BAB 13

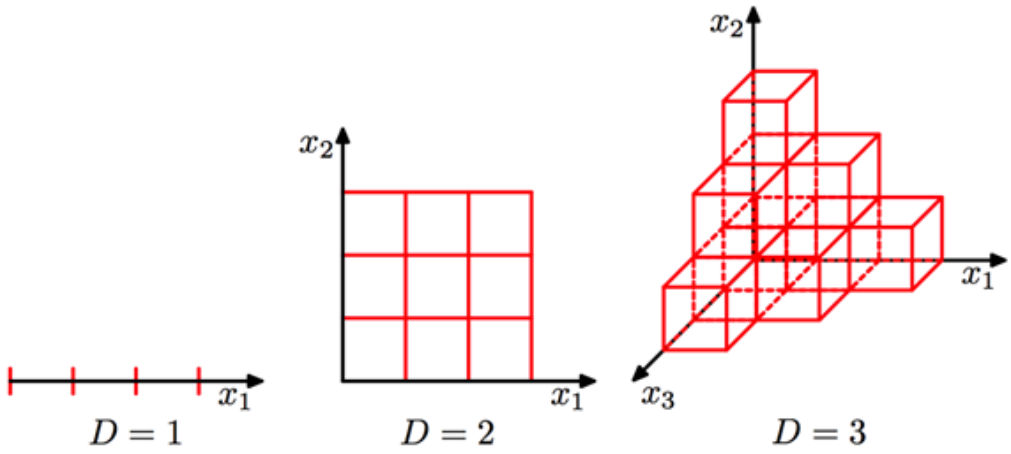
Principal Component Analysis (PCA)

Principal Component Analysis (PCA) termasuk kedalam teknik *unsupervised learning* teknik statistik non-parametrik terutama digunakan untuk pengurangan dimensi (*Dimensionality Reduction*) dalam machine learning.

1. Reduksi / Pengurangan Dimensi

Secara singkat reduksi dimensi merupakan Teknik untuk mengurangi variable input (fitur). pada sekumpulan dataset. Dataset seringkali direpresentasikan sebagai kolom x baris. Dimana kolom merupakan variable input (fitur) dan baris merupakan jumlah instance/data. Jika direpresentasikan variable input (kolom) yang mewakili dimensi dari pada ruang fitur n-dimensi dan baris data sebagai titik dalam ruang tersebut. Ini adalah interpretasi geometris dari kumpulan data. Semakin banyak jumlah variable input maka akan menghasilkan dimensi yang semakin tinggi.

Dimensi yang tinggi berarti bahwa dataset yang memiliki sejumlah fitur yang banyak. Permasalahan yang terjadi ketika melakukan pelatihan pada data yang memiliki dimensi yang tinggi adalah pemrosesan yang mahal dalam artian dibutuhkannya memori yang lebih banyak dan juga waktu pemrosesan yang meningkat sesuai dengan meningkatnya dimensi fitur. Selain itu masalah yang muncul adalah mungkin saja menjadi lebih sulit untuk mendeteksi kemiripan data di ruang dimensi tinggi karena data sparse. Masalah utama terkait dengan dimensi tinggi pada machine learning adalah model *overfitting*, yang mengurangi kemampuan untuk model dalam melakukan prediksi diluar data training. Richard Bellman menggambarkan fenomena ini pada tahun 1961 sebagai *Curse of Dimensionality* di mana "Banyak algoritma yang bekerja dengan baik dalam dimensi rendah akan tetapi menjadi tidak dapat diselesaikan dengan baik ketika inputnya berdimensi tinggi." Terdapat beberapa metode yang dapat digunakan untuk mengurangi dimensi fitur yang tinggi. Salah satu algoritma yang popular adalah *Principal Component Analysis (PCA).*



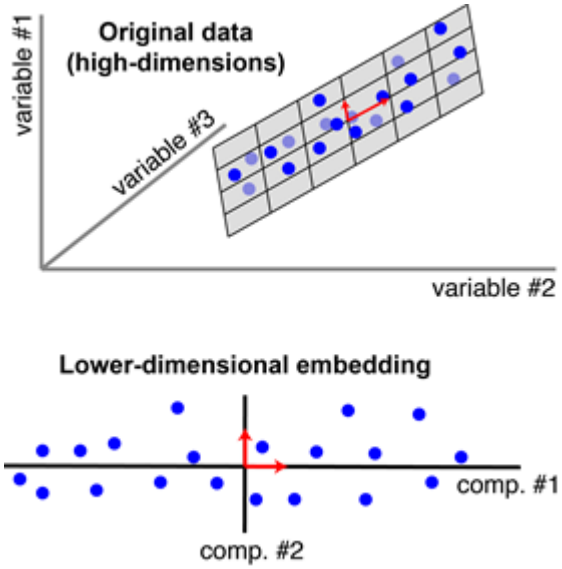
Gambar 13. 1 Ilustrasi Dimensi

2. Principal Component Analysis (PCA)

Menurut buku Biship, C. M. (2007). Pattern recognition and machine learning (information science and statistics). PCA didefinisikan "PCA can be defined as the orthogonal projection of the data onto a lower dimensional linear space, known as the principal subspace, such that the variance of the projected data is maximized". PCA merupakan salah satu metode reduksi dimensi fitur yang digunakan untuk mengurangi dimensi variable input dari sekumpulan kumpulan data yang memiliki sejumlah variable input yang besar, dengan mengubah kumpulan variabel yang besar menjadi variabel yang lebih kecil yang masih berisi sebagian besar informasi dalam kumpulan besar. PCA adalah sebuah teknik untuk membangun variable-variable baru yang merupakan kombinasi linear dari variable-variable asli. Jumlah maximum dari variable-variable baru ini akan sama dengan jumlah dari variable lama, dan variable-variable baru ini tidak saling berkorelasi satu sama lain.

PCA digunakan untuk :

a. Mengurangi satu set variable yang saling berkorelasi dan berdimensi tinggi menjadi satu set variable yang berdimensi lebih rendah yang biasanya disebut sebagai Principal Component. Variable-variable yang dihasilkan oleh PCA tidak saling berkorelasi satu sama lain. Pada gambar 13.2 dapat dilihat bahwa Setiap titik biru merupakan titik data, dan setiap Principal Component mengurangi tiga dimensi menjadi dua dimensi. Algoritma menemukan sepasang vektor ortogonal (panah merah) yang mendefinisikan ruang dimensi lebih rendah (bidang abu-abu) untuk menangkap varian sebanyak mungkin dari kumpulan data asli.



Gambar 13. 2 Ilustrasi principal component

b. Mengurangi masalah yang disebabkan oleh *Curse of Dimensionality*
c. Mengecilkan ukuran data dan meminimalkan jumlah informasi yang hilang sebelum digunakan oleh estimator lain.
d. Melakukan visualisasi kumpulan data dimensi tinggi dalam dua dimensi.

Contoh penggunaan PCA yang dilakukan oleh penelitian Martono, G. H., Adji, T. B., & Setiawan, N. A. (2012). Penggunaan Metode Analisa Komponen Utama (PCA) untuk Mereduksi Faktor-Faktor yang Mempengaruhi Penyakit Jantung Koroner.

Pada penelitian tersebut terdapat 13 fitur yang digunakan untuk mendeteksi penyakit jantung coroner. Fitur-fitur yang digunakan seperti pada table 13.1. Dari 13 variabel yang ada pada dataset, setelah dilakukan reduksi dengan menggunakan metodologi PCA diperoleh empat variabel baru yang menentukan penyakit jantung coroner yang terlihat pada table 13.2 .

Tabel 13. 1 Contoh Fitur

Variabel	**Keterangan**
Age	Age in years
Sex	1 : male, 0 : female
Cp	Chest Pain 1: typical angina; 2 : atypical angina; 3 : non anginal pain; 4 : asymptomatic
Trestbps	resting blood pressure (in mm Hg on admission to the hospital)
Chol	serum cholestoral in mg/dl
Fbs	(fasting blood sugar > 120 mg/dl) (1 = true; 0 = false)
Restecg	resting electrocardiographic results 0:normal; 1: having ST-T wave abnormality (T wave inversions and/or ST elevation or depression of > 0.05 mV); 2: showing probable or definite left ventricular hypertrophy by Estes' criteria
Thalach	maximum heart rate achieved
Exang	exercise induced angina (1 = yes; 0 = no)
Oldpeak	ST depression induced by exercise relative to rest
Slope	the slope of the peak exercise ST segment. 1:upsloping; 2:flat; 3:downsloping
Ca	number of major vessels (0-3) colored by flourosopy
Thal	3 = normal; 6 = fixed defect; 7 = reversable defect

Tabel 13. 2 Variabel baru dari metode PCA

Principal Component (PC)	**Nama Variabel**	**Faktor Loading**	**Varian yang dijelaskan**
PC 1 : Peredaran Darah	Cp Thalach Exang	0,784 -0,583 0,617	23,695
PC 2 : Tekanan Darah	Age Tresbps	0,661 0,613	12,349
PC 3 : Denyut Jantung	Oldpeak Slope	0,762 0,867	9,604
PC 4 : Jenis Kelamin	Sex Thal	0,831 0,589	8,516

3. Tahapan Algoritma **Principal Component Analysis (PCA)**

Tahapan – tahapan dari **Principal Component Analysis (PCA)** adalah sebagai berikut:

a. Lakukan data standardisasi (lihat bab 2)

PCA digunakan untuk mengidentifikasi komponen dengan varians maksimum, dan kontribusi setiap variabel untuk komponen didasarkan pada besarnya varians. Cara terbaik sebelum menerapkan PCA terhadap dimensi fitur yang tinggi sebaiknya standardisasi data menggunakan *z-score(feature scaling)* karena data yang memiliki skala dengan unit pengukuran yang berbeda dapat mendistorsi perbandingan relatif varians di seluruh fitur.

$$standardized\ value\ \ or\ \ z\text{-}score \longrightarrow \frac{x - \overline{x}}{\sigma}$$

b. Hitung rata-rata (mean) setiap kolom
Untuk menghitung rata-rata gunakan persamaan berikut :

$$\mu = \frac{1}{p}\sum_{k=1}^{p} x_k$$

Dimana p adalah jumlah data, x adalah fitur matrix
Contoh :
Jika terdapat data

$$A = \begin{matrix} a_{11} & a_{12} \\ a_{21} & a_{22} \\ a_{31} & a_{32} \end{matrix}$$

Maka mean :

$$M = \begin{bmatrix} (a_{11} + a_{21} + a_{31})/3 \\ (a_{12} + a_{22} + a_{32})/3 \end{bmatrix}$$

Selanjutnya, perlu dipusatkan nilai di setiap kolom dengan mengurangi nilai kolom rata-rata.

$$C = A - M$$

c. Hitung matriks kovarian
Untuk menghitung matriks kovarian gunakan persamaan berikut :

$$cov(X,Y) = \frac{\sum_{i=1}^{n}(X_i - \bar{x})(Y_i - \bar{y})}{n-1}$$

Matriks kovarians adalah matriks simetris p × p (di mana p adalah jumlah dimensi) Misalnya, untuk kumpulan data 3 dimensi dengan 3 variabel x1, x2, dan x3, matriks kovariansnya adalah matriks 3x3 dari:

$$C = \begin{bmatrix} cov(x_1,x_1) & cov(x_1,x_2) & cov(x_1,x_3) \\ cov(x_2,x_1) & cov(x_2,x_2) & cov(x_2,x_3) \\ cov(x_3,x_1) & cov(x_3,x_2) & cov(x_3,x_3) \end{bmatrix}$$

Kovarian memiliki hukum komutatif sehingga (Cov(a,b)=Cov(b,a))

- jika nilai kovarian positif maka: kedua variabel naik atau turun bersama-sama (berkorelasi)
- jika nilai kovarian negatif maka: salah satu variabel bertambah ketika yang lain berkurang (berkorelasi terbalik)

contoh perhitungan covarian, terdapat matrix berikut :

X	Y	Z
Height	Score	Age
64.0	580.0	29.0
66.0	570.0	33.0
68.0	590.0	37.0
69.0	660.0	46.0
73.0	600.0	55.0

Mean(M) = 68.0 untuk X
600.0 untuk Y
40.0 untuk Z

n=5

Covar(XY) =[(64-68.0)*(580-600.0) + (66-68.0)*(570-600.0) + (68-68.0)*(590-600.0) + (69-68.0)*(660-600.0) + (73-68.0)*(600-600.0)] / (5-1) =
=[80.0 + 60.0 + 0 + 60.0 + 0] / 4
=200 / 4
= 50.0

Sehinga didapatkan kovarian dari matrix adalah:

	X	Y	Z
X	11.50	50.00	34.75
Y	50.00	1250.00	205.00
Z	34.75	205.00	110.00

d. Mencari eigendecomposition atau menggunakan Single Value Decomposition. Hitung nilai eigen value dan eigen vector untuk setiap matrix kovarian

1) Untuk menghitung nilai eigen gunakan persamaan berikut :

$$(A - \lambda . I) = 0$$

Contoh

$$\mathbf{A} = \begin{bmatrix} 0 & 1 \\ -2 & -3 \end{bmatrix}$$

Sehingga

$$\left|\mathbf{A} - \lambda \cdot \mathbf{I}\right| = \left|\begin{bmatrix} 0 & 1 \\ -2 & -3 \end{bmatrix} - \begin{bmatrix} \lambda & 0 \\ 0 & \lambda \end{bmatrix}\right| = 0$$

$$\left|\begin{bmatrix} -\lambda & 1 \\ -2 & -3-\lambda \end{bmatrix}\right| = \lambda^2 + 3\lambda + 2 = 0$$

Didapatkan lambda λ1=-1, λ2=-2

2) Hitung vektor eigen

$$[A - \lambda . I][v] = [0]$$

Dari contoh perhitungan point d(1) didapatkan :

- Dengan menggunakan λ1=-1

$$\mathbf{A} \cdot \mathbf{v}_1 = \lambda_1 \cdot \mathbf{v}_1$$

$$(\mathbf{A} - \lambda_1) \cdot \mathbf{v}_1 = 0$$

$$\begin{bmatrix} -\lambda_1 & 1 \\ -2 & -3-\lambda_1 \end{bmatrix} \cdot \mathbf{v}_1 = 0$$

$$\begin{bmatrix} 1 & 1 \\ -2 & -2 \end{bmatrix} \cdot \mathbf{v}_1 = \begin{bmatrix} 1 & 1 \\ -2 & -2 \end{bmatrix} \cdot \begin{bmatrix} v_{1,1} \\ v_{1,2} \end{bmatrix} = 0$$

Berdasarkan matrik diatas didapatkan :

Dari baris pertama

$$v_{1,1} + v_{1,2} = 0, \quad \text{so}$$

$$v_{1,1} = -v_{1,2}$$

Dari baris kedua

$$-2 \cdot v_{1,1} + -2 \cdot v_{1,2} = 0, \quad \text{so again}$$

$$v_{1,1} = -v_{1,2}$$

$$\mathbf{v}_1 = k_1 \begin{bmatrix} +1 \\ -1 \end{bmatrix}$$

- Dengan menggunakan λ2=-2

$$\mathbf{A} \cdot \mathbf{v}_2 = \lambda_2 \cdot \mathbf{v}_2$$

$$(\mathbf{A} - \lambda_2) \cdot \mathbf{v}_2 = \begin{bmatrix} -\lambda_2 & 1 \\ -2 & -3-\lambda_2 \end{bmatrix} \cdot \mathbf{v}_2 = \begin{bmatrix} 2 & 1 \\ -2 & -1 \end{bmatrix} \cdot \begin{bmatrix} v_{2,1} \\ v_{2,2} \end{bmatrix} = 0 \quad \text{so}$$

$$2 \cdot v_{2,1} + 1 \cdot v_{2,2} = 0 \quad (\text{or from bottom line: } -2 \cdot v_{2,1} - 1 \cdot v_{2,2} = 0)$$

$$2 \cdot v_{2,1} = -v_{2,2}$$

$$\mathbf{v}_2 = k_2 \begin{bmatrix} +1 \\ -2 \end{bmatrix}$$

Eigendecomposition dari suatu matriks adalah jenis dekomposisi yang melibatkan penguraian matriks menjadi satu set vektor eigen dan nilai eigen. Pada Bahasa pemrograman python Eigendecomposition dapat dihitung dalam NumPy menggunakan fungsi eig().

e. Tentukan variabel baru (principal component)
Urutkan nilai eigen dalam urutan terbesar ke kecil dan pilih k eigen vektor teratas yang sesuai dengan k nilai eigen terbesar (k akan menjadi jumlah dimensi dari subruang fitur baru dengan syarat k ≤ d, d adalah jumlah dimensi asli). Sejumlah k eigen vector ini disebut principal component.

f. Bangun matriks proyeksi P dari k vektor Eigen yang dipilih.
P = B^T.A

Dimana A adalah original data (matrix awal) dan B^T adalah transpose dari principal component dan P adalah proyeksi matrix dari A

4. Kode program menggunakan Python

A. Principal Component Analysis Secara Manual

```
1 from numpy import array
2 from numpy import mean
3 from numpy import cov
4 from numpy.linalg import eig
5 # define a matrix
6 A = array([[1, 2], [3, 4], [5, 6]])
7 print(A)
8 # calculate the mean of each column
9 M = mean(A.T, axis=1)
10 print(M)
11 # center columns by subtracting column means
12 C = A - M
13 print(C)
14 # calculate covariance matrix of centered matrix
15 V = cov(C.T)
16 print(V)
17 # eigendecomposition of covariance matrix
18 values, vectors = eig(V)
19 print(vectors)
20 print(values)
21 # project data
22 P = vectors.T.dot(C.T)
23 print(P.T)
```

Keterangan

- baris ke 6 merupakan matrix awal yang akan diproyeksi

```
[[1 2]
 [3 4]
 [5 6]]
```

- baris 9 merupakan proses mencari rata-rata dari matrik

```
[3. 4.]
```

- baris 12 merupakan proses mencari pusat kolom

```
[[-2. -2.]
 [ 0.  0.]
 [ 2.  2.]]
```

- baris 15 merupakan proses untuk menghitung matrik kovarian

```
[[4. 4.]
 [4. 4.]]
```

- baris 18 merupakan proses eigendecomposisition dari eigen value dan eigen vector

```
[[ 0.70710678 -0.70710678]
 [ 0.70710678  0.70710678]]
[8. 0.]
```

- baris 22 akan menghasilkan proyeksi data

```
[[-2.82842712  0.        ]
 [ 0.          0.        ]
 [ 2.82842712  0.        ]]
```

B. Principal Component Analysis Menggunakan SKlearn

```
1 # Principal Component Analysis
2 from numpy import array
3 from sklearn.decomposition import PCA
4 # define a matrix
5 A = array([[1, 2], [3, 4], [5, 6]])
6 print(A)
7 # create the PCA instance
8 pca = PCA(2)
9 # fit on data
10 pca.fit(A)
11 # access values and vectors
12 print(pca.components_)
13 print(pca.explained_variance_)
14 # transform data
15 B = pca.transform(A)
16 print(B)
```

PCA dapat dihitung menggunakan library yang telah disediakan oleh sklean dengan menggunakan class PCA(). Manfaat dari pendekatan ini adalah bahwa setelah proyeksi dihitung, dapat diterapkan ke data baru lagi dan lagi dengan cukup mudah. Saat membuat class dengan memanggil class PCA(), jumlah komponen dapat ditentukan sebagai parameter. Fitting data dilakukan dengan memanggil fungsi fit(), dan kemudian kumpulan data asli atau data lain dapat diproyeksikan ke dalam subruang dengan jumlah dimensi yang dipilih dengan memanggil fungsi transform().

Setelah fit, nilai eigen dan komponen utama dapat diakses pada kelas PCA melalui atribut explain_variance_ dan components_.

Output kode program:

```
[[1 2]
 [3 4]
 [5 6]]
[[ 0.70710678  0.70710678]
 [-0.70710678  0.70710678]]
[8. 0.]
[[-2.82842712e+00 -2.22044605e-16]
 [ 0.00000000e+00  0.00000000e+00]
 [ 2.82842712e+00  2.22044605e-16]]
```

C. Principal Component Analysis Menggunakan SKlearn diterapkan pada data heart-disease.

Download data pada link berikut : https://www.kaggle.com/ronitf/heart-disease-uci

Jalankan kode program berikut :

```
1 import pandas as pd
2 import io
3
4 from google.colab import files
5 uplod=files.upload()
```

Materi

```
Choose Files heart.csv
• heart.csv(application/vnd.ms-excel) - 11328 bytes, last modified: 6/11/2021 - 100% done
Saving heart.csv to heart.csv
```

```
[17] 1 import pandas as pd
     2 patients_data = pd.read_csv(io.BytesIO(uplod['heart.csv']),delimiter=',')
     3 print(patients_data.head())
```

```
   age  sex  cp  trestbps  chol  fbs  ...  exang  oldpeak  slope  ca  thal  target
0   63    1   3       145   233    1  ...      0      2.3      0   0     1       1
1   37    1   2       130   250    0  ...      0      3.5      0   0     2       1
2   41    0   1       130   204    0  ...      0      1.4      2   0
3   56    1   1       120   236    0  ...      0      0.8      2   0
4   57    0   0       120   354    0  ...      1      0.6      2   0

[5 rows x 14 columns]
```

Materi

```
[28] 1 from sklearn.model_selection import train_test_split # Import train_test_split function
     2 import numpy as np
     3 import matplotlib.pyplot as plt
     4 from pandas.plotting import scatter_matrix
     5
     6 patients_data_train, patients_data_test = train_test_split(patients_data,random_state= 42,test_size=0.30, stratify=patients_data['target'])
     7 X_train = patients_data_train.drop('target',axis=1)
     8 y_train = patients_data_train['target']
     9 X_test = patients_data_test.drop('target',axis=1)
    10 y_test = patients_data_test['target']
    11
    12 from sklearn.preprocessing import StandardScaler
    13 scaler = StandardScaler()
    14 X_train=scaler.fit_transform(X_train)
    15 X_test=scaler.fit_transform(X_test)
    16
```

Materi

```
 1 from sklearn.decomposition import PCA
 2 from sklearn.neighbors import KNeighborsClassifier
 3 from sklearn.tree import DecisionTreeClassifier # Import Decision Tree Classifier
 4
 5 pca = PCA().fit(X_train)
 6 pca = PCA(n_components=5)
 7 X_reduced_train = pd.DataFrame(pca.fit_transform(X_train))
 8 X_reduced_test = pd.DataFrame(pca.fit_transform(X_test))
 9
10 knn = KNeighborsClassifier(n_neighbors=14)
11 KnnPCA=knn.fit(X_reduced_train, y_train)
12 print("Train set score using KNN as classifier:{:.2f}".format(knn.score(X_reduced_train, y_train)))
13 print("Test set score using KNN as classifier: {:.2f}".format(knn.score(X_reduced_test, y_test)))
14
15 knnNon = knn.fit(X_train,y_train)
16 y_pred = knn.predict(X_test)
17 print("Test set score using KNN non PCA as classifier: {:.2f}".format(knn.score(X_test, y_test)))
18
19 clf = DecisionTreeClassifier()
20 heart_disease_DT = clf.fit(X_reduced_train, y_train)
21 print("Train set score using CART as classifier:{:.2f}".format(clf.score(X_reduced_train, y_train)))
22 print("Test set score using CART as classifier: {:.2f}".format(clf.score(X_reduced_test, y_test)))
23
24 clfNon = clf.fit(X_train,y_train)
25 print("Test set score using CART non PCA as classifier: {:.2f}".format(clf.score(X_test, y_test)))
```

Materi

Output :

```
Train set score using KNN as classifier:0.85
Test set score using KNN as classifier: 0.81
Test set score using KNN non PCA as classifier: 0.80
Train set score using CART as classifier:1.00
Test set score using CART as classifier: 0.81
Test set score using CART non PCA as classifier: 0.69
```

Berdasarkan output yang pada gambar diperoleh hasil bahwa PCA dapat menghasilkan akurasi yang lebih baik dibandingkan dengan tanpa menggunakan PCA pada data heart-disease.