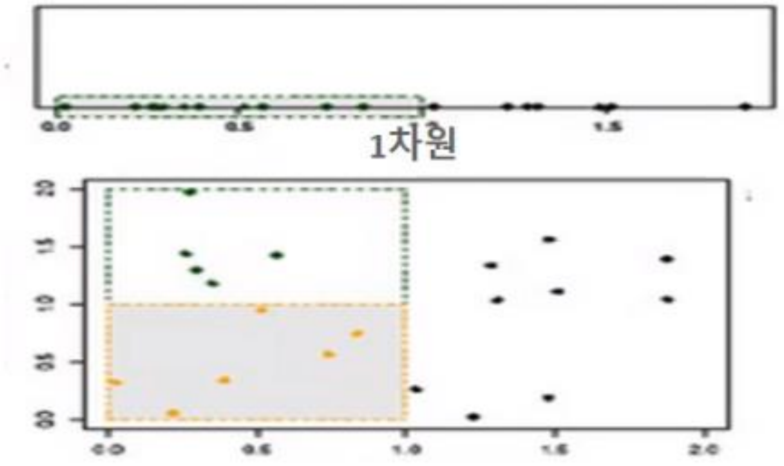
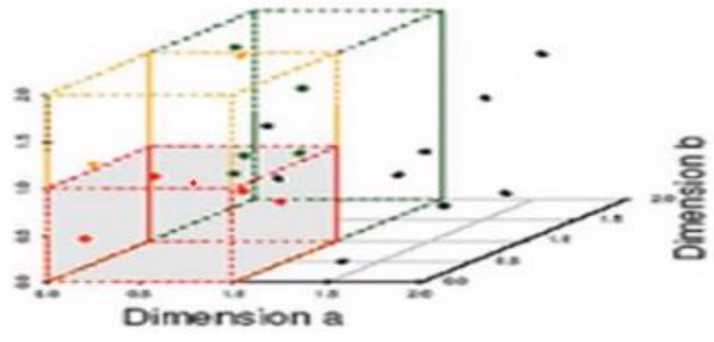
차원 축소 알고리즘

차원의 저주



특정 면적 공간에 6개의 데이터



특정 입체 공간에 4개의 데이터

- 차원이 커지면 커질수록 데이터 포인트들간 거리가 크게 늘어나며, 데이터가 희소화(Sparse)되기 시작한다.
- 수백~수천 개 이상의 Feature로 구성된 포인트들간 거리에 기반한 머신러닝 알고리즘들이 무력화 된다.
- 또한 Feature가 많을 경우 개별 Feature간에 상관관계가 높아 선형 회귀 같은 모델에서는 다중공선성 문제로 모델의 예측 성능이 저하될 가능성이 매우 높다.

차원 축소의 장점

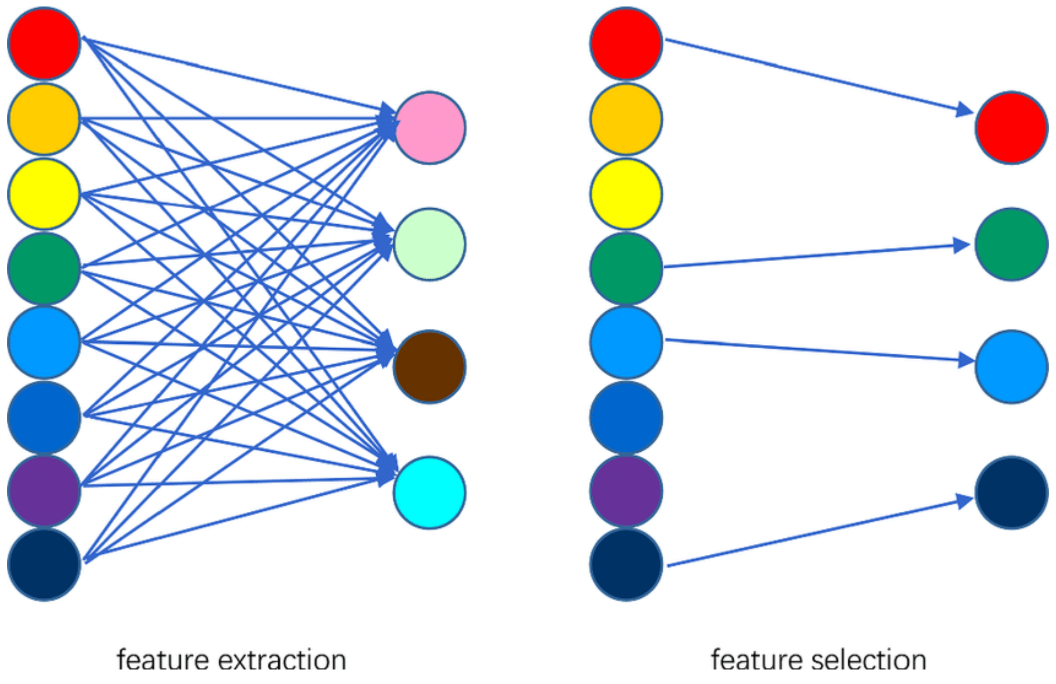
- 수십~ 수백 개의 Feature들을 작은 수의 Feature 들로 축소한다면?
  - 학습 데이터 크기를 줄여서 학습 시간을 절약할 수 있다.
  - 불필요한 Feature들을 줄여서 모델 성능 향상에 기여할 수 있다.
  - 다차원 데이터를 3차원 이하의 차원축소를 통해 시각적으로 보다 쉽게 데이터 패턴을 인지할 수 있다.

차원 축소의 목적은 어떻게 하면 원본 데이터의 정보를 최대한으로 유지한 채로 차원 축소를 수행할 것인가? 이다.

Feature Selection, Feature Extraction

- 일반적으로 차원 축소는 특성 선택(feature selection)과 특성 추출(feature extraction)로 나눌 수 있다.

- 특성 선택(Feature Selection)
  - 특정 특성에 종속성이 강한 불필요한 특성은 아예 제거하고 데이터의 특징을 잘 나타내는 주요 특성만 선택하는 것
- 특성 추출(Feature Extraction)
  - 특성 추출은 기존 특성을 저차원의 중요 특성으로 압축해서 추출하는 것이다. 새롭게 추출된 중요 특성은 기존의 특성을 반영해 압축된 것이지만 새로운 특성으로 추출하게 된다.



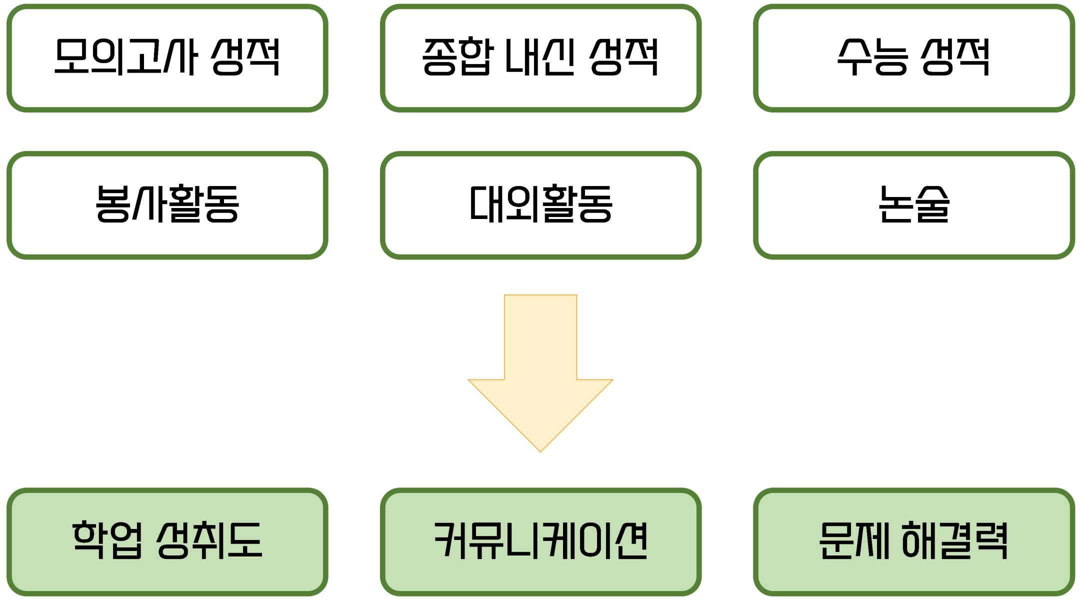
특성 추출(Feature Extraction)

- 특성 추출은 기존 특성을 단순하게 압축하는 것이 아닌 특성을 함축적으로 더 잘 설명할 수 있는 또 다른 공간으로 매핑해 추출하는 것이다.

모의고사 성적	종합 내신 성적	수능 성적
봉사활동	대외활동	논술

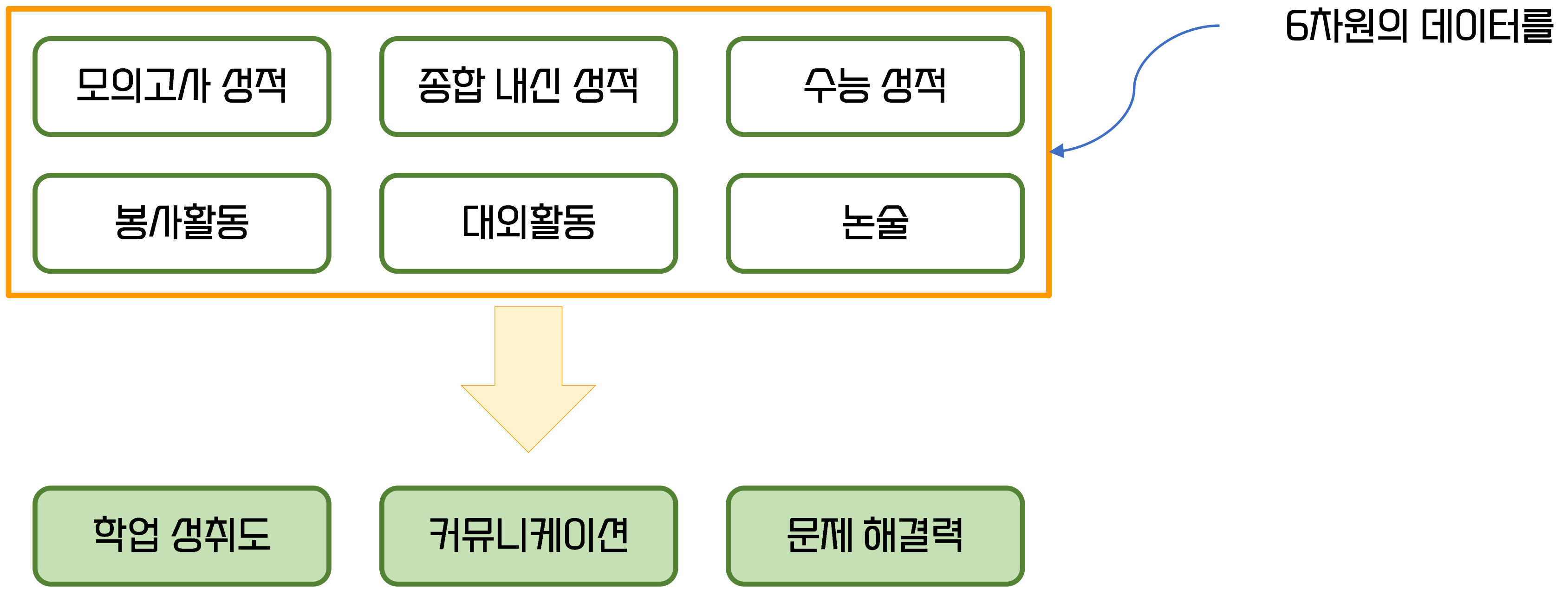
특성 추출(Feature Extraction)

- 특성 추출은 기존 특성을 단순하게 압축하는 것이 아닌 특성을 함축적으로 더 잘 설명할 수 있는 또 다른 공간으로 매핑해 추출하는 것이다.



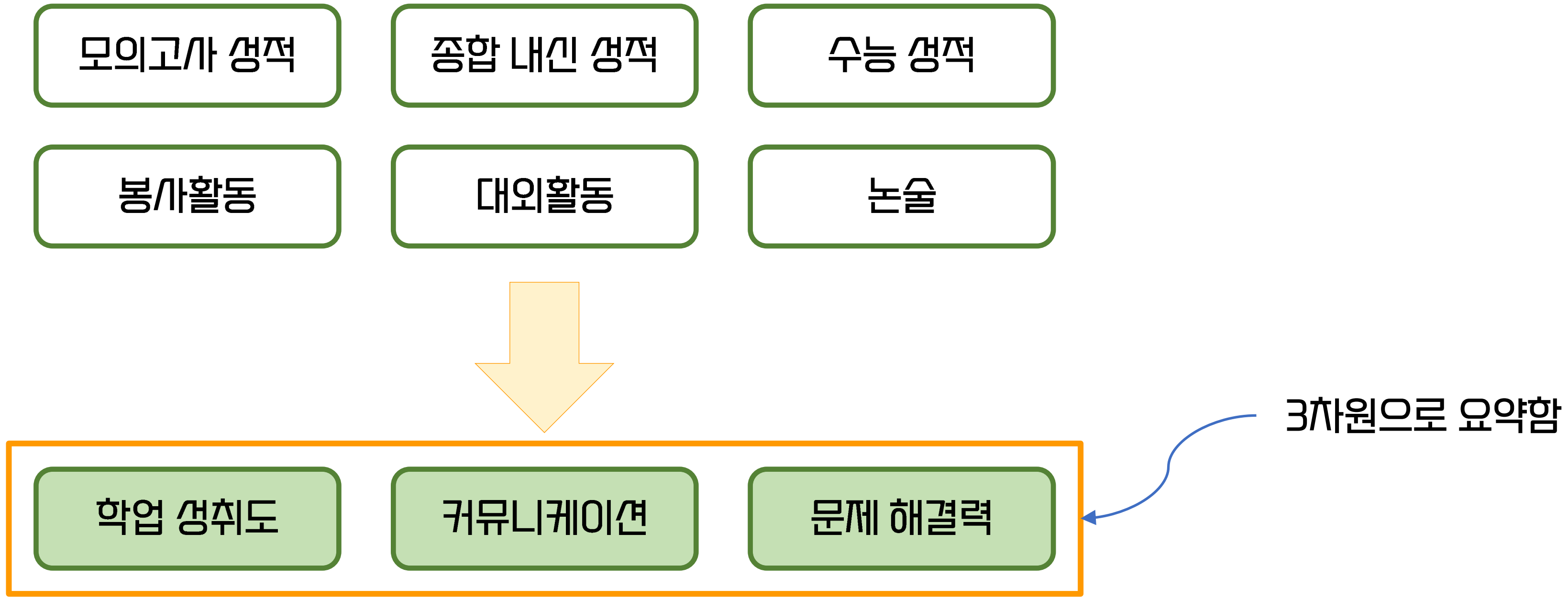
특성 추출(Feature Extraction)

- 특성 추출은 기존 특성을 단순하게 압축하는 것이 아닌 특성을 함축적으로 더 잘 설명할 수 있는 또 다른 공간으로 매핑해 추출하는 것이다.



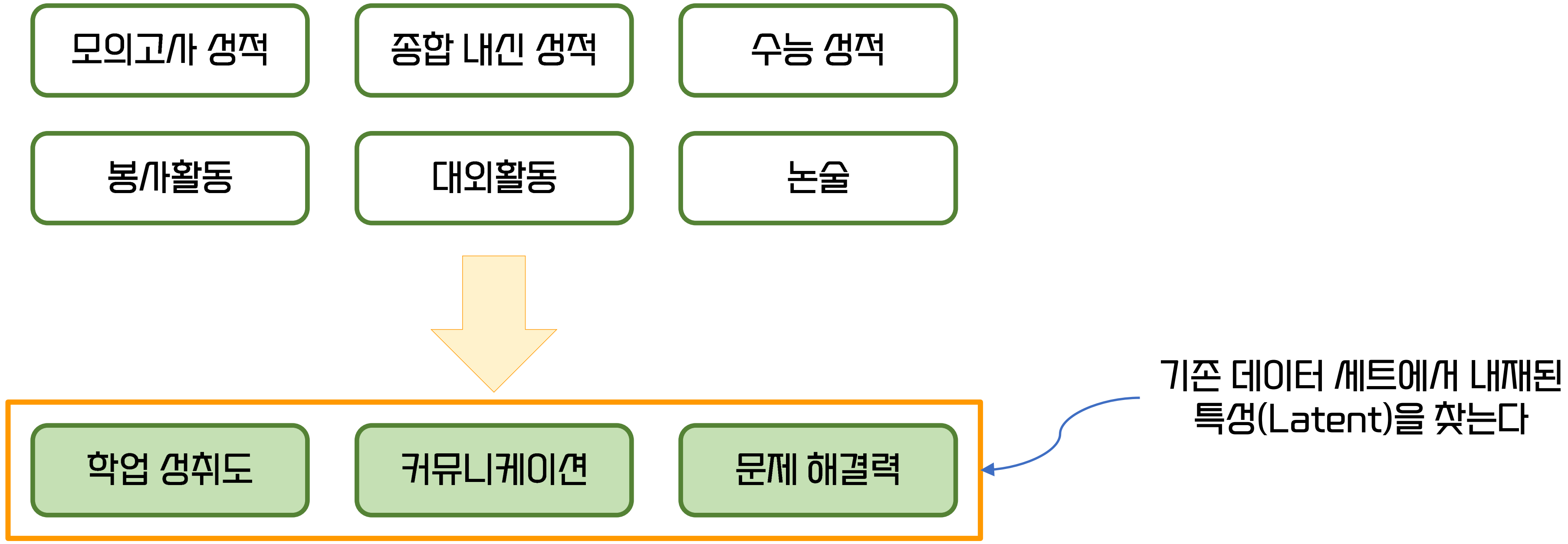
특성 추출(Feature Extraction)

- 특성 추출은 기존 특성을 단순하게 압축하는 것이 아닌 특성을 함축적으로 더 잘 설명할 수 있는 또 다른 공간으로 매핑해 추출하는 것이다.



특성 추출(Feature Extraction)

- 특성 추출은 기존 특성을 단순하게 압축하는 것이 아닌 특성을 함축적으로 더 잘 설명할 수 있는 또 다른 공간으로 매핑해 추출하는 것이다.



차원 축소의 의미

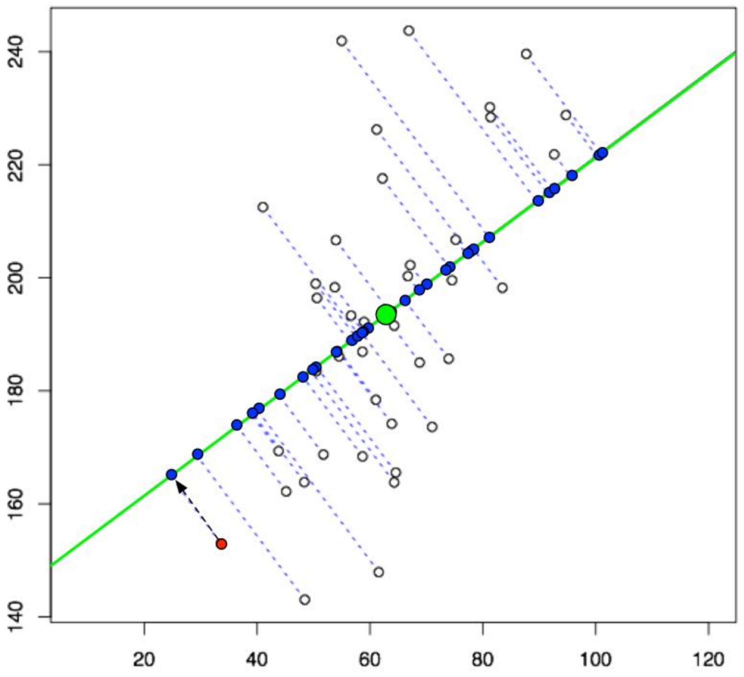
- 차원 축소는 단순히 데이터의 압축을 의미하는 것이 아니다. 더 중요한 의미는 차원 축소를 통해 좀 더 데이터를 잘 설명할 수 있는 잠재적(Latent)인 요소를 추출하는 데에 있다.

- 대표적으로 추천엔진, 이미지 분류 및 변환, 문서 토픽 모델링 등에서 차원 축소가 사용된다.

PCA

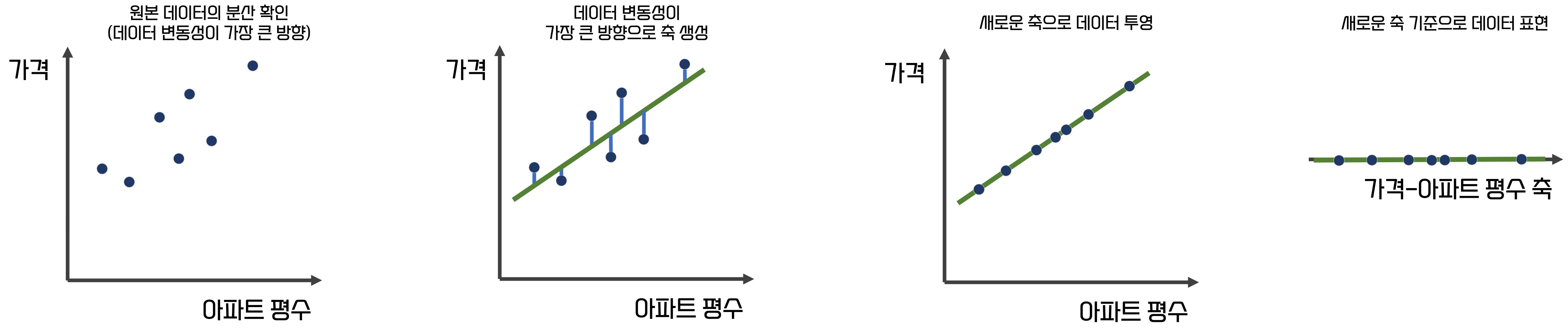
PCA(Principal Component Analysis)의 이해

- 고차원의 원본 데이터를 저차원의 부분 공간으로 **투영**하여 데이터를 축소하는 기법
  - 10차원의 데이터를 2차원의 부분 공간으로 투영하여 데이터를 축소
- PCA는 원본 데이터가 가지는 데이터 변동성(분산)을 가장 중요한 정보로 간주하며 이 변동성에 기반한 원본 데이터 투영으로 차원 축소를 수행



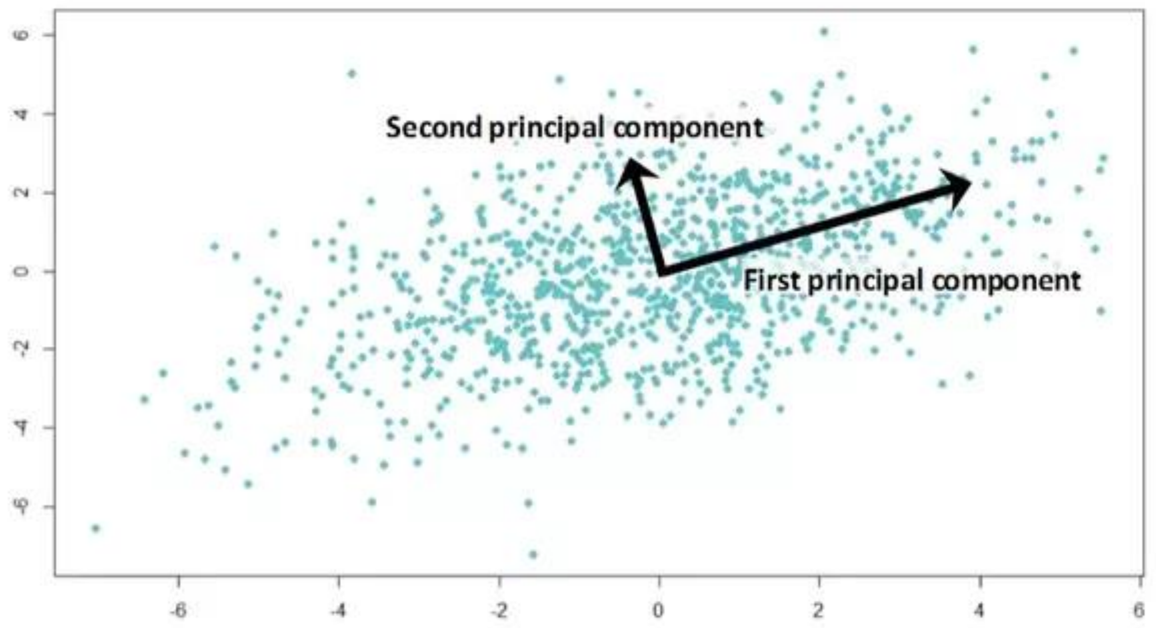
PCA(Principal Component Analysis)의 이해

PCA는 원본 데이터 변동성이 가장 큰 방향으로 순차적으로 축들을 생성하고 이렇게 생성된 축으로 데이터를 투영한다.



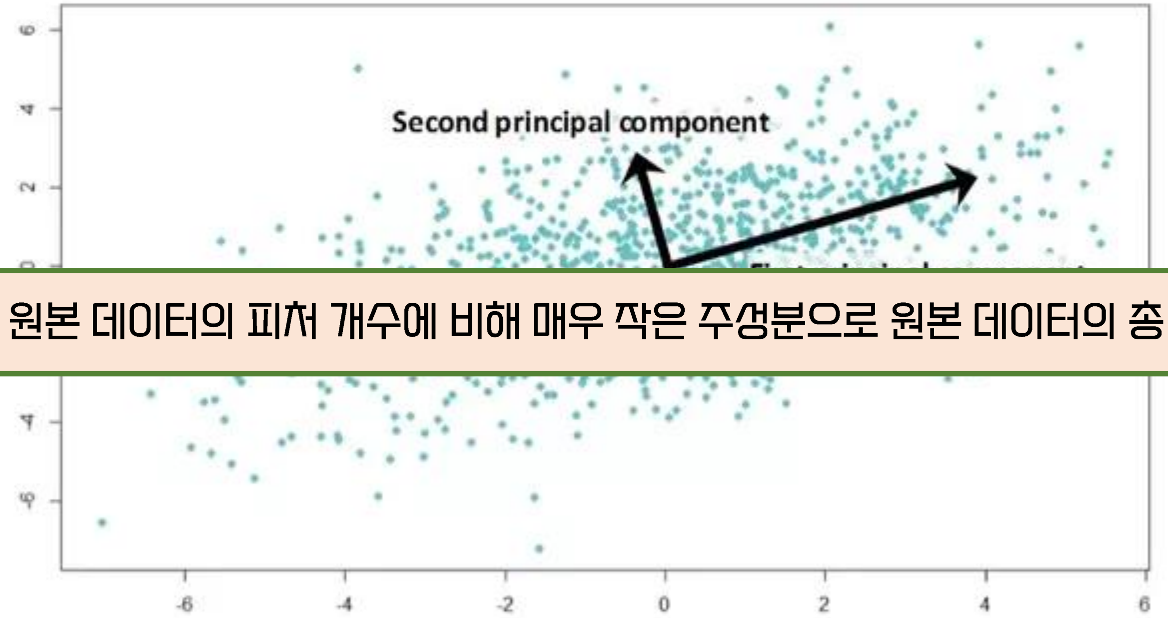
PCA(Principal Component Analysis)의 이해

- PCA는 제일 먼저 원본 데이터의 가장 큰 데이터 변동성(분산)을 기반으로 첫 번째 벡터 축을 생성하고, 두 번째 축은 첫 번째 축을 제외하고 그 다음 변동성(분산)이 큰 축을 설정한다. 이 때 두 번째 축은 첫 번째 축과 직각이 되는 벡터(직교 벡터) 축이 된다. 세 번째 축은 다시 직각이 되는 벡터를 설정하는 방식으로 축을 생성하게 된다. 이렇게 생성된 벡터 축에 원본 데이터를 투영하면 벡터 축의 개수 만큼 차원으로 원본 데이터가 차원 축소 된다.



PCA(Principal Component Analysis)의 이해

- PCA는 제일 먼저 원본 데이터의 가장 큰 데이터 변동성(분산)을 기반으로 첫 번째 벡터 축을 생성하고, 두 번째 축은 첫 번째 축을 제외하고 그 다음 변동성(분산)이 큰 축을 설정한다. 이 때 두 번째 축은 첫 번째 축과 직각이 되는 벡터(직교 벡터) 축이 된다. 세 번째 축은 다시 직각이 되는 벡터를 설정하는 방식으로 축을 생성하게 된다. 이렇게 생성된 벡터 축에 원본 데이터를 투영하면 벡터 축의 개수 만큼 차원으로 원본 데이터가 차원 축소 된다.



PCA, 즉 주성분 분석은 이처럼 원본 데이터의 피처 개수에 비해 매우 작은 주성분으로 원본 데이터의 총 변동성을 대부분 설명할 수 있다.

선형대수 관점의 PCA 변환

- 선형대수 관점으로 PCA 변환을 보면 입력 데이터의 공분산 행렬(Covariance Matrix)을 고유값 분해 하고, 이렇게 구한 고유벡터에 입력 데이터를 선형 변환 하는 것이다.

원본 데이터의 공분산 행렬 추출 → 공분산 행렬을 고유벡터와 고유값 분해 → 원본 데이터를 고유벡터로 선형변환 → PCA 변환 값 도출

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

공분산 행렬의 고윳값 분해

- 공분산 행렬은 정방행렬이며, 대칭행렬이다. 대칭행렬은 고윳값 분해와 관련해 매우 좋은 특징이 있다. 대칭행렬은 항상 고유벡터를 직교행렬로, 고윳값을 정방 행렬로 대각화 할 수 있다는 것이다.

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

- P는 $N \times N$의 직교행렬이며, Σ는 $N \times N$ 정방행렬, P^T는 정방행렬 P의 전치행렬이다.

$$C = \begin{bmatrix} e_1 & \cdots & e_N \end{bmatrix} \begin{bmatrix} \lambda_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \lambda_N \end{bmatrix} \begin{bmatrix} e_1^T \\ \vdots \\ e_N^T \end{bmatrix}$$

- 공분산 C는 고유벡터 직교행렬, 고윳값 정방행렬, 고유벡터 직교행렬의 전치행렬로 분해된다.
- e_i는 i번째 고유벡터를, λ_i는 i번째 고유벡터의 크기를 의미한다. 이 고유벡터가 바로 PCA의 축이 된다.
- e_1은 가장 분산이 큰 방향을 가진 고유벡터이며, e_2는 e_1에 수직이면서 다음으로 가장 큰 분산이 큰 방향을 가진 고유벡터이다.

PCA 변환과 수행 절차

입력 데이터의 공분산 행렬이 고유벡터와 고윳값으로 분해될 수 있으며, 이렇게 분해된 고유벡터를 이용해 입력 데이터를 선형 변환한다.

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

LDA

LDA(Linear Discriminant Analysis)

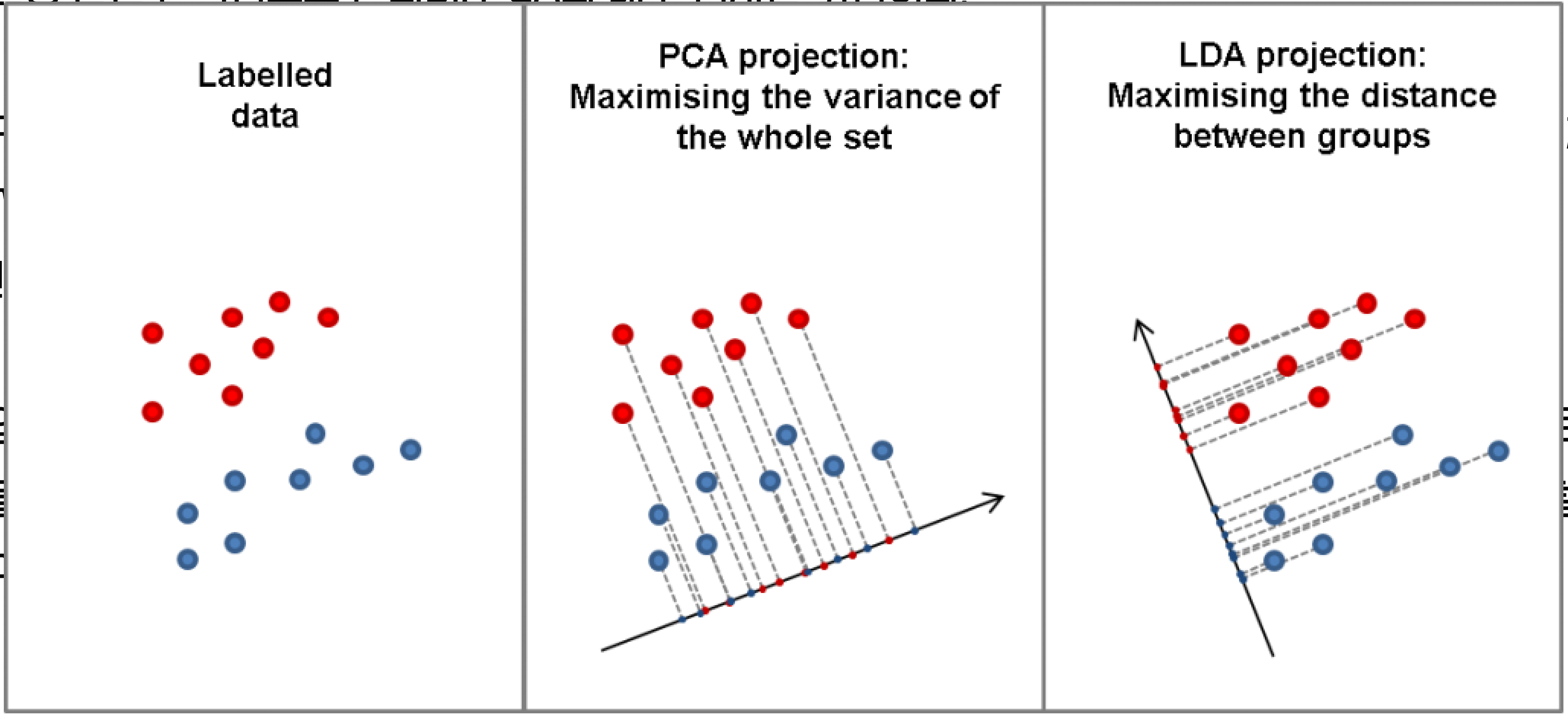
- LDA는 선형 판별 분석법으로 불리며, PCA와 매우 흡사한 기법이다.
- LDA는 PCA와 유사하게 입력 데이터 세트를 저차원 공간에 투영해 차원을 축소하는 기법이지만, 중요한 차이는 LDA는 지도학습의 분류(Classification)에서 사용하기 쉽도록 개별 클래스를 분별할 수 있는 기준을 최대한 유지하면서 차원을 축소한다.
- PCA는 입력 데이터의 변동성이 가장 큰 축을 찾지만, LDA는 입력 데이터의 결정 값 클래스를 최대한으로 분리할 수 있는 축을 찾는다. 즉 LDA는 같은 클래스의 데이터는 최대한 근접해서, 다른 클래스의 데이터는 최대한 떨어뜨리는 축 매핑을 수행한다.

LDA(Linear Discriminant Analysis)

- LDA는 선형 판별 분석법으로 불리며, PCA와 매우 흡사한 기법이다.

- LDA는 P … 차이는
LDA는 … 대한
유지하면 …

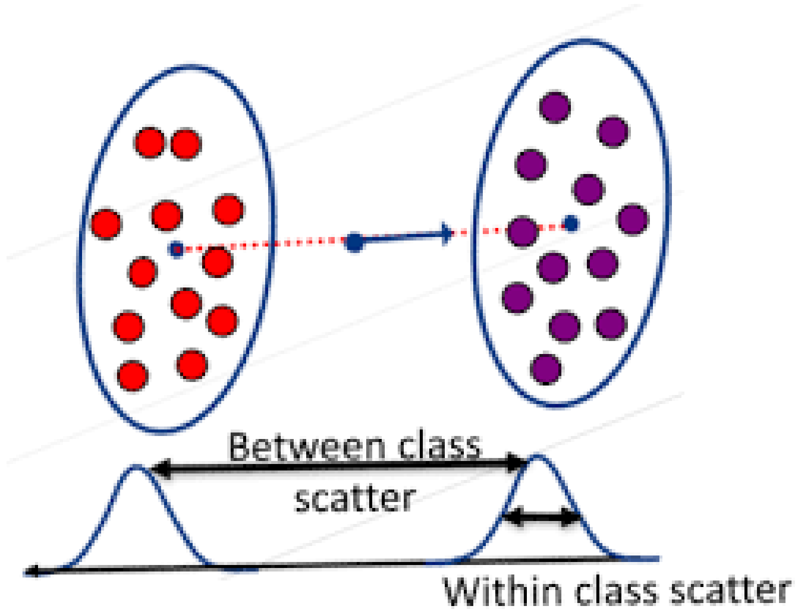
- PCA는 … 로 분리할 수
있는 축을 … 떨어뜨리는 축
매핑을 수 …



LDA 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 축소한다.

- 즉, 클래스 간 분산은 최대한 크게 가져가고, 클래스 내부 분산은 최대한 작게 가져간다.



LDA 절차

LDA를 구하는 것은 PCA와 매우 유사하나, 가장 큰 차이점은 공분산 행렬이 아닌 클래스간 분산과 내부 분산 행렬을 생성한 뒤 이 행렬에 기반해 고유벡터를 구하고 입력 데이터를 투영한다는 것이다.

1. 클래스 내부와 클래스 간 분산 행렬 구하기.
   - 이 두 개의 행렬은 입력 데이터의 클래스 별로 개별 Feature의 평균벡터를 기반으로 구한다.
2. 클래스 내부 분산 행렬(S_W), 클래스 간 분산 행렬(S_B)를 고유벡터로 분해
   - $$S_W^T S_B = \begin{bmatrix} e_1 & \cdots & e_N \end{bmatrix} \begin{bmatrix} \lambda_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \lambda_N \end{bmatrix} \begin{bmatrix} e_1^T \\ \vdots \\ e_N^T \end{bmatrix}$$
3. 고윳값이 가장 큰 순으로 K개(LDA 변환 차수만큼) 추출
4. 고윳값이 가장 큰 순으로 K개 추출, 고윳값이 가장 큰 순으로 추출된 고유벡터를 이용해 새롭게 입력 데이터를 변환

SVD

특잇값 분해 – SVD(Singular Value Decomposition)

- 고유분해와 더불어 대표적인 행렬 분해 방법이다.
- 고유분해는 정방행렬에 대해서만 분해가 가능하지만, 특잇값 분해는 행과 열의 크기가 다른 행렬도 분해가 가능하다.

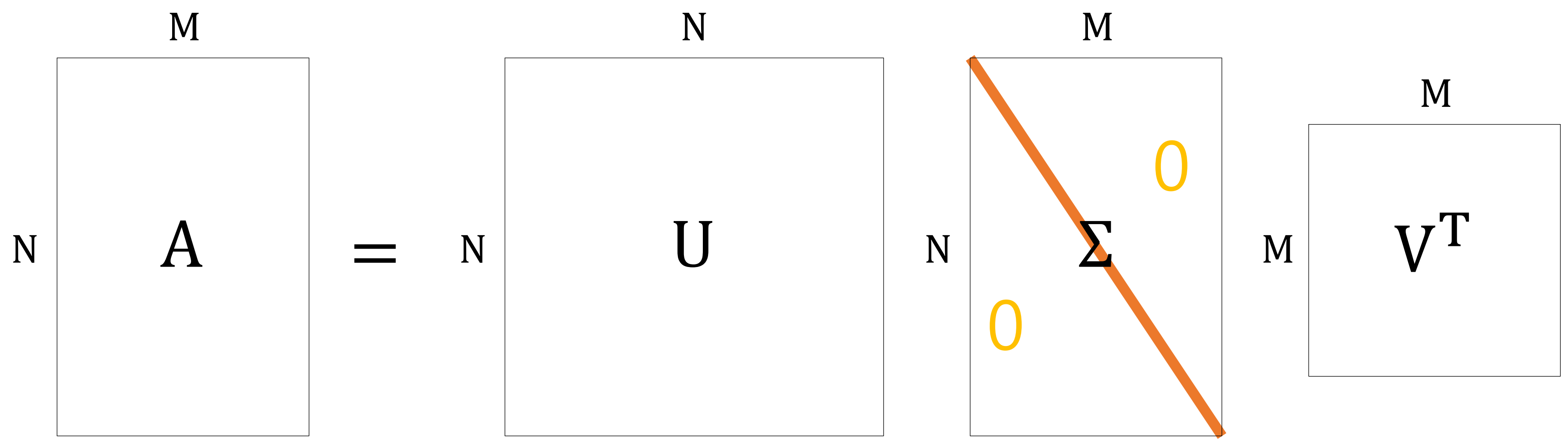
$$A = U\Sigma V^T$$

- 분해된 행렬 U를 왼쪽 특이행렬(왼쪽 직교행렬), Σ를 대각행렬, V^T를 오른쪽 특이행렬(오른쪽 직교행렬)이라고 한다.
- 행렬 U와 V에 속한 벡터를 특이벡터(Singular Vector)라고 하며, 모든 특이벡터는 서로 직교하는 성질을 갖는다.

$$U^T U = I, V^T V = I$$

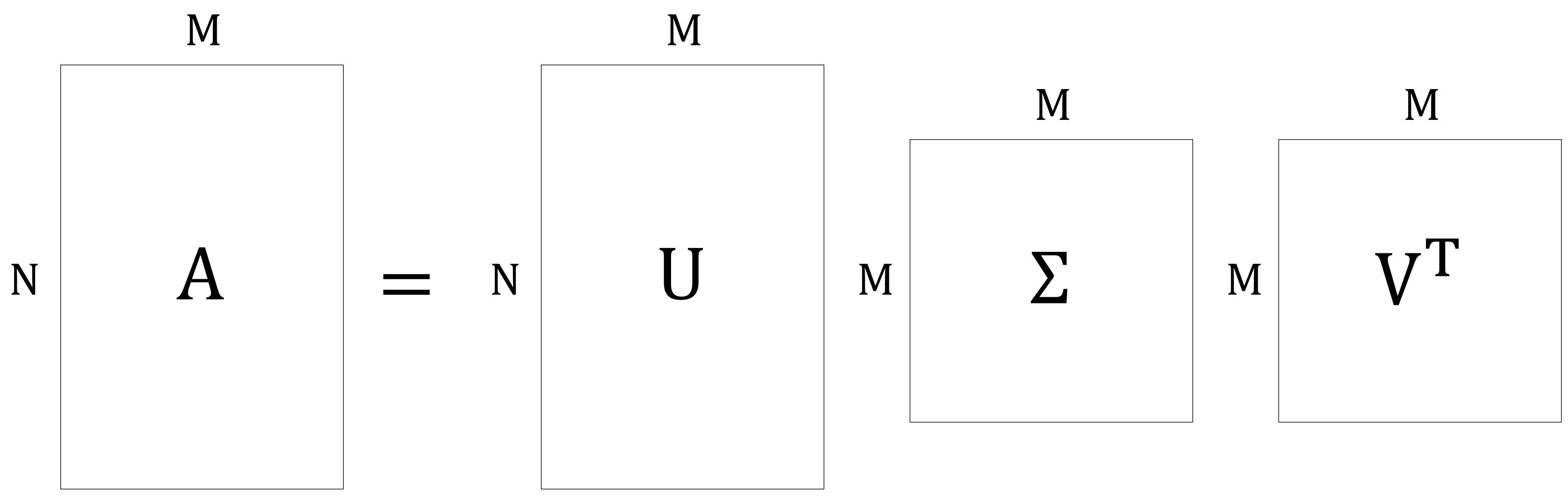
- Σ는 대각행렬이며, 행렬의 대각에 위치한 값만 0이 아니고 나머지 위치의 값은 모두 0이 된다.
- Σ가 위치한 0이 아닌 값이 행렬 A의 특잇값이 된다.

SVD 유형 – Full SVD



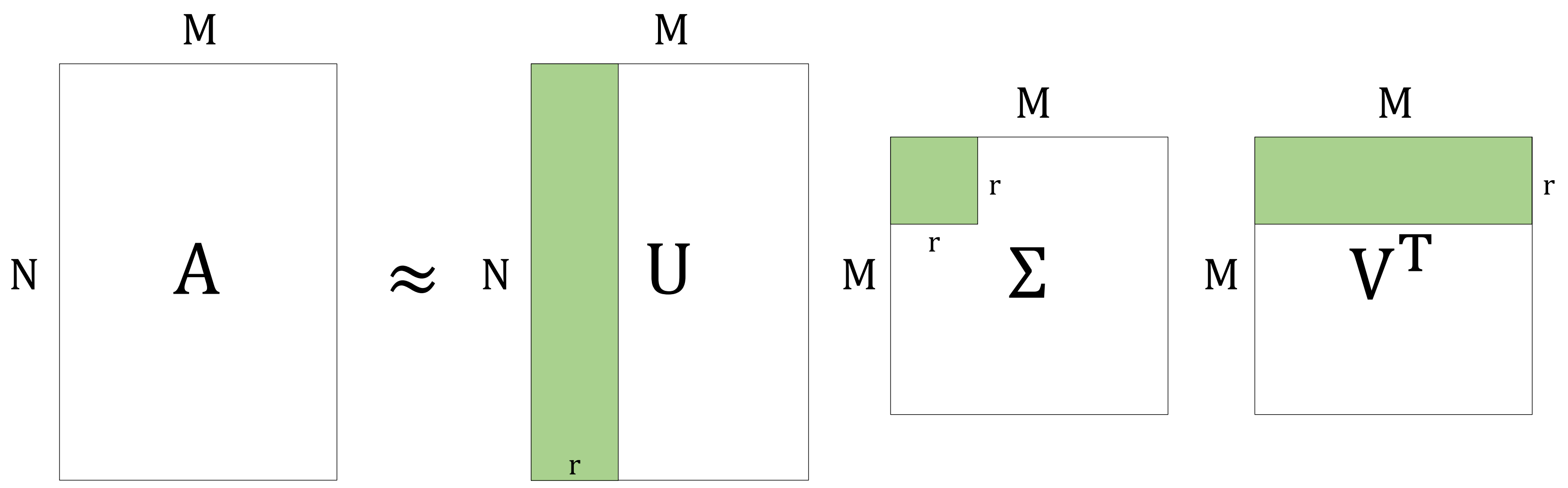
SVD 유형 – Compact SVD

- 비대각 부분과 대각 원소가 0인 부분을 제거한다.



SVD 유형 – Truncated SVD

- 대각 원소 가운데 상위 r개만 추출하여 차원을 축소한다.



Truncated SVD 행렬 분해의 의미

- SVD는 차원 축소를 위한 행렬 분해를 통해 Latent Factor(잠재 요인)을 찾을 수 있다. 이렇게 찾아진 Latent Factor는 많은 분야에 활용된다.(추천 엔진, 문서의 잠재의미 등)
- SVD로 차원 축소 해열 분해된 후 다시 분해된 행렬을 이용하여 원복된 데이터 셋은 잡음이 제거된 형태로 재구성 된다.
  - 데이터를 표현하기 위한 중요한 성질만 남게 된다.
- 사이킷런에서는 TruncatedSVD로 차원 축소할 때 원본 데이터에 를 적용하여 차원을 축소한다.