



# Chap6. 차원 축소

## 1. 차원 축소(Dimension Reduction) 개요



대표적인 차원 축소 알고리즘

→ PCA, LDA, SVD, NMF

### ? 차원 축소 ?

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

→ 수백 개 이상의 피처로 구성된 데이터 세트의 경우 상대적으로 적은 차원에서 학습된 모델보다 예측 신뢰도가 떨어짐

→ 피처가 많을 수록 개별 피처 간에 상관관계 높을 가능성 有(선형 모델에서는 피처 간 상관관계가 높을수록 다중 공선성 문제로 인해 예측 성능 저하)

⇒ 차원 축소를 통해 피처 개수를 줄이면 직관적으로 데이터 해석 가능

⇒ 차원 축소를 통해 좀 더 데이터를 잘 설명할 수 있는 잠재적인 요소를 추출

(이미지, 텍스트에서 차원 축소를 통해 잠재적인 의미를 찾아주는 데 PCA, SVD, NMF 등이 많이 사용)

- 차원 축소 - 피처 선택, 피처 추출

### 1. 피처 선택

→ 특정 피처에 종속성이 강한 불필요한 피처는 아예 제거하고, 데이터의 특징을 잘 나타내는 주요 피처만 선택

⇒ 기존 피처를 저차원의 중요 피처로 압축해서 추출하는 것(기존의 피처가 압축된 것이므로 기존의 피처와는 완전히 다른 값이 됨)

## 2. 피처 추출

→ 단순 압축이 아닌 피처를 함축적으로 더 잘 설명할 수 있는 또 다른 공간으로 매핑해 추출 (모의고사 성적, 종합 내신성적, 수능 성적, 봉사활동 → 학업 성취도, 커뮤니케이션 능력, 문제 해결력 등으로 함축적인 요약 특성으로 추출)

⇒ 기존 피처가 전혀 인지하기 어려웠던 잠재적인 요소를 추출하는 것

## 2. PCA(Principal Component Analysis)

⇒ 여러 변수 간에 존재하는 상관관계를 이용해 이를 대표하는 주성분(Principal Component)을 추출해 차원을 축소하는 기법

가장 높은 분산을 가지는 데이터(PCA의 주성분)의 축을 찾아 이 축으로 차원을 축소함

→ 분산이 데이터의 특성을 가장 잘 나타내는 것으로 간주

⇒ 원본 데이터의 피처 개수에 비해 매우 작은 주성분으로 원본 데이터의 총 변동성을 대부분 설명할 수 있는 분석법

- <선형대수 관점>

⇒ 입력 데이터의 공분산 행렬을 고유값 분해하고 구한 고유 벡터에 입력 데이터를 선형 변환하는 것

**고유벡터** - PCA의 주성분 벡터, 입력 데이터의 분산이 큰 방향을 나타냄, 어떤 행렬을 곱하더라도 방향은 변하지 않고 크기만 변화, 정방 행렬은 최대 그 차원 수 만큼의 고유벡터를 가짐(2x2이면 2개의 고유 벡터) ⇒ 행렬이 작용하는 힘의 방향과 관계가 있어 행렬을 분해하는데 사용

**고유값** - 고유벡터의 크기, 동시에 입력 데이터의 분산을 나타냄

**선형 변환** - 특정 벡터에 행렬 A를 곱해 새로운 벡터로 변환하는 것

**공분산 행렬** - 여러 변수와 관련된 공분산을 포함하는 정방(열과 행이 같은 행렬) 및 대칭 행렬(대각 원소를 중심으로 원소 값이 대칭되는 행렬) ⇒ 대칭 행렬은 항상 고유벡터를 직교 행렬로, 고유값을 정방 행렬로 대각화 할 수 있음

$$C = P\Sigma P^T$$

1. 입력 데이터 세트의 공분산 행렬 생성
2. 공분산 행렬의 고유벡터와 고유값 계산
3. 고유값이 가장 큰 순으로 K개(PCA 변환 차수만큼)만큼 고유벡터 추출
4. 고유값이 가장 큰 순으로 추출된 고유벡터를 이용해 새롭게 입력 데이터 변환

### 3. LDA(Linear Discriminant Analysis)

⇒ 선형 판별 분석법, PCA와 유사하게 입력 데이터 세트를 저차원 공간에 투영해 차원을 축소하지만, LDA는 지도학습의 분류(Classification)에서 사용하기 쉽도록 개별 클래스를 분별할 수 있는 기준을 최대한 유지하며 차원을 축소

(PCA는 입력 데이터의 변동성이 가장 큰 축을 찾지만 LDA는 입력 데이터의 결정 값 클래스를 최대한으로 분리할 수 있는 축을 찾음)

⇒ 클래스 분리를 최대화하는 축을 찾기 위해 클래스 간 분산(between-class scatter)과 클래스 내부 분산(within-class scatter)의 비율을 최대화하는 방식으로 차원 축소

(클래스 간 분산은 최대한 크게, 클래스 내부 분산은 최대한 작게)

PCA와 다르게 공분산 행렬이 아닌 클래스 간 분산과 클래스 내부 분산 행렬을 생성한 뒤 이 행렬에 기반해 고유벡터를 구하고 입력 데이터를 투영함

1. 클래스 내부와 클래스 간 분산 행렬을 구함. 이 두 개의 행렬은 입력 데이터의 결정 값 클래스별로 개별 피처의 평균 벡터(mean vector)를 기반으로 구함.
2. 클래스 내부 분산 행렬을  $S_w$ , 클래스 간 분산 행렬을  $S_b$ 라고 하면, 다음 식으로 두 행렬을 고유벡터로 분해할 수 있음
3. 고유값이 가장 큰 순으로 K개(LDA 변환 차수만큼) 추출
4. 고유값이 가장 큰 순으로 추출된 고유벡터를 이용해 새롭게 입력 데이터 변환

### 4. SVD(Singular Value Decomposition)

⇒ PCA와 유사한 행렬 분해 기법 이용, PCA의 경우 정방행렬만을 고유 벡터로 분해할 수 있지만 SVD는 정방행렬 뿐만 아니라 행과 열의 크기가 다른 행렬에도 적용할 수 있음

(일반적으로 SVD는  $m \times n$  크기의 행렬  $A$ 를 다음과 같이 분해하는 것을 의미)

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

행렬  $U$ 와  $V$ 에 속한 벡터는 특이벡터(singular vector), 모든 특이벡터는 서로 직교하는 성질

$\Sigma$ 는 대각행렬, 행렬 대각에 위치한 값만 0이 아니고 나머지 위치의 값을 모두 0

$\Rightarrow \Sigma$ 에 위치한 0이 아닌 값이 바로 행렬  $A$ 의 특이값

SVD는  $A$ 의 차원이  $m \times n$ 일 때,  $U$ 의 차원을  $m \times m$ ,  $\Sigma$ 의 차원을  $m \times n$ ,  $V^T$ 의 차원을  $n \times n$ 으로 분해

$$\begin{array}{c}
 \boxed{A_{m \times n}} = \boxed{U_{m \times m}} \times \boxed{\Sigma_{m \times n}} \times \boxed{V_{n \times n}^T} \\
 (m < n)
 \end{array}$$

$$\begin{array}{c}
 \boxed{A_{m \times n}} = \boxed{U_{m \times m}} \times \boxed{\Sigma_{m \times n}} \times \boxed{V_{n \times n}^T} \\
 (m > n)
 \end{array}$$

✚ Truncated SVD

## Truncated SVD

$$A' = U_t \Sigma_t V_t^T$$

$\Sigma$ 의 대각원소 중 상위 몇 개( $t$ 개)만 추출하여 여기에 대응하는  $U$ 와  $V$ 의 원소도 함께 제거해 더욱 차원을 줄인 형태로 분해하는 것

$U_t$ 의 크기는  $m \times t$ 이며,  $\Sigma_t$ 의 크기는  $t \times t$ , 그리고 Transpose of  $V_t$ 의 크기는  $t \times n$

Truncated SVD로 한번 분해를 하면 원본 행렬인  $A$ 로 원복할 수 없음

(Full SVD는 분해를 해도 모든 원소가 남아있기 때문에 원본 행렬  $A$ 로 원복이 가능하지만 Truncated SVD는 원소를 손실하기 때문에 원복이 불가함, 근사하게는 일치)

$t$ 가 클수록 원래의 행렬  $A$ 와 가까워지며,  $t$ 가 작아질수록 원본 행렬  $A$ 와 차이가 많이 남.

## 5. NMF(Non-Negative Matrix Factorization)

⇒ Truncated SVD와 같이 낮은 랭크를 통한 행렬 근사(Low-Rank Approximation) 방식의 변형

**원본 행렬 내의 모든 원소 값이 모두 양수**라는게 보장되면 좀 더 간단하게 두 개의 기반 양수 행렬로 분해될 수 있는 기법

SVD와 NMF는 매우 많은 피쳐 데이터를 가진 고차원 행렬을 두 개의 저차원 행렬로 분리하는 행렬 분해 기법 → 원본 행렬에서 잠재된 요소를 추출하기에 토픽 모델링이나 추천 시스템에서 활발하게 사용