데이터 과학

L11.1: PCA Practice

Kookmin University

Iris dataset

- 아이리스(붓꽃) 데이터
  - 붓꽃 종류별로 꽃받침과 꽃잎의 길이 및 너비를 측정한 데이터
  - https://archive.ics.uci.edu/ml/datasets/Iris

```
4.6,3.2,1.4,0.2,Iris-setosa
5.3,3.7,1.5,0.2,Iris-setosa
5.0,3.3,1.4,0.2,Iris-setosa
7.0,3.2,4.7,1.4,Iris-versicolor
6.4,3.2,4.5,1.5,Iris-versicolor
6.9,3.1,4.9,1.5,Iris-versicolor
5.5,2.3,4.0,1.3,Iris-versicolor
```

Loading Iris dataset

- scikit-learn의 load_iris 활용

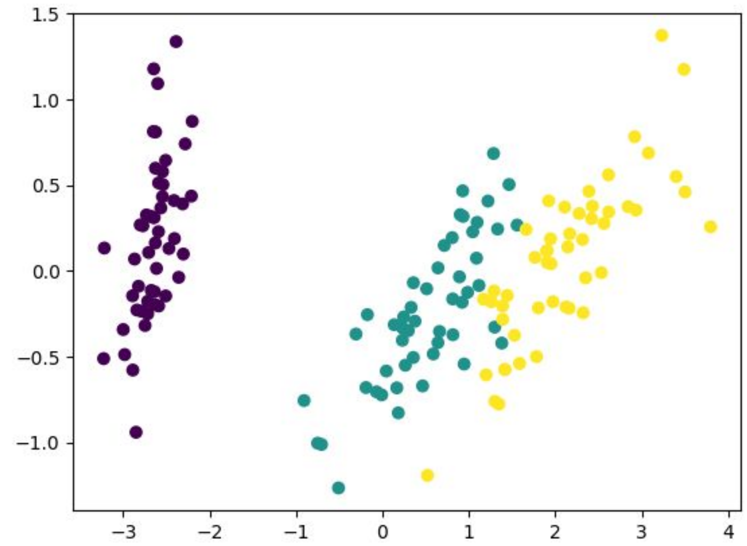
```
from sklearn.datasets import load_iris

iris = load_iris()
```

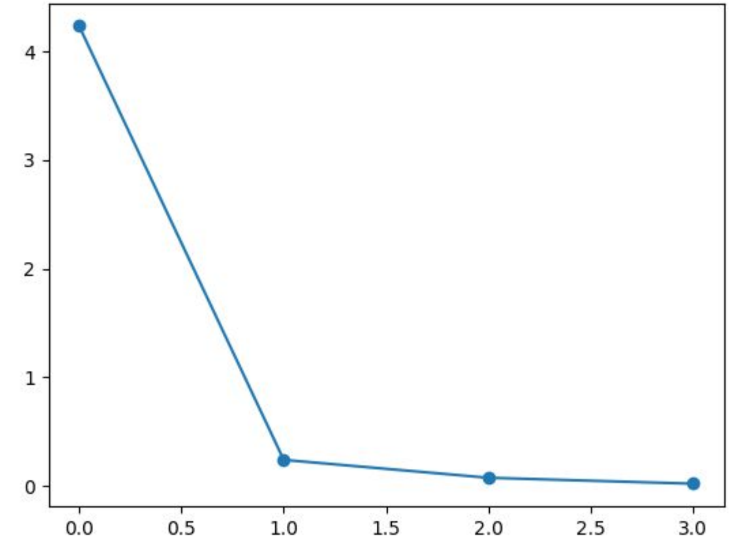
PCA in scikit-learn

```
from sklearn.decomposition import PCA
import matplotlib.pyplot as plt

pca = PCA(n_components=4)

Y = pca.fit_transform(iris['data'])

plt.scatter(Y[:,0], Y[:,1], c=iris['target'])
plt.show()

plt.plot(pca.explained_variance_, "-o")
plt.show()
```

PCA in scikit-learn

```
from sklearn.decomposition import PCA
import matplotlib.pyplot as plt

pca = PCA(n_components=4)

Y = pca.fit_transform(iris['data'])

plt.scatter(Y[:,0], Y[:,1], c=iris['target'])
plt.show()

plt.plot(pca.explained_variance_, "-o")
plt.show()
```





PCA via Eigen Decomposition

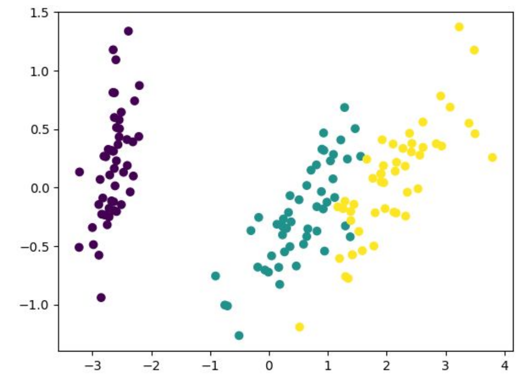
- Eigen Decomposition
  - n x n 행렬 M을 다음 조건에 맞게 분해

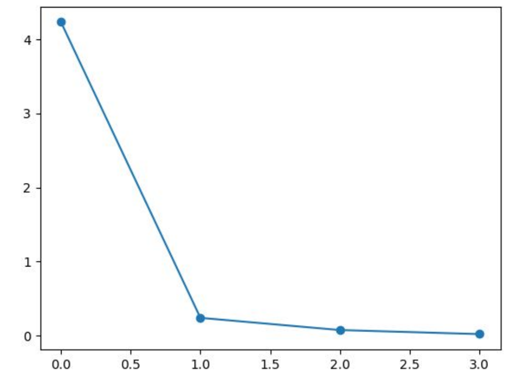
$$\mathbf{M} = \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^\top$$

  - Q: n x n 크기의 Orthogonal Matrix
  - Λ: n x n 크기의 Diagonal Matrix

PCA via Eigen Decomposition

```python
from numpy import linalg
import matplotlib.pyplot as plt

X = iris.data
X = X - X.mean(axis=0)

C = (X.T @ X) / X.shape[0] # covariance matrix
L, Q = linalg.eigh(C) # eigen decomposition

print("L:", L)
print("Q:", Q)

plt.scatter(X @ Q[:,3], X @ Q[:,2], c=iris.target)
plt.show()

plt.plot(sorted(L, reverse=True), "-o")
plt.show()
```





출력:

```
L: [0.02367619 0.0776881  0.24105294 4.20005343]
Q: [[ 0.31548719  0.58202985  0.65658877 -0.36138659]
 [-0.3197231  -0.59791083  0.73016143  0.08452251]
 [-0.47983899 -0.07623608 -0.17337266 -0.85667061]
 [ 0.75365743 -0.54583143 -0.07548102 -0.3582892 ]]
```

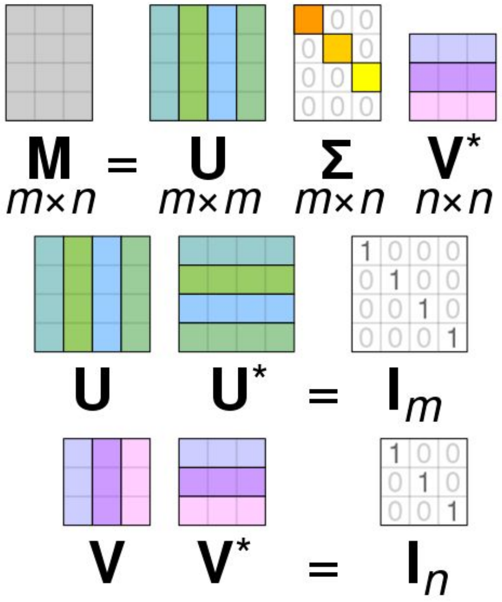
PCA via SVD

- Singular Value Decomposition
  - m x n 행렬 X을 다음 조건에 맞게 분해

$$\mathbf{X} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^\top$$

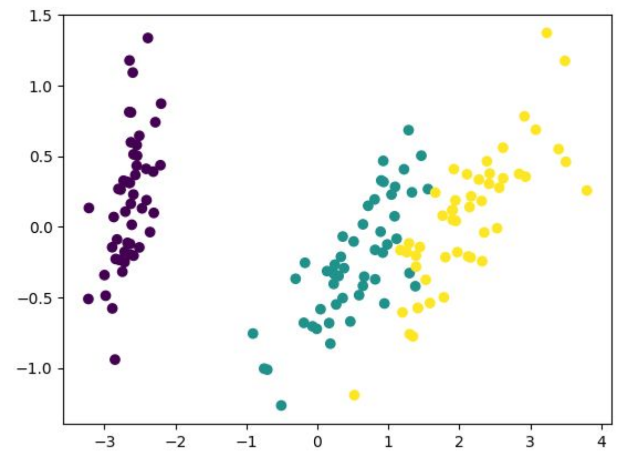
  - U: m x m 크기의 Orthogonal matrix
  - Σ: m x n 크기의 Diagonal matrix
  - V: n x n 크기의 Orthogonal matrix
- Eigen Decomposition과의 관계

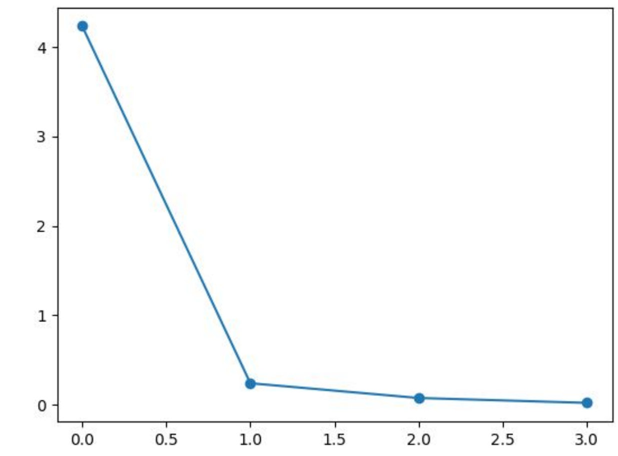
$$\begin{aligned}\mathbf{X}^\top\mathbf{X} &= (\mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^\top)^\top\mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^\top \\ &= \mathbf{V}\boldsymbol{\Sigma}^\top\mathbf{U}^\top\mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^\top \\ &= \mathbf{V}\boldsymbol{\Sigma}^\top\boldsymbol{\Sigma}\mathbf{V}^\top \\ &= \mathbf{V}\boldsymbol{\Sigma}^2\mathbf{V}^\top\end{aligned}$$



PCA via SVD

```
from numpy import linalg

X = iris.data
X = X - X.mean(axis=0)

U, S, VT = linalg.svd(X)

plt.scatter(X @ VT[0], X @ VT[1], c=iris.target)
plt.show()

variances = (S ** 2) / X.shape[0]
plt.plot(variances, "-o")
plt.show()
```





PCA via Power Method

```python
from numpy import linalg
import matplotlib.pyplot as plt
import numpy as np

X = iris.data
X = X - X.mean(axis=0)

# covariance matrix
C = (X.T @ X) / X.shape[0]

v = np.random.randn(C.shape[0], 1)
v = v / linalg.norm(v) # normalize

M = C.copy()

L = []
Q = []
```

```python
for dim in range(4):
   for epoch in range(20):
       vp = M @ v
       lmd = linalg.norm(vp)
       vp = vp / lmd
       v = vp

   M = M - lmd * (v @ v.T)

   L.append(lmd)
   Q.append(v)

L = np.array(L)
Q = np.hstack(Q)

print("L:", L)
print("Q:", Q)
```

출력:

```
L: [4.20005343 0.24105294 0.0776881  0.02367619]
Q: [[ 0.36138659  0.65658877 -0.58202985 -0.31548719]
 [-0.08452251  0.73016143  0.59791083  0.3197231 ]
 [ 0.85667061 -0.17337266  0.07623608  0.47983899]
 [ 0.3582892  -0.07548102  0.54583143 -0.75365743]]
```

Questions?