Implementing Minimum Error Rate Classifier .

Rakib Hossain Rifat
Department of Computer Science and Engineering
Ahsanullah University of Science and Technology
Dhaka, Bangladesh
160204099@aust.edu

***Abstract*—Main objective of this experiment is classify data points and minimize the error.**

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

I. Introduction

Minimum Error Rate Classifier is used for minimize the error rate at the time of classification. In this experiment I need to classify data points of two class using Minimum Error Rate Classifier.

II. Methodology

In this experiment I need to perform several tasks.

A. Plotting Test Data of Both Classes

For this I need to plot training data of both classes from "test-Minimum-Error-Rate-Classifier.txt", and samples belonging to same class should have same marker and color. for this we will use normal distribution formula given below-

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

Normal distribution expressed by mean and sigma. Where sigma is the variance.

B. Draw Decision Boundary

For this I need to use the equation given bellow.

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

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

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

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

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 the decision boundary for minimum error rate classifier.

III. Result Analysis

A. Plotting Training Data of Both Classes

Here I have plotted data of both classes with different marker and color.
Class 1= [1, 1.0], [1, -1.0], [0, 2.0]
Class 2= [4, 5.0], [-2, 2.5], [2, -3.0]

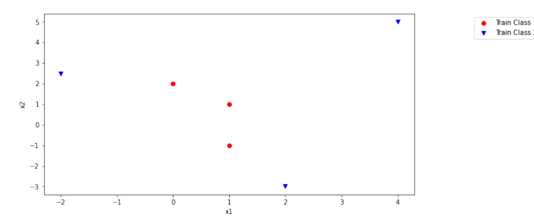

Fig. 1. Test Data of Class One and Class Two

B. Plotting contour graph along with its decision boundary.

Here I have plotted the contour graph along with its decision boundary.

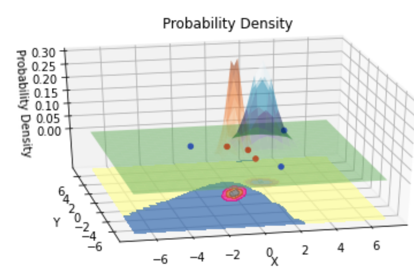

Fig. 2. contour plot along with its decision boundary.

IV. Conclusion

From the experiment it can be stated that data points can be classified using their normal distribution and using minimum error rate. And this classifier is based on probability.

V. CODE

```
# -*- coding: utf-8 -*-
"""160204099_B2_03.ipynb

Automatically generated by Colaboratory.

Original file is located at
    https://colab.research.google.com/drive/1
    psNqDxX4zlvdOAEbA-1lT69DKEvE68c6
"""

import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import math
from matplotlib import cm
from mpl_toolkits.mplot3d import Axes3D

train_df=pd.read_csv("/content/test-Minimum-Error-
    Rate-Classifier.txt",sep="," ,names=["x1","x2"])
train_set=np.array(train_df)

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

class1 = []
class2 = []
class1_X=[]
class1_Y=[]
class2_X=[]
class2_Y=[]
pri_omega_one=0.5
pri_omega_two=0.5
miu_one=np.array([0,0])
miu_two=np.array([2,2])
sigma_one=np.array([[.25,.3],[.3,1]])
sigma_two=np.array([[.5,0],[0,.5]])
D=2

for i in range(0,len(train_set)):


  term_one_1=pow((2*math.pi),D)

  term_one_2=np.linalg.det(sigma_one)

  term_one_one=1/math.sqrt(term_one_1 * term_one_2)

  term_one_two = -0.5 * np.dot(np.transpose(np.
    subtract(train_set[i] , miu_one) ), np.dot(np.
    linalg.inv(sigma_one), (np.subtract(train_set[i]
     , miu_one)) ))

  normal_one=term_one_one*np.exp(term_one_two)


  term_two_1=pow((2*math.pi),D)

  term_two_2=np.linalg.det(sigma_two)

  term_two_one=1/math.sqrt(term_two_1* term_two_2)

  term_two_two = -0.5 * np.dot(np.transpose(np.
    subtract(train_set[i] , miu_two) ), np.dot(np.
    linalg.inv(sigma_two), (np.subtract(train_set[i]
     , miu_two)) ))
  normal_two=term_two_one*np.exp(term_two_two)


  post1=normal_one*pri_omega_one
  post2=normal_two*pri_omega_two
  if (post1 > post2) :
    class1.append([train_set[i][0],train_set[i
    ][1],1])

  else :
     class2.append([train_set[i][0],train_set[i
    ][1],2])

for i in range(len(class1)):
  class1_X.append(class1[i][0])
  class1_Y.append(class1[i][1])
for i in range(len(class2)):
  class2_X.append(class2[i][0])
  class2_Y.append(class2[i][1])

plt.figure(figsize=(10,5))
plt.plot(class1_X,class1_Y,'ro' , label = 'Train
    Class 1')
plt.plot(class2_X,class2_Y,'bv' , label = 'Train
    Class 2')
plt.xlabel('x1')
plt.ylabel('x2')
plt.legend(loc="upper center",bbox_to_anchor=(1.25,
    1))
plt.show()

X = np.linspace(-7, 7, 32)
Y = np.linspace(-7, 7, 32)
X, Y = np.meshgrid(X, Y)
pos = np.empty(X.shape + (2,))
pos[:, :, 0] = X
pos[:, :, 1] = Y
Z = multivariate_gaussian(pos, miu_one, sigma_one)
Z1 = multivariate_gaussian(pos, miu_two, sigma_two)
db=(Z-Z1)
fig = plt.figure()
ax = fig.gca(projection='3d')
fig.set_figheight(4)
fig.set_figwidth(8)
z=0
ax.scatter(class1_X,class1_Y,color='red',alpha=1)
ax.scatter(class2_X,class2_Y,color='blue',alpha=1)
ax.plot_surface(X, Y, Z, rstride=1, cstride=1,
    linewidth=1, antialiased=True,
                cmap=cm.Oranges,alpha=.3)
ax.plot_surface(X, Y, Z1, rstride=1, cstride=1,
    linewidth=1, antialiased=True,
                cmap=cm.ocean,alpha=.3)
ax.contourf(X, Y, db, zdir='z', offset=-.15,cmap=cm.
    Accent,alpha=0.7)
ax.set_title('Probability Density')
ax.set_xlabel('X')
ax.set_ylabel('Y')
ax.set_zlabel('Probability Density')
ax.set_zlim(-0.15,.2)
ax.set_zticks(np.linspace(.30,0.0,7))
ax.view_init(30, -102)
plt.show()
```