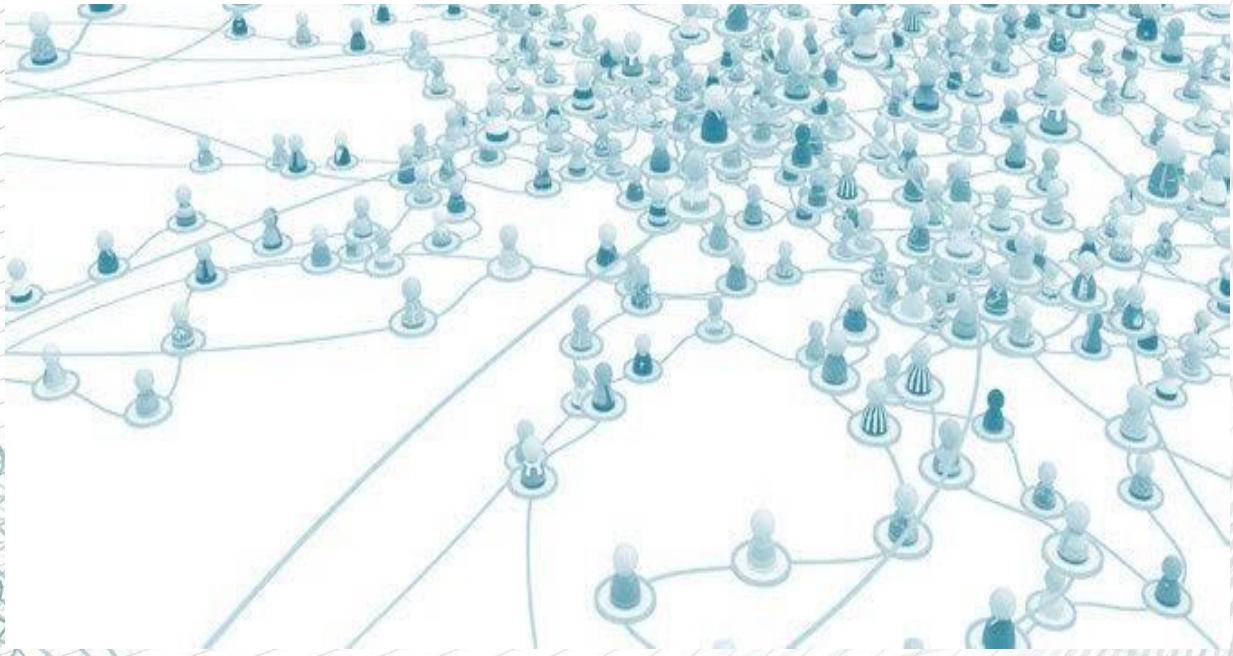
MODUL PRAKTIKUM

SD3107-Pembelajaran Mesin



Program Studi Sains Data
Fakultas Sains
Institut Teknologi Sumatera

2024

MODUL 6

Principal Component Analysis (PCA)

A. Konsep Dasar

Pada sebuah model klasifikasi atau regresi, data yang digunakan terkadang memiliki fitur yang banyak sehingga menyebabkan dimensi datanya juga tinggi. Beberapa akibat dari tingginya dimensi data diantaranya yaitu penurunan performa akibat adanya beberapa fitur yang tidak relevan, sulitnya melakukan visualisasi karena data memiliki dimensi yang tinggi (lebih dari dua atau tiga fitur), serta komputasi yang menjadi lebih kompleks.

Untuk menangani masalah tersebut, dilakukan teknik reduksi dimensi. Reduksi dimensi merupakan teknik mengurangi jumlah fitur atau dimensi dengan tetap mempertahankan informasi aslinya sebanyak mungkin dengan hanya mengubah sedikit integritas atau informasi asli dari datanya. Terdapat dua cara dalam mereduksi dimensi, yaitu seleksi fitur (*feature selection)* dan penggabungan fitur (*feature extraction)*.

Seleksi fitur merupakan teknik reduksi dimensi dengan memilih subset dari fitur asli yang memiliki relevansi tinggi dengan masalah yang dihadapi, sedangkan teknik penggabungan fitur merupakan teknik reduksi dimensi dengan membuat sebuah data baru yang berasal dari kumpulan fitur-fitur data sebelumnya. Contohnya adalah data perumahan dengan fitur (ukuran, banyak kamar tidur, banyak kamar mandi, sekolah terdekat, rumah sakit terdekat) dapat diubah menjadi (Fitur ukuran, Fitur lokasi). Ada beberapa jenis teknik penggabungan fitur, diantaranya adalah *Principal Component Analysis* (PCA), *Linear Discriminant Analysis* (LDA), dan *t-distributed Stochastic Neighbor Estimated* (t-SNE).

PCA merupakan salah teknik reduksi dimensi yang sering digunakan. PCA mereduksi suatu set variabel yang berdimensi tinggi menjadi lebih rendah, namun masih mengandung sebagian besar informasi dari data awal. Proses dari PCA ini adalah untuk mendapatkan beberapa vektor eigen (utama) dan nilai eigen (utama) dari matriks kovariansi dari sekumpulan data yang berdimensi **d** dimana matriks kovariansi yang diperoleh haruslah berukuran **d x d**. Algoritma dari metode PCA adalah sebagai berikut:

1. Standarisasi data dengan terlebih dahulu menghitung nilai *mean* atau rata-rata dan standar deviasi dari tiap fitur. Hasil tersebut kemudian digunakan untuk mencari data standarisasi dengan persamaan berikut:

$$Z = \frac{x - \overline{x}}{\sigma}$$

Dengan Z adalah data hasil standarisasi, x adalah fitur, $\overline{x}$ adalah rata-rata dari fitur, serta σ adalah standar deviasi.

2. Hitung nilai matriks kovarian dengan persamaan berikut ini.

$$Cov = \frac{1}{n-1} Z^T . Z$$

3. Hitung nilai eigen dan vektor eigen dari matriks kovarian. Hubungan antara nilai eigen, vektor eigen, dan matriks kovarian dapat dilihat dari persamaan berikut.

$$Cx = \lambda x$$

Dengan $\boldsymbol{C}$ adalah matriks kovarian, $\boldsymbol{\lambda}$ adalah nilai eigen dan $\boldsymbol{x}$ adalah vektor eigen yang berkoresponden dengan $\boldsymbol{\lambda}$, nilai eigen dapat dicari dengan membuat persamaan karakteristik seperti berikut ini.

$$\det(C - \lambda I) = 0$$

Dari hasil persamaan karakterisitk di atas, kemudian dicari akar-akar polinomialnya. Akar-akar tersebutlah yang merupakan nilai eigen dari matriks $\boldsymbol{C}$. Selanjutnya pada setiap nilai eigen, vektor eigen dapat dicari menggunakan persamaan berikut.

$$(C - \lambda I)x = 0$$

4. Pilih komponen utama berdasarkan beberapa eigen terbesar.
5. Konstruksi matriks proyeksi berdasarkan kumpulan vektor eigen. Misal sudah dipilih nilai eigen terbesar sebagai komponen utamanya, maka vektor eigen dari nilai eigen tersebut yang akan digunakan pada tahapan selanutnya.
6. Transformasi data dengan vektor eigen dari nilai eigen yang terpilih melalui persamaan berikut.

$$Y = Z.W$$

Dimana *Y* adalah hasil dari reduksi dimensi, *Z* matriks dari data, dan *W* adalah vektor eigen yang dipilih.

.

B. Tujuan Praktikum

I. Tujuan Instruksional Umum

Praktikum bertujuan untuk menerapkan teori *Principal Component Analysis* (PCA) untuk mereduksi dimensi data yang tinggi.

II. Tujuan Instruksional Khusus

1. Mahasiswa mampu menguasai konsep dasar *Principal Component Analysis* (PCA).
2. Mahasiswa mampu menyelesaikan studi kasus mereduksi dimensi fitur data yang tinggi menggunakan PCA.
3. Mahasiswa mampu menganalisis hasil dari studi kasus menggunakan metode PCA.

C. Dataset dan Bahasa Pemrograman Python

C.1 Dataset

Dataset yang digunakan pada praktikum ini dapat dilihat pada tabel berikut.

	Fitur 1	Fitur 2	Fitur 3	Fitur 4
0	2.5	2.4	3.5	3.6
1	0.5	0.7	1.2	1.3
2	2.2	2.9	3.0	3.4
3	1.9	2.2	2.5	2.7
4	3.1	3.6	4.1	4.0
5	2.3	2.7	3.1	3.0
6	2.0	1.6	2.3	2.4
7	1.0	1.1	1.5	1.6
8	1.5	1.6	2.0	2.1
9	1.1	0.9	1.3	1.4

C.2 Bahasa Pemrograman Python

Pada praktikum modul 3 ini, kita akan menggunakan bahasa pemrograman python dan beberapa library atau pustaka untuk memudahkan implementasi program. Pustaka yang akan digunakan diantaranya numpy, sklearn, dan matplotlib untuk visualisasi data.

a. NumPy: digunakan untuk operasi matematika berbasis array dan matriks. NumPy sangat penting dalam komputasi numerik dengan Python pada praktikum ini.
b. Scikit-learn (sklearn): library untuk implementasi berbagai algoritma machine learning seperti klasifikasi, regresi, clustering, dan reduksi dimensi. Pada praktikum ini, kita akan menggunakan pustaka sklearn.svm.
c. Matplotlib dan Seaborn: digunakan untuk proses visualisasi data. Matplotlib dapat digunakan untuk membuat visualisasi data dalam bentuk grafik, bagan, peta, dan lainnya. Pustaka lain yang dapat digunakan untuk visualisasi data agar lebih estetik adalah Seaborn.

D. Implementasi Principal Component Analysis untuk Reduksi Dimensi

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

D.1 PCA Menggunakan Numpy

1) Import pustaka atau library yang akan digunakan

```
# import library yang akan digunakan
import numpy as np
```

2) Masukkan dataset yang akan digunakan.

```
# import data yang akan direduksi menggunakan numpy
data = np.array([
    [2.5, 2.4, 3.5, 3.6],
    [0.5, 0.7, 1.2, 1.3],
    [2.2, 2.9, 3.0, 3.4],
    [1.9, 2.2, 2.5, 2.7],
    [3.1, 3.6, 4.1, 4.0],
    [2.3, 2.7, 3.1, 3.0],
    [2.0, 1.6, 2.3, 2.4],
    [1.0, 1.1, 1.5, 1.6],
    [1.5, 1.6, 2.0, 2.1],
    [1.1, 0.9, 1.3, 1.4]
])
```

3) Lakukan standarisasi data.

```
# Standarisasi data
mean = np.mean(data, axis=0)
std_dev = np.std(data, axis=0)
Z = (data - mean) / std_dev
```

4) Hitung matriks kovarian dari Z.

```
# Hitung matriks kovarians
cov_matrix = np.cov(Z, rowvar=False)
```

5) Hitung nilai eigen dan vektor eigen dari matriks kovarian `cov_matrix`

```
# Hitung nilai dan vektor eigen
eigenvalues, eigenvectors = np.linalg.eig(cov_matrix)
```

6) Hitung proporsi variansi dari nilai eigen

```
# Hitung proporsi variansi dari nilai eigen
total_variance = np.sum(eigenvalues)
explained_variance_ratio = eigenvalues / total_variance
```

7) Hitung kontribusi variansi dari nilai eigen

```
# Hitung kontribusi variansi dari nilai eigen
cumulative_variance = np.cumsum(explained_variance_ratio)
```

```
# Tampilkan hasil
print("Nilai Eigen:", eigenvalues)
print("Proporsi Variansi dari setiap komponen:", explained_variance_ratio)
print("Kontribusi Variansi Kumulatif:", cumulative_variance)
```

8) Tentukan ambang batas atau *threshold* dalam pemilihan nilai eigen

```
# Tentukan ambang batas pemilihan eigen
# Biasanya diatas 70%
threshold = 0.90
n_components = np.argmax(cumulative_variance >= threshold) + 1
print(n_components) # tampilkan jumlah komponen
```

```
1
```

9) Urutkan nilai dan vektor eigen

```
# Urutkan nilai dan vektor eigen
sorted_indices = np.argsort(eigenvalues)[::-1]  # Urutkan dari besar ke kecil
sorted_eigenvalues = eigenvalues[sorted_indices]
sorted_eigenvectors = eigenvectors[:, sorted_indices]
```

10) Ambil komponen sesuai `n_components` sebelumnya.

```
# Ambil komponen utama sesuai n_components
n_components = 1
W = sorted_eigenvectors[:, :n_components]
```

11) Lakukan transformasi data

```
# Transformasi data
Y = Z @ W  # Z adalah data standar, W adalah matriks vektor eigen
```

12) Tampilkan hasil reduksi data PCA

```
Data setelah PCA:
[[-1.85364122]
 [ 2.95159575]
 [-1.54579331]
 [-0.29769242]
 [-3.46574526]
 [-1.33423136]
 [ 0.24222221]
 [ 2.066633  ]
 [ 0.90644968]
 [ 2.33020293]]
```

D.2 PCA Menggunakan Sklearn

1) Import pustaka atau library yang akan digunakan.

```
# import library yang akan digunakan
import numpy as np
```

2) Masukkan data yang akan direduksi

```
# import data yang digunakan
X = np.array([
    [2.5, 2.4, 3.5, 3.6],
    [0.5, 0.7, 1.2, 1.3],
    [2.2, 2.9, 3.0, 3.4],
    [1.9, 2.2, 2.5, 2.7],
    [3.1, 3.6, 4.1, 4.0],
    [2.3, 2.7, 3.1, 3.0],
    [2.0, 1.6, 2.3, 2.4],
    [1.0, 1.1, 1.5, 1.6],
    [1.5, 1.6, 2.0, 2.1],
    [1.1, 0.9, 1.3, 1.4]
])
```

3) Lakukan standarisasi data menggunakan StandardScaler dari library sklearn.

```
# standarisasi data
scaler = StandardScaler()
X_scaled = scaler.fit_transform(data)
```

4) Lakukan reduksi dimensi menggunakan PCA dari library sklearn

```
# reduksi dimensi menggunakan PCA
# dipilih satu komponen sesuai n_components pada bg. 1
pca = PCA(n_components=1)
X_pca = pca.fit_transform(X_scaled)
```

5) Tampilkan hasil reduksi PCA menggunakan sklearn

```
# print hasil reduksi
print("Data setelah PCA:")
print(X_pca)
```

```
Data direduksi menggunakan PCA:
[[ 1.85364122]
 [-2.95159575]
 [ 1.54579331]
 [ 0.29769242]
 [ 3.46574526]
 [ 1.33423136]
 [-0.24222221]
 [-2.066633  ]
 [-0.90644968]
 [-2.33020293]]
```

Catatan. Untuk melihat lebih jelas hasil dari tiap proses perhitungan, dapat ditampilkan melalui perintah **`print(nama_variabel).`**