Foundation of Machine Learning 13주차

정재헌, 우지수 / 2023.05.17

Computational Data Science LAB

CONTENTS

Binary & Multiclass

- Binary classification(이진 분류)
  - ✓ Input space : $x \in \mathbb{R}^d$
  - ✓ Output space : $y \in \{-1, 1\}$

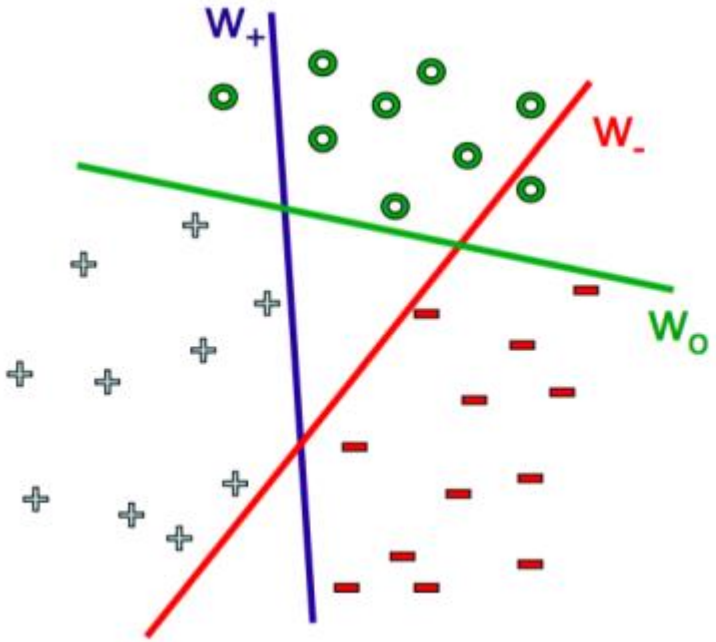
- Multiclass classification(다중 클래스 분류)
  - ✓ Input space : $x \in \mathbb{R}^d$
  - ✓ Output space : $y \in \{1, 2, \dots, k\}$

What is One vs Rest

- One vs Rest
  - ✓ Decision Boundary를 기준으로 $\{-1, +1\}$인 binary classification을 k개 학습
  - ✓ $h_1, \dots, h_k : \mathcal{X} \rightarrow R$이라고 가정했을 때,

$$h(x) = \underset{i \in \{1,\dots,k\}}{\operatorname{argmax}} h_i(x)$$

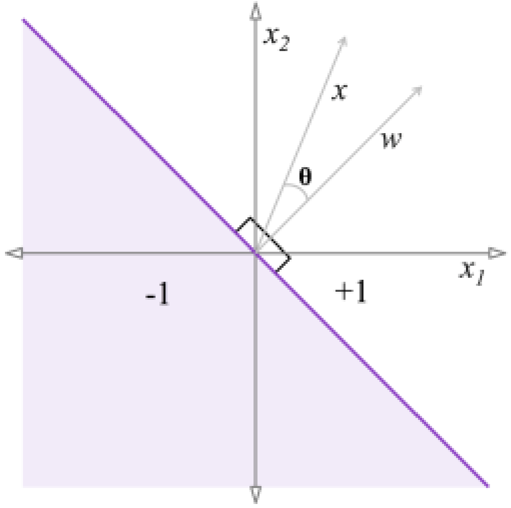
  - ✓ 각 분류기의 출력값 중 가장 큰 값을 가지는 클래스를 선택



02 One vs Rest

Linear Binary Classifier Review

- Linear Binary Classifier
  - ✓ Input space : $x \in \mathbb{R}^d$
  - ✓ Output space : $y \in \{-1, 1\}$
  - ✓ Linear classifier score function:
    $$\boldsymbol{f}(\boldsymbol{x}) = <\boldsymbol{w}, \boldsymbol{x}> = \boldsymbol{w}^T\boldsymbol{x}$$
  - ✓ Final classification prediction: $\boldsymbol{sign}(\boldsymbol{f}(\boldsymbol{x}))$

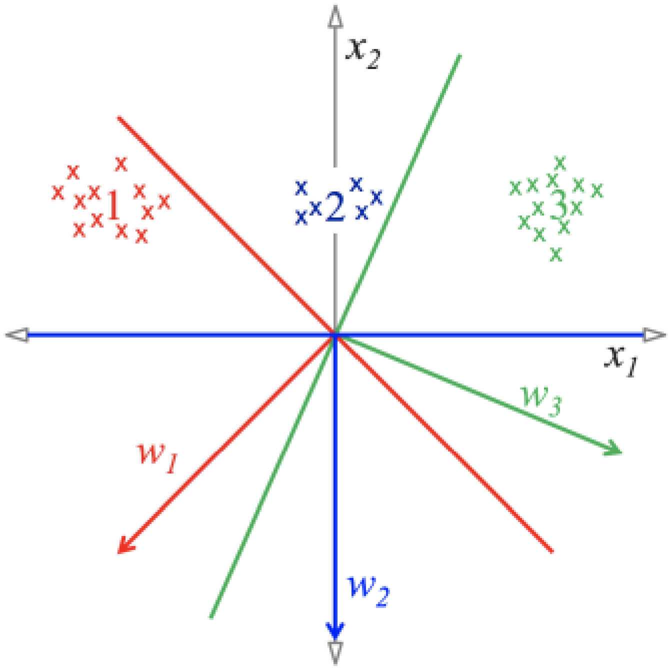


- Suppose $|| w || > 0$ and $|| x || > 0$:
  - ✓ $f(x) = <w, x> = || w || \, || x || \cos\theta$
  - ✓ $f(x) > 0 \leftrightarrow cos\theta > 0 \leftrightarrow \theta \in (-90°, 90°)$
  - ✓ $f(x) < 0 \leftrightarrow cos\theta < 0 \leftrightarrow \theta \notin [-90°, 90°]$

02 One vs Rest

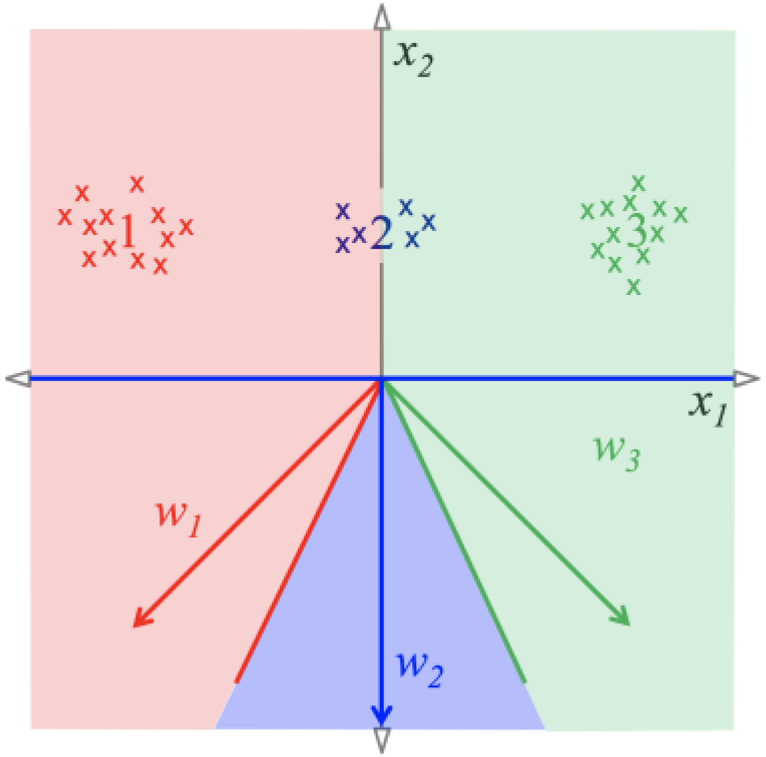
One-vs-Rest: Three Class Example

- Base hypothesis space
  - ✓ $\mathcal{H} = \{f(x) = w^T x \mid x \in R^2\}$
  - ✓ Class 1 vs Rest : $\boldsymbol{f_1(x) = w_1^T x}$
  - ✓ Class 2 vs Rest : 해당 sample에서 Class 2를 제대로 분류하기 어려움
    따라서 not 2인 부분으로 예측하는 것이 전체적인 정확도를 높일 수 있음
- Prediction
  - ✓ Class i 에 대한 점수(score):
    $$f(x) = < w, x > = || w || \, || x || \cos\theta$$
  - ✓ 이때, θ_i는 x와 w_i사이 각도



One-vs-Rest: Three Class Example

- Class Boundaries
  - ✓ $|| w_1 || = || w_2 || = || w_3 ||$ 이라고 가정
  - ✓ 내적을 통해 계산되는 각 클래스 점수는 각도(θ)에 의해서 결정
  - ✓ 입력벡터 x는 각 가중치벡터에서 각도가 가장 작은 Class에 할당
    - → $cos\theta_i$에서 θ_i가 0에 가까울 수록 $cos\theta$가 1에 가까이 가기 때문



Multiclass Hypothesis Space

- Base Hypothesis Space:
  - ✓ $\mathcal{H} = \{h: \mathcal{X} \to R\}$ (score functions)
- Multiclass Hypothesis Space (for k classes):
  - ✓ $\mathcal{F} = \left\{x \to \underset{i}{\operatorname{argmax}}\, h_i(x) \mid h_1, \ldots, h_k \in \mathcal{H}\right\}$
  - ✓ $h_i(x)$ score는 입력벡터 x가 class i에 얼마나 가까운지에 대한 정보
    - ❖ 모델이 k개의 클래스에 각각에 대한 별도의 score 함수를 학습해야함
    - ❖ 단점은 클래스의 수(k)가 많아지면 학습이 힘듦, 연산적으로 비효율적

Multiclass Hypothesis Space: Reframed

- Discrete Output Space : $\mathcal{Y}$ (e.g. $\mathcal{Y} = \{1, \dots, k\}$ for multiclass)
- 새로운 접근법
  - ✓ input x 와 output y 사이의 **호환성(compatibility) score**를 제공하는 $\boldsymbol{h(x, y)}$를 사용
  - ✓ 이 때, 호환성 score란 특정 input x 가 특정 class y 에 얼마나 잘 맞는지 나타내는 척도
  - ✓ 함수 $f(x)$는 input x 에 대해 가장 호환성 score가 높은 class y를 최종 예측값으로 선택:

$$f(x) = \underset{y \in \mathcal{Y}}{\operatorname{argmax}}\, h(x, y)$$

- 새로운 접근법에서의 Hypothesis Space
  - ✓ $\mathcal{H} = \{h: \mathcal{X} * \mathcal{Y} \to R\}$ (score functions)
  - ✓ $\mathcal{F} = \left\{x \to \underset{y \in \mathcal{Y}}{\operatorname{argmax}}\, h(x, y) \mid h \in \mathcal{H}\right\}$

03 | Multiclass

Multiclass Hypothesis Space: In Math

- Training Data: $(x_1, y_1), \ldots (x_n, y_n)$
  - ✓ input x가 레이블 y를 가질 때 호환성 함수 $h(x, y)$가 크게, 그렇지 않을 때는 작게 하기 원함
- $h(x, y)$가 input (x_i, y_i)를 가질 때 :
  - ✓ $h(x_i, y_i) > h(x_i, y) \; \forall y \neq y_i$
  - ✓ 즉, 실제 레이블에 대한 점수가 다른 레이블에 대한 점수보다 클 때 정확한 분류
- 해당 조건을 다른 방식으로 작성해보면:
  - ✓ $h(x_i, y_i) > \max_{y \neq y_i} h(x_i, y)$
  - ✓ $\boldsymbol{m_i = h(x_i, y_i) - \max_{y \neq y_i} h(x_i, y)}$
  - ✓ m_i를 실제 레이블에 대한 score와 다른 모든 레이블에 대한 **최대 score와의 차이로 정의**
  - ✓ 분류는 m_i가 0보다 클 때 정확하다고 할 수 있음(일반적으로 m_i를 크게 하는 것이 목표)

03 | Multiclass

Linear Multiclass Prediction Function

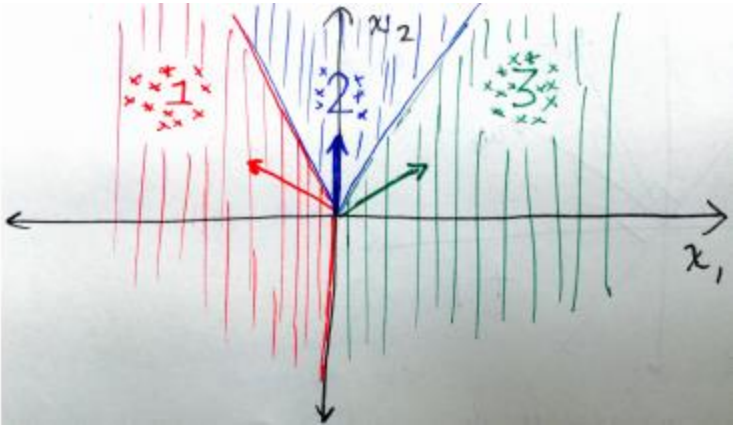
- linear compatibility score function: $h(x,y) =< w, \psi(x,y) >$
  - ✓ 여기서 $\boldsymbol{\psi}(\boldsymbol{x},\boldsymbol{y})$는 **호환성 특징 맵**
  - ✓ $\psi(x,y)$는 input x와 레이블 y의 호환성에 관련된 특징을 추출
  - ✓ 최종 호환성 score는 $\psi(x,y)$의 선형함수
- Example: $\mathcal{X} = R^2, \mathcal{Y} = \{1,2,3\}$
  - ✓ $w_1 = \left(-\frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2}\right), w_2 = (0,1), w_3 = \left(\frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2}\right)$
  - ✓ Prediction function: $(x_1, x_2) \to \underset{i \in \{1,2,3\}}{\operatorname{argmax}} < w_i, (x_1, x_2) >$
  - ✓ 가중치 벡터 w_i를 stack(다중 벡터 구조):

$$w = \left(-\frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2}, 0, 1, \frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2}\right)$$

  - ✓ **매핑**함수 정의

$$\psi : R^2 * \{1,2,3\} \to R^6$$

$$\psi(x,1) \coloneqq (x_1, x_2, 0,0,0,0)$$
$$\psi(x,2) \coloneqq (0,0, x_1, x_2, 0,0)$$
$$\psi(x,3) \coloneqq (0,0,0,0, x_1, x_2)$$



04 | Linear Multiclass SVM

The Margin for Multiclass

- $h: \mathcal{X} * \mathcal{Y} \rightarrow R$ 을 호환성 score 함수라고 했을 때,
- 옳은 class와 다른 class 사이의 margin:

$$\boldsymbol{m_{i,y}(h) = h(x_i, y_i) - h(x_i, y)}$$

  - ✓ 각각의 데이터 포인트 i에 대해, 실제 class y_i와 다른 class y간의 마진 $m_{i,y}$
  - ✓ 이 마진이 크다는 의미는 올바른 클래스 y_i에 대한 score가 다른 class y의 score보다 훨씬 높다는 것을 의미
  - ✓ $m_{i,y}(h)$가 크고 **양수**이길 원함
- 선형 가설 공간의 경우
  - ✓ **매핑함수($\boldsymbol{\psi}$)**를 이용해서 고차원의 벡터를 계산해줌

$$m_{i,y}(w) = < w, \psi(x_i, y_i) > \; - < w, \psi(x_i, y) >$$

Multiclass SVM

- Binary SVM :

✓ $$\min_{w\in R^d} \frac{1}{2}||w||^2 + \frac{c}{n}\sum_{i=1}^{n} max(0,1-\underbrace{y_i w^T x_i}_{\text{margin}})$$

- Multiclass SVM :

✓ $$\min_{w\in R^d} \frac{1}{2}||w||^2 + \frac{c}{n}\sum_{i=1}^{n} max_{\{y\neq y_i\}}\left[max\left(0,1-m_{i,y}(w)\right)\right]$$

where, $m_{i,y}(w) = < w, \psi(x_i, y_i) > \; - < w, \psi(x_i, y) >$

✓ $m_{i,y}(w)$는 i번째 sample에 대해 class y와 실제 class y_i 사이의 마진을 나타냄

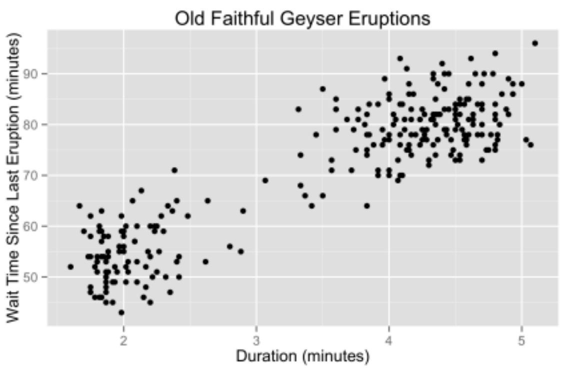
→ 두 SVM 모두 마진을 최대화 하려는 목표지만 다중 클래스 SVM의 경우 **각 클래스 간의 마진**을 고려하게 됨

Multiclass & k-means clustering

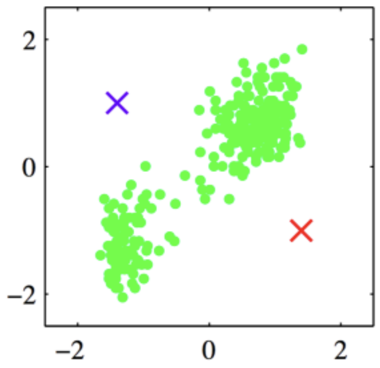
- Multiclass classification:
  - ✓ 주어진 입력에 대해 여러 개의 가능한 클래스 중 하나를 예측하는데 사용되는 **지도 학습** 방법
  - ✓ 각 입력 데이터 하나에 레이블이 함께 제공, 레이블을 학습한 후 예측
- k-means clustering :
  - ✓ 주어진 입력 데이터를 k개의 **클러스터**로 **그룹화**하는데 사용되는 **비지도 학습** 방법
  - ✓ 레이블 제공되지 않음, 데이터의 구조와 분포를 사용해 유사한 데이터 **그룹화**
  - ✓ 단점 : 알고리즘 초기에 중심점을 무작위로 선택하는 위치에 따라 “local minimum(지역 최적해)”에 빠질 수 있음

05 | K-Means Clustering

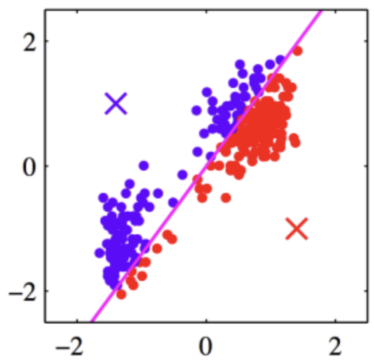
k-means clustering



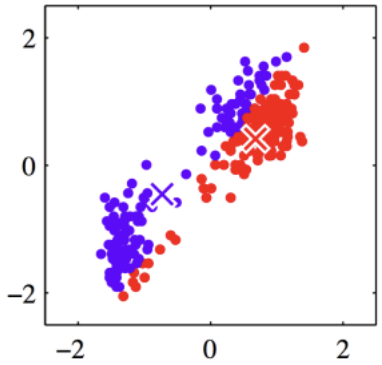
→ 두개의 클러스터 존재



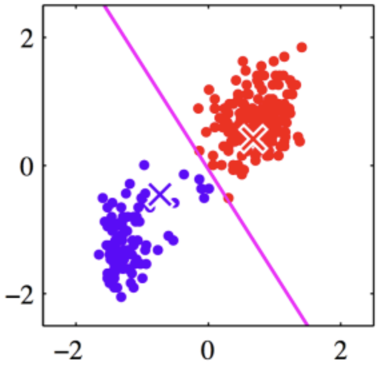
→ 데이터 표준화, 클러스터 센터 선정



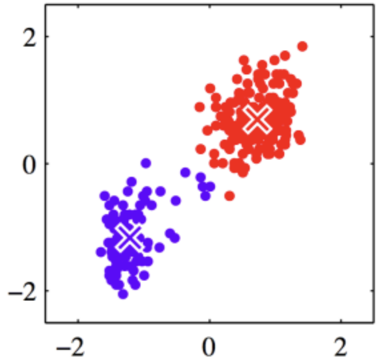
→ 각 데이터를 가장 가까운 중심점에 소속



→ 중심점에 할당된 데이터들의
평균 중심으로 이동



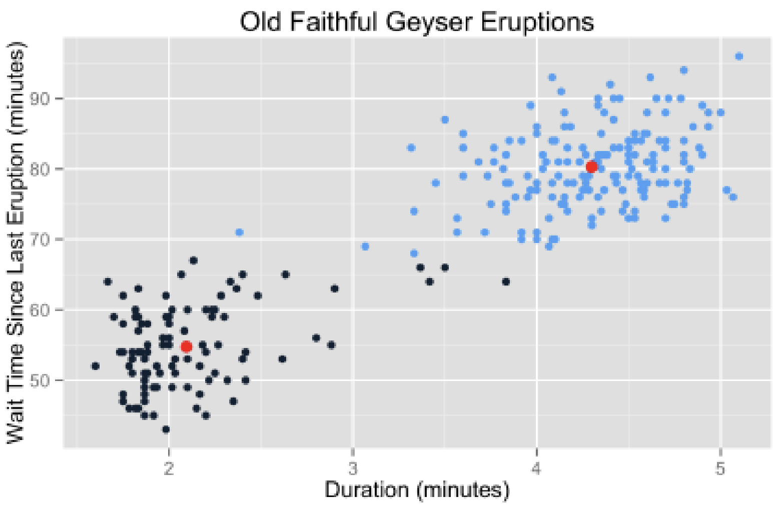
→ 각 데이터들은 이동된 중심점 기준으로
가장 가까운 중심점에 소속



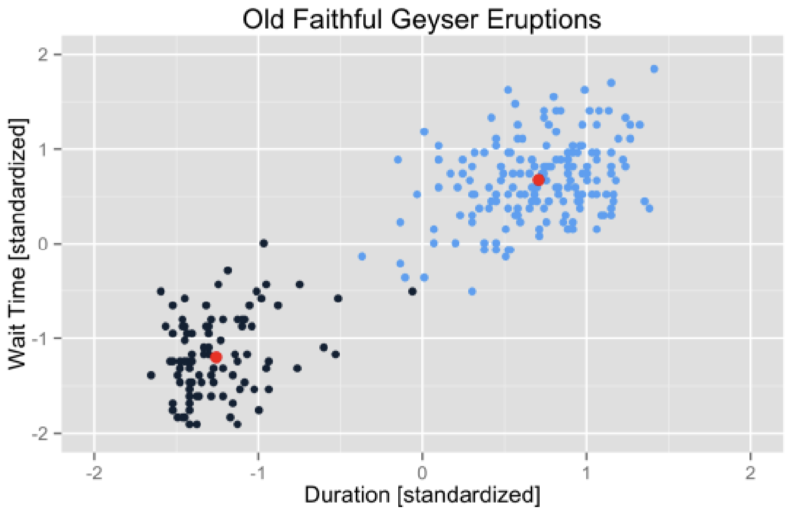
→ 다시 중심점에 할당된 데이터들의 평균
중심으로 중심점 이동

05 | K-Means Clustering

Standardizing the data



→ without standardizing



→ with standardizing

- k-means clustering :
  - ✓ 거리기반 알고리즘이기에 데이터의 scale에 매우 민감해짐 **표준화를 통해 동일한 스케일로 변환**
  - ✓ 거리 측정할 때 유클리디안 거리를 기반으로 계산하는데 특성마다 범위가 다르면 거리계산이 왜곡 가능
  - ✓ K-means의 식은 각 특성의 값들이 거리에 제곱 되어 더해지므로 각 특성의 값의 범위가 서로 다르면 **거리 계산에 영향을 미침**

Formalized

- Input Space : $\mathcal{X} = R^d$
- 거리 식 : $d(x, x') = || x - x' ||$
  - ✓ 이때, d는 임의의 거리 metric
- Cluster 중심을 구하는 식 : $\mu_i = \mu(C_i) = \underset{\mu \in \mathcal{X}}{\operatorname{argmin}} \sum_{x \in C_i} d(x, \mu)^2$
- k-means의 목적함수 : 클러스터링의 결과를 평가하기 위한 **잔차 제곱합(거리 제곱합)**

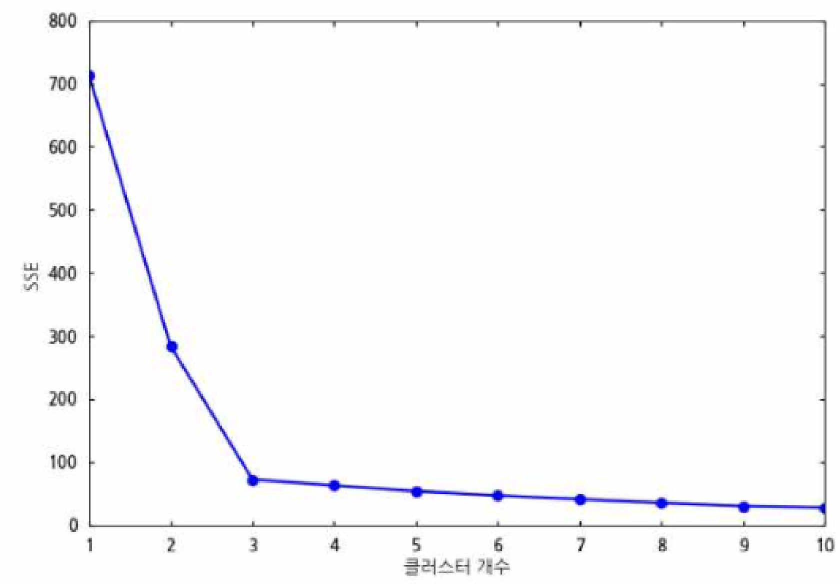
$$J_{k-means}(C_1, \dots, C_k) = \sum_{i=1}^{k} \sum_{x \in C_i} d\big(x, \mu(C_i)\big)^2$$

$$= \min_{\mu_1, \dots, \mu_k \in \mathcal{X}} \sum_{i=1}^{k} \sum_{x \in C_i} d(x, \mu_i)^2$$

- 각 데이터까지의 거리의 제곱합을 최소화하는 방법으로 초기 군집수 결정

05 K-Means Clustering

Formalized



- k가 증가함에 따라 오차제곱합이 감소
- 감속의 폭이 일정하게 유지되기 시작하는 적정 k를 적정 군집으로 선택

Q&A

감사합니다.