Week 10

Chapter 06 차원 축소

6.1 차원 축소(Dimension Reduction) 개요

- 차원 축소란

  매우 많은 피처로 구성된 다차원 데이터 세트의 차원을 축소해 새로운 차원의 데이터 세트를 생성하는 것

  → 차원 축소해 피처 수를 줄이면 더 직관적으로 데이터를 해석할 수 있음

  - 피처 선택(feature selection) : 특정 피처에 종속성이 강한 불필요한 피처는 아예 제거하고 데이터의 특징을 잘 나타내는 주요 피처만 선택하는 것
  - 피처 추출(feature extraction) : 기존 피처를 저차원의 중요 피처로 압축해서 추출하는 것 (기존 피처와는 완전히 다른 값이 됨)

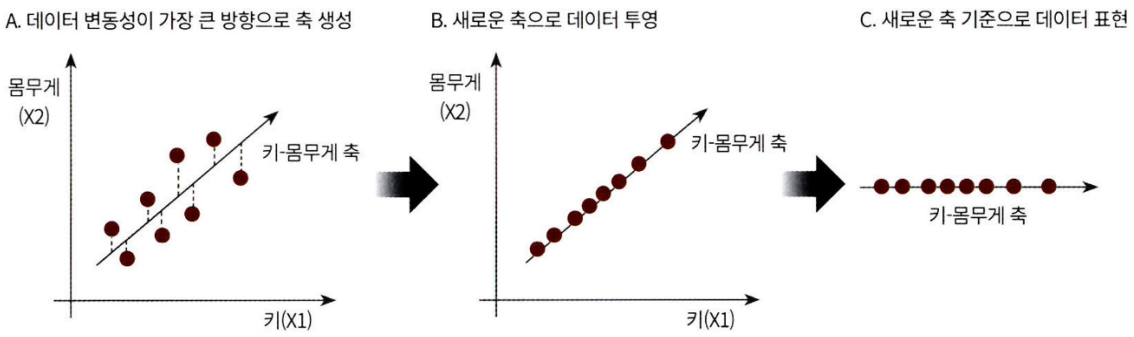
  PCA, SVD, NMF가 가장 대표적인 차원 축소 알고리즘

6.2 PCA(Principal Component Analysis)

- PCA 개요

  여러 변수 간에 존재하는 상관관계를 이용해 이를 대표하는 주성분(Principal Component)을 추출해 차원을 축소.

  *기존 데이터 정보 유실을 최소화하기 위해 가장 높은 분산을 가지는 데이터의 축을 찾아 이 축으로 차원을 축소함



- PCA와 선형대수

  PCA는 입력 데이터의 공분산 행렬을 고유값 분해하고 고유벡터에 입력 데이터를 선형 변환하는 것.

  → 고유벡터가 PCA의 주성분 벡터, 고윳값은 고유벡터의 크기 및 입력 데이터의 분산

$$C = P\sum P^T$$

$$C = [e_1 \cdots e_n]\begin{bmatrix} \lambda_1 & \cdots & 0 \\ \cdots & \cdots & \cdots \\ 0 & \cdots & \lambda_n \end{bmatrix}\begin{bmatrix} e_1^t \\ \cdots \\ e_n^t \end{bmatrix}$$

직교행렬 $*$ 고유값 정방행렬 $*$ 고유벡터 직교행렬의 전치행렬
e_i : i번째 고유벡터
λ_i : i번째 고유벡터의 크기
e_1은 분산이 큰 방향을 가진 고유벡터
e_2는 e_1에 수직이면서 다음으로 가장 분산이 큰 방향을 가진 고유벡터

- PCA 수행 단계

  step1. 입력 데이터 세트의 공분산 행렬을 생성

  step2. 공분산 행렬의 고유벡터와 고유값 계산

  step3. 고유값이 가장 큰 순서로 K개만큼 고유벡터를 추출

  step4. 고유값이 가장 큰 순으로 추출된 고유벡터를 이용해 새롭게 입력 데이터를 변환

- PCA 변환 코드

```
from sklearn.decomposition import PCA

pca = PCA(n_components=2)
pca.fit(data)
data_pca = pca.transform(data
```

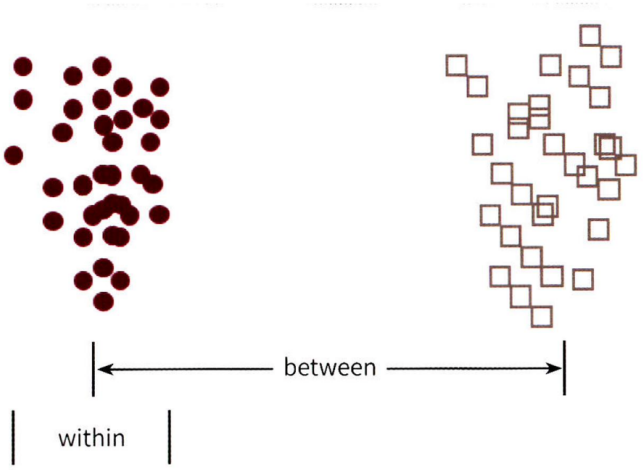
  `n_components` : PCA로 변환할 차원의 수

6.3 LDA(Linear Discriminant Analysis)

- LDA 개요

입력 데이터 세트를 지도학습의 분류에서 사용하기 쉽도록 분별할 수 있는 기준을 최대한 유지하며 저차원 공간에 투영해 차원을 축소

✱ 특정 공간상에서 클래스 분리를 최대화하는 축을 찾기 위해 클래스 간 분산과 클래스 내부 분산의 비율을 최대화하는 방식



- LDA 수행 단계

  step1. 클래스 내부와 클래스 간 분산 행렬을 구함. 두 개의 행렬은 입력 데이터의 결정값 클래스별로 개별 피처의 평균 벡터를 기반으로 구함

  step2. 클래스 내부 분산 행렬을 S_W, 클래스 간 분산 행렬을 S_B라고 하면 다음 식으로 두 행렬을 고유벡터로 분해할 수 있음

$$S_W^T S_B = \begin{bmatrix} e_1 & \cdots & e_n \end{bmatrix} \begin{bmatrix} \lambda_1 & \cdots & 0 \\ \cdots & \cdots & \cdots \\ 0 & \cdots & \lambda_n \end{bmatrix} \begin{bmatrix} e_1^T \\ \cdots \\ e_n^T \end{bmatrix}$$

  step3. 고유값이 가장 큰 순으로 K개 추출

  step4. 고유값이 가장 큰 순으로 추출된 고유벡터를 이용해 새롭게 입력 데이터 변환

- LDA 변환 코드

```
from sklearn.decomposition import LinearDiscriminatAnalysi

lda = LinearDiscriminatAnalysis(n_components=2)
lda.fit(data)
data_lda = lda.transform(data)
```

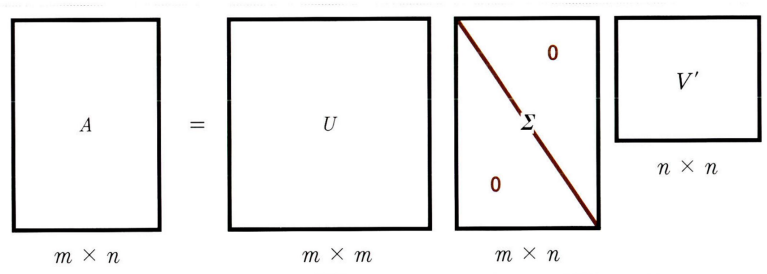
6.4 SVD(Singular Value Decomposition)

- SVD 개요

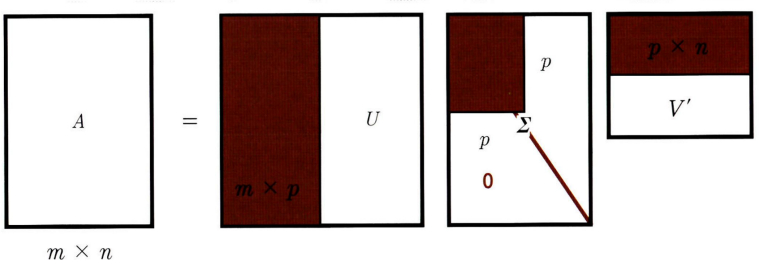
  PCA와 유사한 행렬 분해 기법을 이용. SVD는 정방행렬 뿐만 아니라 행과 열의 크기가 다른 행렬에도 적용 가능

$$A = U \sum V^T$$

$U,\ V$: 특이벡터$(singular\ vector)$
Σ : 대각행렬(대각원소는 행렬A의 특이값을 나타냄)



  일반적으로는 아래와 같이 Σ의 비대각인 부분과 대각원소 중에 특이값이 0인 부분도 모두 제거하고 제거된 Σ에 대응되는 U와 V원소도 함께 제거해 차원을 줄인 형태인 Truncated SVD를 적용함.



- Truncated SVD 분해 코드

```
from scipy.sparse.linalg import svds

num_components = 4
U_tr, Sigma_tr, Vt_tr =  svds(matrix, k=num_components)
```

- Truncated SVD 변환 코드

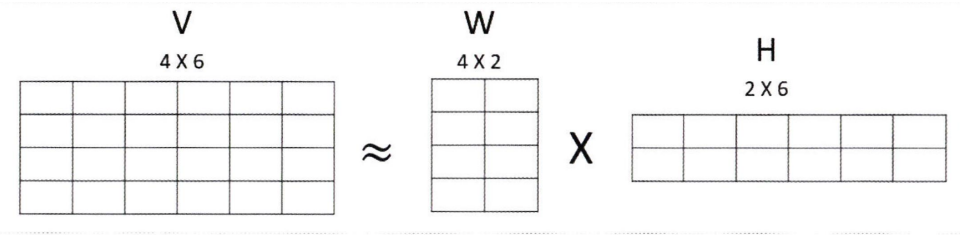
```
from sklearn.decomposition import TruncatedSVD

tsvd = TruncatedSVD(n_components=2)
tsvd.fit(data)
data_tsvd = tsvd.transform(data)
```

6.5 NMF(Non-Negative Matrix Factorization)

- NMF 개요

  낮은 랭크를 통한 행렬 근사 방식의 변형.

  → 원본 행렬 내의 모든 원소 값이 모두 양수라는 것이 보장되면 아래와 같이 두 개의 기반 양수 행렬로 분해될 수 있는 기법



W : 원본 행에 대해서 잠재 요소의 값이 얼마나 되는지에 대응
H : 잠재 요소가 원본 열로 어떻게 구성되었는지를 나타냄

- NMF 변환 코드

```
from sklearn.decomposition import NMF

nmf = NMF(n_components=2)
nmf.fit(data)
data_nmf = nmf.transform(data)
```