BÀI THỰC HÀNH SỐ 3

I. Mục đích:

Xây dựng classifier và biệt thức dựa trên luật Bayes.

II. Báo cáo:

Mỗi nhóm sẽ làm báo kết quả riêng, nộp kèm file source chương trình.

III. Nội dung:

1. Xây dựng bộ phân lớp và trực quan hóa biên phân lớp đối với tập dữ liệu iris:
   - Tập dữ liệu có 3 lớp với các nhãn là Setosa, Versicolor, và Virginica.
   - Sử dụng 2 đặc trưng: sepal length và sepal width.
   - Load dữ liệu:

```
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
import matplotlib.colors as colors
import seaborn as sns
import itertools
from scipy.stats import norm
import scipy.stats
from sklearn.naive_bayes import GaussianNB

#%matplotlib inline
sns.set()
#Load the data set
iris = sns.load_dataset("iris")
iris = iris.rename(index = str, columns =
{'sepal_length':'1_sepal_length','sepal_width':'2_sepal_width',
'petal_length':'3_petal_length', 'petal_width':'4_petal_width'})

#Plot the scatter of sepal length vs sepal width
sns.FacetGrid(iris, hue="species",
height=7) .map(plt.scatter,"1_sepal_length",
"2_sepal_width", )  .add_legend()
plt.title('Scatter plot')
plt.show()
```

   - Xây dựng hàm đánh giá:

```
def predict_NB_gaussian_class(X,mu_list,std_list,pi_list):
```

```
    #Returns the class for which the Gaussian Naive Bayes objective function
has greatest value
    scores_list = []
    classes = len(mu_list)

    for p in range(classes):
        score = (norm.pdf(x = X[0], loc = mu_list[p][0][0], scale =
std_list[p][0][0] )
                * norm.pdf(x = X[1], loc = mu_list[p][0][1], scale =
std_list[p][0][1] )
                * pi_list[p])
        scores_list.append(score)

    return np.argmax(scores_list)

#---------------------------------------------------
def predict_Bayes_class(X,mu_list,sigma_list):
    #Returns the predicted class from an optimal bayes classifier -
distributions must be known
    scores_list = []
    classes = len(mu_list)

    for p in range(classes):
        score = scipy.stats.multivariate_normal.pdf(X, mean=mu_list[p],
cov=sigma_list[p])
        scores_list.append(score)

    return np.argmax(scores_list)
```

- Plot biên phân lớp

```
df1 = iris[["1_sepal_length", "2_sepal_width",'species']]
#Estimating the parameters
mu_list = np.split(df1.groupby('species').mean().values,[1,2])
std_list = np.split(df1.groupby('species').std().values,[1,2], axis = 0)
pi_list = df1.iloc[:,2].value_counts().values / len(df1)

# Our 2-dimensional distribution will be over variables X and Y
N = 100
X = np.linspace(4, 8, N)
Y = np.linspace(1.5, 5, N)
X, Y = np.meshgrid(X, Y)

#fig = plt.figure(figsize = (10,10))
#ax = fig.gca()
color_list = ['Blues','Greens','Reds']
```

```
my_norm = colors.Normalize(vmin=-1.,vmax=1.)

g = sns.FacetGrid(iris, hue="species", height=10, palette =
'colorblind') .map(plt.scatter, "1_sepal_length",
"2_sepal_width",)  .add_legend()
my_ax = g.ax


#Computing the predicted class function for each value on the grid
zz = np.array(  [predict_NB_gaussian_class( np.array([xx,yy]).reshape(-1,1),
mu_list, std_list, pi_list)
                    for xx, yy in zip(np.ravel(X), np.ravel(Y)) ] )
'''
zz1 = np.array(  [predict_Bayes_class( np.array([xx,yy]).reshape(-1,1),
mu_list, std_list)
                    for xx, yy in zip(np.ravel(X), np.ravel(Y)) ] )
'''
#Reshaping the predicted class into the meshgrid shape
Z = zz.reshape(X.shape)


#Plot the filled and boundary contours
my_ax.contourf( X, Y, Z, 2, alpha = .1, colors = ('blue','green','red'))
my_ax.contour( X, Y, Z, 2, alpha = 1, colors = ('blue','green','red'))

# Addd axis and title
my_ax.set_xlabel('Sepal length')
my_ax.set_ylabel('Sepal width')
my_ax.set_title('Gaussian Naive Bayes decision boundaries')

plt.show()
```

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). Tham khảo bài giảng.

- **Đánh giá**

- Đánh giá trên tập dữ liệu test. Tính độ chính xác cho từng tập và xây dựng confusion matrix.

	Class A	Class B
Class A		
Class B		

- 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%.

3. Xây dựng bộ classifier với 2 lớp, 2 đặc trưng. Giả sử tập dữ liệu có dạng phân bố Gauss. Tập dữ liệu huấn luyện là class1_train.mat và class2_train.mat. Tập dữ liệu đánh giá là class1_test.mat và class2_test.mat.

- *Load data:*

- Load 2 file tương ứng cho 2 class là: *class1_train.mat* và *class2_train.mat*. Sử dụng lệnh:

- Plot dữ liệu.

- *Xây dựng classifier**:***

- Tính mean, và covariance tương ứng cho từng class.

- Xây dựng biệt thức (discriminant function).

- Đánh giá trên tập dữ liệu test, *class1_test.mat* và *class2_test.mat*. Tính độ chính xác cho từng tập và xây dựng confusion matrix.

	Class1	Class2
Class1		
Class2		

- Plot dữ liệu testing của 2 lớp và đường biên phân lớp trên cùng một hình.