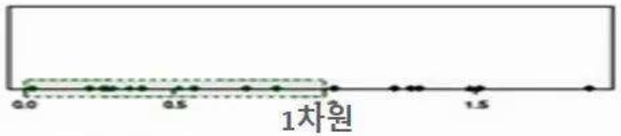
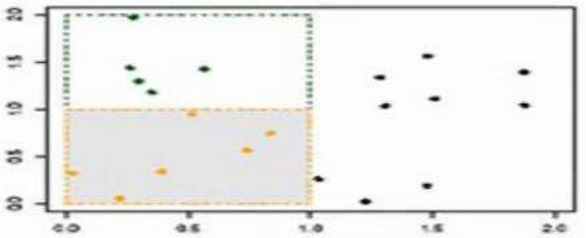
6장 차원 축소(Dimension Reduction)

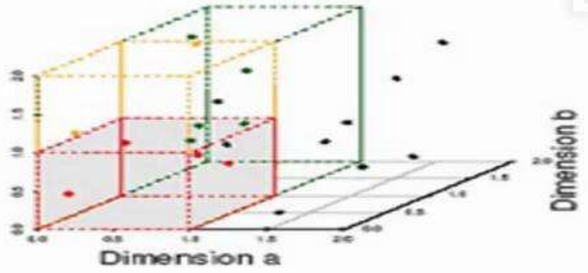
inflearn
4445185
[비공식] 파이썬 머신러닝 완벽 가이드
권 철민

차원의 저주





특정 면적 공간에 6개의 데이터



특정 입체 공간에 4개의 데이터

차원이 커질수록

- 데이터 포인트들간 거리가 크게 늘어남
- 데이터가 희소화(Sparse) 됨

- 수백~수천개 이상의 피처로 구성된 포인트들간 거리에 기반한 ML 알고리즘이 무력화됨.
- 또한 피처가 많을 경우 개별 피처간에 상관관계가 높아 선형 회귀와 같은 모델에서는 다중 공선성 문제로 모델의 예측 성능이 저하될 가능성이 높음

1445185
[개정판] 파이썬 머신러닝 완벽 가이드

차원 축소의 장점

수십~수백개의 피처들을 작은 수의 피처들로 축소한다면 ?

- 학습 데이터 크기를 줄여서 학습 시간 절약
- 불필요한 피처들을 줄여서 모델 성능 향상에 기여(주로 이미지 관련 데이터)
- 다차원의 데이터를 3차원 이하의 차원 축소를 통해서 시각적으로 보다 쉽게 데이터 패턴 인지

[개정판] 파이썬 머신러닝 완벽 가이드
권 철민

어떻게 하면 원본 데이터의 정보를 최대한으로 유지한 채로 차원 축소를 수행할 것인가?

피처 선택과 피처 추출

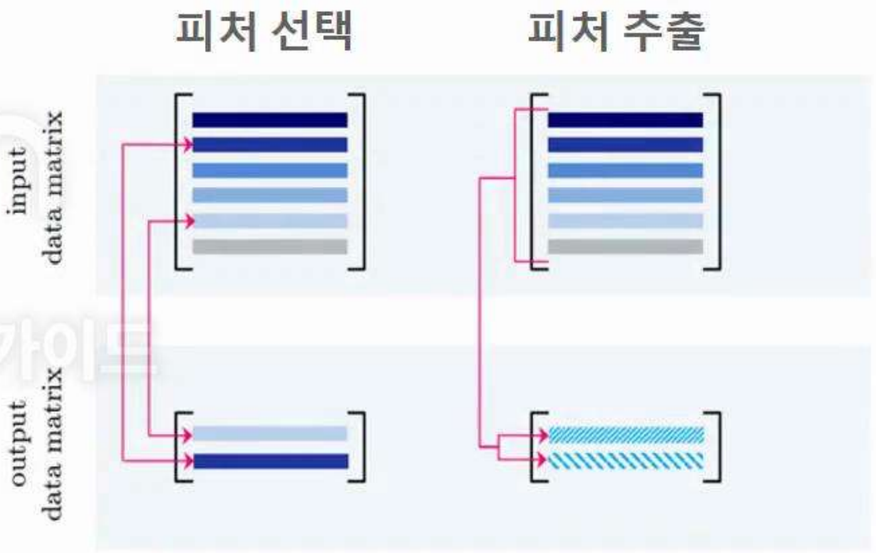
일반적으로 차원 축소는 피처 선택(feature selection)과 피처 추출(feature extraction)로 나눌 수 있습니다.

피처 선택 (Feature Selection)

특정 피처에 종속성이 강한 불필요한 피처는 아예 제거하고, 데이터의 특징을 잘 나타내는 주요 피처만 선택하는 것입니다.

피처 추출 (Feature Extraction)

피처(특성) 추출은 기존 피처를 저차원의 중요 피처로 압축해서 추출하는 것입니다. 이렇게 새롭게 추출된 중요 특성은 기존의 피처를 반영해 압축된 것이지만 새로운 피처로 추출하는 것입니다.



1445185

피처 추출(Feature Extraction)

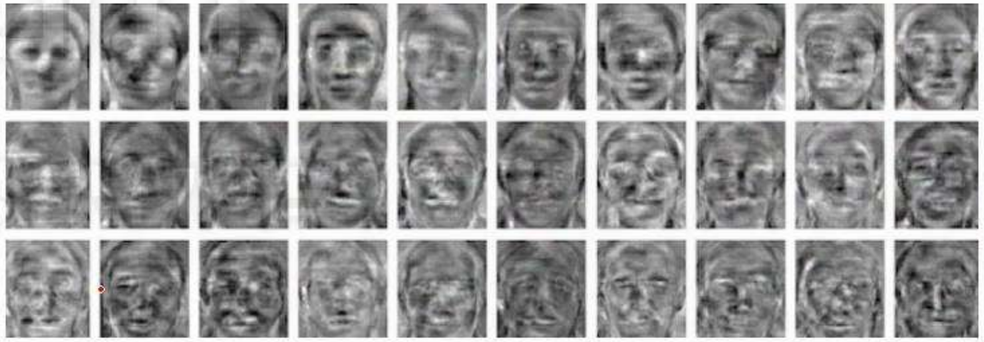
피처 추출은 기존 피처를 단순 압축이 아닌, 피처를 함축적으로 더 잘 설명할 수 있는 또 다른 공간으로 매핑해 추출하는 것입니다

[제정판]파이썬 머신러닝 완벽 가이드
권 철민

차원 축소의 의미

차원 축소는 단순히 데이터의 압축을 의미하는 것이 아닙니다. 더 중요한 의미는 차원 축소를 통해 좀 더 데이터를 잘 설명할 수 있는 잠재적(Latent)인 요소를 추출하는 데에 있습니다

- 추천 엔진
- 이미지 분류 및 변환
- 문서 토픽 모델링



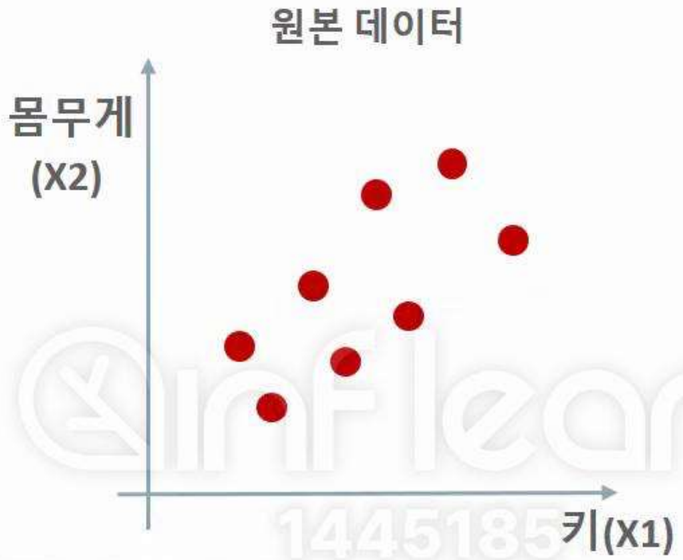
inflearn
1445185
[개정판] 파이썬 머신러닝
권 철민

PCA(Principal Component Analysis)의 이해

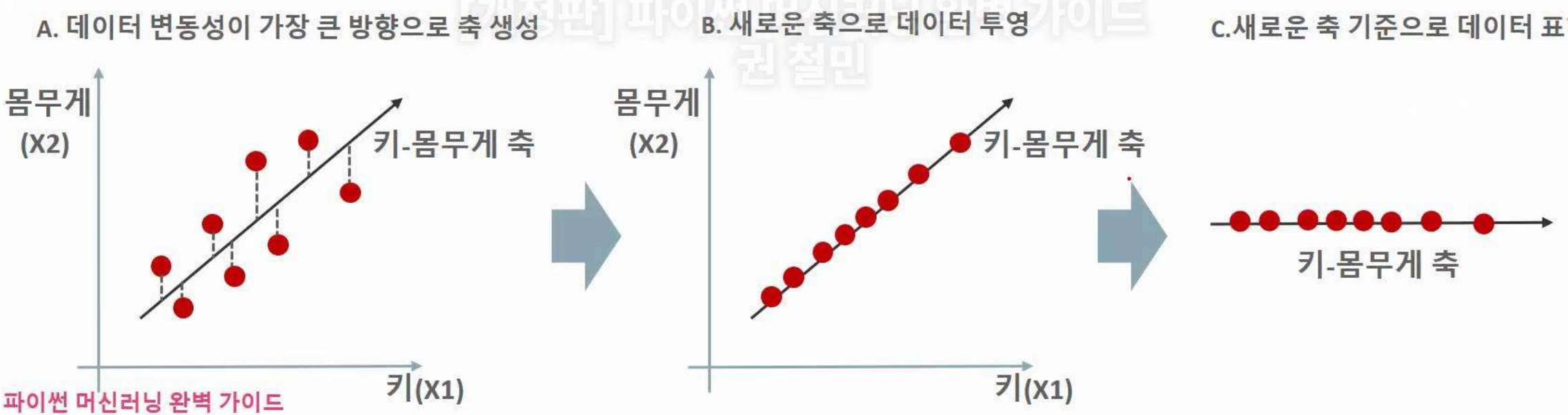
- 고차원의 원본 데이터를 저 차원의 부분 공간으로 투영하여 데이터를 축소하는 기법
- 예를 들어 10차원의 데이터를 2차원의 부분 공간으로 투영하여 데이터를 축소
- PCA는 원본 데이터가 가지는 데이터 변동성을 가장 중요한 정보로 간주하며 이 변동성에 기반한 원본 데이터 투영으로 차원 축소를 수행

@inflearn
[개정판] 파이썬 머신러닝 완벽 가이드
권 철민

PCA(Principal Component Analysis)의 이해



PCA는 원본 데이터 변동성이 가장 큰 방향으로 순차적으로 축들을 생성하고, 이렇게 생성된 축으로 데이터를 투영하는 방식입니다.



1445185
[개정판] 파이썬 머신러닝 완벽 가이드
권 철민

PCA 변환

PCA를 선형대수 관점에서 해석해 보면, 입력 데이터의 공분산 행렬(Covariance Matrix)을 고유값 분해하고, 이렇게 구한 고유벡터에 입력 데이터를 선형 변환하는 것입니다.

inflearn
14451185
[개정판] 파이썬 머신러닝 완벽 가이드
권 철민

- **고유벡터는 PCA의 주성분 벡터로서 입력 데이터의 분산이 큰 방향을 나타냅니다.**
- **고윳값(eigenvalue)은 바로 이 고유벡터의 크기를 나타내며, 동시에 입력 데이터의 분산을 나타냅니다.**

공분산 행렬

보통 분산은 한 개의 특정한 변수의 데이터 변동을 의미하나, 공분산은 두 변수 간의 변동을 의미합니다. 즉, 사람 키 변수를 X, 몸무게 변수를 Y라고 하면 공분산 Cov(X, Y) > 0은 X(키)가 증가할 때 Y(몸무게)도 증가한다는 의미입니다.

	X	Y	Z
X	**3.0**	-0.71	-0.24
Y	-0.71	**4.5**	0.28
Z	-0.24	0.28	**0.91**

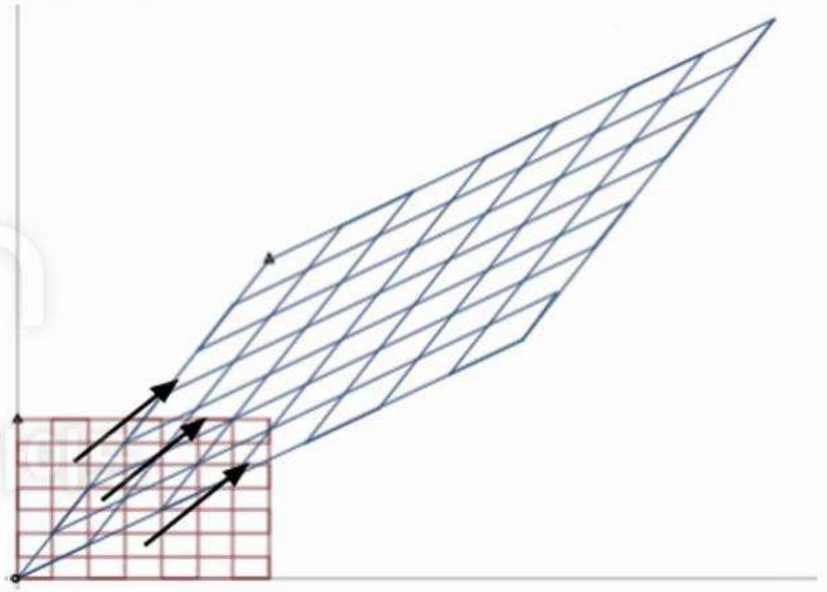
공분산 행렬은 여러 변수와 관련된 공분산을 포함하는 정방형 행렬이며 대칭 행렬 입니다.

정방행렬은 열과 행이 같은 행렬을 지칭하는데, 정방행렬 중에서 대각 원소를 중심으로 원소 값이 대칭되는 행렬, 즉 $A^T = A$인 행렬을 대칭행렬이라고 부릅니다

선형 변환과 고유 벡터/고유값

- 일반적으로 선형 변환은 특정 벡터에 행렬 A를 곱해 새로운 벡터로 변환하는 것을 의미합니다. 이를 특정 벡터를 하나의 공간에서 다른 공간으로 투영하는 개념으로도 볼 수 있으며, 이 경우 이 행렬을 바로 공간으로 가정하는 것입니다.

- 고유벡터는 행렬 A를 곱하더라도 방향이 변하지 않고 그 크기만 변하는 벡터를 지칭합니다. 즉, Ax = ax(A는 행렬, x는 고유벡터, a는 스칼라값)입니다. 이 고유벡터는 여러 개가 존재하며, 정방 행렬은 최대 그 차원 수만큼의 고유벡터를 가질 수 있습니다. 예를 들어 2x2 행렬은 두 개의 고유벡터를, 3x3 행렬은 3개의 고유벡터를 가질 수 있습니다. 이렇게 고유벡터는 행렬이 작용하는 힘의 방향과 관계가 있어서 행렬을 분해하는 데 사용됩니다



https://deeplearning4j.org/kr/eigenvector

공분산 행렬의 고유값 분해

	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 행렬은 정방행렬(Diagonal Matrix)이며 대칭행렬(Symmetric Matrix)입니다. 정방행렬은 열과 행이 같은 행렬을 지칭하는데, 정방행렬 중에서 대각 원소를 중심으로 원소 값이 대칭되는 행렬, 즉 $A^T = A$인 행렬을 대칭행렬이라고 부릅니다
- 대칭행렬은 고유값 분해와 관련해 매우 좋은 특성이 있습니다. 대칭행렬은 항상 고유벡터를 직교행렬(orthogonal matrix)로, 고유값을 정방 행렬로 대각화할 수 있다는 것입니다.

1445185
[개정판] 파이썬 머신러닝 완벽 가이드
권 철민

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

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

- P는 n x n의 직교행렬이며, ∑는 n x n 정방행렬, P^T는 행렬 P의 전치 행렬입니다

- 공분산 C는 고유벡터 직교 행렬, 고유값 정방 행렬 * 고유벡터 직교 행렬의 전치 행렬로 분해됩니다.
- *ei*는 i번째 고유벡터를, *λi*는 i번째 고유벡터의 크기를 의미합니다. 고유 벡터는 바로 PCA의 축입니다.
- *e1*는 가장 분산이 큰 방향을 가진 고유벡터이며, *e2*는 *e1*에 수직이면서 다음으로 가장 분산이 큰 방향을 가진 고유벡터입니다.

PCA 변환과 수행 절차

PCA 변환

입력 데이터의 공분산 행렬이 고유벡터와 고유값으로 분해될 수 있으며, 이렇게 분해된 고유벡터를 이용해 입력 데이터를 선형 변환하는 방식

PCA 변환 수행 절차

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

onflearn
1445185
[개정판] 파이썬 머신러닝 완벽 가이드
권 철민

사이킷런 PCA

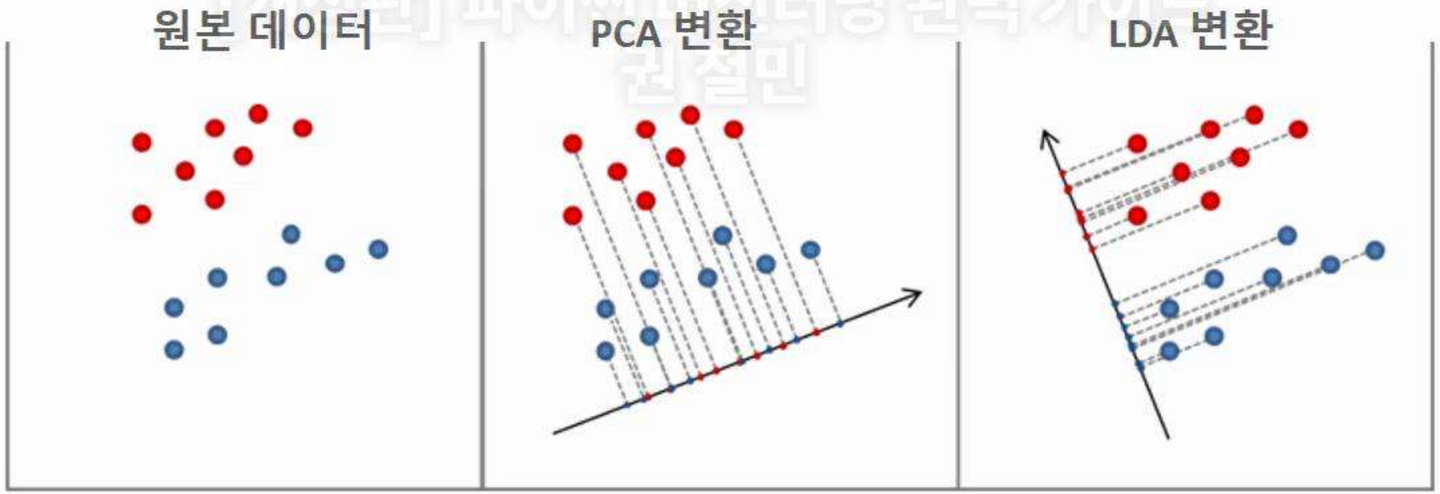
사이킷런은 PCA를 위해 PCA 클래스를 제공합니다.

sklearn.decomposition.PCA(*n_components=None, copy=True, whiten=False, svd_solver='auto', tol=0.0, iterated_power='auto', random_state=None*)

- n_components 는 PCA 축의 개수 즉 변환 차원을 의미합니다.
- PCA를 적용하기 전에 입력 데이터의 개별 피처들을 스케일링해야 합니다. PCA는 여러 피처들의 값을 연산해야 하므로 피처들의 스케일에 영향을 받습니다. 따라서 여러 속성을 PCA로 압축하기 전에 각 피처들의 값을 동일한 스케일로 변환하는 것이 필요합니다. 일반적으로 평균이 0, 분산이 1인 표준 정규 분포로 변환합니다.
- PCA 변환이 완료된 사이킷런 PCA 객체는 전체 변동성에서 개별 PCA 컴포넌트별로 차지하는 변동성 비율을 explained_variance_ratio_ 속성으로 제공합니다.

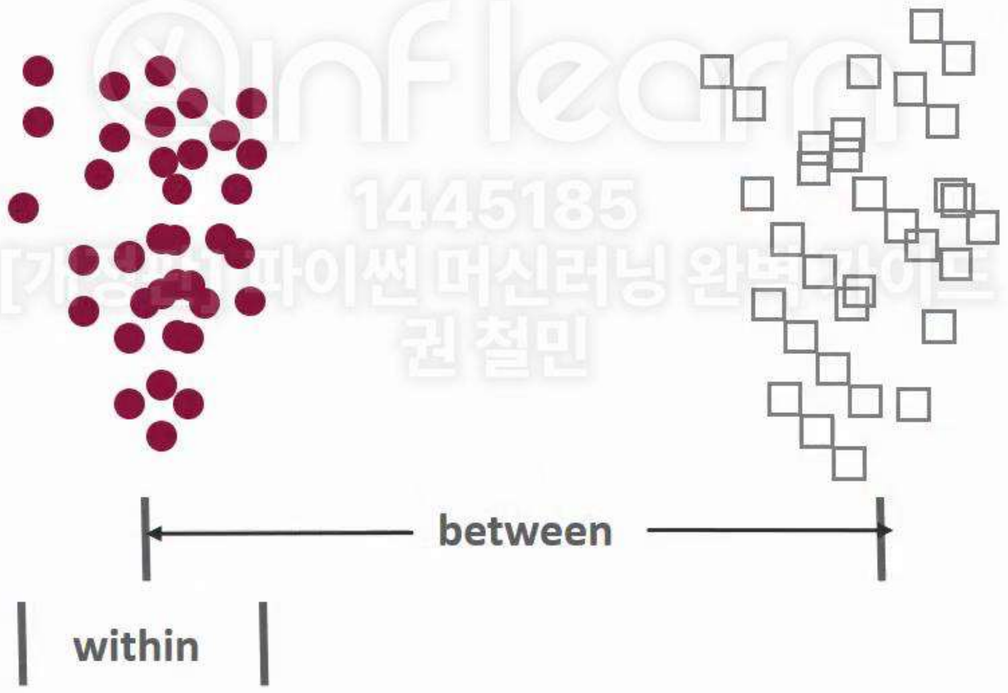
LDA(Linear Discriminant Analysis)

- LDA(Linear Discriminant Analysis)는 선형 판별 분석법으로 불리며, PCA와 매우 유사합니다.
- LDA는 PCA와 유사하게 입력 데이터 세트를 저차원 공간에 투영해 차원을 축소하는 기법이지만, 중요한 차이는 LDA는 지도학습의 분류(Classification)에서 사용하기 쉽도록 개별 클래스를 분별할 수 있는 기준을 최대한 유지하면서 차원을 축소합니다. PCA는 입력 데이터의 변동성의 가장 큰 축을 찾았지만, LDA는 입력 데이터의 결정 값 클래스를 최대한으로 분리할 수 있는 축을 찾습니다.
- LDA는 같은 클래스의 데이터는 최대한 근접해서, 다른 클래스의 데이터는 최대한 떨어뜨리는 축 매핑을 합니다.



LDA(Linear Discriminant Analysis) 차원 축소 방식

LDA는 특정 공간상에서 클래스 분리를 최대화하는 축을 찾기 위해 클래스 간 분산(between-class scatter)과 클래스 내부 분산(within-class scatter)의 비율을 최대화하는 방식으로 차원을 축소합니다. 즉, 클래스 간 분산은 최대한 크게 가져가고, 클래스 내부의 분산은 최대한 작게 가져가는 방식입니다



onflearn
1445185
[파이썬 머신러닝 완벽 가이드
권 철민

LDA 절차

일반적으로 LDA를 구하는 스텝은 PCA와 유사하나 가장 큰 차이점은 공분산 행렬이 아니라 앞에서 설명한 클래스 간 분산과 클래스 내부 분산 행렬을 생성한 뒤, 이 행렬에 기반해 고유벡터를 구하고 입력 데이터를 투영한다는 점입니다.

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

특이값 분해 - SVD(Singular Value Decomposition)

대표적인 행렬 분해 방법

고유값 분해
Eigen-Decomposition

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

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

- 정방행렬(즉, 행과 열의 크기가 같은 행렬)만을 고유벡터로 분해
- PCA는 분해된 고유 벡터에 원본 데이터를 투영하여 차원 축소

특이값 분해
Singular Value Decomposition

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

- SVD는 정방행렬뿐만 아니라 행과 열의 크기가 다른 m × n 행렬도 분해 가능

1445185
[개정판] 파이썬 머신러닝 완벽 가이드
권 철민

특이값 분해 - SVD(Singular Value Decomposition)

특이값 분해
Singular Value Decomposition

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

왼쪽 직교행렬 / 대각 행렬 / 오른쪽 직교행렬

행렬 *U*와 *V*에 속한 벡터는 특이벡터(singular vector)이며, 모든 특이벡터는 서로 직교하는 성질을 가집니다

$$U^T U = I$$

$$V^T V = I$$

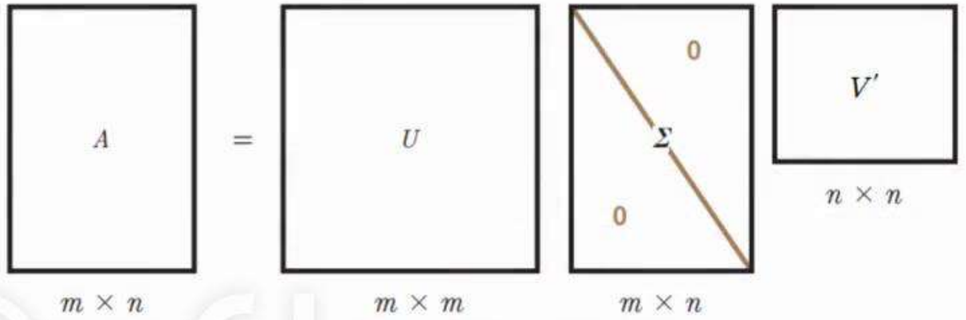
Σ 는 대각행렬이며, 행렬의 대각에 위치한 값만 0이 아니고 나머지 위치의 값은 모두 0입니다.

Σ 이 위치한 0이 아닌 값이 바로 행렬 *A*의 특이값입니다

1445185 [파이썬 머신러닝 완벽 가이드 권철민]

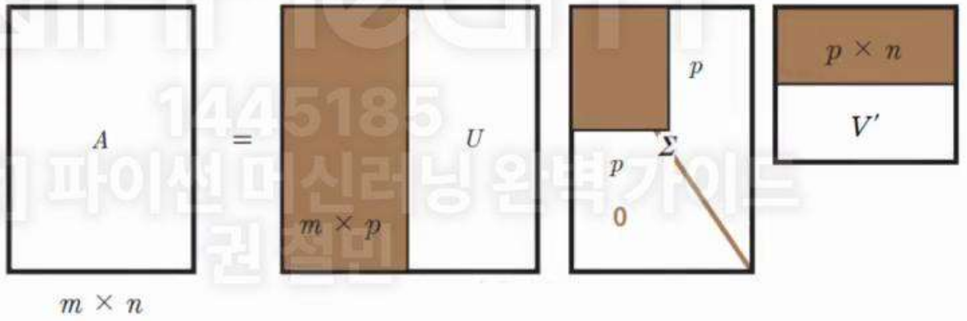
SVD 유형

Full SVD



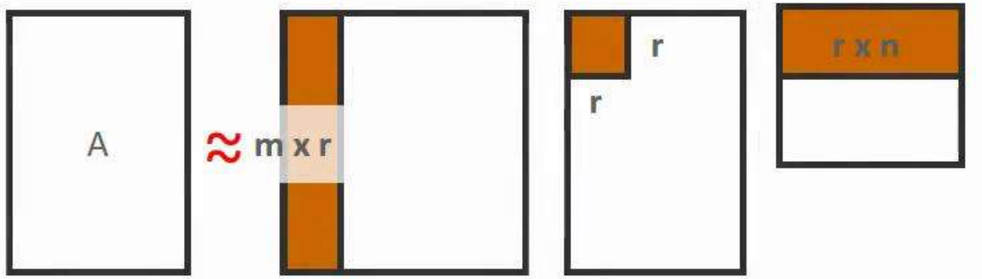
Compact SVD

비대각 부분과 대각 원소가 0인 부분을 제거

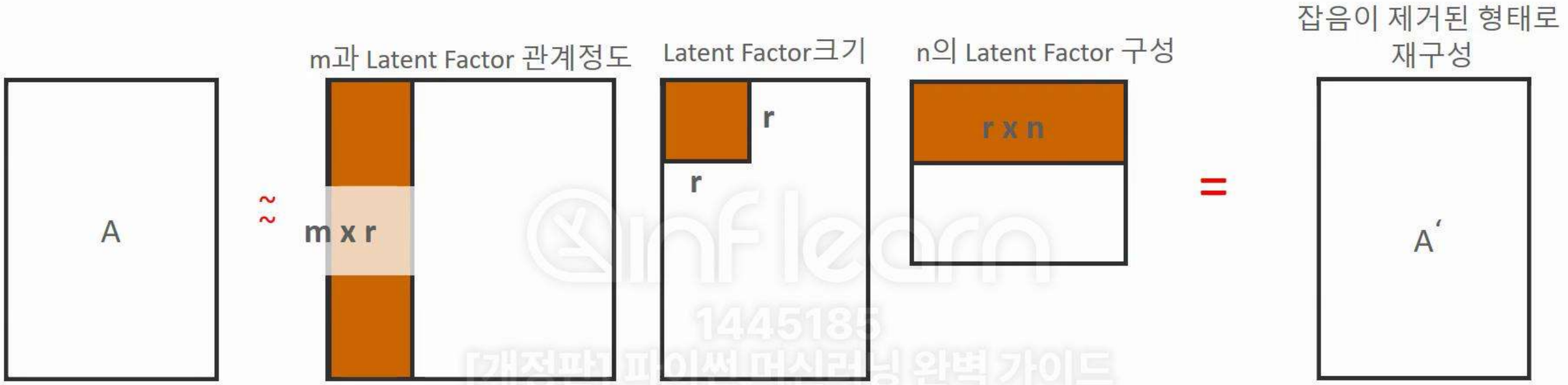


Truncated SVD

대각 원소 가운데 상위 r 개만 추출하여 차원 축소



Truncated SVD 행렬 분해 의미



1445185
[개정판] 파이썬 머신러닝 완벽 가이드
권철민

- SVD 는 차원 축소를 위한 행렬 분해를 통해 Latent Factor(잠재 요인)를 찾을 수 있는데 이렇게 찾아진 Latent Factor는 많은 분야에 활용 (추천 엔진, 문서의 잠재 의미 분석 등)
- SVD로 차원 축소 행렬 분해된 후 다시 분해된 행렬을 이용하여 원복된 데이터 셋은 잡음(Noise)이 제거된 형태로 재 구성될 수 있음
- 사이킷런에서는 Truncated SVD로 차원을 축소할 때 원본 데이터에 $U\Sigma$를 적용하여 차원 축소

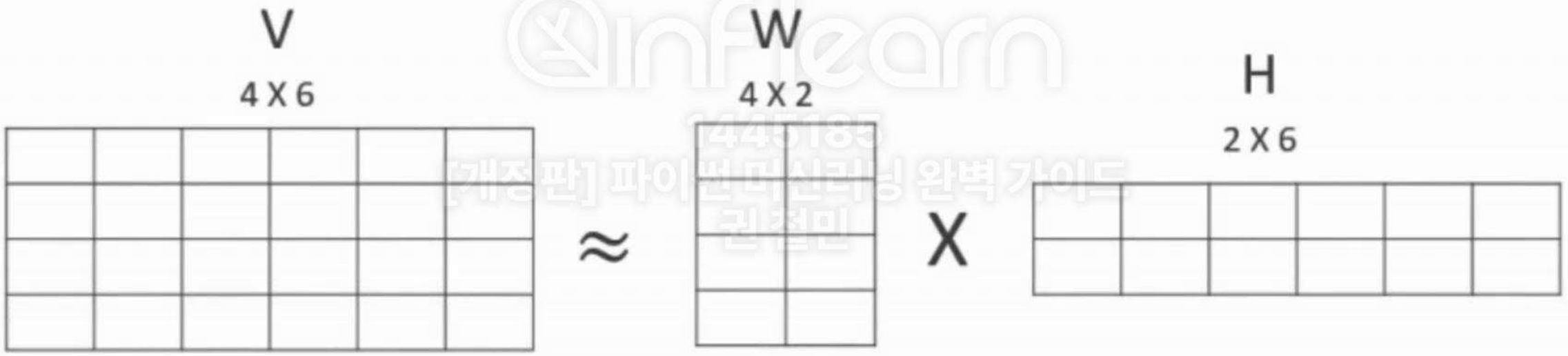
SVD 활용

- 이미지 압축/변환
- 추천 엔진
- 문서 잠재 의미 분석
- 의사 역행렬을 통한 모델 예측

inflearn
1445185
[개정판] 파이썬 머신러닝 완벽 가이드
권 철민

NMF(Non Negative Matrix Factorization)

NMF는 원본 행렬 내의 모든 원소 값이 모두 양수(0 이상)라는 게 보장되면 다음과 같이 좀 더 간단하게 두 개의 기반 양수 행렬로 분해될 수 있는 기법을 지칭합니다.



행렬 분해(Matrix Factorization)

행렬 분해(MatrixFactorization)는 일반적으로 SVD와 같은 행렬 분해 기법을 통칭하는 것입니다. 이처럼 행렬 분해를 하게 되면 W 행렬과 H 행렬은 일반적으로 길고 가는 행렬 W(즉, 원본 행렬의 행 크기와 같고 열 크기보다 작은 행렬)와 작고 넓은 행렬 H(원본 행렬의 행 크기보다 작고 열 크기와 같은 행렬)로 분해됩니다. 이렇게 분해된 행렬은 Latent Factor(잠재 요소)를 특성으로 가지게 됩니다. 분해 행렬 W는 원본 행에 대해서 이 잠재 요소의 값이 얼마나 되는지에 대응하며, 분해 행렬 H는 이 잠재 요소가 원본 열(즉, 원본 속성)로 어떻게 구성됐는지를 나타내는 행렬입니다.