NAME: SRINIVASO NALLA
SUBJECT: PATTERN RECOGNITION
ASSIGNMENT: 1
INSTRUCTOR: DR. HARKEERAT KAUR

Q.1. prepare a Formula Sheet for —

(a). Notations & Symbols for prior probability, class-conditional probability density, evidence, posterior probability and Baye's Formula.

Ans:) Baye's Formula: $P(\omega_j|x) = \frac{P(x|\omega_j)P(\omega_j)}{P(x)}$

$$\Rightarrow \text{posterior} = \frac{\text{Likelihood} \times \text{prior}}{\text{Evidence}}$$

$P(\omega_j)$ is called prior probability

$P(x|\omega_j)$ is called Likelihood or Conditional probability density

$P(x)$ is called Evidence

$P(\omega_j|x)$ is called Posterior probability

—x—

(b). what do you understand by the likelihood $p(x|\omega_j)$?

Ans: Likelihood is also called as conditional probability. It says accuracy. If likelihood is higher then we can say accuracy is more.

—x—

(c). what is P(error)?

Ans: Average probability of error is defined as

$$P(\text{error}) = \int_{-\infty}^{\infty} P(\text{error}, x)\, dx = \int_{-\infty}^{\infty} P(\text{error}|x)\, P(x) dx$$

for every $x$,

we ensure that $P(\text{error}|x)$ is as small as possible then the integral must be small.

Here is another analysis,

$$P(\text{error}|x) = \begin{cases} P(\omega_1|x) & \text{if we decide } \omega_2 \\ P(\omega_2|x) & \text{if we decide } \omega_1 \end{cases}$$

where $\omega_1, \omega_2$ are state of nature, and these are variables to represent.

—x—

(page-1)

(d). What is the formula for Baye's decision rule for minimizing probability of error?

ANS:

$$P(error) = \int_{-\infty}^{\infty} P(error, x)\, dx = \int_{-\infty}^{\infty} P(error|x)\, P(x)\, dx$$

for every $x$.

We insure that $P(error|x)$ as small as possible, so integral must be small and thus $P(error)$ is small.

We have justified the following Baye's decision rule for minimizing the probability of error:

~~Decide $\omega_1$ if $P(\omega_1|x)$ >~~

Decide $\omega_1$ if $P(\omega_1|x) > P(\omega_2|x)$; otherwise decide $\omega_2$

and under this rule, it is

$$P(error|x) = \min\{P(\omega_1|x), P(\omega_2|x)\}$$

— x —

(e). What is conditional risk? How it is defined in notations?

ANS: In decision-theoretic terminology, ~~an~~ an unexpected loss is called a Risk. $R(\alpha_i|x)$ is called the Conditional Risk.

The overall risk is given by,

$$R = \int R(\alpha|x)\, P(x)\, dx$$

Clearly, if $\alpha (= \alpha(x))$ is chosen so that $R(\alpha_i|x)$ is as small as possible for every $x$, then the overall Risk can be minimized.

To minimize the overall Risk, compute the conditional Risk as:

$$R(\alpha_i|x) = \sum_{j=1}^{c} \lambda(\alpha_i|\omega_j)\, P(\omega_j|x) \quad \text{for } i = 1, 2, \ldots, \alpha \text{ and}$$

select the value of $\lambda$, for which $R(\alpha_i|x)$ is minimum.

The resulting minimum overall risk is called Baye's Risk.

— x —

(f). What is the formula for Baye's minimum Risk?

ANS:

To minimize overall risk, use

$$R(\alpha_i|x) = \sum_{j=1}^{c} \lambda(\alpha_i|\omega_j)\, P(\omega_j|x) \quad \text{for } i = 1, 2, \ldots, \alpha$$

and $R(\alpha_i|x)$ should be as minimum as possible.

The resulting minimum overall risk is called Bayes Risk

— x —

(page-2)

(g). How do you write Baye's Risk Rule for two-category classification problem over continuous feature vector?

Ans: Let us consider case when applied for two-category classification problem.

From conditional risk,

$$R(\alpha_i|x) = \sum_{j=1}^{c} \lambda(\alpha_i|\omega_j)\, P(\omega_j|x)$$

we derive,

$$R(\alpha_1|x) = \lambda_{11} P(\omega_1|x) + \lambda_{12} P(\omega_2|x)$$

and,

$$R(\alpha_2|x) = \lambda_{21} P(\omega_1|x) + \lambda_{22} P(\omega_2|x)$$

Here, considered $\lambda_{ij} = \lambda(\alpha_i|\omega_j)$.

The fundamental rule is to decide $\omega_1$ if $R(\alpha_1|x) < R(\alpha_2|x)$.

In terms of the posterior probability, we decide $\omega_1$ if

$$(\lambda_{21} - \lambda_{11})\, P(\omega_1|x) > (\lambda_{12} - \lambda_{22})\, P(\omega_2|x)$$

and $\omega_2$, otherwise

Another alternative, under reasonable assumption that $\lambda_{21} > \lambda_{11}$, is to decide $\omega_1$ if

$$\frac{P(x|\omega_1)}{P(x|\omega_2)} > \frac{\lambda_{12} - \lambda_{22}}{\lambda_{21} - \lambda_{11}}\, \frac{P(\omega_2)}{P(\omega_1)}$$

This form of the decision rule focuses on the x-dependence of the probability densities.

Thus, the Baye's decision rule can be interpreted as calling for deciding $\omega_1$ if the likelihood ratio exceeds a threshold value, that is independent of the observation, x.

— x —

(h). Derive formula for minimum error rate classifier?

Ans: The ~~loss~~ (loss) function of interest for this case is hence, so called 'symmetrical' or 'zero-one' loss function.

$$\lambda(\alpha_i|\omega_j) = \begin{cases} 0 & i = j \\ 1 & i \neq j \end{cases} \quad \text{where } i, j = 1, 2, \ldots, c$$

The risk corresponding to this loss function is precisely average probability of error, since the conditional risk is

$$R(\alpha_i|x) = \sum_{j=1}^{c} \lambda(\alpha_i|\omega_j)\, P(\omega_j|x)$$

$$= \sum_{j \neq i} P(\omega_j|x) = 1 - P(\omega_i|x) \qquad \text{As } P(\omega_i|x) + P(\omega_j|x) = 1$$

To minimize the average probability of error, we should select 'i' that maximizes the posterior probability, $P(\omega_i|x)$.

In other words, for minimum error rate:

Decide $\omega_i$ if $P(\omega_i|x) > P(\omega_j|x)$ for all $i \neq j$.

— x —

(Page - 3)

(i). What is the formula for univariate NDF and multivariate NDF?

ANS:

① For univariate NDF,

$$P(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left\{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2\right\}$$

where $\mu$ is mean, $\sigma^2$ is variance, $\sigma$ is standard deviation

② For multivariate NDF,

$$P(x) = \frac{1}{(2\pi)^{d/2}(|\Sigma|)^{1/2}} \exp\left\{-\frac{1}{2}(x-\mu)^t \Sigma^{-1}(x-\mu)\right\}$$

where $\mu$ is a d-dimensional mean vector,

$\Sigma$ is d-by-d covariance matrix,

$|\Sigma|$ and $\Sigma^{-1}$ are determinant & inverse respectively.

—⁎—

(j). Write the discriminate functions for multivariate NDF for all three cases?

ANS: ① Case-1: $\Sigma_i = \sigma^2 I$

$$g_i(x) = \frac{-\|x-\mu_i\|^2}{2\sigma^2} + \ln P(\omega_i)$$

where $\|\cdot\|$ is Euclidean norm, and, $\|x-\mu_i\|^2 = (x-\mu_i)^t(x-\mu_i)$

Also, $g_i(x) = \omega_i^t x + \omega_{i0}$

where, $\omega_i = \frac{1}{\sigma^2} \cdot \mu_i$

and $\omega_{i0} = -\frac{1}{2\sigma^2} \mu_i^t \mu_i + \ln P(\omega_i)$

② Case-2: $\Sigma_i = \Sigma$

$$g_i(x) = \omega_i^t x + \omega_{i0}$$

where, $\omega_i = \Sigma^{-1}\mu_i$

and $\omega_{i0} = -\frac{1}{2}\mu_i^t \Sigma^{-1}\mu_i + \ln P(\omega_i)$

③ Case-3: $\Sigma_i$ = arbitrary

$$g_i(x) = x^t W_i x + \omega_i^t x + \omega_{i0}$$

where, $W_i = -\frac{1}{2}\Sigma_i^{-1}$, $\omega_i = \Sigma_i^{-1}\mu_i$

and $\omega_{i0} = -\frac{1}{2}\mu_i^t \Sigma_i^{-1}\mu_i + -\frac{1}{2}\ln|\Sigma_i| + \ln P(\omega_i)$

—⁎—

For questions 2, can see Python programs.

—⁎—

Q). (c).: Out of the three cases, which one can apply for multivariate NDF?

ANS: ① If the covariance matrices of all three classes are equal and diagonal (for $\Sigma_i = \sigma^2 I$) then the discriminant function is linear.

② If the covariance matrices of all three classes are equal but not diagonal (for $\Sigma_i = \Sigma$) then the discriminant function is quadratic

③ If the covariance matrices of all three classes are arbitrary and different from each other then the discriminant function is quadratic

⋆ In our case, it belongs to 3rd case, i.e., $\Sigma_i$ = arbitrary ⋆

— END of Notes —
(Page-4)

# assignment-1

July 5, 2023

[19]:
```python
#Q.2 (a). : Compute Mean vector  over all samples, and class means 1,  2 and  3.
######TO FIND THE MEAN FOR TABULAR DATA FOR x1 & x2 IS:
# Python code to demonstrate the working of mean()

# importing statistics to handle statistical operations
import statistics

# initializing list
x1w1 = [2.1,1.1,1.4,3.3]
x1w2 = [4.4,3.4,4.5,4.1]
x1w3 = [-1.3,-3.2,-3.2,-2.1]
x2w1 = [-2.5,-3.1,-2.1,-1.8]
x2w2 = [6.5,5.8,7.2,5.65]
x2w3 = [-2.3,-4.5,-4.5,-3.3]
#Find Mean for x1 & x2 classes
x1 = [2.1,1.1,1.4,3.3,4.4,3.4,4.5,4.1,-1.3,-3.2,-3.2,-2.1]
x2 = [-2.5,-3.1,-2.1,-1.8,6.5,5.8,7.2,5.65,-2.3,-4.5,-4.5,-3.3]

# using mean() to calculate average of list of elements
print ("The average of list values of x1 for w1 is : ",end="")
print (statistics.mean(x1w1))
print ("The average of list values of x1 for w2 is : ",end="")
print (statistics.mean(x1w2))
print ("The average of list values of x1 for w3 is : ",end="")
print (statistics.mean(x1w3))
print ("The average of list values of x2 for w1 is : ",end="")
print (statistics.mean(x2w1))
print ("The average of list values of x2 for w2 is : ",end="")
print (statistics.mean(x2w2))
print ("The average of list values of x2 for w3 is : ",end="")
print (statistics.mean(x2w3))
print ("The average of list values of x1 class is : ",end="")
print (statistics.mean(x1))
print ("The average of list values of x2 class is : ",end="")
print (statistics.mean(x2))
```

```
The average of list values of x1 for w1 is : 1.975
The average of list values of x1 for w2 is : 4.1
```

```
The average of list values of x1 for w3 is : -2.45
The average of list values of x2 for w1 is : -2.375
The average of list values of x2 for w2 is : 6.2875
The average of list values of x2 for w3 is : -3.65
The average of list values of x1 class is : 1.2083333333333333
The average of list values of x2 class is : 0.08750000000000004
```

[15]:
```python
#Q,2.(b). Use numpy.cov() to compute covariance matrix Σ for each class Σ1, Σ2,␣
↪Σ3.
####Covariance of each class for 3 ws: E1 E2 E3
# Python code to demonstrate the use of numpy.cov
import numpy as np
x1w1 = [2.1,1.1,1.4,3.3]
x1w2 = [4.4,3.4,4.5,4.1]
x1w3 = [-1.3,-3.2,-3.2,-2.1]
x2w1 = [-2.5,-3.1,-2.1,-1.8]
x2w2 = [6.5,5.8,7.2,5.65]
x2w3 = [-2.3,-4.5,-4.5,-3.3]
x1 = np.array([ [2.1,1.1,1.4,3.3],    [4.4,3.4,4.5,4.1],    [-1.3,-3.2,-3.2,-2.
↪1]])
x2 = np.array([ [-2.5,-3.1,-2.1,-1.8],   [6.5,5.8,7.2,5.65],   [-2.3,-4.5,-4.
↪5,-3.3]])

print("Covariance matrix of x1w1 is :\n", np.cov(x1w1))
print("Covariance matrix of x1w2 is :\n", np.cov(x1w2))
print("Covariance matrix of x1w3 is :\n", np.cov(x1w3))
print("Covariance matrix of x2w1 is :\n", np.cov(x2w1))
print("Covariance matrix of x2w2 is :\n", np.cov(x2w2))
print("Covariance matrix of x2w3 is :\n", np.cov(x2w3))
print("Shape of array of x1 is :\n", np.shape(x1))
print("Shape of array of x2 is :\n", np.shape(x2))
print("Covariance matrix of x1 is :\n", np.cov(x1))
print("Covariance matrix of x2 is :\n", np.cov(x2))
```

```
Covariance matrix of x1w1 is :
 0.9558333333333332
Covariance matrix of x1w2 is :
 0.24666666666666676
Covariance matrix of x1w3 is :
 0.8566666666666667
Covariance matrix of x2w1 is :
 0.3158333333333333
Covariance matrix of x2w2 is :
 0.5072916666666667
Covariance matrix of x2w3 is :
 1.1300000000000001
Shape of array of x1 is :
```

```
 (3, 4)
Shape of array of x2 is :
 (3, 4)
Covariance matrix of x1 is :
 [[0.95583333 0.14       0.565     ]
 [0.14       0.24666667 0.19      ]
 [0.565      0.19       0.85666667]]
Covariance matrix of x2 is :
 [[ 0.31583333  0.07041667  0.13833333]
 [ 0.07041667  0.50729167 -0.09916667]
 [ 0.13833333 -0.09916667  1.13      ]]
```

[20]:
```
#Q.2. (c). : Out of the three cases which case applies for computing␣
 ↪discriminant function of
#             Multivariate NDF?
#To differentiate the three cases of discriminant functions, you can look at␣
 ↪the covariance matrices of the three classes. The three cases are:
#---(1): If the covariance matrices of all three classes are equal and diagonal␣
 ↪(i.e., E = o^2I),
# ---    then the discriminant function is linear.
#---(2): If the covariance matrices of all three classes are equal but not␣
 ↪diagonal (i.e., E = E),
# ---    then the discriminant function is linear.
#---(3): If the covariance matrices of all three classes are arbitrary and␣
 ↪different from each other (i.e., E is arbitrary),
# ---    then the discriminant function is quadratic.
###############From this, says it is belongs to␣
 ↪case-3######################################
```

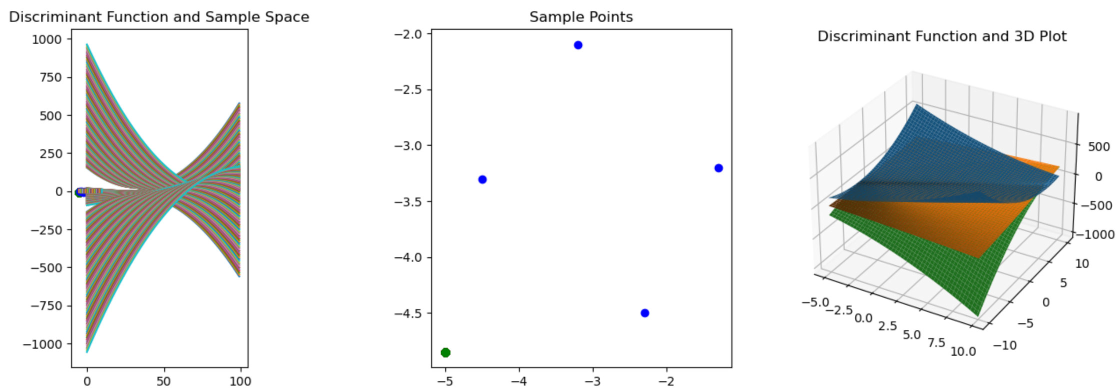
[4]:
```
#Q.2. (d). : Let p( 1)= 0.4, p( 2)=0.35, p( 3)=0.25 .Can you write a python␣
 ↪function for computing the
#             discriminant functions defined in part c?
#Q.2. (e). : Compute and plot discriminant functions g1(X), g2 (X), g3(X) and␣
 ↪the sample points in 2
#             dimensions.
# NOTE:       The discriminant functions are probability density functions that␣
 ↪describe
############ the likelihood of a sample belonging to each class.
############ They can be quite complex and difficult to interpret, especially␣
 ↪for high-dimensional data.
############ If you are having trouble understanding the discriminant␣
 ↪functions, you can try visualizing the decision boundaries.
############ between the classes instead.

######################NOTE#####################
#The formula to calculate the mean is the sum of all the data points divided by␣
 ↪the number of data points.
```

```python
#The Q.2(a) code is using the formula statistics.mean(x1w1) where x1w1 is the
 ↪list of data points for x1 class of w1.
#This code is using the formula mean1 = np.mean(X1, axis=0) where X1 is the
 ↪data points for w1.
#Both codes are correct and here the results are different as uses different
 ↪formula in both the cases.
#################################################

#Defining libraries/modules in python to plot & use statistics & probability
import numpy as np
from scipy.stats import multivariate_normal
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

# Defining the data in matrix form
X1 = np.array([[2.1,1.1],[1.4,3.3],[-2.5,-3.1],[-2.1,-1.8]]) #X1 is same as W1
X2 = np.array([[4.4,3.4],[4.5,4.1],[6.5, 5.8],[7.2, 5.65]]) #X2 same as W2
X3 = np.array([[-1.3,-3.2],[-3.2,-2.1],[-2.3,-4.5],[-4.5, -3.3]]) #X3 same as W3

# Computing the mean vectors for each class
mean1 = np.mean(X1, axis=0)
mean2 = np.mean(X2, axis=0)
mean3 = np.mean(X3, axis=0)
print("The Mean of w1,w2,w3 are:" , mean1,mean2,mean3)

# Computing the covariance matrices for each class
cov1 = np.cov(X1.T)
cov2 = np.cov(X2.T)
cov3 = np.cov(X3.T)
print("The variance of w1,w2,w3 are:" , (cov1,cov2,cov3))

# Defining the prior probabilities for each class
p1 = 0.4
p2 = 0.35
p3 = 0.25

# Defining the discriminant functions for each class
def discriminant(x):
    g1 = multivariate_normal(mean=mean1, cov=cov1).logpdf(x) + np.log(p1)
    g2 = multivariate_normal(mean=mean2, cov=cov2).logpdf(x) + np.log(p2)
    g3 = multivariate_normal(mean=mean3, cov=cov3).logpdf(x) + np.log(p3)
    return g1, g2, g3

# Computing and plotting the discriminant functions and sample points in x1 &
 ↪x2.
x1 = np.linspace(-5, 10, 100)
y1 = np.linspace(-10, 10, 100)
```

```
X1,Y1 = np.meshgrid(x1,y1)
x2 = np.linspace(-5, 10, 100)
y2 = np.linspace(-10, 10, 100)
X2,Y2 = np.meshgrid(x2,y2)

#Define Z1,Z2,Z3 before loop starts
Z1,Z2,Z3 = np.zeros((100,100)),np.zeros((100,100)),np.zeros((100,100))

#For loop to read discriminate function for g1,g2,g3
for i in range(100):
    for j in range(100):
        x = [X2[i,j], Y2[i,j]]
        g1,g2,g3 = discriminant(x)
        Z1[i,j] = g1
        Z2[i,j] = g2
        Z3[i,j] = g3

#This will print an array of discriminant function values for each sample in␣
 ↪the input data.
g_values = discriminant(x)
print("Prints the discriminant functions for case-3 for g1,g2,g3 are:",g_values)

#Plotting commands
fig=plt.figure(figsize=(15,5))

#Uncomment this sample Points and comment the below one , else if you keep same␣
 ↪for both Discrete function & Sample space.
ax=fig.add_subplot(151)
ax.scatter(X1[:,0], X1[:,1], color='r')
ax.scatter(X2[:,0], X2[:,1], color='g')
ax.scatter(X3[:,0], X3[:,1], color='b')
ax.plot(X2,Y2,Z1-Z2)
ax.plot(X2,Y2,Z1-Z3)
ax.plot(X2,Y2,Z2-Z3)
ax.set_title('Discriminant Function and Sample Space')

'''
ax=fig.add_subplot(131)
ax.contour(X2,Y2,Z1-Z2,[0])
ax.contour(X2,Y2,Z1-Z3,[0])
ax.contour(X2,Y2,Z2-Z3,[0])
ax.scatter(X1[:,0], X1[:,1], color='r')
ax.scatter(X2[:,0], X2[:,1], color='g')
ax.scatter(X3[:,0], X3[:,1], color='b')
ax.set_title('Discriminant Functions')
'''
```

```
ax=fig.add_subplot(132)
ax.scatter(X1[:,0], X1[:,1], color='r')
ax.scatter(X2[:,0], X2[:,1], color='g')
ax.scatter(X3[:,0], X3[:,1], color='b')
ax.set_title('Sample Points')

ax=fig.add_subplot(133, projection='3d')
ax.plot_surface(X2,Y2,Z1-Z2)
ax.plot_surface(X2,Y2,Z1-Z3)
ax.plot_surface(X2,Y2,Z2-Z3)
ax.set_title('Discriminant Function and 3D Plot')

#Below function to plot the graph
plt.show()
```

```
The Mean of w1,w2,w3 are: [-0.275 -0.125] [5.65   4.7375] [-2.825 -3.275]
The variance of w1,w2,w3 are: (array([[5.57583333, 6.1075    ],
       [6.1075    , 8.29583333]]), array([[2.00333333, 1.57416667],
       [1.57416667, 1.385625  ]]), array([[ 1.84916667, -0.30916667],
       [-0.30916667,  0.9625    ]]))
Prints the discriminant functions for case-3 for g1,g2,g3 are:
(-13.714869512800131, -18.4442718011498, -178.4765015182157)
```




[ ]:

[ ]: