Shantanu Pande 181060057

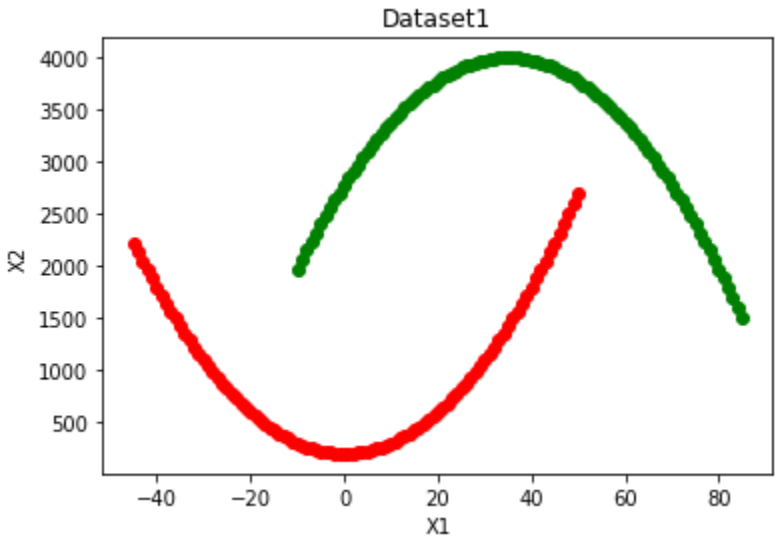
DS Lab Assignment 4

Problem Statement: During our classroom discussions and previous lab sessions, we have explored the utility of basis functions and Eigen analysis in context of classification. Please find attached the data file and and try to find a linear separator using the techniques studied in the class. The submission will contain the data plots with correct labels in the original space, the script, analytical form of the classifier, classified result, your observations and comments (the most important!).

Code for original basis plot:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from scipy.linalg import eigh

def plot_in_org():
    df = pd.DataFrame(pd.read_excel("D4.xlsx"))
    df_1 = df.dropna(axis=1,how='all')
    data1 = df_1
    colors = {'C1':'r', 'C2':'g'}
    fig, ax = plt.subplots()
    for i in range(len(data1['X1'])):
        ax.scatter(data1['X1'][i], data1['X2'][i],color=colors[data1['Class'][i]])
    ax.set_title('Dataset1')
    ax.set_xlabel('X1')
    ax.set_ylabel('X2')
    return data1

data1 = plot_in_org()
```

Plot in original basis:

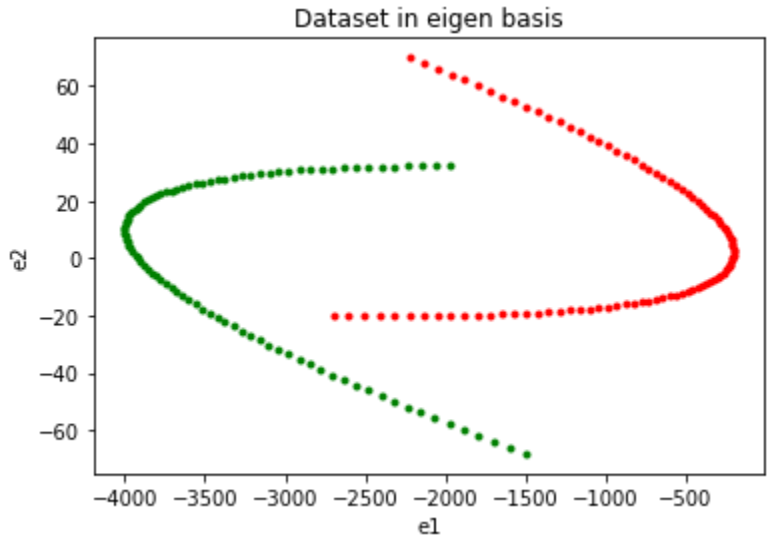


Observation: The above plot shows the given dataset D4 and we can clearly see that the classes are not separable in the original basis using any linear separator.

Code for eigen analysis:

```
def calculate_eigen(data):
    d1_array = data.to_numpy()
    X = np.array([[],[]])
    for i in range (len(d1_array)):
        X = np.append(X,[[d1_array[i][1]],[d1_array[i][2]]], axis= 1)
    cov = np.cov(X,ddof=0)
    e,trans = np.linalg.eig(cov)
    sort_index = np.argsort(-1*e)
    e = e[sort_index]
    trans = trans[:,sort_index]
    trans = trans.T
    Y = np.matmul(trans,X)
    Y = np.array(Y)
    print(trans)
    class1 = np.array([[],[]])
    class2 = np.array([[],[]])
    eig_class1 = np.array([[],[]])
    eig_class2 = np.array([[],[]])
    for i in range(len(d1_array)):
        if d1_array[i][3] == "C1":
            class1 = np.append(class1, [[d1_array[i][1]], [d1_array[i][2]]], axis=1)
            eig_class1 = np.append(eig_class1, [[Y[0][i]], [Y[1][i]]], axis=1)
        else:
            class2 = np.append(class2, [[d1_array[i][1]], [d1_array[i][2]]], axis=1)
            eig_class2 = np.append(eig_class2, [[Y[0][i]], [Y[1][i]]], axis=1)
    plt.plot(eig_class1[0],eig_class1[1],'r.')
    plt.plot(eig_class2[0],eig_class2[1],'g.')
    plt.title("Dataset in eigen basis")
    plt.xlabel('e1')
    plt.ylabel('e2')
    plt.show()
    return class1, class2, eig_class1, eig_class2
class1,class2, eig_class1, eig_class2 = calculate_eigen(data1)
```

Plot in eigen basis:

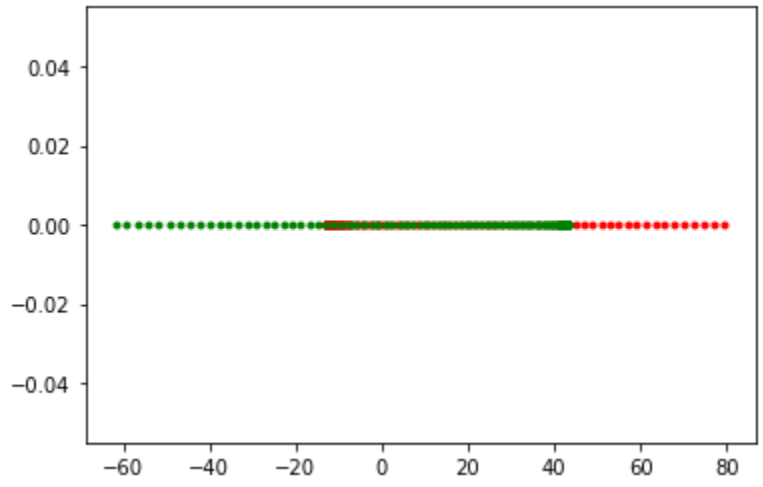


Observations: It is evident that this dataset cannot be separated in the eigen basis also, as we are unable to find a linear separator for the same.

Code for generalised eigen:

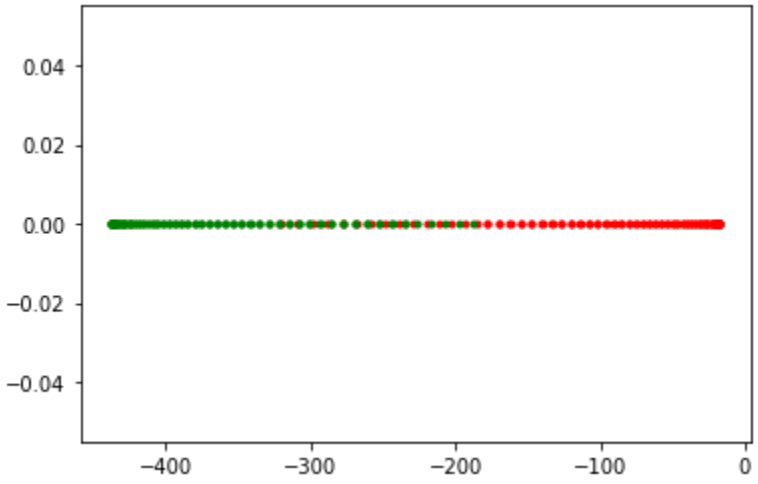
```
def generalized_eigen():
    cw1 = np.cov(class1)
    cw2 = np.cov(class2)

    cw = (cw1+cw2)/2

    data = np.concatenate((class1,class2),axis=1)

    ca = np.cov(data)

    eigvals, eigvecs = eigh(ca, cw, eigvals_only=False)

    return eigvals, eigvecs, ca, cw, data

val, vec, ca, cw, data = generalized_eigen()

print("eigen vectors",vec)
print("eigen values", val)

s1 = val[0]
m1 = (ca-(s1*cw))
# s2 = val[1]
# m2 = (ca-(s2*cw))
d = null_space(m1)
print("d",d)

out1 = np.dot(d.T,class1)
out2 = np.dot(d.T,class2)
y = np.zeros([1,96])
# out1 = m2_null * class1
# out2 = m2_null * class2
plt.plot(out1,y,"r.")
plt.plot(out2,y,"g.")
plt.show()
```

Plot using value of s=s1, after generalized eigen method:



Observation: Using the generalized eigen method we can see from the above plot that for s=0.9947 the projections of classes C1 and C2 overlap and that the dataset D4 is not separable using this method.

Plot using value of s=s2, after generalized eigen method:



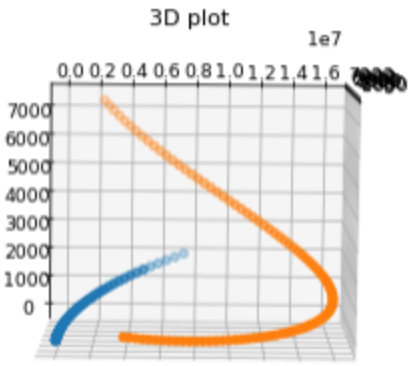
Observation: Using the generalized eigen method we can see from the above plot that for s=3.96047 the projections of classes C1 and C2 overlap and that the dataset D4 is not separable using this method.

So, the given dataset is too complex to be separated in the original dimension and that we need to increase the dimensionality in order to get the data separated linearly.

Code for visualization in higher dimensions:

```
%matplotlib notebook
def data_in_3d(class1, class2):
#     class1 = class1.T
#     class2 = class2.T
    f1 = np.power(class1[0],[2])
    f2 = np.power(class1[1],[2])
    f3 = np.multiply(f1, f2) * 2
    ax = plt.axes(projection ='3d')
    ax.scatter3D(f1, f2, f3, 'red')
    g1 = np.power(class2[0],[2])
    g2 = np.power(class2[1],[2])
    g3 = np.multiply(g1, g2) * 2
    ax.scatter3D(g1, g2, g3, 'green')
    ax.set_title('3D plot')
    plt.show()
data_in_3d(class1, class2)
```

Plot in 3D:



Here orange is for C1 and blue for C2 instead for red and green, due to scatter3D functions problem the colors are not plotted correctly.

Observations: The above plot represents C1 and C2 in 3D using
$f1 = x1^2$, $f2 = x2^2$, $f3 = x1x2$ for C1 and
$g1 = x1^2$, $g2 = x2^2$, $g3 = x1x2$
for C2. Also, various combinations of x1 and x2 were used in order to plot in such a way that the data can be separated using a linear separator in higher dimension. Also, using various combinations for f1, f2, f3 and g1, g2, g3 did not yield a result that could be linearly separated in three dimensions.
So according to the operations and transformations it seems that the data is not separable or it appears to be too complex to be separated in this dimension using the above used operations.