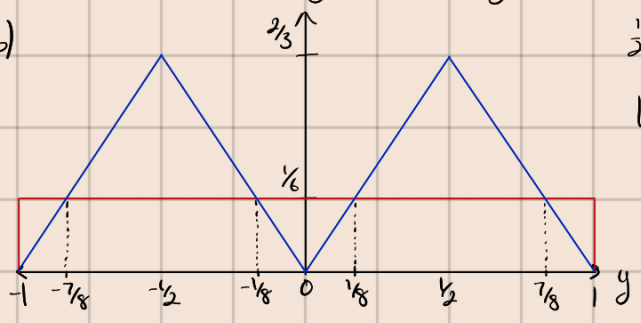
Jilin Zheng // 4492587796

EE381 Homework #8

Problem 8.1

(a) $$D^{ML}(y)=\begin{cases}1, & -1\le y\le \frac{-3}{4} \text{ or } \frac{-1}{4}\le y\le\frac{1}{4} \text{ or } \frac{3}{4}\le y\le 1\\ 0, & \frac{-3}{4}<y<\frac{-1}{4} \text{ or } \frac{1}{4}<y<\frac{3}{4}\end{cases}$$

(b)



$\frac{1}{2}\cdot\frac{1}{3}=\frac{1}{6}$ $\quad 1\cdot\frac{2}{3}=\frac{2}{3}$

$f(y)=my+b$ $\quad -\frac{2}{3}\cdot\frac{2}{1}=-\frac{4}{3}$

$\frac{1}{6}=-\frac{4}{3}y+\frac{4}{3}$

$y=\left(\frac{1}{6}-\frac{8}{6}\right)\cdot\frac{-3}{4}=\frac{-7}{6_2}\cdot\frac{-3^1}{4}=\frac{7}{8}$

$\frac{1}{6}=\frac{4}{3}y+0$

$y=\frac{1}{6}\cdot\frac{3}{4}=\frac{1}{8}$

$$D^{MAP}(y)=\begin{cases}1, & -1\le y\le -7/8 \text{ or } -1/8\le y\le \frac{1}{8} \text{ or } 7/8\le y\le 1\\ 0, & -7/8<y<-1/8 \text{ or } 1/8<y<7/8\end{cases}$$

(c) $P_{FA}=P[Y\in A_1|H_0]$

$=\frac{1}{4}\left(\frac{1}{2}\right)\left(\frac{1}{2}\right)\cdot 4$

$=\frac{1}{4}$

$P_{MD}=P[Y\in A_0|H_1]$

$=\left[\frac{1}{2}\left(\frac{1}{2}\right)\right]2$

$=\frac{1}{2}$

$P_e=P_{FA}P[H_0]+P_{MD}P[H_1]=\frac{1}{4}\left(\frac{2}{3}\right)+\frac{1}{2}\left(\frac{1}{3}\right)=\frac{1}{6}+\frac{1}{6}=\frac{1}{3}$

(d) $P_{FA}=P[Y\in A_1|H_0]$

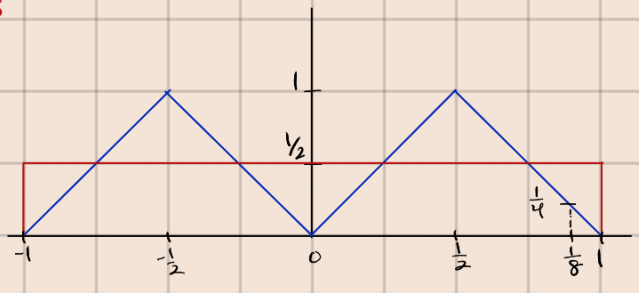
$=\frac{1}{8}\left(\frac{1}{4}\right)\left(\frac{1}{2}\right)\cdot 4$

$=\frac{1}{16}$

$P_{MD}=P[Y\in A_0|H_1]$

$=\frac{6}{8}\left(\frac{1}{2}\right)\cdot 2$

$=\frac{3}{4}$



$P_e=P_{FA}P[H_0]+P_{MD}P[H_1]$

$=\frac{1}{16}\cdot\frac{2}{3}+\frac{3}{4}\cdot\frac{1}{3}=\frac{7}{24}$

Problem 8.2

(a) $D^{ML} = \begin{cases} 1, & 0 \le y < \infty \\ 0, & -\infty < y < 0 \end{cases}$ $\quad D^{MAP} = \begin{cases} 1, & -0.347 \le y < \infty \\ 0, & -\infty < y < -0.347 \end{cases}$

(b) Exponential: $f_X(x) = \begin{cases} \lambda e^{-\lambda x}, & x \ge 0 \\ 0, & x < 0 \end{cases}$

Likelihood Ratio $= L(y) = \frac{P_{Y|H_1}(y)}{P_{Y|H_0}(y)} = \frac{2e^{-2y}}{1e^{-y}} = \frac{e^{y}}{2e^{2y}} = \frac{1}{2e^{y}}$

$D^{ML} = \begin{cases} 1, & L(y) \ge 1 \\ 0, & L(y) < 1 \end{cases} = \begin{cases} 1, & \frac{1}{2e^{y}} \ge 1 \\ 0, & \frac{1}{2e^{y}} < 1 \end{cases}$

$P_{FA} = P[Y \in A_1 | H_0] = \int_{0.693}^{\infty} 2e^{-2y}\,dy = \frac{1}{4}$

$P_{MD} = P[Y \in A_0 | H_1] = \int_{0}^{0.693} e^{-y}\,dy = \frac{1}{2}$

$P_e = P_{FA}\left(\frac{1}{2}\right) + P_{MD}\left(\frac{1}{2}\right) = \frac{1}{4}\left(\frac{1}{2}\right) + \frac{1}{2}\left(\frac{1}{2}\right) = \frac{1}{8} + \frac{2}{8} = \frac{3}{8}$

(c) Likelihood Ratio $= L(y) = \frac{\binom{3}{y}\left(\frac{1}{2}\right)^{y}\left(\frac{1}{2}\right)^{3-y}}{\binom{4}{y}\left(\frac{1}{2}\right)^{y}\left(\frac{1}{2}\right)^{4-y}} = \frac{\binom{3}{y}\left(\frac{1}{2}\right)^{3}}{\binom{4}{y}\left(\frac{1}{2}\right)^{4}} = \frac{\frac{1}{8}\binom{3}{y}}{\frac{1}{16}\binom{4}{y}} = 2\frac{\binom{3}{y}}{\binom{4}{y}}$ $\quad \frac{P[H_0]}{P[H_1]} = \frac{\frac{2}{3}}{\frac{1}{3}} = 2$

$D^{ML} = \begin{cases} 1, & 2\frac{\binom{3}{y}}{\binom{4}{y}} \ge 1 \\ 0, & 2\frac{\binom{3}{y}}{\binom{4}{y}} < 1 \end{cases}$ $\quad D^{MAP} = \begin{cases} 1, & 2\frac{\binom{3}{y}}{\binom{4}{y}} \ge 2 \\ 0, & 2\frac{\binom{3}{y}}{\binom{4}{y}} < 2 \end{cases}$

$P_e = P_{FA}\left(\frac{2}{3}\right) + P_{MD}\left(\frac{1}{3}\right) = 0.3125$ for ML $= \frac{5}{16}$

$P_{FA} = P[Y \in A_1 | H_0]$

$= 0.125$

$P_{MD} = 0.0625 + 0.25 + 0.375$

$= 0.6875$

$P_e = P_{FA}\left(\frac{2}{3}\right) + P_{MD}\left(\frac{1}{3}\right)$

$P_{FA} = 0.042 + 0.125 \cdot 2$

$= \frac{7}{24}$

$P_{MD} = 0.042 = \frac{1}{24}$

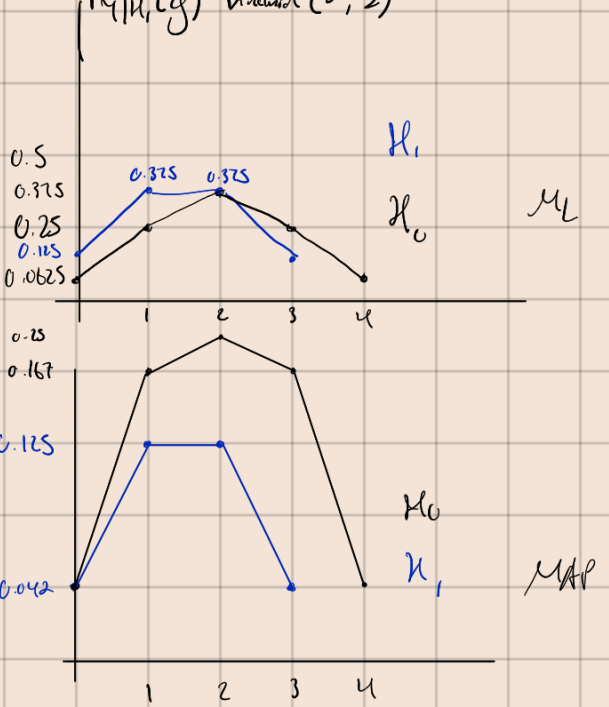
$P_e = \frac{7}{24}\left(\frac{2}{3}\right) + \frac{1}{24}\left(\frac{1}{3}\right)$

$= \frac{7}{36} + \frac{1}{72} = \frac{15}{72} = \frac{5}{24}$ for MAP

$P_X(x) = \binom{n}{x} p^{x}(1-p)^{n-x}, \quad x = 0 \ldots n$

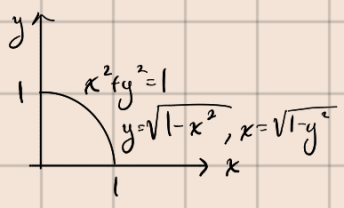
$P_{Y|H_0}(y) = \text{Binomial}\left(4, \frac{1}{2}\right) = \binom{4}{y}\left(\frac{1}{2}\right)^{y}\left(\frac{1}{2}\right)^{4-y} = \binom{4}{y}\frac{1}{16}$

$P_{Y|H_1}(y) = \text{Binomial}\left(3, \frac{1}{2}\right)$



Problem 8.3

$$f_{X,Y}(x,y)=\begin{cases}\frac{4}{\pi}, & x\ge 0, y\ge 0, x^2+y^2\le 1\\ 0, & \text{otherwise}\end{cases}$$



(a) $\hat{X}_{MMSE}(y)=E[X|Y=y]=\int_{-\infty}^{\infty} x f_{X|Y}(x|y)\,dx$

$f_Y(y)=\int_0^{\sqrt{1-y^2}}\frac{4}{\pi}dx=\frac{4}{\pi}\sqrt{1-y^2}$, $0\le y\le 1$

$f_{X|Y}(x|y)=\begin{cases}\frac{1}{\sqrt{1-y^2}}, & 0\le x\le\sqrt{1-y^2}, 0\le y\le 1\\ 0, & \text{otherwise}\end{cases}$

$=\int_0^{\sqrt{1-y^2}} x\,\frac{\frac{4}{\pi}}{\frac{4}{\pi}\sqrt{1-y^2}}\,dx=\frac{1}{\sqrt{1-y^2}}\left[\frac{x^2}{2}\Big|_0^{\sqrt{1-y^2}}\right]=\frac{1}{2\sqrt{1-y^2}}(1-y^2)=\frac{\sqrt{1-y^2}}{2}$

(b) $E\left[(X-\hat{X}_{MMSE}(Y))^2\right]=E[X^2]-E\left[\hat{X}^2_{MMSE}(Y)\right]$

$=\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}(x-\hat{X}_{MMSE}(y))^2 f_{X,Y}(x,y)\,dx\,dy$

$=\int_0^1\int_0^{\sqrt{1-y^2}}\left(x-\frac{\sqrt{1-y^2}}{2}\right)^2\frac{4}{\pi}\,dx\,dy=\frac{1}{16}$

(c) $\hat{X}_{LLSE}(y)=E[X|Y]+\frac{Cov[X,Y]}{Var[Y]}(y-E[Y])$

$f_X(x)=\int_0^{\sqrt{1-x^2}}\frac{4}{\pi}dy=\frac{4}{\pi}\sqrt{1-x^2}$, $0\le x\le 1$

$f_Y(y)=\int_0^{\sqrt{1-y^2}}\frac{4}{\pi}dx=\frac{4}{\pi}\sqrt{1-y^2}$, $0\le y\le 1$

$E[X]=\frac{4}{3\pi}$

$E[Y]=\int_0^1 y\left(\frac{4}{\pi}\sqrt{1-y^2}\right)dy=\frac{4}{3\pi}$

$E[XY]=\int_0^1\int_0^{\sqrt{1-y^2}} xy\left(\frac{4}{\pi}\right)dx\,dy=\frac{4}{3\pi}\cdot y$

$E[Y^2]=\int_0^1 y^2\left(\frac{4}{\pi}\sqrt{1-y^2}\right)dy=\frac{1}{4}$

$Cov[X,Y]=E[XY]-E[X]E[Y]=y\cdot\frac{4}{3\pi}-\left(\frac{4}{3\pi}\right)^2$

$Var[Y]=E[Y^2]-(E[Y])^2=\frac{1}{4}-\left(\frac{4}{3\pi}\right)^2=\frac{1}{4}-\frac{16}{9\pi^2}\approx 0.0699$

$=\frac{4}{3\pi}+\frac{y\cdot\frac{4}{3\pi}-\frac{16}{9\pi^2}}{\frac{1}{4}-\frac{16}{9\pi^2}}\left(y-\frac{4}{3\pi}\right)$

(d) $MSE_{LLSE}=Var[X]-\frac{(Cov[X,Y])^2}{Var[Y]}$

$E[X^2]=E[Y^2]=\frac{1}{4}$, $Var[X]=Var[Y]=\frac{1}{4}-\frac{16}{9\pi^2}$

$=\frac{1}{4}-\frac{16}{9\pi^2}-\frac{\left(y\cdot\frac{4}{3\pi}-\left(\frac{4}{3\pi}\right)\right)^2}{\left(\frac{1}{4}-\frac{16}{9\pi^2}\right)}$

Problem 8.4

(a) $\hat{X}_{MMSE}(y) = \mathbb{E}[X|Y=y]$

$= \mu_X + \frac{Cov[X,Y]}{Var[Y]}(y-\mu_Y) = 0 + \frac{1}{2}(y-0) = \frac{1}{2}y$

(b) $\hat{Z}_{1,LSE}(u) = \mathbb{E}[Z_1] + \frac{Cov[U,Z]}{Var[Z]}(u - \mathbb{E}[U]) = \frac{1}{5}\left(\frac{21}{10}\right)u = \frac{21}{50}u$

$\mathbb{E}[Z_1] = \mathbb{E}[U^3+V] = \mathbb{E}[U^3] + \mathbb{E}[V] = 0$

$\mathbb{E}[U^3] = \int_{-1}^{1} u^3 \frac{1}{2}\,du = \frac{1}{2}\left[\frac{u^4}{4}\Big|_{-1}^{1}\right] = \frac{1}{8}(1-1) = 0$

$\mathbb{E}[Z_1^2] = \mathbb{E}[U^6 + 2U^3V + V^2] = \frac{1}{7} + \frac{1}{3} = \frac{3}{21} + \frac{7}{21} = \frac{10}{21}$

$\mathbb{E}[U^6] = \int_{-1}^{1} u^6 \frac{1}{2}\,du = \frac{1}{2}\left[\frac{u^7}{7}\Big|_{-1}^{1}\right] = \frac{1}{14}(1-(-1)) = \frac{1}{7}$

$2\mathbb{E}[U^3V] = \int_{-1}^{1}\int_{-1}^{1} u^3 v \frac{1}{2}\cdot\frac{1}{2}\,du\,dv = 0$

$\int_{-1}^{1} u^3 v \frac{1}{4}\,du = \frac{1}{4}v\left[\frac{u^4}{4}\Big|_{-1}^{1}\right] = \frac{1}{16}v(1-1) = 0$

$\mathbb{E}[V^2] = \int_{-1}^{1} v^2 \frac{1}{2}\,dv = \frac{1}{2}\left[\frac{v^3}{3}\Big|_{-1}^{1}\right] = \frac{1}{6}[1-(-1)] = \frac{1}{3}$

$Var[Z] = \mathbb{E}[Z_1^2] - (\mathbb{E}[Z_1])^2 = \frac{10}{21} - 0^2 = \frac{10}{21}$

$Cov[U,Z] = Cov[U, U^3+V] = \mathbb{E}[U^4+UV] - \mathbb{E}[U]\mathbb{E}[U^3+V]$

$\mathbb{E}[U^4] = \int_{-1}^{1} u^4 \frac{1}{2}\,du = \frac{1}{2}\left[\frac{u^5}{5}\Big|_{-1}^{1}\right] = \frac{1}{10}(2) = \frac{1}{5}$

(c) $MSE_{LSE} = Var[Z] - \frac{(Cov[Z,U])^2}{Var[U]}$

$Var[U] = \mathbb{E}[U^2] - (\mathbb{E}[U])^2 = \frac{1}{3} - 0 = \frac{1}{3}$

$\mathbb{E}[U^2] = \int_{-1}^{1} u^2 \frac{1}{2}\,du = \frac{1}{2}\left[\frac{u^3}{3}\Big|_{-1}^{1}\right] = \frac{1}{6}(2) = \frac{1}{3}$

$= \frac{10}{21} - \frac{\frac{1}{25}}{\frac{1}{3}} = \frac{187}{525} \approx 0.356 = MSE_{MMSE}$

Problem 8.5

(a) 0.068

(b) 0.05

(c) 0.012

(d) 0.385

(e) 0.085

(f) 0.065

(g) 0.015

```
# Import Necessary Modules
import glob
import matplotlib.pyplot as plt
from skimage import io
import numpy as np
import math
import scipy.stats as stats
%matplotlib inline

#This function takes in the column vectors of guesses for H0 and H1
#data and outputs the probability of error, assuming H0 and H1 are
#equally likely.

def proberror(testguess0,testguess1):
    #Calculate sizes.
    n0test = testguess0.shape[0]
    n1test = testguess1.shape[0]
    #The number of false alarms is the total number of ones in
testguess0.
    num_false_alarm =
np.sum(np.not_equal(testguess0,np.zeros((n0test,1))))
    #Divide by total number of guesses to estimate the probability of
false alarm.
    P_FA = num_false_alarm / n0test
    #The number of missed detections is the total number of zeros in
testguess1.
    num_missed_detection =
np.sum(np.not_equal(testguess1,np.ones((n1test,1))))
    #Divide by total number of guesses to estimate the probability of
missed detection.
    P_MD = num_missed_detection / n1test
    Pe = P_FA * 0.5 + P_MD * 0.5
    return Pe

#Read in synthetic data for 8.5(a), 8.5(b), 8.5(c)
dataset0 = np.genfromtxt("syntheticH0.csv", delimiter = ",")
dataset1 = np.genfromtxt("syntheticH1.csv", delimiter = ",")

#Read in pet classificaton data for 8.4(h)
# dataset0,dataset1 = read_cats_dogs()

#Determine number of samples and dimension.
n0,d0 = dataset0.shape
n1,d1 = dataset1.shape
if (d0 == d1):
    d = d0
else:
    raise Exception("dataset0 and dataset1 do not have the same number
of columns.")
```

```
#Split dataset into training and test data.
train0 = dataset0[0:math.floor(n0/2),:]
test0 = dataset0[math.floor(n0/2):n0,:]
train1 = dataset1[0:math.floor(n1/2),:]
test1 = dataset1[math.floor(n1/2):n1,:]
n0train = train0.shape[0]
n1train = train1.shape[0]
n0test = test0.shape[0]
n1test = test1.shape[0]

#Estimate mean vectors and covariance matrices from training data.
mu0 = np.mean(train0, axis=0)
mu1 = np.mean(train1, axis=0)
sigma0 = np.cov(train0, rowvar=False)
sigma1 = np.cov(train1, rowvar=False)

#This function takes in a row vector currentdata as well as two mean
#vectors mu0 and m1. It outputs 1 if currentdata is closer to mu1
#than mu0, and 0 if mu0 is closer. In the case of a tie, it outputs 1.

def closest_average(currentdata,mu0,mu1):
    #Calculate distances.
    distance_mu0 = np.math.sqrt(np.sum((currentdata-mu0)**2))
    distance_mu1 = np.math.sqrt(np.sum((currentdata-mu1)**2))
    #Decide based on smaller distance.
    if (distance_mu0 < distance_mu1):
        guess = 0
    elif (distance_mu0 >= distance_mu1):
        guess = 1
    return guess

#8.5(a) Apply Gaussian ML rule for identity covariance matrix by
plugging into the PDFs.
H0guesses_idcov = np.zeros((n0test,1))
H1guesses_idcov = np.zeros((n1test,1))

for i in range(n0test):
    currentdata = test0[i,:]
    h0 = stats.multivariate_normal.pdf(currentdata, mu0,
np.identity(d0))
    h1 = stats.multivariate_normal.pdf(currentdata, mu1,
np.identity(d0))
    if h1>= h0:
        H0guesses_idcov[i] =  1

for i in range(n1test):
    currentdata = test1[i,:]
    h0 = stats.multivariate_normal.pdf(currentdata, mu0,
np.identity(d1))
    h1 = stats.multivariate_normal.pdf(currentdata, mu1,
```

```
np.identity(d1))
    if h1>= h0:
        H1guesses_idcov[i] = 1

Pe_idcov = proberror(H0guesses_idcov,H1guesses_idcov)
print("Probability of error for identity covariance matrix is " +
str(Pe_idcov) + ".")
```

```
Probability of error for identity covariance matrix is 0.068.
```

```
#8.5 (b) Apply Gaussian ML rule for the same covariance matrix by
plugging into the PDFs.
H0guesses_samecov = np.zeros((n0test,1))
H1guesses_samecov = np.zeros((n1test,1))

sigmas = (1/(n0train+n1train-2))*((n0train-1)*sigma0+(n1train-
1)*sigma1)

for i in range(n0test):
    currentdata = test0[i,:]
    h0 = stats.multivariate_normal.pdf(currentdata, mu0, sigmas)
    h1 = stats.multivariate_normal.pdf(currentdata, mu1, sigmas)
    if h1>=h0:
        H0guesses_samecov[i] = 1

for i in range(n1test):
    currentdata = test1[i,:]
    h0 = stats.multivariate_normal.pdf(currentdata, mu0, sigmas)
    h1 = stats.multivariate_normal.pdf(currentdata, mu1, sigmas)
    if h1>=h0:
        H1guesses_samecov[i] = 1

Pe_samecov = proberror(H0guesses_samecov,H1guesses_samecov)
print("Probability of error for the same covariance matrices is " +
str(Pe_samecov) + ".")
```

```
Probability of error for the same covariance matrices is 0.05.
```

```
#8.5 c Apply Gaussian ML rule for different covariance matrices by
plugging into the PDFs.
H0guesses_diffcov = np.zeros((n0test,1))
H1guesses_diffcov = np.zeros((n1test,1))

for i in range(n0test):
    currentdata = test0[i,:]
    h0 = stats.multivariate_normal.pdf(currentdata, mu0, sigma0)
    h1 = stats.multivariate_normal.pdf(currentdata, mu1, sigma1)
    if h1>=h0:
        H0guesses_samecov[i] = 1

for i in range(n1test):
```

```
    currentdata = test1[i,:]
    h0 = stats.multivariate_normal.pdf(currentdata, mu0, sigma0)
    h1 = stats.multivariate_normal.pdf(currentdata, mu1, sigma1)
    if h1>=h0:
        H1guesses_diffcov[i] = 1

Pe_diffcov = proberror(H0guesses_diffcov,H1guesses_diffcov)
print("Probability of error for different covariance matrices is " +
str(Pe_diffcov) + ".")
```

```
Probability of error for different covariance matrices is 0.012.
```

```
## For problem 8.5(d), load a new data set:

# Read in breast cancer data
dataset0 = np.genfromtxt("benignfull.csv", delimiter = ",")
dataset1 = np.genfromtxt("malignantfull.csv", delimiter = ",")

# compute the statistics
n0,d0 = dataset0.shape
n1,d1 = dataset1.shape
if (d0 == d1):
    d = d0
else:
    raise Exception("dataset0 and dataset1 do not have the same number
of columns.")

#Split dataset into training and test data.
train0 = dataset0[0:math.floor(n0/2),:]
test0 = dataset0[math.floor(n0/2):n0,:]
train1 = dataset1[0:math.floor(n1/2),:]
test1 = dataset1[math.floor(n1/2):n1,:]
n0train = train0.shape[0]
n1train = train1.shape[0]
n0test = test0.shape[0]
n1test = test1.shape[0]

#Estimate mean vectors and covariance matrices from training data.
mu0 = np.mean(train0, axis=0)
mu1 = np.mean(train1, axis=0)
sigma0 = np.cov(train0, rowvar=False)
sigma1 = np.cov(train1, rowvar=False)

#8.5(d) Apply Gaussian ML rule for identity covariance matrix by
plugging into the PDFs for the cancer data
#  Note: you shoud reuse the code from 8.5(a)
H0guesses_idcov = np.zeros((n0test,1))
H1guesses_idcov = np.zeros((n1test,1))

for i in range(n0test):
    currentdata = test0[i,:]
```

```
    h0 = stats.multivariate_normal.pdf(currentdata, mu0,
np.identity(d0))
    h1 = stats.multivariate_normal.pdf(currentdata, mu1,
np.identity(d0))
    if h1>=h0:
        H0guesses_idcov[i] = 1

for i in range(n1test):
    currentdata = test1[i,:]
    h0 = stats.multivariate_normal.pdf(currentdata, mu0,
np.identity(d1))
    h1 = stats.multivariate_normal.pdf(currentdata, mu1,
np.identity(d1))
    if h1>=h0:
        H1guesses_idcov[i] = 1

Pe_idcov = proberror(H0guesses_idcov,H1guesses_idcov)
print("Probability of error for identity covariance matrix is " +
str(Pe_idcov) + ".")
```

```
Probability of error for identity covariance matrix is 0.385.
```

```
#Now we start testing with the log-likelihood tests
# Problem 8.5(e)  Apply Gaussian ML rule for identity covariance
matrix by using the closest average classifier.
# Recall that function was defined above.

H0guesses_idcov = np.zeros((n0test,1))
H1guesses_idcov = np.zeros((n1test,1))

for i in range(n0test):
    currentdata = test0[i,:]
    H0guesses_idcov[i] = closest_average(currentdata, mu0, mu1)

for i in range(n1test):
    currentdata = test1[i,:]
    H1guesses_idcov[i] = closest_average(currentdata,mu0,mu1)

Pe_idcov = proberror(H0guesses_idcov,H1guesses_idcov)
print("Probability of error for identity covariance matrix is " +
str(Pe_idcov) + ".")
```

```
Probability of error for identity covariance matrix is
0.08499999999999999.

/var/folders/pf/619m_yfj3w5cwc52r3yxbb_40000gn/T/
ipykernel_46189/403505849.py:7: DeprecationWarning: `np.math` is a
deprecated alias for the standard library `math` module (Deprecated
Numpy 1.25). Replace usages of `np.math` with `math`
  distance_mu0 = np.math.sqrt(np.sum((currentdata-mu0)**2))
```

```
/var/folders/pf/619m_yfj3w5cwc52r3yxbb_40000gn/T/ipykernel_46189/403505849.py:8: DeprecationWarning: `np.math` is a deprecated alias for the standard library `math` module (Deprecated Numpy 1.25). Replace usages of `np.math` with `math`
  distance_mu1 = np.math.sqrt(np.sum((currentdata-mu1)**2))
```

```
#8.5 (f) Apply Gaussian ML rule for the same covariance matrix by using the log-likelihood ratio
# using poled covariances
H0guesses_samecov = np.zeros((n0test,1))
H1guesses_samecov = np.zeros((n1test,1))

sigmas = (1/(n0train+n1train-2))*((n0train-1)*sigma0+(n1train-1)*sigma1)
sigmainv = np.linalg.pinv(sigmas)

for i in range(n0test):
    currentdata = test0[i,:]
    value0 = np.matmul(np.matmul((currentdata-mu0),sigmainv),np.transpose(currentdata-mu0))
    value1 = np.matmul(np.matmul((currentdata-mu1),sigmainv),np.transpose(currentdata-mu1))
    if value1<=value0:
        H0guesses_samecov[i] = 1

for i in range(n1test):
    currentdata = test1[i,:]
    value0 = np.matmul(np.matmul((currentdata-mu0),sigmainv),np.transpose(currentdata-mu0))
    value1 = np.matmul(np.matmul((currentdata-mu1),sigmainv),np.transpose(currentdata-mu1))
    if value1<=value0:
        H1guesses_samecov[i] = 1

Pe_samecov = proberror(H0guesses_samecov,H1guesses_samecov)
print("Probability of error for the same covariance matrices is " + str(Pe_samecov) + ".")
```

```
Probability of error for the same covariance matrices is 0.065.
```

```
#8.5(g) Apply Gaussian ML rule for different covariance matrices by using the log-likelihood ratio.
H0guesses_diffcov = np.zeros((n0test,1))
H1guesses_diffcov = np.zeros((n1test,1))

sigma0inv = np.linalg.pinv(sigma0)
sigma1inv = np.linalg.pinv(sigma1)
offset = math.log(np.linalg.det(sigma0))-math.log(np.linalg.det(sigma1))
```

```
for i in range(n0test):
    currentdata = test0[i,:]
    value0 = np.matmul(np.matmul((currentdata-
mu0),sigma0inv),np.transpose(currentdata-mu0))
    value1 = np.matmul(np.matmul((currentdata-
mu1),sigma1inv),np.transpose(currentdata-mu1))
    if value1<=(value0+offset):
        H0guesses_diffcov[i] = 1

for i in range(n1test):
    currentdata = test1[i,:]
    value0 = np.matmul(np.matmul((currentdata-
mu0),sigma0inv),np.transpose(currentdata-mu0))
    value1 = np.matmul(np.matmul((currentdata-
mu1),sigma1inv),np.transpose(currentdata-mu1))
    if value1<=(value0+offset):
        H1guesses_diffcov[i] = 1

Pe_diffcov = proberror(H0guesses_diffcov,H1guesses_diffcov)
print("Probability of error for different covariance matrices is " +
str(Pe_diffcov) + ".")
```

```
Probability of error for different covariance matrices is 0.015.
```