6장 차원 축소

날짜	@2025년 11월 10일
카테고리	개념 정리

01 차원 축소 Dimension Reduction 개요

개념

- 많은 피처(feature)로 구성된 **다차원(고차원) 데이터**를 **낮은 차원**으로 줄이는 기법
- 차원을 줄이면서 **데이터의 주요 정보(변동성)**를 최대한 보존

다차원/고차원 데이터 문제점

- **데이터 희소성(Sparsity) / 희소한 구조** : 차원이 커질수록 데이터 포인트 간 거리가 멀어짐
- **다중 공선성(Multicollinearity) 문제** : 피처 간 상관관계가 커져 선형 모델 예측력 저하
- **과적합(Overfitting)** : 불필요한 피처가 많을수록 학습 데이터에 과하게 맞춰짐
- **시각화 어려움** : 3차원 이상은 직관적 시각화가 불가능

차원 축소의 목적

- **모델 성능 향상** : 불필요한 피처 제거로 일반화 성능 개선
- **시각화 가능성** : 2~3차원 공간으로 데이터 구조 파악
- **연산 효율성 증가** : 계산량 및 저장공간 감소

차원 축소 방법

구분	설명	대표 알고리즘
Feature Selection (특성 선택)	중요하지 않거나 중복된 피처 제거	필터, 래퍼, 임베디드 기법
Feature Extraction (특성 추출)	피처를 변환·압축해 새로운 저차원 특성 생성	**PCA, LDA, SVD, NMF**

- **피처 추출**
  : 기존 피처를 단순 압축 X, 피처를 함축적으로 더 잘 설명할 수 있는 또 다른 공간으로 '매핑'해 추출하는 것
  - ex) 학생 평가 요소: 모의고사 성적, 종합 내신성적, 수능성적, 봉사활동, 대외활동 ...
    → 학업 성취도 / 커뮤니케이션 능력 / 문제 해결력 등 더 함축적인 **요약 특성**으로 추출
    → 기존 피처가 인지하기 어려웠던 잠재적인 요소 (Latent Factor)를 추출하는 것

영역

1. **이미지 데이터**
    - 많은 픽셀 데이터 … 이미지 데이터에서 잠재된 특성 피처로 도출
      → 함축적 형태의 이미지 변환 & 압축
      ⇒ 과적합 영향력 적어져 예측 성능 끌어올림
2. **텍스트 문서의 숨겨진 의미 추출**
    - 단어들의 구성에 숨겨져 있는 시맨틱 의미나 토픽을 잠재 요소로 간주하고 찾기 가능
    - SVD, NMF … 시맨틱토픽 모델링을 위한 기반 알고리즘으로 사용됨

---

02 PCA (Principal Component Analysis)

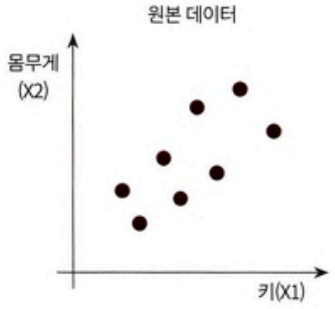
PCA 개요

- 서로 **상관관계가 높은 다차원 데이터를 선형 결합**하여,
  상관성이 없는, 대표하는 주성분(Principal Components) 을 추출하는 방법
- 데이터의 분산(variance)이 **가장 큰 방향**을 찾아 축을 재설정
- 데이터의 정보 손실을 최소화하면서 **차원을 축소**

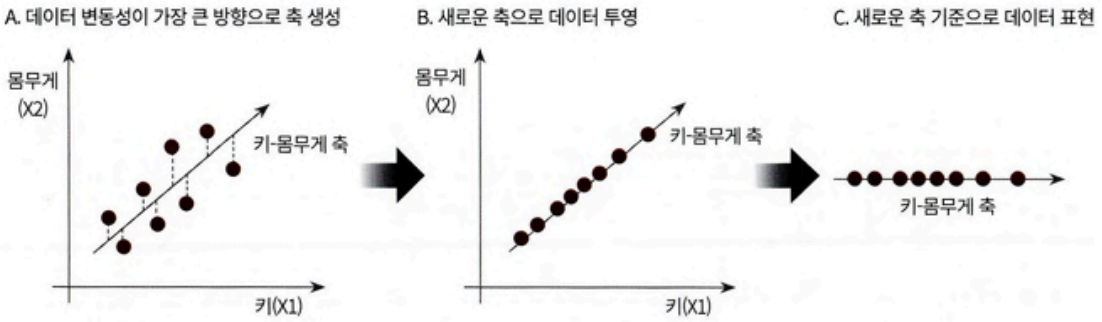
ex) 키- 몸무게 feature

→ 2개의 피처를 한 개의 주성분을 가진 데이터 세트로 차원 축소

- 데이터 변동성이 가장 큰 방향을 축으로 생성
- 새롭게 생성된 축으로 데이터를 투영하는 방식

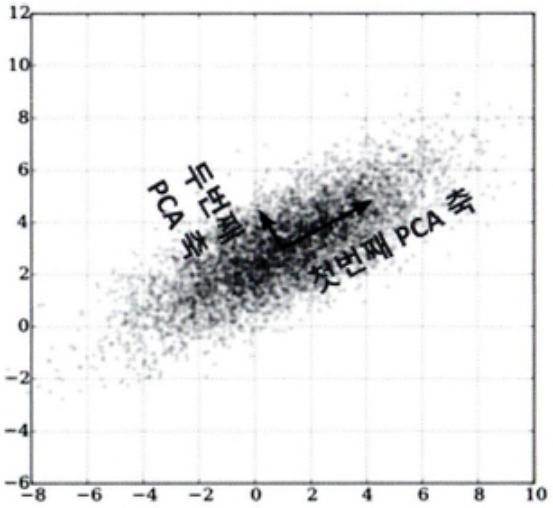


이 2개의 피처를 한 개의 주성분을 가진 데이터 세트로 차원 축소를 할 수 있습니다. 데이터 변동성이 가장 큰 방향으로 축을 생성하고, 새롭게 생성된 축으로 데이터를 투영하는 방식입니다.



PCA 과정

1. 입력 데이터의 **정규화 (Standardization)**
2. **공분산 행렬(Covariance Matrix)** 계산 / 생성
   : 두 변수 간의 변동을 의미
3. **공분산 행렬의 고유 백터와 고유값 계산**
4. 고유값이 가장 큰 순으로 K개 (PCA 변환 차수만큼) **고유 벡터 추출**
5. 고유값이 가장 큰 순으로 추출된 고유벡터를 이용해, 새롭게 입력 데이터 변환
   = **원본 데이터를 새로운 축(주성분)으로 변환**



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

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

공분산 행렬

- 공분산은 두 변수 간의 변동 Cov(X, Y) > 0 이면 X 증가할 때 Y도 증가한다는 의미
  - 정방행렬, 대칭행렬

POINT

- **첫 번째 주성분**: 분산이 가장 큰 방향
- **두 번째 주성분**: 첫 번째 주성분과 직교하며, 남은 분산이 큰 방향
- 이후 주성분들도 동일한 원리로 생성
- `n_components` 매개변수를 통해 축소 차원 수 지정

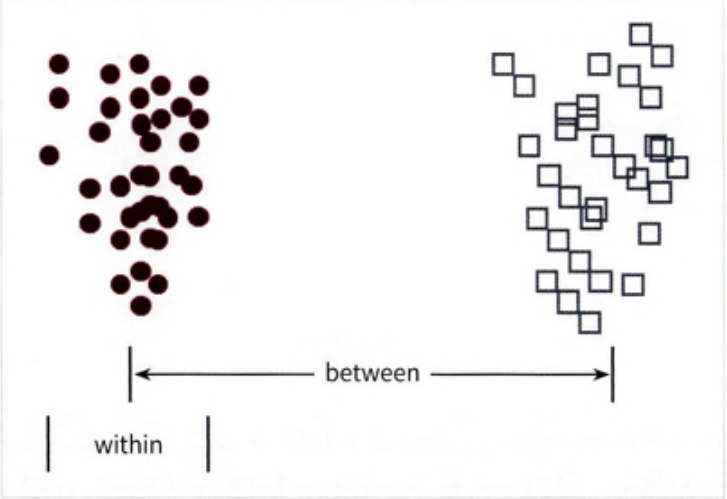
---

03 LDA (Linear Discriminant Analysis)

LDA 개요

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

- **PCA와의 공통점** : 입력 데이터 세트를 저차원 공간으로 투영하여 차원 축소 수행
- **PCA와의 차이점**
  - PCA → 데이터의 **분산(Variance)** 이 가장 큰 축을 찾음 (비지도 학습)
  - LDA → **클래스 간 분리를 최대화**하는 축을 찾음 (지도 학습)



- **목표** : 클래스 간 분산(between-class scatter)은 최대화하고, 클래스 내 분산(within-class scatter)은 최소화
- 즉, **서로 다른 클래스는 멀리**, **같은 클래스는 가깝게** 배치되도록 투영

> 좋은 LDA 변환 → 클래스 간 분산 ↑, 클래스 내 분산 ↓

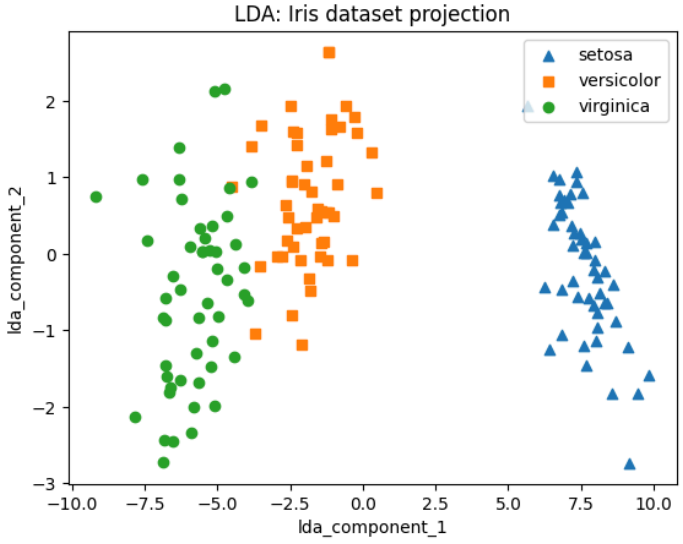
LDA 수행 단계

- 각 클래스별 **평균 벡터(mean vector)** 계산
- 이를 바탕으로 **클래스 내부 분산 행렬**과 **클래스 간 분산 행렬** 계산
- 두 행렬의 고유벡터(eigenvector)와 고유값(eigenvalue) 계산
- **가장 큰 고유값**에 해당하는 **K개 고유벡터**를 선택
- 선택된 고유벡터로 데이터를 새로운 축에 **투영(transform)**

붓꽃 데이터 세트에 LDA 적용

```
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.preprocessing import StandardScaler
from sklearn.datasets import load_iris

# 데이터 로드 및 스케일링
iris = load_iris()
iris_scaled = StandardScaler().fit_transform(iris.data)
```

```
lda = LinearDiscriminantAnalysis(n_components=2)
lda.fit(iris_scaled, iris.target)
iris_lda = lda.transform(iris_scaled)
print(iris_lda.shape)

import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline

# LDA 결과를 DataFrame으로 변환
lda_columns = ['lda_component_1', 'lda_component_2']
irisDF_lda = pd.DataFrame(iris_lda, columns=lda_columns)
irisDF_lda['target'] = iris.target

# 클래스별 산점도 시각화
markers = ['^', 's', 'o']  # setosa, versicolor, virginica

for i, marker in enumerate(markers):
    x_axis = irisDF_lda[irisDF_lda['target'] == i]['lda_component_1']
    y_axis = irisDF_lda[irisDF_lda['target'] == i]['lda_component_2']
    plt.scatter(x_axis, y_axis, marker=marker, label=iris.target_names[i])

plt.legend(loc='upper right')
plt.xlabel('lda_component_1')
plt.ylabel('lda_component_2')
plt.title('LDA: Iris dataset projection')
plt.show()
```



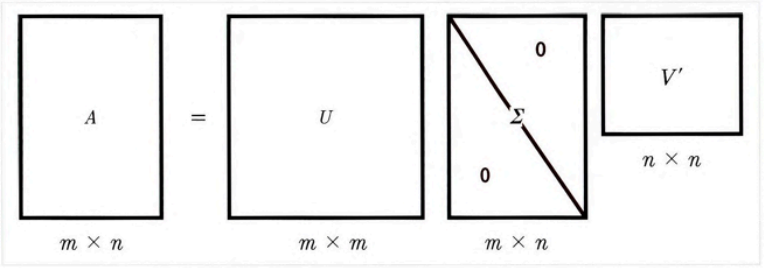
04 SVD (Singular Value Decomposition)

SVD 개요

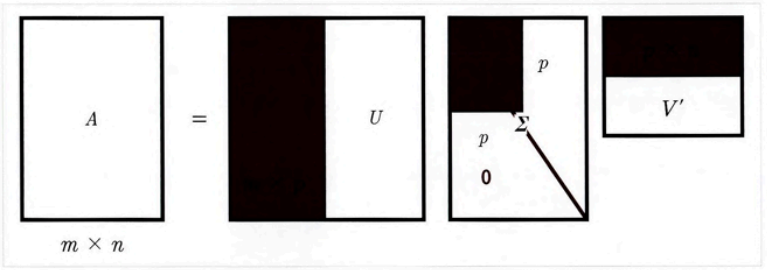
- **SVD(특이값 분해)** 는 PCA와 매우 밀접한 관계를 가진 **행렬 분해(Matrix Factorization)** 기법
- 복잡한 고차원 데이터를 더 단순한 형태로 분해하여,

  차원 축소, **노이즈 제거**, **데이터 압축**, **잠재 요인 분석(latent factor analysis)** 등에 널리 활용

** 수학 개념 참고 **



하지만 일반적으로는 다음과 같이 Σ의 비대각인 부분과 대각원소 중에 특이값이 0인 부분도 모두 제거하고 제거된 Σ에 대응되는 U와 V 원소도 함께 제거해 차원을 줄인 형태로 SVD를 적용합니다. 이렇게 컴팩트한 형태로 SVD를 적용하면 A의 차원이 m × n일 때, U의 차원을 m × p, Σ의 차원을 p × p, V^T의 차원을 p × n으로 분해합니다.



SVD는 임의의 행렬 A 를 다음과 같이 세 행렬의 곱으로 분해합니다.

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

- A: m×n 행렬 (원본 데이터)
- U: m×m 직교 행렬 (왼쪽 특이벡터, left singular vectors)
- Σ: m×n 대각 행렬 (특이값 singular values)
- V^T: n×n 직교 행렬 (오른쪽 특이벡터, right singular vectors)

- 특이값(σ)은 항상 **양수이며 내림차순**으로 정렬
- 큰 특이값일수록 데이터의 **변동성(variance)** 을 잘 설명
- PCA는 공분산 행렬을 SVD로 분해한 것과 수학적으로 동일한 결과를 냄

Key

- SVD는 **데이터의 근본적인 구조를 파악**하고

  불필요한 잡음(noise)을 제거하는 데 매우 효과적
- **PCA, 추천시스템, 이미지 압축, 자연어 처리(LSA)** 등

  다양한 분야에서 핵심적인 역할을 함
- SVD는 PCA보다 **수학적으로 일반적이며, 더 다양한 행렬에 적용 가능**

사이킷런 TruncatedSVD 클래스를 이용한 변환

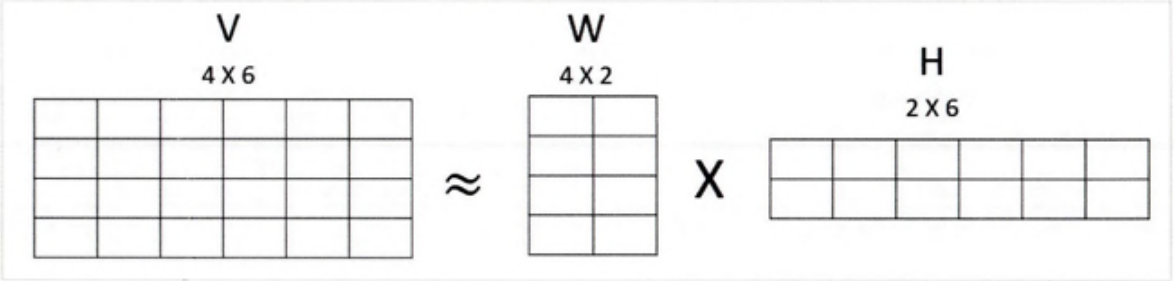
- 사이파이의 svds와 같이 Truncated SVD 연산을 수행해 원본 행렬을 분해한 U, Sigma, Vt 행렬을 반환하지 않음
- PCA 클래스와 유사하게 fit() transform()을 호출해 원본 데이터를 몇 개의 주요 컴포넌트로 차원 축소해 변환

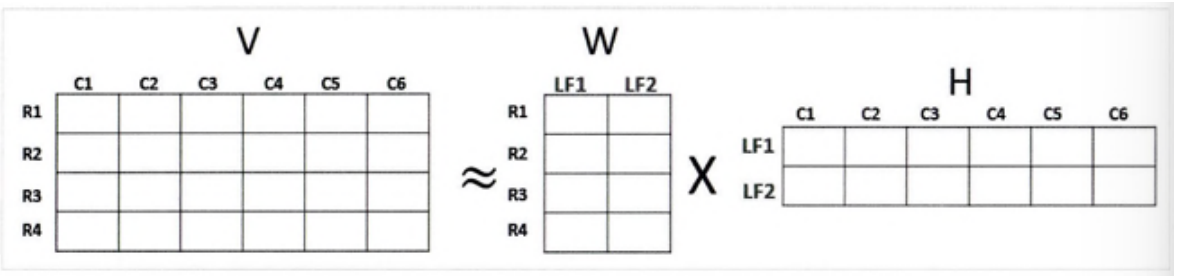
---

05 NMF (Non-Negative Matrix Factorization)

NMF 개요

- Truncated SVD와 같이 낮은 랭크를 통한 행렬 근사방식의 변형
- 원본 행렬 내의 모든 원소 값이 모두 양수라는 게 보장되면, 두 개의 기반 양수 행렬로 분해될 수 있는 기법을 지칭함
- SVD와 유사하게 차원 축소를 통한 잠재 요소 도출
  - 이미지 변환 및 압축
  - 텍스트의 토픽 도출에 활





정리

- PCA, LDA, SVD, NMF
- 많은 피처로 이뤄진 데이터 세트를 차원 축소를 통해 직관적으로 이해 가능
- 단순히 피처 개수를 줄이는 개념보다는 이를 통해 데이터를 잘 설명할 수 있는 <잠재적 요소>를 추출하는 데 큰 의미가 있음
- 이미지 / 텍스트 (많은 차원 가지는 데이터)에서 차원 축소 알고리즘이 활발히 사용됨
- `PCA` 입력 데이터의 변동성이 가장 큰 축을 구하고 → 이 축에 직각인 축을 반복적으로 축소하는 차원 개수만큼 구한뒤 → 입력 데이터를 이 축들에 투영해 차원 축소하는 방식
- `LDA` PCA와 유사한 방식, 입력 데이터의 결정값 클래스를 최대한을 분리할 수 있는 축을 찾는 방식으로 차원 축소
- `SVD` & `NMF` 매우 많은 피처 데이터를 가진 고차원 행렬을 두 개의 저차원 행렬로 분리하는 행렬 분해 기법