**BỘ CÔNG THƯƠNG**

**TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP**

**KHOA CÔNG NGHỆ THÔNG TIN**

**\*\*\*\*\*\*🙣🟋🙡\*\*\*\*\***



***BÀI BÁO CÁO THỰC HÀNH***

***MÔN NHẬN DẠNG MẪU***

**Giáo viên hướng dẫn: Hồ Đắc Quán**

**Sinh viên thực hiện: Nguyễn Đức Nam**

**Mã số sinh viên: 17089981**

**Lớp: DHKHMT13A**

**Tp Hồ Chí Minh – 11/2020**

**BÀI THỰC HÀNH SỐ 3**

***Bài tập 2:*** Cho 2 tập dữ liệu class A (classA.mat) và class B (classB.mat). Xây dựng bộ classifier với 2 đặc trưng. Giả sử hai tập dữ liệu có dạng phân bố Gauss có cùng ma trận hiệp phương sai là SIGMA=[1 0; 0 1].

* Load data:
  + Load 2 file tương ứng cho 2 class là: classA.mat và classB.mat.
  + Plot dữ liệu.
  + Xác định số mẫu của mỗi class.
  + Phân chia tập dữ liệu thành 2 tập con: tập huấn luyện (60%) và tập kiểm thử (40%).
* Huấn luyện:
  + Tính mean tương ứng cho từng class.
  + Xây dựng biệt thức (discriminant function).
* **Đánh giá:**
  + Đánh giá trên tập dữ liệu test. Tính độ chính xác cho từng tập và xây dựng confusion matrix.
  + Plot dữ liệu testing của 2 lớp và đường biên phân lớp trên cùng một hình.

Tương tự, sinh viên thực nghiệm với các cách phân chia tập dữ liệu khác nhau. Cụ thể, tập dữ liệu huấn luyện là 70%, 75%, và 80%.

**Bài làm:**

* ***Load data:***
  + Load 2 file tương ứng cho 2 class là: classA.mat và classB.mat.
    1. Load dữ liệu từ file:

**#Load dữ liệu từ 2 file classA.mat và classB.mat**

dataA = scipy.io.loadmat('classA.mat')

dataB = scipy.io.loadmat(‘classB.mat')

* + 1. Chuyển đổi dữ liệu sang cấu trúc DataFrame:

**#Khởi tạo tên cho 2 thuộc tính của dữ liệu**

columnsName = ['Feature1','Feature2']

**#Chuyển dữ liệu được nhập vào sang dạng DataFrame**

classA = pd.DataFrame(dataA['classA'] ,columns = columnsName)

classB = pd.DataFrame(dataB['classB'] ,columns = columnsName)

* + Plot dữ liệu:
    1. Ghép 2 class lại thành 1 class chung để dễ dàng dùng để vẽ plot:

**#Ghép 2 tập dữ liệu lại với nhãn của chúng lần lượt là A và B**

classMerge = pd.concat([classA,classB], keys=['A', 'B'])

**#Đánh thứ tự lại cho từng dòng của dữ liệu**

classMerge = classMerge.reset\_index()

**#Xoá cột "level\_1"**

classMerge = classMerge.drop('level\_1', axis=1)

**#Đổi tên cột 'level\_0' thành 'Class'**

classMerge = classMerge.rename(columns = {'level\_0': 'Class'})

* + 1. Vẽ plot cho 2 tập dữ liệu đã cho:

**#Khởi tạo môi trường seaborn để vẽ plot.**

sns.set()

**#Khởi tạo bảng lưới với dữ liệu đã ghép phân loại theo loại “Class” và độ lớn của plot là 7.**

g = sns.FacetGrid(classMerge, hue="Class", height=7)

**#Vẽ plot có dạng chấm (scatter) theo 2 thuộc tính lần lượt là “Feature1” và “Feature2”.**

g.map(plt.scatter,"Feature1","Feature2")

**#Thêm chú thích vào plot.**

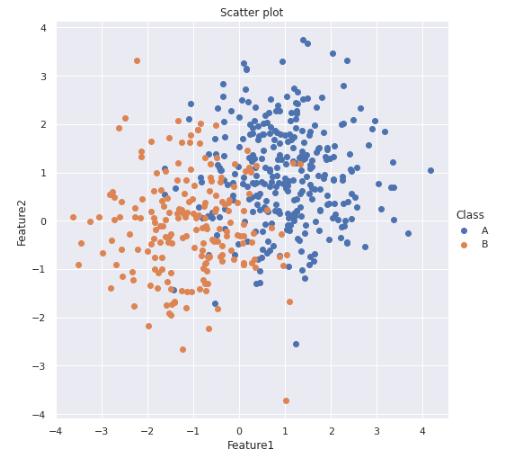
g.add\_legend()

**#Thêm nhãn của plot.**

plt.title('Scatter plot')

**#Hiện plot.**

plt.show()



Hình : Plot đại diện cho 2 tập dữ liệu đã cho.

* + Xác định số mẫu của mỗi class:

**#In số lương mẫu trong mỗi class**

print("Số mẫu của class A là :",len(classA))

print("Số mẫu của class B là :",len(classB))



Hình : Số lượng mẫu trong mỗi class

* + Phân chia tập dữ liệu thành 2 tập con: tập huấn luyện (60%) và tập kiểm thử (40%):

**#Chia tập dữ liệu thành 2 tập con: tập huấn luyện chiếm 60% và tập kiểm thử chiếm 40%**

**#Hàm sẽ xáo trộn ngẫu nhiên tập dữ liệu bạn đầu. Xong chọn ra 60%\*(300+195) = 297 mẫu**

**#sử dụng cho tập huấn luyện và phần còn lại sử dụng cho tập kiểm thử .**

train\_set, test\_set = train\_test\_split(classMerge, train\_size=0.6)

**#In ra số lượng phần tử của mỗi tập.**

print("Số lượng mẫu của tập huấn luyện là :",len(train\_set))

print("Số lượng mẫu của tập kiểm thử là :",len(test\_set))

******

Hình : Số lượng mẫu trong tập huấn luyện và tập kiểm thử

* ***Huấn luyện:***
  + Tính mean tương ứng cho từng class:
    1. Tính trung vị :

**#Tính trung vị của mỗi thuộc tính trong mỗi class của tập huấn luyện.**

mu\_list = np.split(train\_set.groupby('Class').mean().values,[1])

* + 1. Tính hiệp phương sai :

**#Tính hiệp phương sai của mỗi thuộc tính trong mỗi class của tập huấn luyện.**

cov\_list = np.split(train\_set.groupby('Class').cov().values,[2])

* + 1. Tính xác suất phân bố :

**#Tính xác suất phân bố của mỗi thuộc tính trong mỗi class của tập huấn luyện.**

pi\_list = train\_set.iloc[:,0].value\_counts().values / len(train\_set)

* + 1. In ra giá trị trung vị của tập huấn luyện:

**#In ra giá trị trung vị của tập huấn luyện**

print("Mean của từng class A trong train set là : ",mu\_list[0][0])

print("Mean của từng class B trong train set là : ",mu\_list[1][0])



Hình : Trung vị của từng thuộc tính trong mỗi class.

* + Xây dựng biệt thức (discriminant function):
* Ta có biệt hàm của một hàm phân phối chuẩn:
* Mà vì các biệt hàm đều có nên ta có thể lược bỏ đi :
* Để dễ dàng tính toán ta sẽ cho toàn bộ :
* Vì dữ liệu có dạng phân phối chuẩn nên ta có :
* Mà vì các biệt hàm đều có nên ta có thể lược bỏ đi :
* Để dễ dàng ta có thể tóm tắt công thức trên như sau:

Trong đó:

và

* + - Và hàm biệt hàm dưới đây được xây dựng dựa trên công thức trên:

def DF(X,mu\_list,cov\_list,pi\_list):

**#Tạo danh sách để chứa các giá trị của biệt hàm**

  scores\_list = []

**#Lấy số lượng lớp cần phân**

  classes = len(mu\_list)

**#Tính giá trị của biệt hàm với mỗi lớp**

  for p in range(classes):

**#Tính W\_i**

    Wi = (-1/2)\*np.linalg.inv(cov\_list[p])

**#Tính w\_i**

    wi = np.linalg.inv(cov\_list[p])@mu\_list[p][0]

**#Tính w\_i0**

    wi0 = (-1/2)\*np.transpose(mu\_list[p][0])@np.linalg.inv(cov\_list[p])@mu\_list[p][0]

    + (-1/2)\*np.log(np.linalg.norm(cov\_list[p]))

    + np.log(pi\_list[p])

**#Tính giá trị biệt hàm g\_i**

    score = np.transpose(X)@Wi@X + np.transpose(wi)@X + wi0

**#Thêm giá trị biệt hàm vào danh sách**

    scores\_list.append(score)

**#Trả về lớp có giá trị biệt hàm lớn nhất**

  return np.argmax(scores\_list)

* ***Đánh giá:***
  + Đánh giá trên tập dữ liệu test. Tính độ chính xác cho từng tập và xây dựng confusion matrix:
  + Plot dữ liệu testing của 2 lớp và đường biên phân lớp trên cùng một hình: