

# 6장 차원 축소

## 1. 차원 축소 개요

다차원의 데이터 세트의 차원을 축소해 새로운 차원의 데이터 세트를 생성하는 것을 말함.  
왜?

차원이 증가할수록 포인트 간의 거리가 멀어져서 희소한 구조를 가지게 됨

→ 예측 신뢰도가 떨어지는 결과 발생

→ 선형회귀같은 선형 모델에서는 다중공선성 문제 발생

피쳐 선택 vs 피쳐 추출.

피쳐 선택: 중요하지 않은 피쳐 제거

피쳐 추출: 기존 피쳐를 조합/ 변환해 새로운 피쳐로 표현

사용하는 영역 → 이미지 분류, 문서나 텍스트 숨은 의미 찾아내기 위해

## 2. PCA (Principal Component Analysis, 주성분 분석)

- 데이터의 **분산(Variance)** 이 가장 큰 방향(축)을 찾아 그 방향으로 차원을 줄임. 그 다음, 이 벡터의 직각이 되는 벡터를 축으로 함.
- **정보 손실 최소화 + 고유벡터(Eigenvector)** 이용.

### - PCA 수학적 흐름

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

고유벡터 = 데이터 분산이 큰 방향

고유값 = 그 방향의 분산 크기

### 실습 예시 – 붓꽃 데이터 (Iris)

### 1. 표준화(StandardScaler) 적용

→ PCA는 변수 스케일에 영향을 받음

### 2. `PCA(n_components=2)` 로 4차원을 2차원으로 축소

### 3. 시각화:

- Setosa는 잘 구분됨
- Versicolor와 Virginica는 일부 겹침

### 4. 설명된 분산 비율(explained\_variance\_ratio\_)

→ `[0.7296, 0.2285]`

→ 두 성분으로 전체 변동성의 **95% 이상 설명**

### 5. 성능 비교

데이터	평균 정확도
원본 4차원	0.96
PCA 2차원	0.88

| 약 8% 성능 하락이 있지만, 속성이 절반으로 줄어 효율성↑

## 신용카드 데이터 예시

- 속성이 23개인 데이터에서 상관관계가 높은 `BILL_AMT1~6` 속성 6개를  
**PCA 2개 성분으로 축소** → 변동성 95% 유지
- 전체 속성의 1/4 수준인 **6개의 컴포넌트만으로** 정확도 거의 동일 (약 1~2% 감소)

| PCA는 데이터 압축 능력이 뛰어나며, 이미지 인식(예: Eigen-face) 등에 활용됨.

## 3. LDA (Linear Discriminant Analysis, 선형 판별 분석)

- 지도학습용 차원 축소 기법
- 클래스 간 분산(Between-class) 은 크게, 클래스 내부 분산(Within-class) 은 작게 만드는 방향(축)을 찾음.
- 주의해야 할 점: 지도학습임. 즉, 결정값이 변환 시에 필요함.

비교	PCA	LDA
학습유형	비지도	지도

비교	PCA	LDA
목표	분산이 큰 방향	클래스 구분이 잘 되는 방향

## -수행 절차

1. 클래스별 평균벡터 계산
2. 클래스 내부/클래스 간 분산 행렬 생성
3. 두 행렬을 이용해 고유벡터 계산
4. 가장 큰 고유값 방향으로 데이터 투영

## 실습예시- 붓꽃 데이터 예시

- `LinearDiscriminantAnalysis(n_components=2)`
- 시각화: PCA와 유사하나 클래스 간 구분이 더 명확

## 4. SVD (Singular Value Decomposition, 특이값 분해)

### 개념 요약

- PCA와 비슷하지만, **정방행렬이 아닌 임의의 행렬**에도 적용 가능.
- 행렬 A를 다음으로 분해:

$$A = U \times \Sigma \times V^t$$

- **U, V**: 직교행렬 (특이벡터)
- **Σ(Sigma)**: 대각행렬 (특이값)

| Sigma 값이 크면 데이터의 중요한 방향 → 차원 축소 시 이 값 기준으로 선택.

### Truncated SVD

- Σ의 상위 몇 개 특이값만 사용하여 **차원 축소**
- 완벽 복원은 불가능하지만, **근사적 복원** 가능
- **희소 행렬(Sparse Matrix)** 에도 사용 가능
- **텍스트 토픽 모델링 (LSA)** 의 핵심 기반 알고리즘

## Scikit-learn

- `TruncatedSVD(n_components=k)`
- `fit()` + `transform()` → PCA와 거의 동일한 사용법
- 데이터가 표준화되어 있으면 PCA와 결과 거의 동일함.

## 5. NMF (Non-Negative Matrix Factorization, 비음수 행렬 분해)

### 개념 요약

- 모든 값이 **양수(0 이상)** 인 행렬만 분해 가능
- 원본 행렬 **V** 를 두 개의 양수 행렬 **W** 와 **H** 로 근사:

$$V \approx W \times H$$

- **W**: 각 데이터가 잠재요소를 얼마나 가지는가
- **H**: 잠재요소가 원본 속성에 얼마나 기여하는가

### 특징

- 데이터의 **잠재 요인(Latent Factor)** 분석에 강함
- **이미지 압축, 텍스트 토픽 모델링, 추천 시스템(Recommendation)** 에 자주 사용됨  
→ 예: 영화 평가 행렬을 분해해 **사용자 취향**과 **영화 특성**을 추출

알고리즘	학습유형	수학적 기반	주요 목적	활용 예시
<b>PCA</b>	비지도	공분산 행렬, 고유값 분해	분산 최대 방향 찾기	이미지 압축, 시각화
<b>LDA</b>	지도	클래스 간 분산 비율	클래스 구분 극대화	분류 전처리
<b>SVD</b>	비지도	특이값 분해	일반 행렬 차원 축소	텍스트 LSA, 희소 데이터
<b>NMF</b>	비지도	비음수 행렬 분해	잠재요인 추출	추천 시스템, 토픽 모델링