Implementing Minimum Error Rate Classifier

Anas Sikder
Department of CSE
Ahsanullah University of Science and Technology
Dhaka, Bangladesh
160204021@aust.edu

***Abstract*—The experiment is about to implement the Minimum Error Rate Classifier to classify some sample points using the posterior probabilities. For doing so I used Multivariate Gaussian Density Function for calculating the likelihood probabilities and distribute the data points. The reason for this type of classifier is to minimize the error rate during classification. I distributed the data points into two classes and labeled them. As given data points follow Gaussian distribution and visible in a 3D plane that data in a different class is situated in a different region on the decision boundary.**

***Index Terms*—Likelihood,Prior,Prosterior,Probability Density Funtion,Sigma,Mean,Co relation,Co varience etc**

I. Introduction

Bayesian decision theory is a fundamental statistical approach to the problem of pattern classification. As we know, the minimum error rate classifier tries to minimize the error. In the experiment six sample data were given. The likelihood probabilities of a sample are determinate by the normal distribution.Any normal distribution can be express by two significant parameter μ and Σ.

A. Motivation

My main goal is to implement the Minimum Error Rate Classifier on given training sets and performing a distribution on given data and labeled them. Also, plot them in a 3D surface for clear visualization.

B. Theory

1) Posterior , Prior and Likelihood: The decision result based on Bayes depends on the class conditional probability and the prior probability of the training samples given. Bayesian estimates the probability that a certain event may occur in the future through probabilistic knowledge and statistics of existing data, using the method of probability. The posterior probability is a multiplication of likelihood and prior divide by marginal probability. The posterior probability is elaborated as:

$$P(\omega/x) = \frac{P(x/\omega) * P(\omega)}{P(x)}$$

$P(x/\omega)$ and $P(\omega)$ are likelihood and prior .

2) Minimum error rate and Multivariate Normal Distribution: Bayesian decision rules based on minimum error rate is:

$$P(\omega_1/x) > P(\omega_2/x) \qquad x \in \omega_1$$

$$P(\omega_1/x) < P(\omega_2/x) \qquad x \in \omega_2$$

The likelihood $P(x/\omega)$ is written in Normal distribution as $N(\mu_i, \Sigma_i)$.So the equation of Multivariate Normal Distribution is:

$$g_i(x) = -\frac{1}{2}(x-\mu_i)^T\Sigma_i^{-1}(x-\mu_i) - \frac{d}{2}\ln 2\pi - \frac{1}{2}\ln|\Sigma_i| + \ln P(\omega_i)$$

For two class as if $g_1(x) > g_2(x)$ $, x \in g_1(x)$ if not $x \in g_2(x)$. Also 'd' is the dimension of the data-set. For our experiment $d = 2$.Using Multivariate Normal equation data points can be distributed into different class with the help of Σ and μ pair of each class.

3) Sigma(Σ) and Mean(μ) : The value μ is referred to where the Multivariate Normal Density will be shown in the X and Y-axis.And the Σ contains co-relation(σ^2) and spreading direction. For better understanding let take two examples:

$$\Sigma_1 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \mu_1 = \begin{bmatrix} 1 & 1 \end{bmatrix}$$

We can see data, in this case, is equally spread in the X and

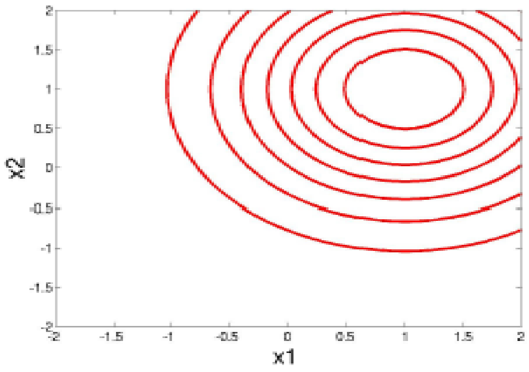

Fig. 1. Mean and Sigma influence in Normal Density

Y axis with no shifting as no co-relation exists and is situated at (1,1) position.

$$\Sigma_2 = \begin{bmatrix} 1 & .9 \\ .9 & 4 \end{bmatrix}, \mu_2 = \begin{bmatrix} 0 & 0 \end{bmatrix}$$

We can see data, in this case, is spread towards Y-axis and

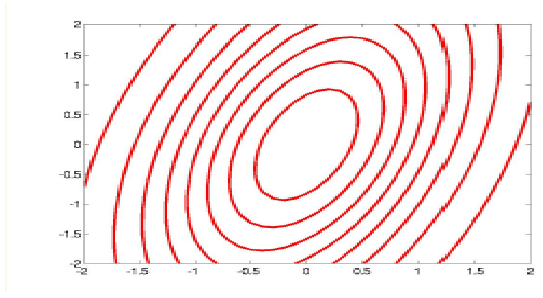

Fig. 2. Mean and Sigma influence in Normal Density

shifting to Y axis as positive co-relation exists and is situated at (0,0) position.

4) Decision Boundary: In a statistical-classification problem with two classes, a decision boundary is a hypersurface that partitions the underlying vector space into two sets, one for each class. The classifier will classify all the points on one side of the decision boundary as belonging to one class and all those on the other side as belonging to the other class. The decision boundary is determined by a hyperquadrics for Multivariate Normal distribution with an arbitrary covariance matrix. For implementing the decision boundary I subtract the two Multivariate Gaussian Density and calculated the decision boundary.

$$g_1(x) = \frac{1}{2\pi\sqrt{|\Sigma_1|}} e^{(-\frac{1}{2}(x-\mu_1)^T \Sigma_1^{-1}(x-\mu_1))}$$

$$g_2(x) = \frac{1}{2\pi\sqrt{|\Sigma_2|}} e^{(-\frac{1}{2}(x-\mu_2)^T \Sigma_2^{-1}(x-\mu_2))}$$

$$g(x) = g_1(x) - g_2(x)$$

II. Experimental Design / Methodology

A. Task 1

Using Discriminant Function for Multivariate Gaussian Density for two different Sigma and mean pair I check the following condition $g_1(x) > g_2(x)$ and classified data points and labeled as 'class 1' otherwise classified data points and labeled as 'class 2'.

B. Task 2

Then I drawn classified samples in 2D plane using different colored markers according to the assigned class label in task 1.

C. Task3

I drew a 3D plane including data points, the corresponding probability distribution function along its contour with the help of Matplotlib specifically using surface and contour plotting.

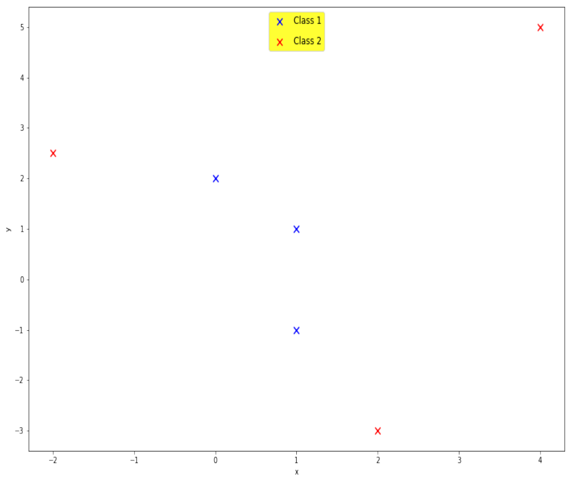

Fig. 3. Distribution of data points as Class 1 and Class 2

D. Task 4

Drawn a decision boundary line in between two contours with the help of the decision boundary equation described in the theory section.

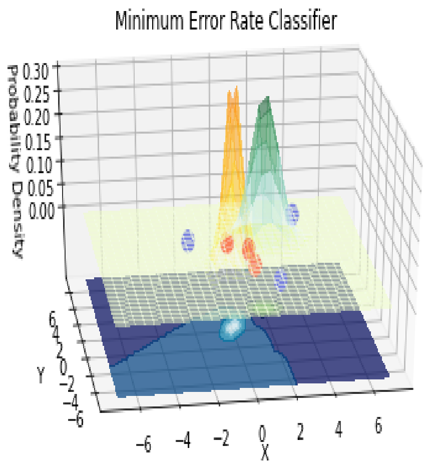

Fig. 4. Probability Distribution Function with data points,contour and decision boundary

III. Result Analysis

For Analysis the probability distribution and decision boundary in between of data points in a 3D plane I used surface plotting and contour plotting combined.

I used the equation of multivariate probability distribution function to calculate probability density along the Z-axis. From the two sets of pair values of mean and sigma two hills are drawn.The first pair of the mean(μ)and sigma(Σ) had higher height in the hill(likelihood) top are drawn in orange color. And the second pair had a lower height in the hill(likelihood) top are drawn in green color. From the top view, both contours are visible. **The first contour had positive co-relation and**

slightly angled to left (Y-axis) and inequal spread and Y-axis treats like a major axis. And the second contour has no co-relation and is equally spread in both the X and Y-axis. Also, the first contour is in (0,0) co-ordinate as the mean is (0,0) and the second one is in (2,2) co-ordinate as the mean(μ) is (2,2).
The data points are labeled in task 2 are equally distributed between two sides of the contour as both classes are in the different regions on the decision boundary. **From the above figure 5, now 3 sample values are classified as class 1 and 3 sample value is classified as class 2.**
The decision boundary is obtained by subtracting the two likelihood values and clearly visible in figure 5. The decision boundary is separated with dark and light blue colors.

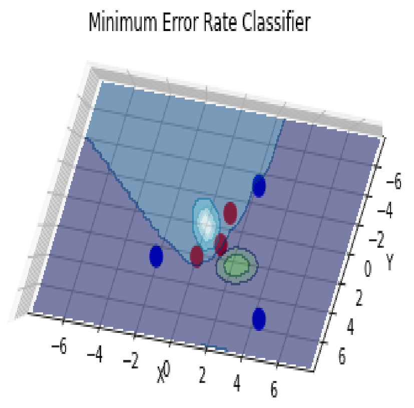

Fig. 5. Decision Boundary

IV. Conclusion

From this experiment,**I had a better understanding of the probability density function of the multivariate Gaussian distribution.**The data point I used and labeled them are meet the Gaussian distribution.So, the data points are **distributed perfectly in the curve region separation of the decision boundary. But, when the data does not meet the Gaussian distribution, how to calculate the class conditional probability density needs further thinking. Also, a linear decision boundary is way more flexible of work with than a curve decision boundary in the case of a large data-set scenario. Also,if I change the parameter of Gaussian distribution the sample values also can be shifted to another class because the likelihood probabilities can also be shifted towards another class.**

V. Algorithm Implementation / Code

Write down your code along with the algorithm (if applicable) here with necessary steps.

```
#import necessary libaries
import pandas
import numpy as np
from matplotlib import cm
import matplotlib.pyplot as plt
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D


#dataset fetching to Numpy Array
dataset = pandas.read_csv('test-Minimum-
                          Error-Rate-
                          Classifier.txt',
                          header = None)
dataset=np.array(dataset)

#define values of mean,sigma
min1 = [0.,0.]
min1=np.array(min1)
min2 = [2.,2.]
min2=np.array(min2)
sigma1 = [[.25,.3],[.3,1.]]
sigma1=np.array(sigma1)
sigma2 = [[.5,0.],[0.,.5]]
sigma2=np.array(sigma2)

#define values of prior
prior1 = 0.5
prior2 = 0.5
a=np.zeros([6,3])

#define empty class
class1=[[]]
class2=[[]]

#copy variable and assign
temp = np.empty_like(dataset)
temp[:] = dataset

#copy data
a[:,0]=dataset[:,0]
a[:,1]=dataset[:,1]

#posterior,likelihood,prior
b=[]
res1 = 0
res2 = 0
print(b)
for i in range(len(temp)):

        res1 = -0.5*np.dot(np.dot((temp[i,:]-min1)
        np.linalg.inv(sigma1)),(temp[i,:]-min1))
        -np.log(2*np.pi)-0.5*np.
        log(np.linalg.det(sigma1))+np.log(prior1)

        res2 = -0.5*np.dot(np.dot((temp[i,:]-min2)
        np.linalg.inv(sigma2)),(temp[i,:]-min2))
        -np.log(2*np.pi)-0.5*np.
        log(np.linalg.det(sigma2))+np.log(prior2)
        if (res1 > res2):
          b.append(1)
        else:
          b.append(2)

print(b)

#assigning class to a
for i in range(len(b)):
    a[i][-1]=b[i]
print(a)

#separating two class
class1 =[([i[0],i[1],i[2]]) for i in a
if i[2] == 1]
class1 = np.array(class1)
class2 =[([i[0],i[1],i[2]]) for i in a
if i[2] == 2]
class2 = np.array(class2)
print(class1)
print(class2)

#plotting dataset in 2d
fig, axis = plt.subplots()
fig.set_figheight(10)
fig.set_figwidth(15)

#scatting train values
axis.scatter(class1[:,0],class1[:,1],
marker='x',color='red',s=70,label='Class 1')
axis.scatter(class2[:,0],class2[:,1],
marker='x',color='blue',s=70,label='Class 2')

#labeling decoration
legend = axis.legend(loc='upper center'
,fontsize='large',labelspacing=1.0)
legend.get_frame().set_facecolor('yellow')
axis.set_xlabel('x')
axis.set_ylabel('y')

#distribution calculation
def mulv(pos,mu,Sigma):
    res1=np.zeros([32,32])
    for i in range(len(X)):
        for j in range(len(X[0])):
            r0 = (np.sqrt(np.linalg.det(Sigma))
            *2*np.pi)
            r1 = 1/r0
            r2 = -.5*np.dot(np.dot((pos[i,j]-mu)
            .T,np.linalg.inv(Sigma)),
            (pos[i,j]-mu))
            r3 = np.exp(r2)
            r4 = r1 * r3
            res1[i][j]=r4
    return res1

#poltting
N = 32
X = np.linspace(-7, 7, N)
Y = np.linspace(-7, 7, N)
X, Y = np.meshgrid(X, Y)
pos = np.empty(X.shape + (2,))
pos[:, :, 0] = X
pos[:, :, 1] = Y
Z = mulv(pos,min1,sigma1)
Z1 = mulv(pos, min2, sigma2)
db=(Z-Z1)
#print(type(db))
#db[:,:]=db[:,:]/40
fig = plt.figure()
ax = fig.gca(projection='3d')
fig.set_figheight(4)
fig.set_figwidth(8)
z=0
ax.scatter(class1[:,0],class1[:,1],
            z,color='blue',alpha=0.9)
ax.scatter(class2[:,0],class2[:,1],
            z,color='red',alpha=0.9)
ax.plot_surface(X, Y, Z, rstride=3, cstride=1,
                linewidth=5, antialiased=False,
                cmap=cm.Wistia,alpha=.5)
ax.plot_surface(X, Y, Z1, rstride=3, cstride=1,
                 linewidth=5, antialiased=False,
                 cmap=cm.BuGn,alpha=.5)
ax.contourf(X, Y, db, zdir='z', offset=-.15,
            cmap=cm.ocean,alpha=0.8)
ax.set_title('Probability Density')
ax.set_xlabel('x')
ax.set_ylabel('y')
ax.set_zlabel('z')
ax.set_zlim(-0.15,0.2)
ax.set_zticks(np.linspace(0,0.2,5))
ax.view_init(27, -21)
plt.show()
```