feature-extraction-selection-pca

May 26, 2025

0.1 Principal Component Analysis (PCA) –

0.1.1 Read Curse of Dimensionality

0.1.2 PCA ?

PCA (Principal Component Analysis) , - -
- ,

0.1.3 Step 1: Standardization

:

$$Z = \frac{X - \mu}{\sigma}$$

- X: (: $n \times p$)
- μ:
- σ:

Sample	X	X
1	2	3
2	4	1
3	6	5

: :

$$\mu_{X_1} = 4, \quad \mu_{X_2} = 3$$

:

$$Z = \begin{bmatrix} \frac{2-4}{\sigma_{X_1}} & \frac{3-3}{\sigma_{X_2}} \\ \frac{4-4}{\sigma_{X_1}} & \frac{1-3}{\sigma_{X_2}} \\ \frac{6-4}{\sigma_{X_1}} & \frac{5-3}{\sigma_{X_2}} \end{bmatrix}$$

0.1.4 Step 2:

:

$$\Sigma = \frac{1}{n} Z^T Z$$

- Σ: (: $p \times p$)

:

$$\Sigma = \begin{bmatrix} \text{Var}(Z_1) & \text{Cov}(Z_1, Z_2) \\ \text{Cov}(Z_2, Z_1) & \text{Var}(Z_2) \end{bmatrix}$$

0.1.5 Step 3:

:

$$\Sigma \mathbf{v} = \lambda \mathbf{v}$$

- λ: Eigenvalue
- $\mathbf{v}$: Eigenvector

:

1. Characteristic Equation:

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

2. $(\Sigma - \lambda I)\mathbf{v} = 0$ $\mathbf{v}$

:

$$\Sigma = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix} \Rightarrow \lambda = 1, 3$$

Eigenvectors:

- $\lambda = 3$: $\begin{bmatrix} 1 \\ 1 \end{bmatrix}$
- $\lambda = 1$: $\begin{bmatrix} -1 \\ 1 \end{bmatrix}$

0.1.6 Step 4: Principal Components

:

$$\text{Cumulative Variance} = \frac{\sum_{i=1}^{k} \lambda_i}{\sum_{i=1}^{p} \lambda_i}$$

:

$$\lambda_1 = 3, \lambda_2 = 1 \Rightarrow \text{Total Variance} = 4 \Rightarrow PC1 = 75\%, \quad PC1 + PC2 = 100\%$$

0.1.7 Step 5:

:

$$Y = ZW_k$$

- W_k: k
- Y:

:

$$Y = Z \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

0.2 Python Code: PCA

```python
import numpy as np
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler

# Step 1: Raw Data
X = np.array([[2, 3], [4, 1], [6, 5]])

# Step 2: Standardization
scaler = StandardScaler()
Z = scaler.fit_transform(X)

# Step 3: PCA
pca = PCA(n_components=1)
X_pca = pca.fit_transform(Z)

# Output
print("          :\n", X_pca)
print("                    :", pca.explained_variance_ratio_)
```

:

```
          :
 [[-1.414]
 [ 0.   ]
 [ 1.414]]
                    : [0.75]
```

0.3 Practical Application: PCA ?

0.4

Q: PCA ? A: ,

Q: PCA - ? A: , **Kernel PCA**

Q: PC ? A: , 85-95% ;

0.4.1

PCA
- PCA !

##More on My Hand Note: **ML-5 and ML-6**

```
[ ]: import numpy as np
     import pandas as pd
     import matplotlib.pyplot as plt
     import seaborn as sns
     import plotly.express as ex
```

```
[ ]: df=pd.read_csv('/content/Titanic-Dataset.csv')
     df.head(3)
```

```
[ ]:    PassengerId  Survived  Pclass  \
     0            1         0       3
     1            2         1       1
     2            3         1       3

                                                     Name     Sex   Age  SibSp  \
     0                            Braund, Mr. Owen Harris    male  22.0      1
     1  Cumings, Mrs. John Bradley (Florence Briggs Th…  female  38.0      1
     2                             Heikkinen, Miss. Laina  female  26.0      0

        Parch            Ticket     Fare Cabin Embarked
     0      0         A/5 21171   7.2500   NaN        S
     1      0          PC 17599  71.2833   C85        C
     2      0  STON/O2. 3101282   7.9250   NaN        S
```

```
[ ]: df=df.iloc[:,[2,5,6,7,9,1]]
     df.head(3)
```

```
[ ]:    Pclass   Age  SibSp  Parch     Fare  Survived
     0       3  22.0      1      0   7.2500         0
     1       1  38.0      1      0  71.2833         1
     2       3  26.0      0      0   7.9250         1
```

[ ]:
```
df.dropna(inplace=True)
```

[ ]:
```
df.isnull().sum()
```

[ ]:
```
Pclass      0
Age         0
SibSp       0
Parch       0
Fare        0
Survived    0
dtype: int64
```

#PCA manually

[ ]:
```
X=df.iloc[:,:5]
y=df['Survived']
```

[ ]:
```
#Setp-1:Standardization Scaling
from  sklearn.preprocessing import StandardScaler
ss=StandardScaler()
new_x=ss.fit_transform(X)
new_x
```

[ ]:
```
array([[ 0.91123237, -0.53037664,  0.52457013, -0.50589515, -0.51897787],
       [-1.47636364,  0.57183099,  0.52457013, -0.50589515,  0.69189675],
       [ 0.91123237, -0.25482473, -0.55170307, -0.50589515, -0.50621356],
       …,
       [-1.47636364, -0.73704057, -0.55170307, -0.50589515, -0.08877362],
       [-1.47636364, -0.25482473, -0.55170307, -0.50589515, -0.08877362],
       [ 0.91123237,  0.15850313, -0.55170307, -0.50589515, -0.50952283]])
```

[ ]:
```
#step2: Co-Varinace Matrix
cov=np.cov(new_x.T)
```

[ ]:
```
#setp3: Eigen value and Eigen Vector
eig=np.linalg.eig(cov)
eig
```

[ ]:
```
EigResult(eigenvalues=array([1.74380104, 1.6159263 , 0.36447732, 0.69209424,
0.59071373]), eigenvectors=array([[-0.62133317,  0.2742927 , -0.71116258,
-0.16677988,  0.0716998 ],
       [ 0.54592049,  0.19719823, -0.28190116, -0.66731845, -0.37188909],
       [-0.31151744, -0.53986787, -0.02187925,  0.03015686, -0.7810963 ],
       [-0.20582771, -0.56301483,  0.15922655, -0.64819931,  0.44173954],
       [ 0.42012825, -0.52671942, -0.62365674,  0.32526724,  0.2265224 ]]))
```

[ ]:
```
eig.eigenvalues
```

```
[ ]: array([1.74380104, 1.6159263 , 0.36447732, 0.69209424, 0.59071373])
```

```
[ ]: #step4:Choose No. of Principle Component(PC)
     W=eig.eigenvectors[:2,]
     W
```

```
[ ]: array([[-0.62133317,  0.2742927 , -0.71116258, -0.16677988,  0.0716998 ],
            [ 0.54592049,  0.19719823, -0.28190116, -0.66731845, -0.37188909]])
```

```
[ ]: #setp5: Projection of Scaled Data on PCs
     pca_matrix=np.dot(new_x,np.transpose(W))
     pca_matrix
```

```
[ ]: array([[-1.03754947,  0.77558953],
            [ 0.83509012, -0.76080572],
            [-0.19564717,  1.12858364],
            …,
            [ 1.18550751, -0.42518742],
            [ 1.3177758 , -0.33009531],
            [-0.08251163,  1.21132184]])
```

```
[ ]: New_df=pd.DataFrame(pca_matrix,columns=['pc1','pc2'])
     New_df['Survived']=y
     New_df.head()


     #5D to 2D te converted
```

```
[ ]:         pc1       pc2  Survived
     0 -1.037549  0.775590       0.0
     1  0.835090 -0.760806       1.0
     2 -0.195647  1.128584       1.0
     3  0.753750 -0.673686       1.0
     4 -0.025418  1.249966       0.0
```

```
[ ]: New_df.isnull().sum()
```

```
[ ]: pc1           0
     pc2           0
     Survived    147
     dtype: int64
```

```
[ ]: ex.scatter_3d(X,x='Age',y='Fare',z='Pclass',color=df['Survived'].astype('str'))
```

```
[ ]: ex.scatter(New_df,x='pc1',y='pc2',color=New_df['Survived'].astype('str'))
```