# 3.2 수식

## 3.2.1 K-평균 클러스터링 (K-means Clustering)

K-means는 주어진 데이터를 사용자가 정한 K개의 그룹(군집)으로 나누는 비지도학습 알고리즘입니다. 각 군집은 하나의 중심점(centroid)을 가지고 있고, 데이터는 이 중심점들과의 거리를 기준으로 군집에 할당됩니다.

이 알고리즘의 핵심 목표는 군집 내 데이터들이 중심점으로부터 얼마나 가까운지를 나타내는 총 거리 제곱합(SSE)을 최소화하는 것입니다.

### 비용 함수

$$J = \sum_{k=1}^{K} \sum_{x_i \in C_k} ||x_i - \mu_k||^2$$

- $x_i$: 데이터 포인트
- $\mu_k$: $k$ 번째 클러스터의 중심점
- $C_k$: $k$ 번째 클러스터에 속한 데이터 집합

이 수식은 각 데이터가 자기 클러스터 중심과 얼마나 떨어져 있는지를 계산하고, 이를 모두 더한 값입니다. 이 값이 작을수록 같은 클러스터 안에 있는 데이터들이 서로 가깝게 모여 있음을 의미합니다.

### 알고리즘 단계

- K개의 중심점을 무작위로 초기화합니다.
- 각 데이터 포인트 $x_i$를 가장 가까운 중심점 $\mu_k$에 할당합니다.
- 새롭게 할당된 군집을 기준으로 중심점을 다시 계산합니다.

$$\mu_k = \frac{1}{|C_k|} \sum_{x_i \in C_k} x_i$$

> $k$ 번째 클러스터에 있는 모든 점들의 좌표 평균을 구하는 것입니다.

- 중심점이 더 이상 크게 이동하지 않을 때까지 2–3번 단계를 반복합니다.

### 특징 및 한계

- 클러스터 수 $K$를 사용자가 정해야 합니다.
- 클러스터의 **초기 중심값**에 따라 결과가 달라질 수 있습니다.
- 둥근(구형) 형태의 군집에 적합합니다.
- 이상치(outlier)에 매우 민감합니다.

## 3.2.2 DBSCAN (Density-Based Spatial Clustering of Applications with Noise)

DBSCAN은 K-means와 달리 클러스터 수 $K$를 미리 정하지 않고, 데이터의 밀도(density)에 기반하여 클러스터를 형성합니다. DBSCAN의 핵심 아이디어는, 밀도가 높은 곳은 하나의 군집으로 보고, 밀도가 희박한 지역은 클러스터가 아닌 잡음(noise)으로 처리한다는 것입니다.

### 주요 개념

- $\varepsilon$: 반경 (이웃으로 간주할 최대 거리)
- minPts: 해당 반경 안에 있어야 할 최소 점 개수

DBSCAN은 각 포인트를 다음 세 가지 중 하나로 분류합니다:

1. **Core Point**: 반경 $\varepsilon$ 안에 minPts 이상의 점이 존재하는 경우

2. **Border Point**: Core Point에 인접하지만, 자기 주변에는 minPts 미만
3. **Noise Point**: 어떤 Core Point에도 속하지 않는 이상치

### 밀도 기반 접근

어떤 점 $x$의 반경 $\varepsilon$ 안에 있는 점들의 집합은 다음과 같이 정의됩니다.

$$N_\varepsilon(x) = \{y \in \mathbb{R}^d \mid ||x - y|| \leq \varepsilon\}$$

> $d$ 차원 공간 위에서, 점 $x$ 로 부터 $\varepsilon$ 이하 거리에 있는 모든 점 $y$ 들을 모은 집합

- $||x - y||$: 두 점 사이의 거리 (보통 유클리디안 거리)
- 이 집합 $N_\varepsilon(x)$ 에 속한 점의 수가 **minPts 이상**이면 $x$는 Core Point

이웃 점의 개수가 minPts 이상이면, 해당 점은 Core Point로 간주하고 군집 확장을 시작합니다. DBSCAN은 이러한 점들을 연결해 가며 밀도가 높은 영역을 하나의 클러스터로 만들어냅니다.

### 장단점

- 클러스터의 모양이 **복잡하거나 불규칙한 경우**에도 잘 작동
- 이상치(Noise)를 따로 분리해낼 수 있음
- $K$값을 정할 필요가 없음
- 하지만 $\varepsilon$ 와 minPts를 잘못 설정하면 성능이 크게 저하될 수 있음

## 3.2.3 PCA (주성분 분석, Principal Component Analysis)

PCA는 고차원 데이터를 핵심 정보만 유지하면서 **저차원으로 변환**하는 차원 축소 기법입니다. 데이터의 분산(variability)을 최대한 보존하는 새로운 좌표축을 찾아 데이터를 이 축들 위로 재표현합니다.

#### 과정

**데이터 중심화**: 평균을 0으로 맞추기 위해 각 데이터에서 평균을 뺍니다. 이 과정을 통해 데이터는 원점을 중심으로 재배열됩니다.

$$x_i' = x_i - \bar{x}$$

$$x_i = \begin{bmatrix} x_i^{(1)} \\ x_i^{(2)} \\ \vdots \\ x_i^{(d)} \end{bmatrix} \in \mathbb{R}^{d \times 1}$$

$$x_i' = \begin{bmatrix} x_i^{(1)} - \bar{x}^{(1)} \\ x_i^{(2)} - \bar{x}^{(2)} \\ x_i^{(3)} - \bar{x}^{(3)} \end{bmatrix} = \begin{bmatrix} a \\ b \\ c \end{bmatrix}$$

**공분산 행렬 계산**:

$$\Sigma = \frac{1}{n} \sum_{i=1}^{n} \boldsymbol{x}_i' (\boldsymbol{x}_i')^T \in \mathbb{R}^{d \times d}$$

$$x_i'(x_i')^T = \begin{bmatrix} a \\ b \\ c \end{bmatrix} \begin{bmatrix} a & b & c \end{bmatrix} = \begin{bmatrix} a^2 & ab & ac \\ ba & b^2 & bc \\ ca & cb & c^2 \end{bmatrix}$$

$$x_i'(x_i')^T = \begin{bmatrix} (x_i^{(1)} - \bar{x}^{(1)})^2 & (x_i^{(1)} - \bar{x}^{(1)})(x_i^{(2)} - \bar{x}^{(2)}) & (x_i^{(1)} - \bar{x}^{(1)})(x_i^{(3)} - \bar{x}^{(3)}) \\ (x_i^{(2)} - \bar{x}^{(2)})(x_i^{(1)} - \bar{x}^{(1)}) & (x_i^{(2)} - \bar{x}^{(2)})^2 & (x_i^{(2)} - \bar{x}^{(2)})(x_i^{(3)} - \bar{x}^{(3)}) \\ (x_i^{(3)} - \bar{x}^{(3)})(x_i^{(1)} - \bar{x}^{(1)}) & (x_i^{(3)} - \bar{x}^{(3)})(x_i^{(2)} - \bar{x}^{(2)}) & (x_i^{(3)} - \bar{x}^{(3)})^2 \end{bmatrix}$$

- 중심화된 각 벡터 $x_i'$에 대해 자기 자신과의 외적(Outer Product)을 계산한 후 평균을 냅니다.

- 결과는 $d \times d$ 크기의 공분산 행렬로, 각 성분은 특성 간의 상관 관계를 나타냅니다.

**고유값 분해**를 통해 분산이 큰 방향(고유벡터)을 찾습니다.

$$\Sigma = Q \Lambda Q^T$$

- $\Sigma$를 고유값 분해하면, $Q$는 고유벡터(Eigenvectors)를 열벡터로 가지는 직교 행렬,
  $\Lambda$는 고유값(Eigenvalues)이 대각선에 위치한 대각 행렬입니다.
- 고유값이 클수록 해당 고유벡터 방향으로의 분산이 큽니다.

**고유값이 큰 순서대로 $k$개를 선택**

- 고유값을 내림차순으로 정렬하고, **상위 $k$개의 고유벡터**를 선택합니다.
- 이 고유벡터들은 데이터가 가장 많이 퍼진 방향을 의미하며, 새로운 축(주성분)이 됩니다.
- 선택된 $k$개의 고유벡터로 구성된 행렬을 $W \in \mathbb{R}^{d \times k}$라고 하면:

**데이터 투영**

$$z_i = W^T x_i'$$

- 중심화된 원래 데이터 를 주성분 축 $W$ 위에 투영하여 새로운 표현 $z_i$를 얻습니다.
- 결과는 $k$-차원 데이터로 변환된 것입니다.

## 해석

- 고유값이 클수록 그 방향에 정보가 많이 담겨 있다는 뜻
- 차원을 줄여도 정보 손실이 적습니다 (적절한 $k$ 선택 시)
- PCA는 **선형 구조를 가정**하며, 전체적인 데이터 분포를 설명하려고 합니다

| 단계 | 설명 | 수식 (미리보기) | 차원 |
|---|---|---|---|
| 1 | 데이터 중심화 (평균 0으로 이동) | $x_i' = x_i - \bar{x}$ | $\bar{x} \in \mathbb{R}^d, \quad x_i' \in \mathbb{R}^d$ |
| 2 | 공분산 행렬 계산 | $\Sigma = \frac{1}{n} \sum_{i=1}^{n} x_i' (x_i')^T$ | $\Sigma \in \mathbb{R}^{d \times d}$ |
| 3 | 고유값 분해 | $\Sigma = Q \Lambda Q^T$ | $Q \in \mathbb{R}^{d \times d}, \quad \Lambda \in \mathbb{R}^{d \times d}$ |
| 4 | 주성분 선택 (상위 k개) | $W = [v_1, v_2, \ldots, v_k]$ | $W \in \mathbb{R}^{d \times k}$ |
| 5 | 데이터 투영 (저차원 표현) | $z_i = W^T x_i'$ | $z_i \in \mathbb{R}^k$ |

# 3.2.4 t-SNE (t-Distributed Stochastic Neighbor Embedding)

t-SNE는 주로 **고차원 데이터를 2D나 3D로 시각화**하기 위한 알고리즘입니다.

PCA와는 달리, 데이터 간의 **전체 구조**보다는 국소 구조(local structure)를 잘 보존하도록 설계되어 있습니다.

즉, **서로 가까운 점은 시각화 결과에서도 가깝게**, 먼 점은 되도록 멀리 유지하는 것이 목표입니다. 고차원 데이터의 구조와 패턴을 유지하면서, 차원 축소를 가능하게 합니다.

## 기본 아이디어

1. 고차원 공간에서 각 점 $x_i$와 $x_j$ 사이의 거리로 유사도 확률 $p_{ij}$ 계산
2. 저차원 공간에서는 $y_i, y_j$로 대응하고, 비슷한 방식으로 $q_{ij}$ 계산
3. 두 확률 분포 사이의 차이(KL Divergence)를 최소화

$$KL(P\|Q) = \sum_{i \neq j} p_{ij} \log \frac{p_{ij}}{q_{ij}}$$

저차원 공간에서는 t-분포를 사용하여, 데이터가 너무 밀집되지 않도록 합니다.

## 특징

- **비선형 차원 축소 기법**이며, 복잡한 데이터 구조 시각화에 탁월
- 데이터가 시각적으로 뚜렷한 클러스터 구조를 띠도록 만들어줌
- 학습 시 랜덤 요소가 있어서 결과가 매번 조금씩 달라질 수 있음
- **속도 느림**, 특히 대용량 데이터에서는 최적화 필요