Question 3

Solve the following small size problem manually to understand the working of the algorithm. Consider four training patterns, one from each region, as follows.

• From Region1, X = (0, –0.5)t ∈ NLClass1. So, Xa = (0, –0.5, 0, 0.25)t . • From Region3, X = (–1, –0.5)t ∈ NLClass1; Xa = (–1, –0.5, 1, 0.25)t. • From Region4, X = (–0.5, 0)t ∈ NLClass2; Xa = (–0.5, 0, 0.25, 0)t. • From Region2, X = (–0.5, –1)t ∈ NLClass2; Xa = (–0.5, –1, 0.25, 1)t. • Start with W0 = (0,0,0,0)t and use the perceptron algorithm on the normalized aug- mented data given by 0,-0.5,0, 0.25 -1,-0.5, 1, 0.25 0.5, 0, -0.25, 0 0.5, 1, -0.25, -1 • After some updates you will get W = (0.5, –0.5, 0.75, –0.75)t. • So, the function is 0.5x1 – 0.5x2 + 0.75x21 – 0.75x2.

In [1]:
```
import pandas as pd
import numpy as np
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D  # noqa: F401 unused import
from matplotlib import cm
import seaborn as sns
from sklearn.model_selection import train_test_split
import pdb

w = 4
h = 4
d = 70
```

In [3]:

```
import pdb
cluster_list_pattern={}
Error_Threshold_List=[]

PATTERNS = 1000
FEATURES = 100

PATTERNS= 9
FEATURES = 3

cluster_leader_list= []

def perceptron_fit_x(x):

    print(x)

    #row_count = len(DataFrame.index)

    # initialize W0
    W0 = [[0,0,0,0]]
    #print('WWWWWW', W0)
    WT = pd.DataFrame(W0)
    print(WT.info(),'\n')
    row_count = len(x.index)

    epcoh = 0
    i = 0
    while (True):
        #print( f"\n Weight { WT[0][0]} {WT[1][0]} {WT[2][0]} {WT[3][0]}")
        dot_prod = WT.iloc[0].dot(x.iloc[i])
        print( f"\n Weight { WT[0][0]} {WT[1][0]} {WT[2][0]} {WT[3][0]}  Feature Index {i} {x[0][i]}
{x[1][i]} {x[2][i]} {x[3][i]} \n dotprod {dot_prod} \n ")
        if dot_prod <= 0:
            W0 = [WT.iloc[0].add(x.iloc[i])]
            WT = pd.DataFrame(W0)
            epcoh = 0
        else:
            epcoh = epcoh + 1

        i = i + 1
        if i >= 4 :
            i = 0
```

```
            print("-------------------------------------------------")
        if epcoh > row_count:
            break;
    return


def perceptron_fit(x_data):
    # initialize W0
    W0 = [[0,0,0,0]]

    WT = pd.DataFrame(W0)
    print(WT.info(),'\n')
    row_count = len(x_data.index)

    #x = x_data.drop(['C1','C2','C3','C4','R13','R14','R24','R23','NC1','NC2'],  axis=1)

    print( f'\n----ooooo------print-----\n')
    x = x.reset_index(drop=True)
    print(x)
    print( f'\n----ooooo------print----- {test_data} \n')



    #x = x.rename(columns = {'X1': '0', 'value': 'Income'}, inplace = False)

    epcoh = 0
    i = 0

    feature = []
    while (True):
        nWT =WT.to_numpy()
        #print( "x---------->\n", x.iloc[i])
        #print("\n WT-------->\n",WT)



        nx = x.iloc[[i]].to_numpy()


        nxf = nx.flatten()
        nWTf = nWT.flatten()

        dot_prod = np.dot(nWTf,nxf)
```

```
        #print( f"\n W [ { WT[0][0] } {WT[1][0]} {WT[2][0]} {WT[3][0]} ] X Index {i} [{x[0][i]} {x[1]
[i]} {x[2][i]} {x[3][i]} ] ==> dp {dot_prod} \n ")

        # X1   X2  X1Square  X2Square
        #print( f"\n W [ { WT[0][0] } {WT[1][0]} {WT[2][0]} {WT[3][0]} ] X Index {i} [{x['X1'][i]} {x
['X2'][i]} {x['X1Square'][i]} {x['X2Square'][i]} ] ==> dp {dot_prod} \n ")

        #print(f'\n WT = {WT}')
        if dot_prod <= 0:
            W0 = np.add(nWTf,nxf)
            WT = pd.DataFrame(W0)
            epcoh = 0
        else:
            epcoh = epcoh + 1
            #return

        i = i + 1
        #if i >= 100 :
        if i >= row_count :

            i = 0
            #print("-----------------------------------------------")
        if epcoh > row_count:
            break;


    print(f' Weight Vector ===> {WT}')

    nWT =WT.transpose().to_numpy()
    nWTf = nWT.flatten()

    print(f' Weight Vector ===> {nWTf[0]} , {nWTf[1]}, {nWTf[2]} , {nWTf[3]}')

    # Test the pattern



    a = nWTf[0]
    b = nWTf[1]
    c = nWTf[2]
```

```
    d = nWTf[3]

    for i in test_data.index:
        # EquivalentlyX∈NC1 ifax1+bx2+cx21+dx2 >0andX∈NC2 if ax1 +bx2 +cx21 +dx2 <0.
        #'X1','X2', 'X1Square', 'X2Square'
        val = test_data.iloc[i]['X1'] * a  + test_data.iloc[i]['X2'] * b + test_data.iloc[i]['X1Square'] * c + test_data.iloc[i]['X2Square'] * d

        #val = test_data.iloc[0]['X1']
        print( f"Predict Val =  {val}")

        if (val > 0):
            print("NC1 YES")
        else :
            print("NC2 NO")

        print (f'{test_data.iloc[i]}')

    return WT.transpose().to_string(index=False, header=False)



data = [[1,1],[3,3],[2,2],[3,4]]
data = [[2,2],[1,1],[3,3],[3,4]]
data = [[1, 2, 2],[2, 1 ,3],[3, 3 ,1],[6 ,1 ,1],[6 ,2, 3],[6 ,3 ,2],[6, 6 ,6],[6 ,6 ,7],[6, 6, 8]]


data = [ [0,-0.5,0, 0.25],
[-1,-0.5, 1, 0.25],
[0.5, 0, -0.25, 0 ],
[0.5, 1, -0.25, -1] ]


DM = pd.DataFrame(data)

print('DM Matrix for Data Points ')
print(f"{DM.to_string(header=None,index=False)}")
print('\n')
Threshhold = 3
perceptron_fit_x(DM)
```

```
DM Matrix for Data Points
 0.0 -0.5  0.00  0.25
-1.0 -0.5  1.00  0.25
 0.5  0.0 -0.25  0.00
 0.5  1.0 -0.25 -1.00


     0    1     2     3
0  0.0 -0.5  0.00  0.25
1 -1.0 -0.5  1.00  0.25
2  0.5  0.0 -0.25  0.00
3  0.5  1.0 -0.25 -1.00
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1 entries, 0 to 0
Data columns (total 4 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   0       1 non-null      int64
 1   1       1 non-null      int64
 2   2       1 non-null      int64
 3   3       1 non-null      int64
dtypes: int64(4)
memory usage: 160.0 bytes
None


 Weight 0 0 0 0  Feature Index 0 0.0 -0.5 0.0 0.25
 dotprod 0.0


 Weight 0.0 -0.5 0.0 0.25  Feature Index 1 -1.0 -0.5 1.0 0.25
 dotprod 0.3125


 Weight 0.0 -0.5 0.0 0.25  Feature Index 2 0.5 0.0 -0.25 0.0
 dotprod 0.0


 Weight 0.5 -0.5 -0.25 0.25  Feature Index 3 0.5 1.0 -0.25 -1.0
 dotprod -0.4375

------------------------------------------------
```

```
 Weight 1.0 0.5 -0.5 -0.75  Feature Index 0 0.0 -0.5 0.0 0.25
 dotprod -0.4375


 Weight 1.0 0.0 -0.5 -0.5  Feature Index 1 -1.0 -0.5 1.0 0.25
 dotprod -1.625


 Weight 0.0 -0.5 0.5 -0.25  Feature Index 2 0.5 0.0 -0.25 0.0
 dotprod -0.125


 Weight 0.5 -0.5 0.25 -0.25  Feature Index 3 0.5 1.0 -0.25 -1.0
 dotprod -0.0625

-----------------------------------------------

 Weight 1.0 0.5 0.0 -1.25  Feature Index 0 0.0 -0.5 0.0 0.25
 dotprod -0.5625


 Weight 1.0 0.0 0.0 -1.0  Feature Index 1 -1.0 -0.5 1.0 0.25
 dotprod -1.25


 Weight 0.0 -0.5 1.0 -0.75  Feature Index 2 0.5 0.0 -0.25 0.0
 dotprod -0.25


 Weight 0.5 -0.5 0.75 -0.75  Feature Index 3 0.5 1.0 -0.25 -1.0
 dotprod 0.3125

-----------------------------------------------

 Weight 0.5 -0.5 0.75 -0.75  Feature Index 0 0.0 -0.5 0.0 0.25
 dotprod 0.0625


 Weight 0.5 -0.5 0.75 -0.75  Feature Index 1 -1.0 -0.5 1.0 0.25
 dotprod 0.3125


 Weight 0.5 -0.5 0.75 -0.75  Feature Index 2 0.5 0.0 -0.25 0.0
```

```
dotprod 0.0625


Weight 0.5 -0.5 0.75 -0.75  Feature Index 3 0.5 1.0 -0.25 -1.0
dotprod 0.3125

-----------------------------------------------
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: