Implementing Minimum Error Rate Classifier

Shimul Paul
Department of Computer Science and Engineering
Ahsanullah University of Science and Technology
Dhaka, Bangladesh
160204014@aust.edu

***Abstract*—This classifier is designed for specify the sample points using posterior probabilities. It uses Gaussian distribution for this task. The main objective of this classifier is to minimize the error rate. So, we can say that this classifier take the decision based on the most posterior probabilities.**

***Index Terms*—Minimum Error Rate, Minimum Classifier, Gaussian Distribution.**

I. Introduction

Minimum Error Rate Classifier is used for decrease the error rate at the time of classification. In this experiment we try to classify the given sample points using normal distribution.Normal distribution expressed by mean and sigma. Where sigma is variance. For posterior probabilities the decision rules are as follows:

$$If\ p(w_1|x) > p(w_2)|x)\ then\ x\epsilon w_1$$

$$If\ p(w_1|x) < p(w_2)|x)\ then\ x\epsilon w_2$$

We can calculate posterior probabilities with the help of likelihood. So, it is as follows:

$$P(w_i|x) = P(x|w_i).P(w_i)$$

$$=> lnP(w_i|x) = lnP(x|w_i).P(w_i)$$

$$=> lnP(w_i|x) = lnP(x|w_i) + P(w_i)$$

Here $P(x|w_i)$ and $P(w_i)$ is likelihood and prior.
We use the following equation for normal distribution:

$$N_k(x_i|\mu_k, \Sigma_k) = \frac{1}{\sqrt{(2\pi)^D|\Sigma_k|)}} e^{\frac{-1}{2}((x_i-\mu_k)^T)\Sigma_k^{-1}(x_i-\mu_k)^T))}$$

Taking ln we can get,

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

Here, N_k is normal disctribution,μ_k is mean, Σ is co-variance matrix,x_i data, and D is 2 for this experiment.

So,the decision boundary is :

$$g_1(x) = g_2(x)$$

$$=> p(w_1|x) = p(w_2|x)$$

$$=> p(w_1|x) - p(w_2|x) = 0$$

$$=> P(x|w_1).P(w_1) - P(x|w_2).P(w_2) = 0$$

Taking ln we can get,

$$=> lnP(x|w_1).P(w_1) - lnP(x|w_2).P(w_2) = 0$$

$$=> lnP(x|w_1) + lnP(w_1) - lnP(x|w_2) - lnP(w_2) = 0$$

$$=> lnP(x|w_1)/lnP(x|w_2) - lnP(w_2)/lnP(w_1) = 0$$

This is the equation of a decision boundary for minimum error rate classifier.

II. Experimental Design / Methodology

- Firstly, I read all the training data using pandas read_csv() function and store it in a array.
- Then, I convert it in numpy arrays.
- Then, I initializes all the μ and Σ with the values provided in the question.
- Then, I take a temporary train class.
- Then, I apply the normal distribution function and classify the data.
- Then, for the training data I store the values of two feature in two arrays if the class is 1 then class1 else in class2.
- After that, I plot the training class1 and training class2 values using matplotlib with marker and color.
- Then, I plot the contour plot with decision boundary.

III. Algorithm Implementation / Code

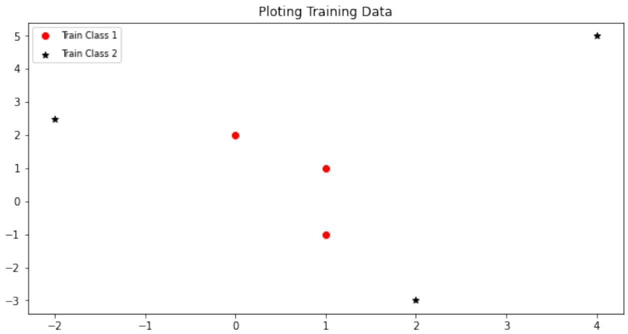
```
import pandas
import numpy as np
import matplotlib.pyplot as matplot
from matplotlib import cm
from mpl_toolkits.mplot3d import Axes3D

train = pandas.read_csv('test-Minimum-Error-Rate-Classifier.txt', header = None);
train = np.array(train)

#initialize mean,sigma(variance),prior
mean1 = [0,0]
mean2 = [2,2]
sigma1 = [[.25,.3],[.3,1]]
sigma2 = [[.5,0],[0,.5]]
prior1 = 0.5
prior2 = 0.5
```

```
#converting in numpy arrays
mean1=np.array(mean1)
mean2=np.array(mean2)
sigma1=np.array(sigma1)
sigma2=np.array(sigma2)

trainNew=np.zeros([6,3])
trainNew[:,0]=train[:,0]
trainNew[:,1]=train[:,1]

tempClass=[]
resultClass1 = 0
resultClass2 = 0
for i in range(len(train)):
    resultClass1 = -0.5*np.dot(np.dot
    ((train[i,:]-mean1).T,np.linalg.inv
    (sigma1)),(train[i,:]-mean1))-np.log
    (2*np.pi)-0.5*np.log(np.linalg.det
    (sigma1))+np.log(prior1)
    resultClass2 = -0.5*np.dot(np.dot
    ((train[i,:]-mean2).T,np.linalg.inv
    (sigma2)),(train[i,:]-mean2))-np.log
    (2*np.pi)-0.5*np.log(np.linalg.det
    (sigma2))+np.log(prior2)

    if (resultClass1 > resultClass2):
        tempClass.append(1)
    else:
        tempClass.append(2)
print(tempClass)

for i in range(len(tempClass)):
    trainNew[i][-1]=tempClass[i]
print(trainNew)

class1 =[([x[0],x[1],x[2]]) for x in
trainNew if x[2] == 1]
class2 =[([x[0],x[1],x[2]]) for x in
trainNew if x[2] == 2]
class1 = np.array(class1)
class2 = np.array(class2)

fig ,ax = matplot.subplots()
ax.set_title('Ploting Training Data')
fig.set_figheight(5)
fig.set_figwidth(10)
ax.scatter(class1[:,0],class1[:,1]
,marker='o',color='r',label='Train Class 1')
ax.scatter(class2[:,0],class2[:,1]
,marker='*'
,color='k',label='Train Class 2')
legend = ax.legend(loc='upper left'
, shadow=False,fontsize='small',
labelspacing=1)
legend.get_frame().set_facecolor('None')
matplot.show()

X = np.linspace(-8, 8, 32)
Y = np.linspace(-8, 8, 32)
X, Y = np.meshgrid(X, Y)
pos = np.empty(X.shape + (2,))

pos[:, :, 0] = X
pos[:, :, 1] = Y


def gaussian(pos, mean, Sigma):
    d = mean.shape[0]
    Sigma_det = np.linalg.det(Sigma)
    Sigma_inv = np.linalg.inv(Sigma)
    N = np.sqrt((2*np.pi)**d * Sigma_det)
    fac = np.einsum('...k,kl,...l->...'
    , pos-mean, Sigma_inv, pos-mean)
    return np.exp(-fac / 2) / N

Z = gaussian(pos, mean1, sigma1)
Z1 = gaussian(pos, mean2, sigma2)

fig = matplot.figure()
fig.set_figheight(10)
fig.set_figwidth(15)
ax = fig.gca(projection='3d')
db=Z-Z1
z=0
ax.scatter(class1[:,0],class1[:,1]
,z,color='red')
ax.scatter(class2[:,0],class2[:,1]
,z,color='blue')

ax.plot_surface(X,Y,Z,rstride=1,cstride=1
,linewidth=1,antialiased=False,
cmap=cm.BuGn,alpha=.3)
ax.plot_surface(X,Y,Z1,rstride=1,cstride=1
,linewidth=1,antialiased=False,
cmap=cm.coolwarm,alpha=.3)
ax.contourf(X,Y,db,zdir='z'
,offset=-.15,cmap=cm.ocean,alpha=0.7)
ax.set_title('Probability Density')

ax.set_xlabel('X')
ax.set_ylabel('Y')
ax.set_zlabel('Z')

ax.set_zlim(-0.20,0.2)
ax.set_zticks(np.linspace(0,0.2,5))
ax.view_init(27, -21)

matplot.show()
```

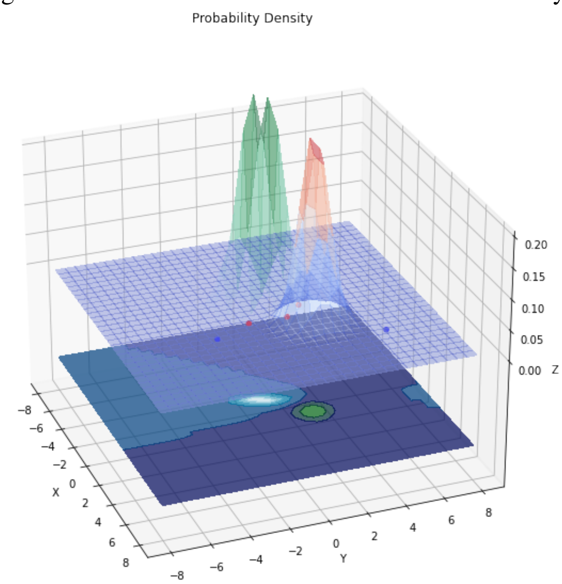
IV. Result Analysis

The results is shown below:

- Firstly, I applied the normal distribution formula for classify the data and I got this output: [1, 1, 2, 2, 1, 2]
- Secondly, I plot the classes using matplotlib and the got this output:



- Thirdly, I plot the contour plot along with its decision boundary.



CONCLUSION

Here in this experiment we see that how to solve the classify problem using normal distribution and how to minimize the error.From this classifier it can be seen that this classifier based on probability.