Chapter06. 차원축소

01. 차원 축소(Dimension Reduction) 개요

> 📌 **차원축소**
>
> 매우 많은 피처로 구성된 다차원 데이터 세트의 차원을 축소
>
> → 새로운 차원의 데이터 세트 생성

- 일반적으로 차원 증가 → 데이터 포인트 간 거리 기하급수적으로 멀어짐 → 희소한 구조
- 수백 개 이상의 피처로 구성된 데이터 셋은 상대적으로 적은 차원에서 학습된 모델보다 예측 신뢰도가 떨어짐
- 피처가 많을 경우 개별 피처 간의 상관관계 높을 가능성이 큼
  - 선형회귀 같은 경우 다중 공선성 문제
- 매우 많은 다차원의 피처를 차원 축소해 피처 수를 줄이면 더 직관적인 데이터 해석가능
- 차원 축소를 할 경우 학습 데이터의 크기가 줄어듦(학습에 필요한 처리 능력 감소)

- 차원 축소는 피처 선택(feature slection)과 피처 추출(feature extraction)으로 나뉨
  - 피처 선택: 특정 피처에 종속성이 강한 불필요한 피처는 아예 제거, 데이터의 특징이 잘 나타나는 주요 피처만 선택
  - 피처 추출: 기존 피처를 저차원의 중요 피처로 압축해 추출
    - 새롭게 추출된 중요 특성은 기존의 피처가 압축된 것이므로 기존의 피처값과 완전 다른 값
    - 기존 피처를 단순 압축 X
    - 피처를 함축적으로 더 잘 설명할 수 있는 또다른 공간으로 매핑해 추출
    - 이런 함축적인 특성 추출은 잠재적인 요소 추출을 의미

- 차원 축소는 단순히 데이터 압축 X, 차원 축소를 통해 잠재적인 요소 추출
  - PCA, SVD, NMF는 잠재적인 요소를 찾는 대표적인 차원 축소 알고리즘

- 매우 많은 픽셀로 이뤄진 이미지 데이터 → 잠재된 특성을 피처로 도출해 함축적 형태의 이미지 변환과 압축 수행
  - 분류 수행 시 과적합 영향력이 작아짐 → 예측 성능 up

- 차원 축소 알고리즘이 자주 사용되는 또다른 영역 : 텍스트 문서의 숨은 의미 추출
  - 문서 내 단어들의 구성에서 숨겨진 시맨틱 의미나 토픽을 잠재요소로 간주하고 이를 찾아낼 수 있음
  - SVD와 NMF은 시맨틱 토픽 모델링을 위한 기반 알고리즘으로 사용

02. PCA(Principal Component Analysis)

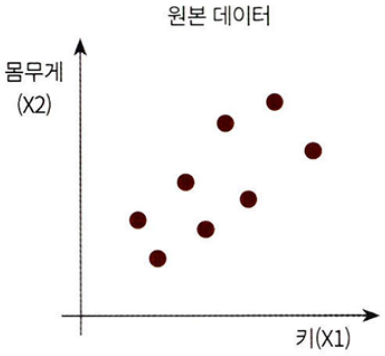
1) PCA 개요

> 📌 **PCA(Principal Component Analysis)**
>
> 여러 변수 간에 존재하는 상관관계를 이용해 이를 대표하는 주성분(Principal Component)을 추출해 차원을 축소하는 기법

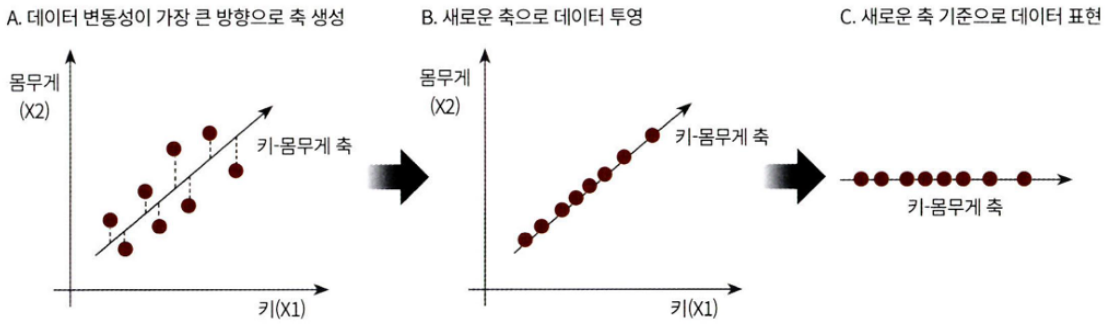
- PCA로 차원 축소 시 기존 데이터의 정보 유실이 최소화 됨
- 가장 높은 분산을 갖는 데이터의 축을 찾아 이 축으로 차원 축소

  → PCA의 주성분이 됨(분산이 데이터의 특성을 가장 잘 나타낸다고 간주)

- 키와 몸무게 2개 피처의 데이터 세트가 다음과 같이 구성돼있다고 가정

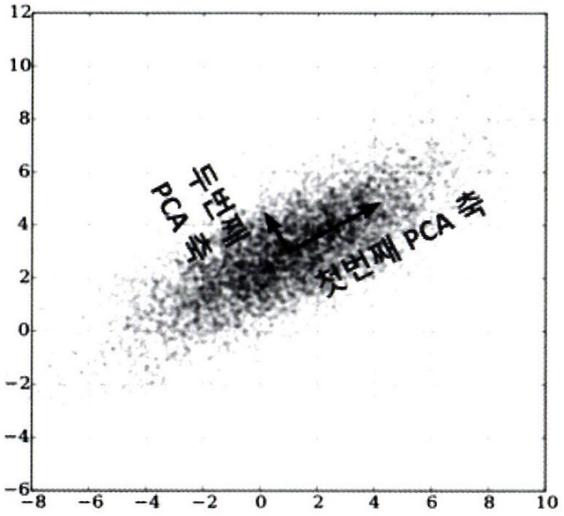


- 2개의 피처를 한 개의 주성분을 가진 데이터 셋으로 차원 축소
- 가장 큰 방향으로 축 생성 후 새롭게 생성된 축으로 데이터 투영



1. 가장 큰 데이터 변동성(Variance)을 기반으로 첫 번째 벡터 축 생성
2. 두 번째 벡터 축은 1 벡터 축에 직각이 되는 벡터(직교 벡터)를 축으로
3. 세 번째 벡터는 두 번째 벡터와 직각이 되는 벡터를 설정하는 방식으로 축 생성

⇒ 이렇게 생성된 벡터 축에 원본 데이터를 투영하면 벡터 축의 개수만큼의 차원으로 원본 데이터가 차원 축소



- PCA는 원본 데이터의 피처 개수에 비해 매우 작은 주성분으로 원본 데이터의 총 변동성을 대부분 설명할 수 있는 분석법
- 선형대수 관점에서 해석
  - 입력 데이터의 공분산 행렬(Covariance Matrix)을 고유값 분해 → 고유벡터에 입력 데이터를 선형 변환
  - 고유벡터가 PCA의 주성분 벡터로서 입력 데이터의 분산이 큰 방향을 나타냄
  - 고윳값 : 고유벡터의 크기와 동시에 입력 데이터의 분산을 나타냄

- 선형 변환
  - 특정 벡터에 행렬 A를 곱해 새로운 벡터로 변환하는 것
  - 특정 벡터를 하나의 공간 → 다른 공간 투영하는 개념
    - 이 행렬을 바로 공간으로 가정하는 것

- 공분산
  - 두 변수 간의 변동을 의미
  - 여러 변수와 관련된 공분산을 포함하는 정방향 행렬

	X	Y	Z
X	3.0	−0.71	−0.24
Y	−0.71	4.5	0.28
Z	−0.24	0.28	0.91

- 고유벡터
  - 행렬A를 곱하더라도 방향이 변하지 않고 그 크기만 변하는 벡터를 지칭
  - Ax = ax(A:행렬, x: 고유벡터, a:스칼라값)
  - 이 고유벡터는 여러 개 존재
  - 정방 행렬은 최대 그 차원 수만큼의 고유 벡터를 가질 수 있음
    - 2X2 행렬 : 두 개의 고유벡터
    - 3X3행렬 : 세 개의 고유벡터
  - 행렬이 작용하는 힘의 방향과 관계, 행렬을 분해하는 데 사용

- 공분산 행렬
  - 정방행렬(Diagonal Matrix)이며 대칭행렬(Symmetric Matrix)
  - 정방행렬 : 열과 행이 같은 행렬
  - 대칭행렬 : 정방행렬 중 대각 원소를 중심으로 원소 값이 대칭되는 행렬
  - 공분산 행렬은 개별 분산값을 대각 원소로 하는 대칭행렬
    - 항상 고유벡터를 직교행렬로, 고유값을 정방 행렬로 대각화 가능

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

  - 입력 데이터의 공분산 행렬을 C라고 가정
    - P는 n x n의 직교행렬
    - Σ는 n x n의 정방행렬
    - P의 T승은 행렬 P의 전치 행렬

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

- 공분산 C는 고유벡터 직교행렬*고유값 정방 행렬*고유벡터 직교 행렬의 전치 행렬로 분해

- 입력 데이터의 공분산 행렬이 고유벡터와 고유값으로 분해 가능
- 분해된 고유벡터를 이용해 입력 데이터를 선형 변환하는 방식이 PCA

- PCA 수행 스텝

1. 입력 데이터 세트의 공분산 행렬을 생성합니다.
2. 공분산 행렬의 고유벡터와 고유값을 계산합니다.
3. 고유값이 가장 큰 순으로 K개(PCA 변환 차수만큼)만큼 고유벡터를 추출합니다.
4. 고유값이 가장 큰 순으로 추출된 고유벡터를 이용해 새롭게 입력 데이터를 변환합니다.

- 사이킷런의 PCA 변환 : PCA클래스
  - 생성 파라미터 : n_components - PCA로 변환할 차원의 수를 의미
- explained_variance_ratio_ : 전체 변동성에서 개별 PCA 컴포넌트별로 차지하는 변동선 비율 제공
- PCA가 더 활발하게 적용되는 영역 : 컴퓨터 비전(Computer Vision)분야
  - 특히 얼굴 인식의 경우 Eigen-face라고 불리는 PCA 변환으로 원본 얼굴 이미지를 변환해 사용하는 경우가 많음

03. LDA(Linear Discriminant Analysis)

1) LDA 개요

📌 **LDA(Linear Discriminant Analysis) : 선형 판별 분석법**

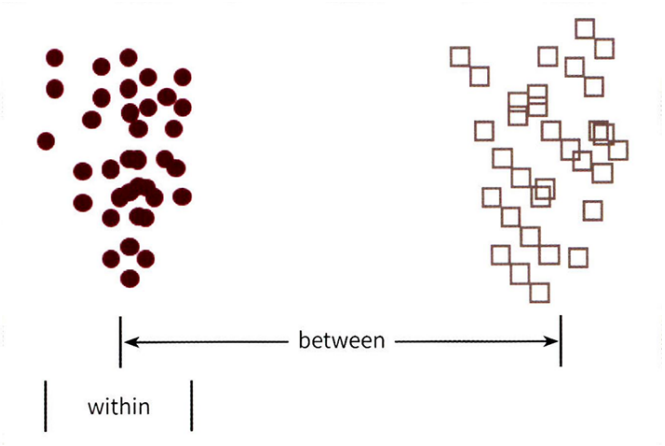
PCA와 유사하게 입력 데이터 세트를 저차원 공간에 투영해 차원 축소 기법

But, 개별 클래스를 분별할 수 있는 기준을 최대한 유지하면서 차원 축소

⇒지도학습의 분류에서 사용하기 쉽도록

- PCA Vs. LDA
  - PCA : 입력 데이터의 변동성의 가장 큰 축을 찾음
  - LDA : 입력 데이터의 결정 값 클래스를 최대한 분리할 수 있는 축을 찾음

- 클래스 간 분산과 클래스 내부 분산의 비율을 최대화하는 방식
  - 클래스 간 분산은 최대한 크게
  - 클래스 내부의 분산은 최대한 작게



- LDA 구현 스텝
  1. 클래스 간 분산과 클래스 내부 분산 행렬 생성

2. 이 행렬에 기반한 고유벡터를 구하고 입력 데이터를 투영

1. 클래스 내부와 클래스 간 분산 행렬을 구합니다. 이 두 개의 행렬은 입력 데이터의 결정 값 클래스별로 개별 피처의 평균 벡터(mean vector)를 기반으로 구합니다.
2. 클래스 내부 분산 행렬을 S_W, 클래스 간 분산 행렬을 S_B라고 하면 다음 식으로 두 행렬을 고유벡터로 분해할 수 있습니다.

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

3. 고유값이 가장 큰 순으로 K개(LDA변환 차수만큼) 추출합니다.
4. 고유값이 가장 큰 순으로 추출된 고유벡터를 이용해 새롭게 입력 데이터를 변환합니다.

2) 붓꽃 데이터 세트에 LDA 적용하기

- 사이킷런의 LDA : LinearDiscriminantAnalysis클래스

> 📣 **PCA 유의점**
>
> PCA와 다르게 지도학습이라는 것 !
>
> 클래스의 결정값이 변환 시에 필요

- lda 객체의 fit() 메서드를 호출할 때 결정값이 입력됐음에 유의하자

04. SVD(Singular Value Decomposition)

1) SVD 개요

📌 **SVD(Singular Value Decomposition)**

PCA와 유사한 행렬 분해 기법 사용

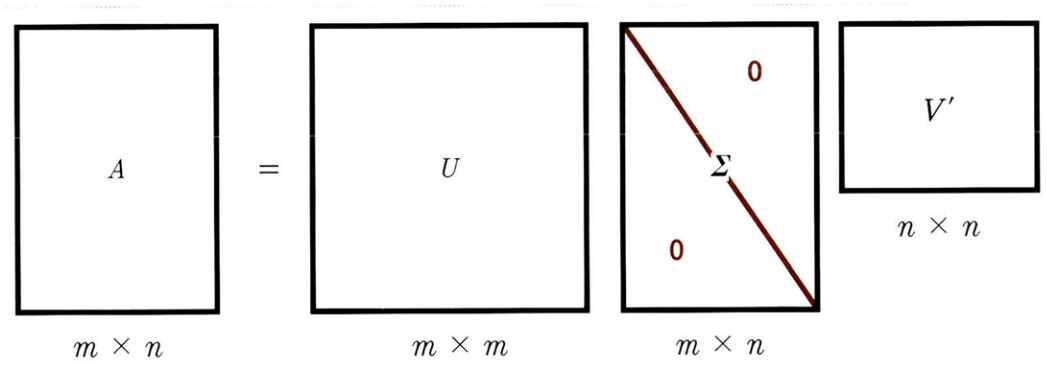
SVD는 정방행렬 뿐만 아니라 행과 열의 크기가 다른 행렬에도 적용

- 일반적으로 SVD는 m x n 크기의 행렬 A를 다음과 같이 분해

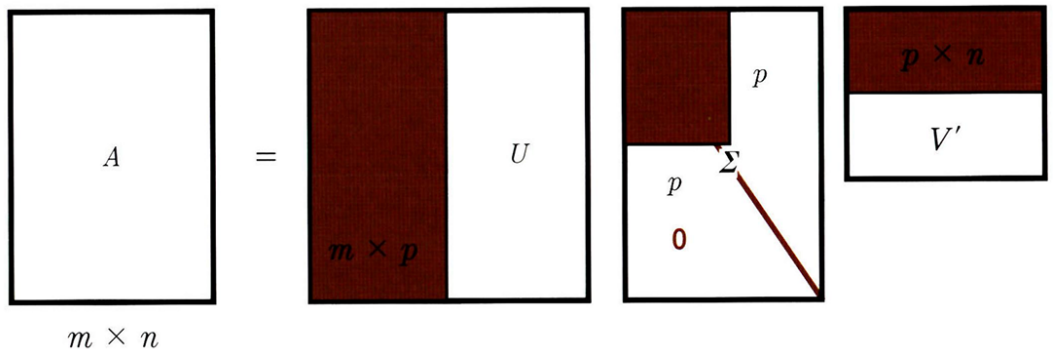
$$A = U \sum V^T$$

- 특이값 분해라고도 불림
- 행렬 U와 V에 속하는 벡터는 특이벡터(singular vector)
- 모든 특이 벡터는 서로 직교하는 성질
- Σ는 대각행렬, 행렬의 대각에 위치한 값만 0이 아니고 나머지 위치 값은 모두 0
- Σ는 다 위치한 0이 아닌 값이 행렬 A의 특이값

- SVD는 A의 차원이 m x n일 때



- 일반적으로 다음과 같이 Σ의 비대각 부분과 대각원소 중 특이값이 0인 부분도 모두 제거
- 제거된 Σ에 대응되는 U와 V원소도 함께 제거해 차원을 줄인 형태로 SVD적용
- 이러한 컴팩츠한 형태로 SVD 적용



- Truncated SVD는 Σ의 대각원소 중에 상위 몇 개만 추출해서 여기에 대응하는 U와 V의 원소도 함께 제거해 더욱 차원을 줄인 형태로 분해
- 보통 SVD는 넘파이나 사이파이 라이브러리 이용

- Sigma행렬의 경우 Σ행렬을 나타내며, 행렬의 대각에 위치한 값만 0이 아님

  그렇지 않은 경우는 모두 0

  0이 아닌 값만 1차원 행렬로 표현(대각에 위치한 값)
- 원본 행렬로 복원할 때 유의점
  - Sigma의 경우 0이 아닌 값만 1차원으로 추출

    ⇒ 다시 0을 포함한 대칭행렬로 변환 후 내적 수행

- Truncated SVD
  - Σ행렬에 있는 대각원소, 즉 특이값 중 상위 일부 데이터만 추출해 분해하는 방식
  - 인위적으로 더 작은 차원의 U, Σ, V^T로 분해 → 원본 행렬로 정확히 복원 X
  - 상당한 수준으로 원본 행렬에 근사 가능

    (원래 차원의 차수에 가깝게 잘라낼수록 원본 행렬에 더 가깝게 복원)

2) 사이킷런 TruncatedSVD 클래스를 이용한 변환

- 사이킷런의 TruncatedSVD 클래스

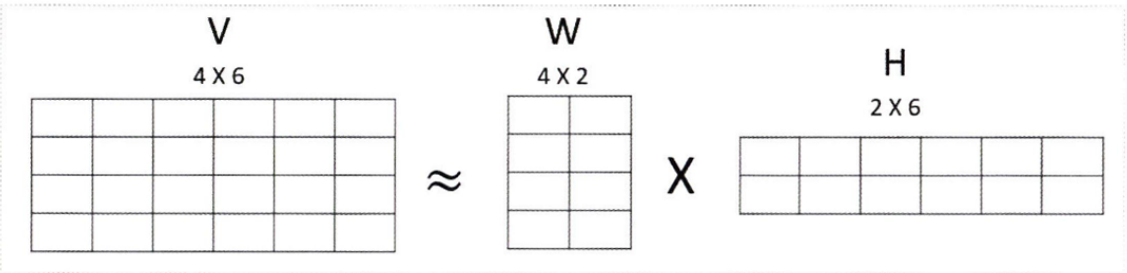
    - 사이파이의 svds와 같이 Truncated SVD 연산을 수행해 원본 행렬을 분해한 U, Sigma, Vt 행렬 반환 X
    - PCA 클래스와 유사하게 fit()와 transform()을 호출해 원본 데이터를 몇 개의 주요 컴포넌트로 차원을 축소해 변환
    - 원본 데이터를 Truncated SVD 방식으로 분해된 U*Sigma 행렬에 선형 변환해 생성
- 데이터 세트가 스케일링으로 데이터 중심이 동일해지면 사이킷런의 SVD와 PCA는 동일한 변환을 수행 → PCA가 SVD 알고리즘으로 구현됨
- PCA는 밀집 행렬에 대한 변환만 가능, SVD는 희소 행렬에 대한 변환도 가능
- SVD는 컴퓨터 비전 영역에서 이미지 압축을 통한 패턴 인식과 신호 처리 분야에 사용
- 텍스트의 토픽 모델링 기법인 LSA의 기반 알고리즘

05. NMF(Non-Negative Matrix Factorization)

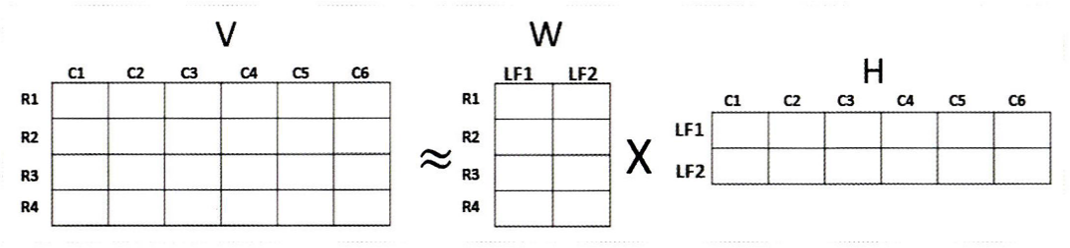
1) NMF 개요

> 📌 **NMF(Non-Negative Matrix Factorization)**
>
> 낮은 랭크를 통한 행렬 근사(Low-Rank Approximation) 방식의 변형

- 원본 행렬 내의 모든 원소 값이 모두 양수가 보장되면 다음과 같이 좀 더 간단하게 두 개의 기반 양수 행렬로 분해될 수 있는 기법을 지칭



- 행렬 분해(Matrix Factorization)는 일반적으로 SVD와 같은 행렬 분해 기법을 통칭
  - 행렬 분해를 하게 되면 일반적으로 길고 가는 행렬 W와 작고 넓은 행렬 H로 분해됨
  - 분해된 행렬은 잠재 요소를 특성으로 가짐
  - 분해 행렬 W는 잠재 요소가 원본 행에 대응, 분해 행렬 H는 잠재요소가 원본 열에 대응



- NMF는 SVD와 유사하게 차원 축소를 통한 잠재 요소 도출로 이미지 변환 및 압축, 텍스트의 토픽 도출 등의 영역에서 사용

- 사이킷런에서 NMF : NMF클래스

- 이미지 압축을 통한 패턴 인식, 텍스트의 토픽 모델링 기법, 문사 유사도 및 클러스터링에 자주 사용
- 영화 추천과 같은 추천 영역에 활발히 적용
  - 높은 순위로 예측된 상품 추천 방식 : 잠재 요소 기반의 추천 방식