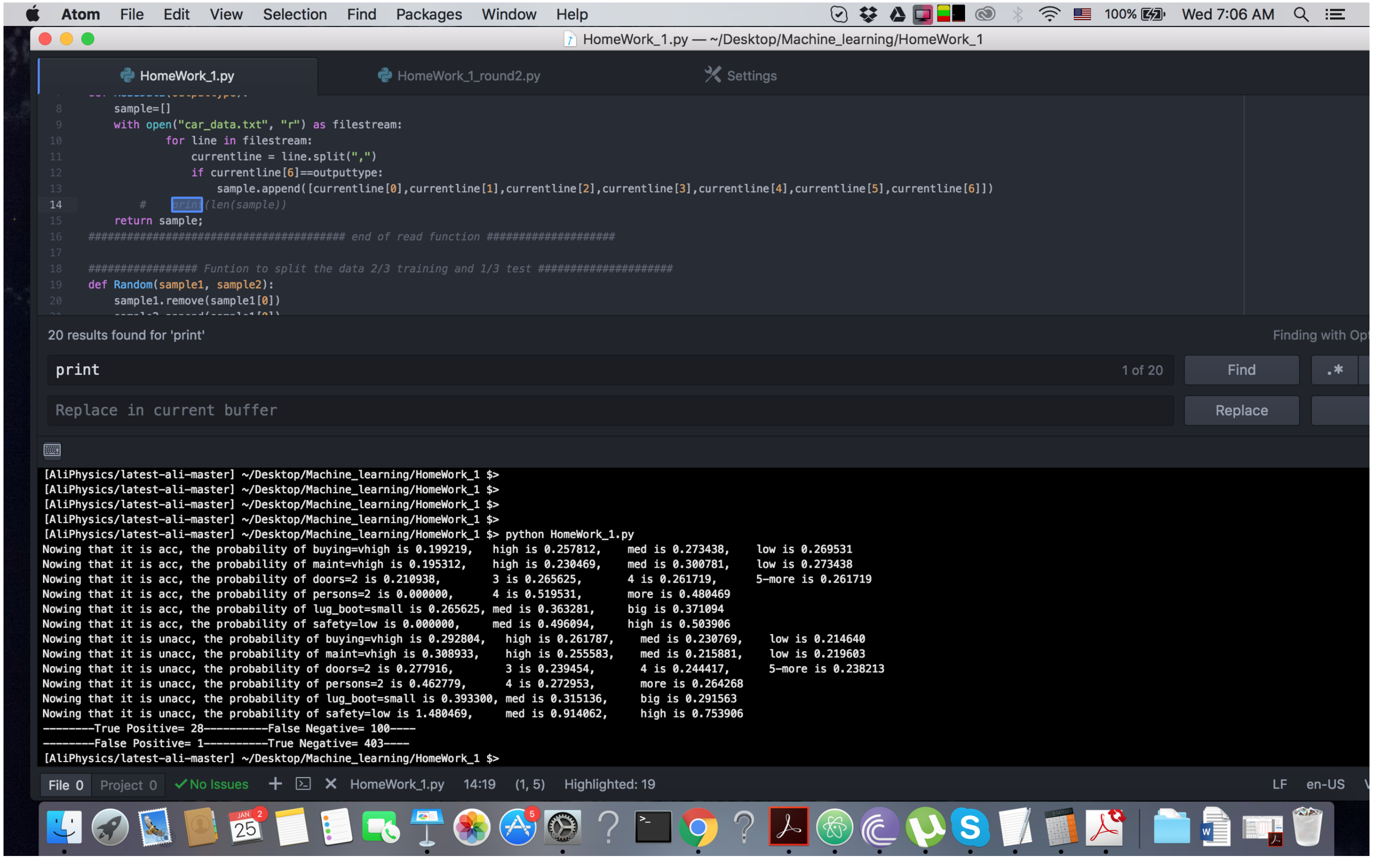

Figure 2: snapshot of my code after compiling. It shows all the requested parameters for the homework.

Question 6:

Below is the syntax for all the parts (a, b, c, d). I did add comments in order to make it more clear. I did also include a snapshot for the compiled code. Sorry incase of not perfectly organised code. I am mainly using C++ in my research but I wrote the syntax in PYTHON to get familiar with the syntax (of course same logic as in C++).

```
import random
import math
from decimal import *
getcontext().prec = 6

################## Funtion to read the data from the text file ######################
def ReadData(outputtype):
    sample=[]
    with open("car_data.txt", "r") as filestream:
            for line in filestream:
                currentline = line.split(",")
                if currentline[6]==outputtype:
                    sample.append([currentline[0], currentline[1], currentline[2], currentline[3],
        #       print(len(sample))
    return sample;
######################################## end of read function ####################

################## Funtion to split the data 2/3 training and 1/3 test ######################
def Random(sample1, sample2):
```

```
    sample1.remove(sample1[0])
    sample2.append(sample1[0])
    while (float(len(sample1))/float(len(sample2)))>2.0:
        import random
        random=random.randint(0,len(sample1)-1)
    #    print(random)
        sample2.append(sample1[random])
        sample1.remove(sample1[random])
    #    print(len(sample2))
    return;
################# end of random function #####################

################# Funtion to optain the conditional Probability using Naive Bayes classifier ##
################## We could done this function more general and loop based but for now it is m
def Byes(sample,sampleoutput,Buying,Maint,Doors,Persons,Boot,Safety):
    for i in range(0,len(sample)):
        if(sample[i][6]==sampleoutput):
            if(sample[i][0]=='vhigh'):
                Buying[0]+=1.0
            if(sample[i][0]=='high'):
                Buying[1]+=1.0
            if(sample[i][0]=='med'):
                Buying[2]+=1.0
            if(sample[i][0]=='low'):
                Buying[3]+=1.0
            if(sample[i][1]=='vhigh'):
                Maint[0]+=1.0
            if(sample[i][1]=='high'):
                Maint[1]+=1.0
            if(sample[i][1]=='med'):
                Maint[2]+=1.0
            if(sample[i][1]=='low'):
                Maint[3]+=1.0
            if(sample[i][2]=='2'):
                Doors[0]+=1.0
            if(sample[i][2]=='3'):
                Doors[1]+=1.0
            if(sample[i][2]=='4'):
                Doors[2]+=1.0
            if(sample[i][2]=='5more'):
                Doors[3]+=1.0
            if(sample[i][3]=='2'):
                Persons[0]+=1.0
            if(sample[i][3]=='4'):
                Persons[1]+=1.0
            if(sample[i][3]=='more'):
                Persons[2]+=1.0
            if(sample[i][4]=='small'):
                Boot[0]+=1.0
            if(sample[i][4]=='med'):
                Boot[1]+=1.0
            if(sample[i][4]=='big'):
                Boot[2]+=1.0
            if(sample[i][5]=='low'):
```

```
                    Safety[0]+=1.0
                if(sample[i][5]=='med'):
                    Safety[1]+=1.0
                if(sample[i][5]=='high'):
                    Safety[2]+=1.0
    return;
################# end of Naive Bayes classifier function #####################




testsample=[]
testsample1=[]
buyingProb=[0]*4
maintProb=[0]*4
doorsProb=[0]*4
personProb=[0]*3
bootProb=[0]*3
safetyProb=[0]*3
buyingProb1=[0]*4
maintProb1=[0]*4
doorsProb1=[0]*4
personProb1=[0]*3
bootProb1=[0]*3
safetyProb1=[0]*3
multiply=[0]*6
multiply1=[0]*6
classifier=[0]*4
accsample=ReadData("acc\n")    ###### store the acc output cases in accsample
unccsample=ReadData("unacc\n") ###### store the unacc output cases in unccsample
Random(accsample,testsample1)    ###### random 2/3 training set stored in accsample while test s
Random(unccsample,testsample)   ###### random 2/3 training set stored in unccsample while test s
testsample.extend(testsample1)    ###### merging the training and testing sample with the same p
unccsample.extend(accsample)

Byes(unccsample,"acc\n",buyingProb1,maintProb1,doorsProb1,personProb1,bootProb1,safetyProb1)
######### trainning the classifier (+ive case)
Byes(unccsample,"unacc\n",buyingProb,maintProb,doorsProb,personProb,bootProb,safetyProb)
######### trainning the classifier (-ive case)

x=sum(buyingProb)
y=sum(buyingProb1)

for i in range(0,4):  ######### normalizing the probabilities
    buyingProb[i]=buyingProb[i]/x
    maintProb[i]=maintProb[i]/x
    doorsProb[i]=doorsProb[i]/x
    buyingProb1[i]=buyingProb1[i]/y
    maintProb1[i]=maintProb1[i]/y
    doorsProb1[i]=doorsProb1[i]/y
for i in range(0,3):
    personProb[i]=personProb[i]/x
    bootProb[i]=bootProb[i]/x
    safetyProb[i]=safetyProb[i]/y
```

```
    personProb1[i]=personProb1[i]/y
    bootProb1[i]=bootProb1[i]/y
    safetyProb1[i]=safetyProb1[i]/y
#
######### Printing all the parameters from training as requested
samplekind1='acc'
print"Nowing that it is %s, the probability of buying=vhigh is %f,   high is %f,    med is %f, 
print"Nowing that it is %s, the probability of maint=vhigh is %f,    high is %f,    med is %f, 
print"Nowing that it is %s, the probability of doors=2 is %f,        3 is %f,       4 is %f,   
print"Nowing that it is %s, the probability of persons=2 is %f,      4 is %f,       more is %f"
print"Nowing that it is %s, the probability of lug_boot=small is %f, med is %f,     big is %f"%
print"Nowing that it is %s, the probability of safety=low is %f,     med is %f,     high is %f"

personProb1[0]=personProb1[0]+0.2    ######### TO avoid zero probability  ###################
safetyProb1[0]=safetyProb1[0]+0.2    ######### TO avoid zero probability  ###################

samplekind='unacc'
print"Nowing that it is %s, the probability of buying=vhigh is %f,   high is %f,    med is %f, 
print"Nowing that it is %s, the probability of maint=vhigh is %f,    high is %f,    med is %f, 
print"Nowing that it is %s, the probability of doors=2 is %f,        3 is %f,       4 is %f,   
print"Nowing that it is %s, the probability of persons=2 is %f,      4 is %f,       more is %f"
print"Nowing that it is %s, the probability of lug_boot=small is %f, med is %f,     big is %f"%
print"Nowing that it is %s, the probability of safety=low is %f,     med is %f,     high is %f"



############## validation check and TP TN FP FN using sample tests ######################
for i in range(0,len(testsample)):
    if testsample[i][0]=='vhigh':
        multiply[0]=buyingProb[0]
        multiply1[0]=buyingProb1[0]
    if testsample[i][0]=='high':
        multiply[0]=buyingProb[1]
        multiply1[0]=buyingProb1[1]
    if testsample[i][0]=='med':
        multiply[0]=buyingProb[2]
        multiply1[0]=buyingProb1[2]
    if testsample[i][0]=='low':
        multiply[0]=buyingProb[3]
        multiply1[0]=buyingProb1[3]
    if testsample[i][1]=='vhigh':
        multiply[1]=maintProb[0]
        multiply1[1]=maintProb1[0]
    if testsample[i][1]=='high':
        multiply[1]=maintProb[1]
        multiply1[1]=maintProb1[1]
    if testsample[i][1]=='med':
        multiply[1]=maintProb[2]
        multiply1[1]=maintProb1[2]
    if testsample[i][1]=='low':
        multiply[1]=maintProb[3]
        multiply1[1]=maintProb1[3]
    if testsample[i][2]=='2':
        multiply[2]=doorsProb[0]
```

```
            multiply1[2]=doorsProb1[0]
        if testsample[i][2]=='3':
            multiply[2]=doorsProb[1]
            multiply1[2]=doorsProb1[1]
        if testsample[i][2]=='4':
            multiply[2]=doorsProb[2]
            multiply1[2]=doorsProb1[2]
        if testsample[i][2]=='5more':
            multiply[2]=doorsProb[3]
            multiply1[2]=doorsProb1[3]
        if testsample[i][3]=='2':
            multiply[3]=personProb[0]
            multiply1[3]=personProb1[0]
        if testsample[i][3]=='4':
            multiply[3]=personProb[1]
            multiply1[3]=personProb1[1]
        if testsample[i][3]=='more':
            multiply[3]=personProb[2]
            multiply1[3]=personProb1[2]
        if testsample[i][4]=='small':
            multiply[4]=bootProb[0]
            multiply1[4]=bootProb1[0]
        if testsample[i][4]=='med':
            multiply[4]=bootProb[1]
            multiply1[4]=bootProb1[1]
        if testsample[i][4]=='big':
            multiply[4]=bootProb[2]
            multiply1[4]=bootProb1[2]
        if testsample[i][5]=='low':
            multiply[5]=bootProb[0]
            multiply1[5]=bootProb1[0]
        if testsample[i][5]=='med':
            multiply[5]=bootProb[1]
            multiply1[5]=bootProb1[1]
        if testsample[i][5]=='high':
            multiply[5]=bootProb[2]
            multiply1[5]=bootProb1[2]
        prob=multiply[0]*multiply[1]*multiply[2]*multiply[3]*multiply[4]*multiply[5]*x/1062
        prob1=multiply1[0]*multiply1[1]*multiply1[2]*multiply1[3]*multiply1[4]*multiply1[5]*y/1062
#       print(prob)
#       print(prob1)
        output='unacc\n'
        if prob1>prob:
            output='acc\n'
        if testsample[i][6]==output:
            if output=='acc\n':
                classifier[0]+=1    ######## True Positive #########
            if output=='unacc\n':
                classifier[1]+=1    ######## True Negative #########
        if testsample[i][6]!=output:
            if output=='acc\n':
                classifier[2]+=1    ######## False Positive #########
            if output=='unacc\n':
                classifier[3]+=1    ######## False Negative  #########
```

```
print"——————True Positive= %d————————False Negative= %d———"%(classifier[0],classifier[3])
print"——————False Positive= %d————————True Negative= %d———"%(classifier[2],classifier[1])
```