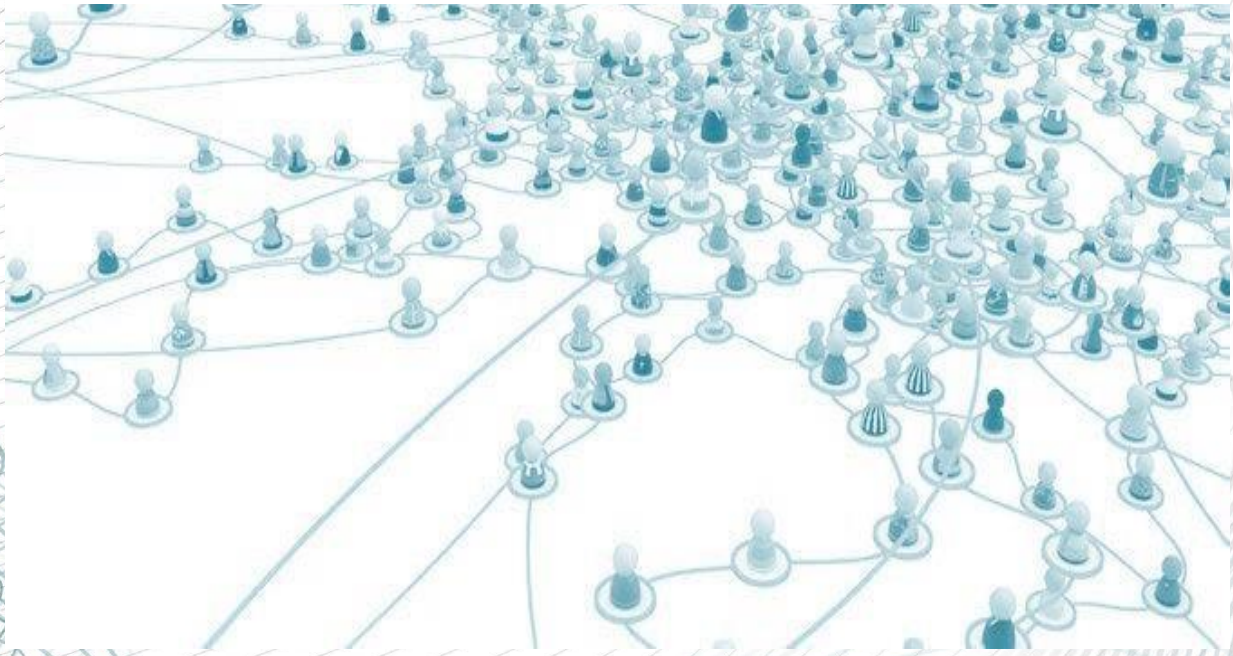
MODUL PRAKTIKUM

SD3107-Pembelajaran Mesin



Program Studi Sains Data
Fakultas Sains
Institut Teknologi Sumatera

2024

MODUL 7

LINEAR DISCRIMINANT ANALYSIS

A. Konsep Dasar

Linear Discriminant Analysis **(LDA)** adalah metode reduksi dimensi yang mempertimbangkan perbedaan antar kelas dari himpunan data. LDA sering digunakan dalam analisis data dan pembelajaran mesin untuk mengurangi jumlah fitur dengan tetap menjaga informasi yang berguna untuk klasifikasi. Langkah-langkah utama dalam LDA adalah sebagai berikut:

1. **Mendefinisikan Mean Kelas**

   LDA membutuhkan perhitungan mean (rata-rata) untuk setiap kelas. Misalkan terdapat himpunan data d-dimensi x: $\{x_1, x_2, \dots, x_n\}$, N_1 di antaranya termuat di kelas ω_1, dan N_2 lainnya berada di kelas ω_2 (Masalah dua kelas, $N_1 + N_2 = n$).

2. **Proyeksi Data ke Dimensi Lebih Rendah**

   Data x diproyeksikan ke dalam sebuah vektor $y = w^T x$, di mana w adalah vektor proyeksi yang memaksimalkan pemisahan antar kelas.

3. **Matriks Scatter Dalam Kelas (Within-Class Scatter Matrix)**

   Linear Discriminant Analysis akan mencari Proyeksi Diskriminan Linier yang akan tetap memisahkan data sesuai dengan kelas walaupun sudah mengurangi dimensinya. Langkah pertama adalah mencari matriks kovariansi dari kumpulan data pada setiap kelas (Misalkan terdapat c kelas). Kemudian untuk setiap matriks kovariansinya, akan dijumlahkan menjadi sebuah matriks baru, dinotasikan dengan S_w.

   S_w ini disebut sebagai Matriks scatter dalam kelas (Within-class scatter matrix). Matriks ini menunjukkan variabilitas dalam data untuk setiap kelas.

4. **Matriks Scatter Antar Kelas (Between-Class Scatter Matrix)**
   Selain variabilitas dalam kelas, LDA juga mempertimbangkan variabilitas antar kelas, yang ditunjukkan oleh matriks scatter antar kelas (Sb). Matriks ini membantu memaksimalkan jarak antara mean kelas-kelas yang berbeda. Definisikan rata-rata untuk kelas ke-i: $m_i = \frac{1}{N_i} \sum_{x \in \omega_i} x$

   Definisikan matriks S_b sebagai Matriks scatter antar kelas (Between-class scatter matrix), yaitu:

$$S_b = \sum_{1 \le i < j \le c} (m_i - m_j)(m_i - m_j)^t$$

di mana m_i adalah mean untuk kelas ke-i.

5. **Persamaan Nilai Eigen**

Vektor proyeksi w diperoleh dengan menyelesaikan persamaan nilai eigen:

$$S_w^{-1} S_b w = \lambda w$$

Dari persamaan ini, dipilih k nilai eigen tertinggi sebagai proyeksi data n dimensi menjadi k dimensi berdasarkan kumpulan vektor eigen yang sesuai dengan k nilai eigen tertinggi.

B. Tujuan Praktikum

I. Tujuan Instruksional Umum

Praktikum bertujuan untuk menerapkan teori *Linear Discriminant Analysis* dalam pembelajaran mesin.

II. Tujuan Instruksional Khusus

1. Mahasiswa mampu menguasai konsep dasar *Linear Discriminant Analysis*
2. Mahasiswa mampu menyelesaikan studi kasus menggunakan metode *Linear Discriminant Analysis*
3. Mahasiswa mampu menganalisis hasil dari studi kasus menggunakan metode *Linear Discriminant Analysis*

C. Dataset dan bahasa pemrograman Python

C.1 Dataset dalam praktikum

Pada modul 7 dataset yang digunakan pada praktikum ini dapat dilihat pada tabel dibawah ini. Data yang diberikan memiliki dua fitur (x_1 dan x_2) dan dua kelas (A dan B). Berikut dataset yang digunakan

x_1	x_2	Kelas
0	2	A
0	0	A
1	1	A
-1	1	A
2	2	B
4	0	B
3	-1	B
5	2	B
1	-2	A
-2	2	A
4	-2	B
3	3	B

C.2 Bahasa Pemrograman Python

Pada praktikum modul 7 ini, kita akan menggunakan bahasa pemrograman python dan beberapa library atau pustaka untuk memudahkan implementasi program. Pustaka yang akan digunakan diantaranya numpy, pandas, dan sklearn.

1. NumPy: digunakan untuk operasi matematika berbasis array dan matriks. NumPy sangat penting dalam komputasi numerik dengan Python pada praktikum ini.

2. Pandas : library untuk pengolahan data berbasis tabel (DataFrame).

3. Scikit-learn (sklearn): library untuk implementasi berbagai algoritma machine learning seperti klasifikasi, regresi, clustering, dan reduksi dimensi.

D. Implementasi Linear Discriminant Analysis

Pada praktikum ini kita akan menggunakan Google colab yang dapat diakses menggunakan tautan https://colab.research.google.com/

1) Import pustaka atau library yang akan digunakan

```
# Import pustaka yang diperlukan
import numpy as np
import pandas as pd
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
```

2) Mendefinisikan Data

```
# Mendefinisikan data fitur (X) dan label kelas (y)
data = {
    'x1': [0, 0, 1, -1, 2, 4, 3, 5, 1, -2, 4, 3],
    'x2': [2, 0, 1, 1, 2, 0, -1, 2, -2, 2, -2, 3],
    'kelas': ['A', 'A', 'A', 'A', 'B', 'B', 'B', 'B', 'A', 'A', 'B', 'B']
}
df = pd.DataFrame(data)

X = df[['x1', 'x2']].values
y = df['kelas'].values
```

Kita menyusun data fitur (X) dan label kelas (y). X adalah matriks data dua dimensi, dan y adalah array yang berisi label kelas untuk setiap data.

3) Menghitung Mean (Rata-Rata) untuk Setiap Kelas

Pisahkan data berdasarkan kelas dan hitung rata-rata untuk kelas A dan B.

```
# Menghitung mean untuk setiap kelas
mean_A = X[y == 'A'].mean(axis=0)
mean_B = X[y == 'B'].mean(axis=0)

print("Mean Kelas A:", mean_A)
print("Mean Kelas B:", mean_B)
```

4) Menghitung Matriks Scatter Dalam Kelas (Within-Class Scatter Matrix, S_w)

Hitung matriks scatter dalam kelas dengan menggunakan selisih antara data dan rata-rata kelasnya.

```
# Menghitung matriks scatter dalam kelas (Sw)
Sw = np.zeros((2, 2))
for i in range(len(X)):
    x = X[i].reshape(2, 1)
    if y[i] == 'A':
        mean_vec = mean_A.reshape(2, 1)
    else:
        mean_vec = mean_B.reshape(2, 1)
    Sw += (x - mean_vec).dot((x - mean_vec).T)

print("Within-Class Scatter Matrix (Sw):\n", Sw)
```

5) Menghitung Matriks Scatter Antar Kelas (Between-Class Scatter Matrix, S_b)

Kita menghitung matriks scatter antar kelas (S_b) dengan mengambil selisih antara rata-rata kelas A dan kelas B. Matriks ini memperkuat pemisahan antara kedua kelas.

```
# Menghitung matriks scatter antar kelas (Sb)
mean_total = np.mean(X, axis=0).reshape(2, 1)
mean_A = mean_A.reshape(2, 1)
mean_B = mean_B.reshape(2, 1)
Sb = 5 * (mean_A - mean_total).dot((mean_A - mean_total).T) + \
     5 * (mean_B - mean_total).dot((mean_B - mean_total).T)

print("Between-Class Scatter Matrix (Sb):\n", Sb)
```

6) Menyelesaikan Persamaan Nilai Eigen

Cari nilai dan vektor eigen dari matriks $S_w^{-1}S_b$

```
# Menyelesaikan persamaan nilai eigen
eig_vals, eig_vecs = np.linalg.eig(np.linalg.inv(Sw).dot(Sb))
print("Nilai Eigen:\n", eig_vals)
print("Vektor Eigen:\n", eig_vecs)
```

7) Memilih Vektor Eigen dengan Nilai Eigen Tertinggi

Pilih vektor eigen yang sesuai dengan nilai eigen tertinggi sebagai vektor proyeksi.

```
# Memilih vektor eigen dengan nilai eigen tertinggi
idx = np.argmax(eig_vals)
w = eig_vecs[:, idx]
print("Vektor Proyeksi (w):\n", w)
```

8) Memproyeksikan Data ke Dimensi yang Lebih Rendah

```
# Memproyeksikan data ke dimensi yang lebih rendah
X_lda = X.dot(w)
print("Data setelah proyeksi LDA:\n", X_lda)
```

9) Implementasi Menggunakan Scikit-Learn

```
# Implementasi LDA menggunakan Scikit-Learn untuk perbandingan
lda = LinearDiscriminantAnalysis(n_components=1)
X_lda_sklearn = lda.fit_transform(X, y)
print("Data setelah proyeksi LDA (Scikit-Learn):\n", X_lda_sklearn)
```