♥ TITLE : 6_차원 축소 ♥ DATE : 2022.10.05

1 차원 축소

- 매우 많은 피처로 구성된 다차원 데이터 세트의 차원을 축소해 새로운 차원의 데이터 세트를 생성하는 것
- 차원 축소를 하는 이유
  - 일반적으로 차원 증가 → 데이터 포인트 간의 거리가 기하급수적으로 멀어짐, 희소한(Sparse) 구조
  - 수백 개 이상의 피처로 구성된 데이터 세트의 경우 상대적으로 적은 차원에서 학습된 모델보다 예측 신뢰도 ↓
  - 피처가 많은 경우 개별 피처 간의 상관관계가 높을 가능성이 ↑ ⇒ 다중 공선성

⇒ 더 직관적으로 데이터를 해석하기 위해서
학습 데이터의 크기 ↓ → 학습에 필요한 처리 능력 ↓

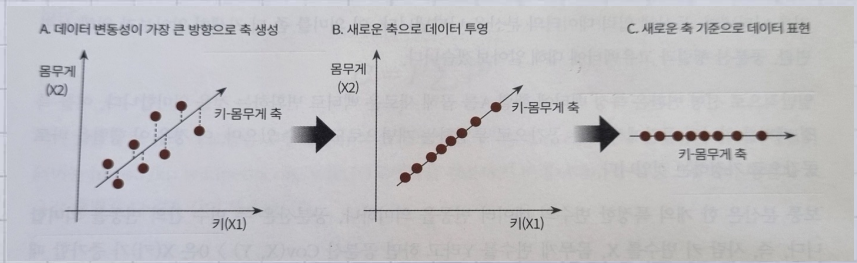
- 차원 축소의 유형
  - i) 피처 선택: 특정 피처에 중요성이 강한 불필요한 피처는 아예 제거
  - ii) 피처 추출: 기존 피처를 저차원의 중요 피처로 압축해서 추출 → 기존과는 완전히 다른 피처

⇒ 기존 피처가 전혀 인지하지 어려웠던 잠재적 요소를 추출

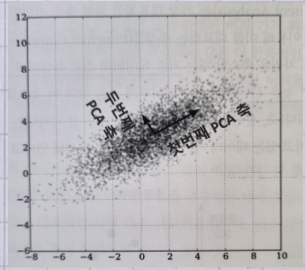
- 활용
  - i) 이미지 데이터 처리
  - ii) 텍스트 문서 처리

2 PCA (Principal Component Analysis)

- 여러 변수 간에 존재하는 상관관계를 이용하여 이를 대표하는 주성분을 추출해 차원을 축소하는 기법
- 가장 높은 분산을 가지는 데이터 축을 선택해 차원 축소 ⇒ 가장 데이터의 정보 유실 최소화
  ↳ 데이터 변동성이 가장 큰



- 가장 큰 데이터 변동성을 기반으로 첫 번째 벡터 축을 생성하고, 두 번째 축은 첫 번째 벡터 축에 직각이 되는 벡터를 축으로 함. ⇒ 계속해서 직교 벡터를 축으로 생성



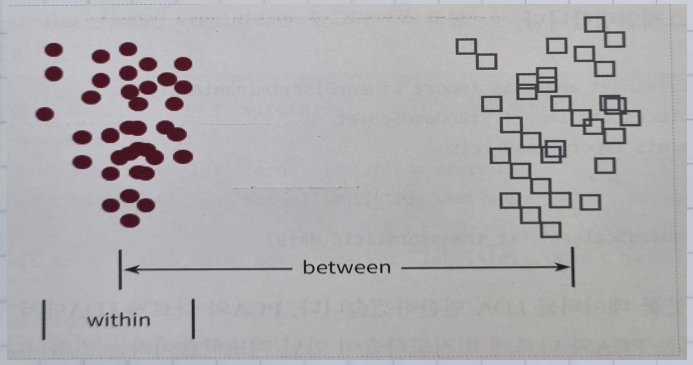
- 입력 데이터의 공분산 행렬이 고유벡터와 고유값으로 분해될 수 있으며, 이렇게 분해된 고유벡터를 이용해 입력 데이터를 선형 변환하는 방식 (*교재 p.380~382)
- 순서
  - i) 입력 데이터 세트의 공분산 행렬을 생성
  - ii) 공분산 행렬의 고유벡터와 고유값을 계산
  - iii) 고유값이 가장 큰 순으로 K개만큼 고유벡터 추출
  - iv) 추출된 고유벡터를 이용해 새롭게 입력 데이터 변환

♥ TITLE : 6_차원 축소 ♥ DATE :

- PCA는 여러 속성의 값을 연산해야 함 → PCA로 압축 전 여러 속성들을 동일한 스케일로 변환하는 것이 필요
- 사이킷런의 PCA 클래스를 이용
  - n_components 파라미터: PCA로 변환할 차원의 수
  - pca.explained_variance_ratio_: 전체 변동성에서 개별 PCA 컴포넌트별로 차지하는 변동성 비율을 제공
- 원본 데이터 세트 대비 예측 정확도는 PCA 변환 차원 개수에 따라 예측 성능이 떨어질 수밖에 없음.
- 높은 상관도를 가진 속성들은 소수의 PCA만으로도 자연스럽게 속성들의 변동성을 수용할 수 O

3 LDA(Linear Discriminant Analysis)

- 선형 판별 분석법
- 지도학습의 분류에서 사용하기 쉽도록 개별 클래스를 분별할 수 있는 기준을 최대한 유지하면서 차원 축소 → 입력 데이터의 결정 값 클래스를 최대한으로 분리할 수 있는 축 찾기
- 클래스 간 분산은 최대한 크게, 클래스 내부 분산은 최대한 작게 유지



- LDA를 구하는 Step
  - i) 클래스 내부와 클래스 간 분산 행렬 구하기
    - ↳ 입력 데이터의 결정값 클래스별 개별 피처의 평균 벡터 기반
  - ii) 클래스 내부 분산 행렬(S_W)과 클래스 간 분산 행렬(S_B)을 고유벡터로 분해

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

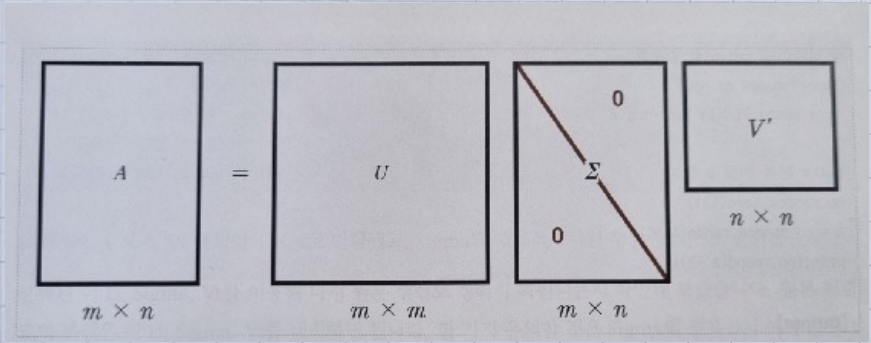
  - iii) 고유값이 가장 큰 순으로 k개(LDA 변환 차수만큼) 추출
  - iv) 추출된 고유벡터를 이용하여 새롭게 입력 데이터 변환
- 사이킷런의 LinearDiscriminantAnalysis 클래스도 제공됨.
- LDA는 지도학습 ⇒ 변환 시 클래스의 결정값 필요

set a record

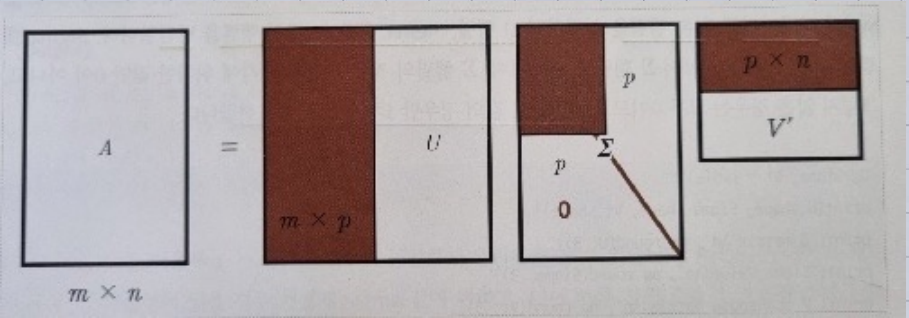
♥ TITLE : 6_차원축소 ♥ DATE : 2022.10.06

4 SVD(Singular Value Decomposition)

- 정방행렬뿐만 아니라 행과 열의 크기가 다른 mxn 크기의 행렬도 분해 가능
- $A = U \Sigma V^T$
  - A: mxn, U: mxm, V^T: nxn
  - Σ: 대각행렬 (singular vector) mxn ← Σ이 위치한 0이 아닌 값이 행렬 A의 특이값



but 일반적으로는 아래와 같이 Σ의 비대각인 부분과 대각원소 중에 특이값이 0인 부분도 모두 제거하고 제거된 Σ에 대응되는 U와 V도 함께 제거
→ 차원 축소된 형태로 SVD 적용

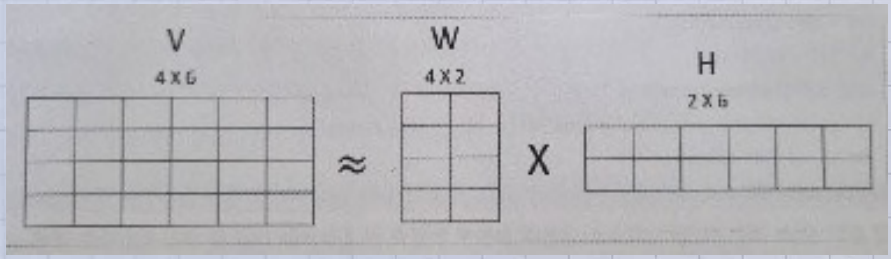


- 일반적인 SVD는 보통 numpy 또는 scipy 라이브러리를 이용해 수행
  - ↳ numpy.linalg.svd(원본행렬) = U행렬, Sigma 행렬, V 전치 행렬
    - Sigma 행렬: 0이 아닌 경우만 1차원 행렬로 표현
- 원본 행렬로의 복원
  - ↳ U, sigma, Vt를 내적
  - 주의 Sigma의 경우 0이 아닌 값만 1차원으로 추출 ⇒ 다시 0을 포함한 대칭행렬로 변환 후 내적을 수행해야 함.
- Truncated SVD
  - ↳ Σ 행렬에 있는 대각원소만 추출하여 분해하는 방식
  - ↳ 인위적으로 더 적은 차원의 U, Σ, V^T로 분해 → 원본행렬을 정확히 다시 복원할 수는 없지만 근사할 수 O
  - ↳ 사이파이 모듈에서만 지원됨. ⇒ scipy.sparse.linalg.svds
- 사이킷런 TruncatedSVD 클래스
  - ↳ fit() 과 transform()을 호출해 원본 데이터를 몇 개의 주요 컴포넌트로 차원 축소해 변환
  - ↳ 원본 데이터를 TruncatedSVD 방식으로 분해된 U*Sigma 행렬에 선형 변환해 생성
  - ↳ TruncatedSVD(n_components = k) : k개의 주요 component로 변환

♥ TITLE : 6_차원 축소 ♥ DATE : 2022.10.08

5 NMF (Non-negative Matrix Factorization)

- Low-Rank Approximation 방식의 변형
- 원본 행렬 내의 모든 원소가 양수(0 포함)라는 게 보장되면 다음과 같이 좀 더 간단하게 두 개의 기반 양수 행렬로 분해될 수 있는 기법을 지칭



✶ 행렬 분해 (Matrix Factorization)

- SVD와 같은 행렬 분해 기법을 통칭
- 일반적으로 길고 가는 행렬 W와 작고 넓은 행렬 H로 분해
  ∈행 크기 유지 ∈열 크기 유지
- 사이킷런의 NMF 클래스를 통해 지원됨.
- 차원 축소를 통한 잠재 요소 도출 ⇒ 이미지 변환·압축, 텍스트의 토픽 도출 등의 영역에서 사용되고 o