|  |  |
| --- | --- |
|  | |
| **CSE 4214 Pattern Recognition Lab**  **Assignment No**:01    **Assignment Topic**:   1. “Implementing the Perceptron algorithm for finding the weights of   a Linear Discriminant function.” | |
| **Date of Performance**: 09/06/2020  **Date of Submission**: 23/06/2022 | **Name**: Atanu Kumar Saha  **Student ID**: 17.02.04.003  **Lab Group**:A1  **Department of CSE, AUST.** |

from google.colab import files

uploaded = files.upload()

import io

import pandas as pd

df = pd.read\_csv(io.BytesIO(uploaded['train-perceptron.txt'] ), sep=" " ,  header = None, dtype = 'float64')

print(df)

import numpy as np

import matplotlib.pyplot as plt

TASK 01

train\_data = pd.read\_csv('train-perceptron.txt', sep = ' ', header = None)

train\_X = train\_data.iloc[:, 0:2]

train\_Y = train\_data.iloc[:, 2]

train\_X = np.array(train\_X)

train\_Y = np.array(train\_Y)

# To check if data is successfully retrieved

print(train\_X)

print(train\_Y)

class1\_X1, class1\_X2, class2\_X1, class2\_X2 = [], [], [], []

for i in range(train\_X.shape[0]):

  if train\_Y[i] == 1:

    class1\_X1.append(train\_X[i, 0])

    class1\_X2.append(train\_X[i, 1])

  else:

    class2\_X1.append(train\_X[i, 0])

    class2\_X2.append(train\_X[i, 1])

# To check if data is successfully retrieved

print(class1\_X1)

print(class1\_X2)

print(class2\_X1)

print(class2\_X2)

limit\_X1 = list(map(int, class1\_X1 + class2\_X1))

limit\_X2 = list(map(int, class1\_X2 + class2\_X2))

plt.figure(figsize = (10, 5))

# Training Data

plt.scatter(class1\_X1, class1\_X2, label = 'Train Class 1', color = 'red', marker = 's')

plt.scatter(class2\_X1, class2\_X2, label = 'Train Class 2', color = 'blue', marker = 'o')

# Plot Accessory

plt.xticks([i for i in range(min(limit\_X1) - 1, max(limit\_X1) + 1)])

plt.yticks([i for i in range(min(limit\_X2) - 1, max(limit\_X2) + 1)])

plt.legend(loc = 'upper center')

plt.show()

TASK 02

trainPoints = np.zeros((train\_X.shape[0], 6))

j = len(class1\_X1)

for i in range(j):

  trainPoints[i, :] = np.array([class1\_X1[i] \*\* 2, class1\_X2[i] \*\* 2,

                                class1\_X1[i] \* class1\_X2[i], class1\_X1[i],

                                class1\_X2[i], 1])

for i in range(len(class2\_X1)):

  trainPoints[i + j, :] = np.array([class2\_X1[i] \*\* 2, class2\_X2[i] \*\* 2,

                                    class2\_X1[i] \* class2\_X2[i], class2\_X1[i],

                                    class2\_X2[i], 1])

# To check if data is successfully retrieved

print(trainPoints)

for i in range(j, 0, -1):

  trainPoints[-i, :] \*= -1

# To check if data is successfully retrieved

print(trainPoints)