**PRACTICAL 5**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Name:** | Harsh Shah | **Semester:** | VII | **Division:** | 6 |
| **Roll No.:** | 21BCP359 | **Date:** | 20-08-24 | **Batch:** | G11 |
| **Aim:** | Understanding Linear Discriminant projection in Datasets. | | | | |

**Compute the Linear Discriminant projection for the following two dimensional dataset.**

* **Samples for class ω1: X1= (x1, x2) = {(6, 4), (4, 5), (3, 4), (5, 7), (6, 6)}**
* **Sample for class ω2: X2= (x1, x2) = {(11, 12), (7, 9), (10, 7), (10, 9), (12, 10)}**

**Linear Discriminant Projection**

Linear Discriminant Projection (LDP) refers to the process of projecting data onto a lower-dimensional space in a way that maximizes the separation between different classes. It is a key part of **Linear Discriminant Analysis (LDA)**, a method used in statistics, pattern recognition, and machine learning for dimensionality reduction and classification.

**Steps:**

1. Define the samples for each class
2. Compute the mean vectors
3. Compute the within-class scatter matrix SW for both classes
4. Compute the between-class scatter matrix SB
5. Compute the eigenvalues and eigenvectors of SW-1 \* SB
   1. First, compute the inverse of SW
   2. Then, compute the matrix SW-1 \* SB
   3. Compute the eigenvalues and eigenvectors
   4. Find the eigenvector corresponding to the largest eigenvalue

**Code**

import numpy as np

*# Step 1: Define the samples for each class*

X1 = np.array([[6, 4], [4, 5], [3, 4], [5, 7], [6, 6]])

X2 = np.array([[11, 12], [7, 9], [10, 7], [10, 9], [12, 10]])

*# Step 2: Compute the mean vectors*

mu1 = np.mean(X1, *axis*=0)

mu2 = np.mean(X2, *axis*=0)

*# Step 3: Compute the within-class scatter matrix SW*

S\_W = np.zeros((2, 2))

*# Compute the scatter matrix for class ω1*

for x in X1:

    diff = (x - mu1).reshape(2, 1)

    S\_W += diff @ diff.T

*# Compute the scatter matrix for class ω2*

for x in X2:

    diff = (x - mu2).reshape(2, 1)

    S\_W += diff @ diff.T

*# Step 4: Compute the between-class scatter matrix SB*

diff\_mu = (mu1 - mu2).reshape(2, 1)

S\_B = diff\_mu @ diff\_mu.T

*# Step 5: Compute the eigenvalues and eigenvectors of SW^-1 \* SB*

*# First, compute the inverse of SW*

S\_W\_inv = np.linalg.inv(S\_W)

*# Then, compute the matrix SW^-1 \* SB*

S\_W\_inv\_S\_B = S\_W\_inv @ S\_B

*# Compute the eigenvalues and eigenvectors*

eigenvalues, eigenvectors = np.linalg.eig(S\_W\_inv\_S\_B)

*# Find the eigenvector corresponding to the largest eigenvalue*

max\_eigenvalue\_index = np.argmax(eigenvalues)

linear\_discriminant\_vector = eigenvectors[:, max\_eigenvalue\_index]

**Output**



