BITAmin 13주차

K-means 군집화

5조
최대상, 조은정, 조예진

K-means

개념

K-means 개념
K-NN과 차이점

과정

K-means
군집화 과정

장단점

K-means
단점 개선

PCA

K-means와
PCA 연관성

01_K-means 개념

02_K-means 과정

03_K-means 장단점

04_K-means&PCA

01

K-means 개념

K-means vs. K-NN

K-means 개념

- 비지도학습
- 데이터를 K개의 군집(Cluster)로 묶는 알고리즘

군집: 비슷한 특성을 지닌 데이터들을 모은 그룹

군집화: 비슷한 특성을 지닌 데이터들을 군집으로 묶는 것

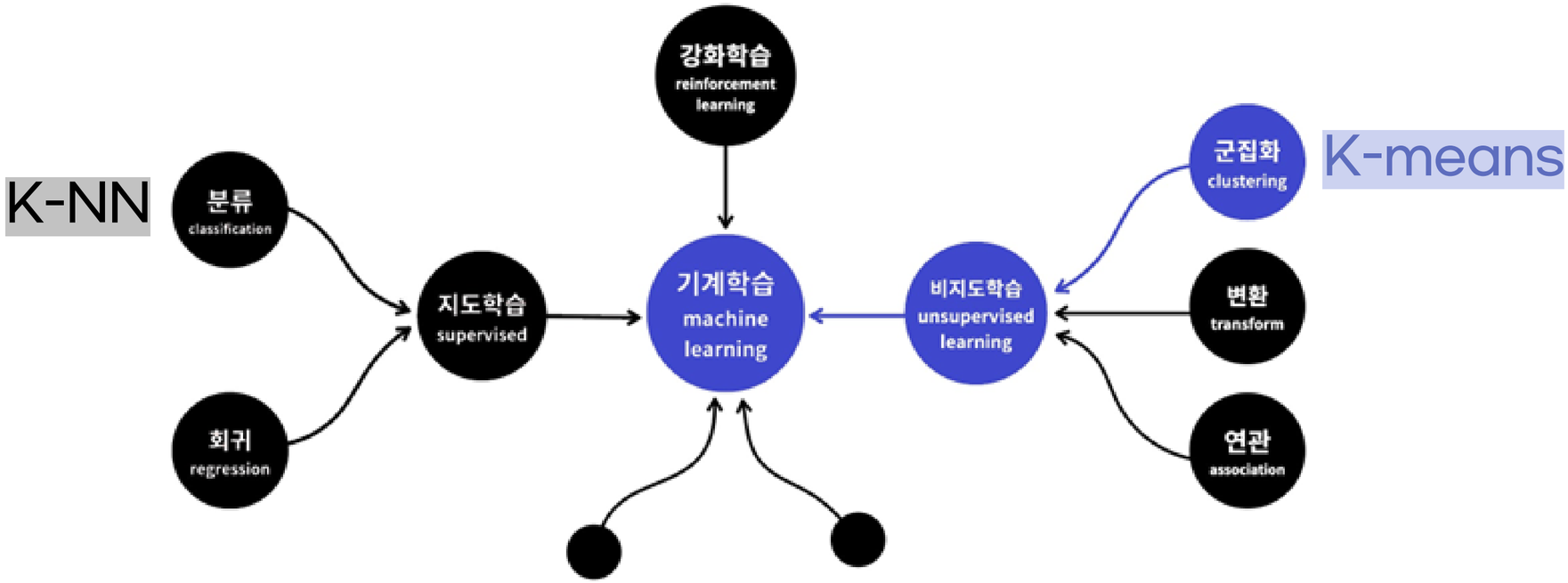
K - means

묶을 군집의 개수 평균

각 군집의 평균을 활용하여 K개의 군집으로 묶는다는 의미

K-means vs. K-NN

두 알고리즘 모두 K개의 점을 지정하여
거리를 기반으로 구현되는 **거리기반 분석 알고리즘**



01_K-means 개념

02_K-means 과정

03_K-means 장단점

04_K-means&PCA

02

K-means 군집화 과정

K-means 과정

Step 1

군집의 개수
K 설정

Step 2

무작위로
K개의 초기
중심점 설정

Step 3

각 데이터에서
가장 가까운
중심을 찾아 할당

Step 4

군집에 속한
데이터의 평균값
으로 군집 중심점
재설정

Step 5

데이터를 군집에
재할당

중심점의 위치가
더 이상 변하지 않을 때까지
Step4, 5 반복

K-means 과정

Step 1

군집의 개수 K 설정

Step 2

무작위로 K개의 초기 중심점 설정

Step 3

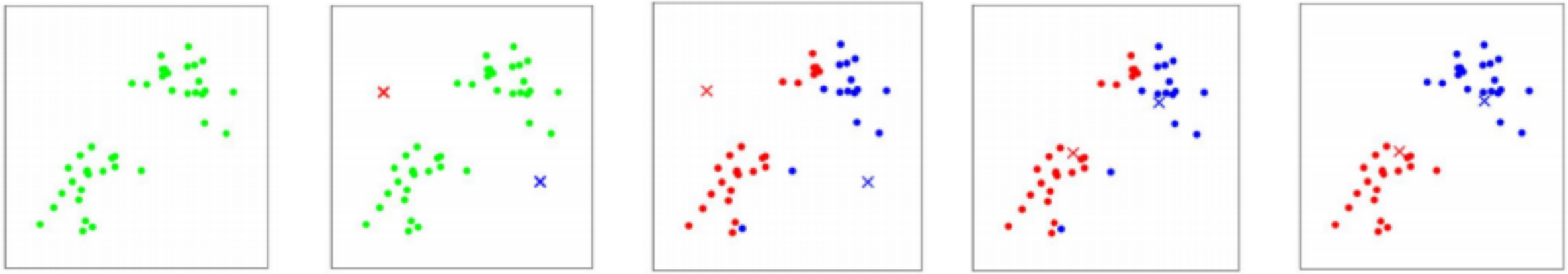
각 데이터에서 가장 가까운 중심을 찾아 할당

Step 4

군집에 속한 데이터의 평균값으로 군집 중심점 재설정

Step 5

데이터를 군집에 재할당



K=2

K-means 과정

Step 1

군집의 개수
K 설정

Step 2

무작위로
K개의 초기
중심점 설정

Step 3

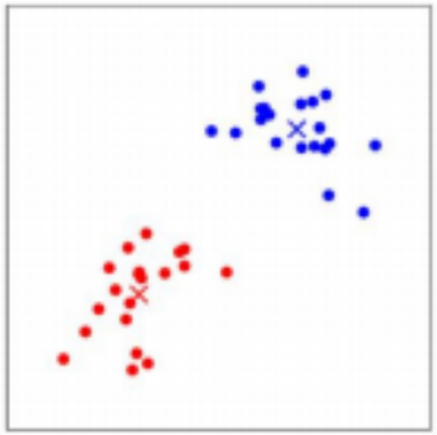
각 데이터에서
가장 가까운
중심을 찾아 할당

Step 4

군집에 속한
데이터의 평균값
으로 군집 중심점
재설정

Step 5

데이터를 군집에
재할당



중심점의 위치가
더 이상 변하지 않을 때까지
Step4, 5 반복

01_K-means 개념

02_K-means 과정

03_K-means 장단점

04_K-means&PCA

03

K-means 장단점

단점 개선 방안

K-means 장단점

장점

- 직관적이고 쉬운 구현
- 비교적 빠르고, 대용량 데이터 세트에도 잘 작동
- 수렴성이 보장

K-means 장단점

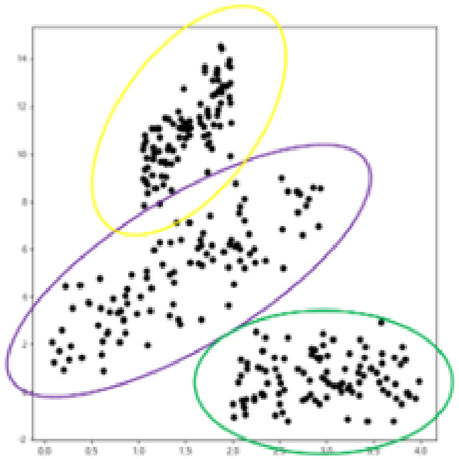
장점

- 직관적이고 쉬운 구현
- 비교적 빠르고, 대용량 데이터 세트에도 잘 작동
- 수렴성이 보장

단점

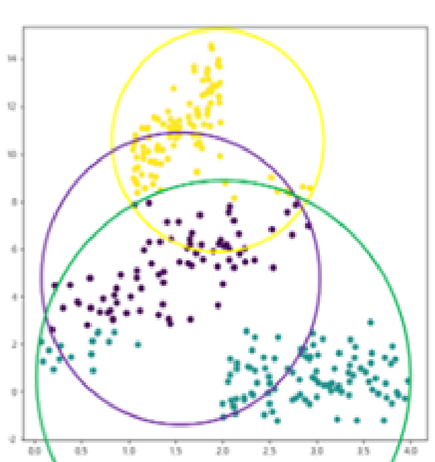
- 중심점을 갱신하는 과정에서 이상치에 영향을 받음
- 그룹 내 분산 구조를 반영할 수 X
- 원형이 아닌 군집을 구분하지 못함
- 차원의 저주에 걸릴 수 있음
- 범주형 변수가 있으면 K-means 불가
  - ㄴ 해결책: K-modes 알고리즘 이용
- 군집의 개수 K를 정해야 함
- 초기값(=초기 중심점)에 민감

세 개의 분산 구조

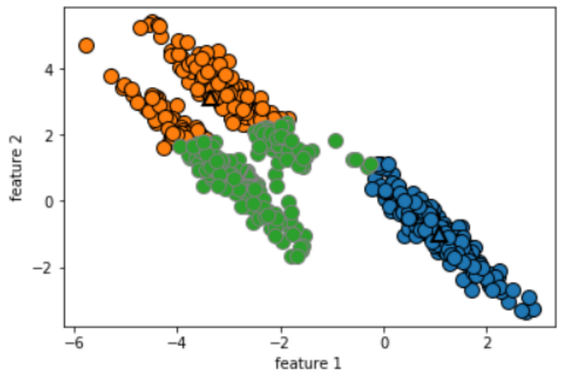


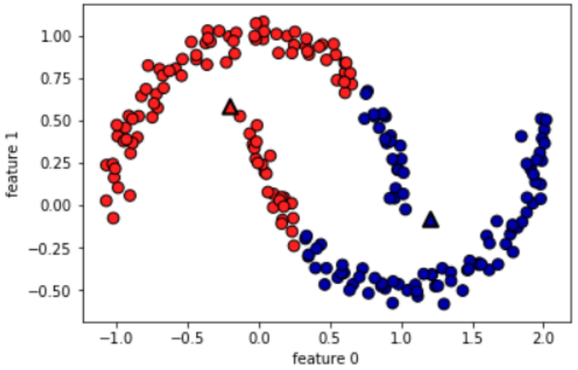
목적: 타원 형태 군집화

실제 K-means 결과



유클리디안 거리의 영향으로
타원 구조가 아닌 원형 구조





K-means 단점 개선

Step 1. 군집의 개수 K 설정

군집의 개수 설정을 어떻게 하는지에 따라 결과가 크게 변화

군집 개수 K 설정 방법론

01. Rule of thumb

$$k \approx \sqrt{n/2}$$

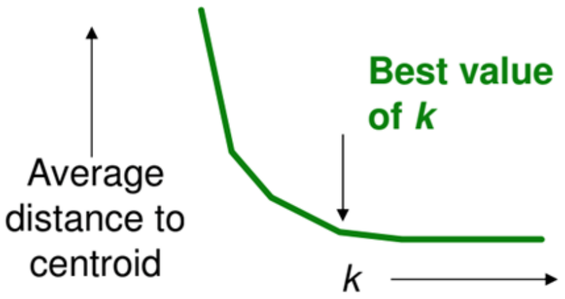
n: 데이터의 개수

02. 정보 기준 접근법

클러스터링 모델에 대해
가능도를 계산하는 것이
가능할 때 사용하는 방법

But. 일반적으로 가능도를
계산할 수 있는 데이터가 많이
없기 때문에 Rule of Thumb,
Elbow Method를 사용

03. Elbow Method



K-means 단점 개선

Step 1. 군집의 개수 K 설정

03. Elbow Method

이너셔 inertia 군집 중심과 군집에 속한 데이터 사이의 거리의 제곱 합
군집에 속한 데이터가 얼마나 가깝게 모여 있는지를 나타내는 값

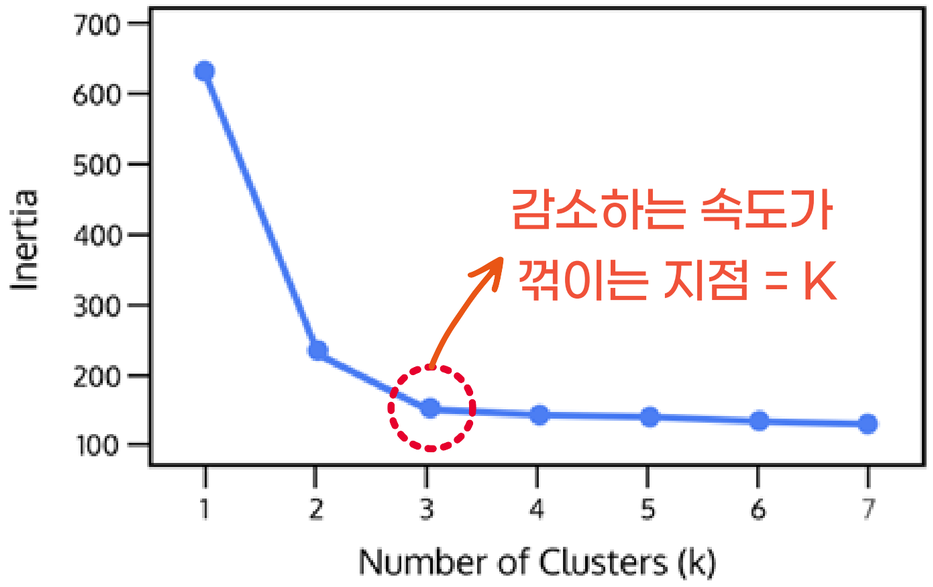
엘보우 방법은 군집의 개수를 늘려가면서 이너셔의 변화를 관찰하여
최적의 군집 개수를 찾는 방법

K-means 단점 개선

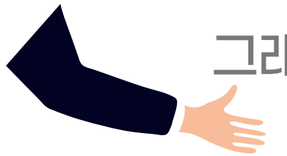
Step 1. 군집의 개수 K 설정

03. Elbow Method

군집 개수를 증가시켰을 때 이너셔의 변화



이 지점부터는 군집 개수를 늘려도 군집에
잘 밀집된 정도가 크게 개선되지 X



그래프가 팔꿈치 모양과 닮아 엘보우 방법

sklearn.cluster 모듈 아래 KMeans 클래스에서
자동으로 이너셔를 계산하여 inertia_ 속성으로 제공

K-means.py

```
from sklearn.cluster import KMeans

def elbow(data, length):
    inertia = []
    for k in range(2,length):
        kmeans = KMeans(n_clusters=k, random_state=42)
        kmeans.fit(data)
        inertia.append(kmeans.inertia_)
    plt.plot(range(2,length), inertia)
    plt.xlabel('Number of Clusters(k)')
    plt.ylabel('inertia')
    plt.show()
elbow(data, length)
```

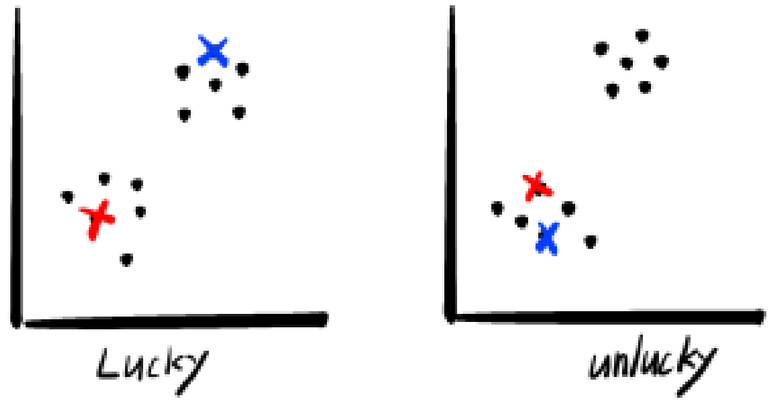
K-means 단점 개선

Step 2. 무작위로 K개의 초기 중심점 설정

K개의 초기 중심점(Center of Cluster, Centroid)을 설정
무게중심

K-means 알고리즘은 초기 중심점으로 어떤 값을 선택하는가에 따라 성능이 크게 달라짐

-> 초기 중심점을 잘 설정해야 함



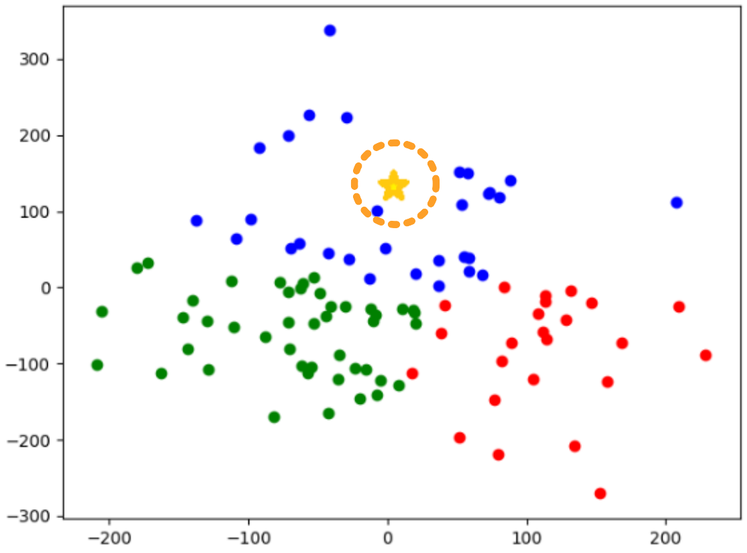
초기 중심점 설정 방법론

Randomly select **Manually assign** **K-means++**

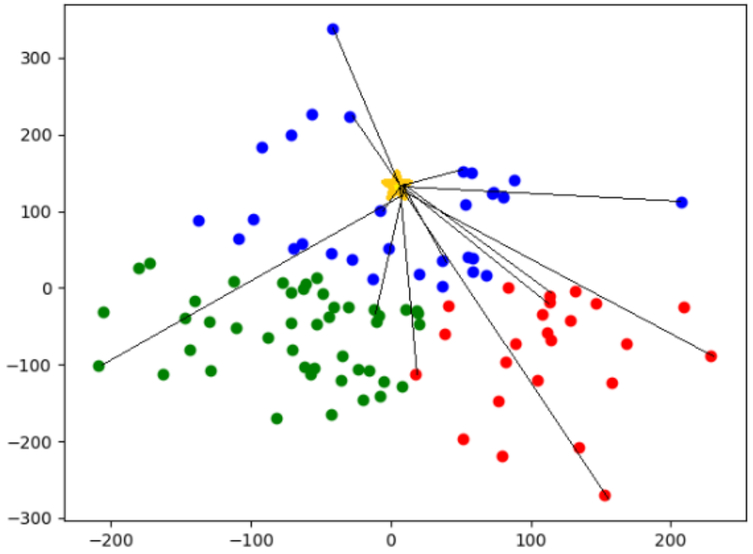
K-means 단점 개선

초기 중심점 설정 방법론

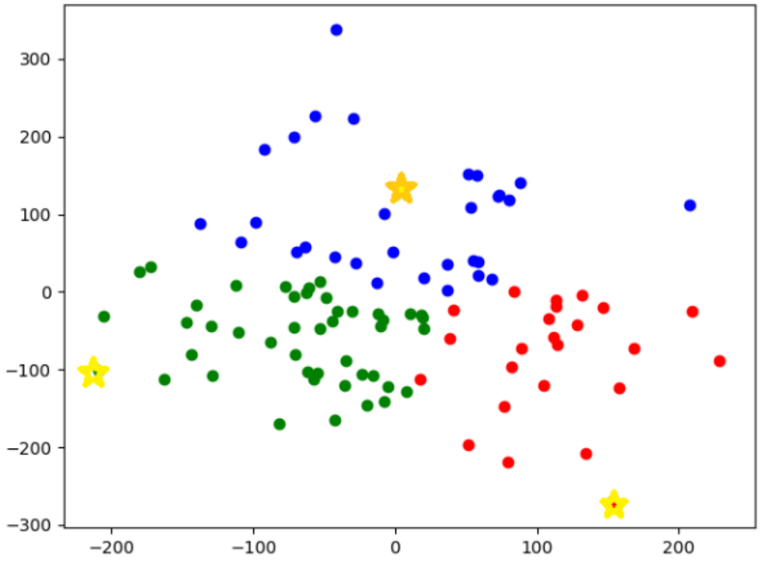
K-means++



중심점을 한 번에 K개를 모두 생성하는 것이 아니라, 데이터 포인터 중에서 무작위로 '1개'를 선택하여, 이 데이터를 첫 번째 중심점으로 지정



나머지 데이터 포인터들과 중심점 사이의 거리를 계산



다음 생성할 중심점들의 위치는,
데이터 포인터들과 전 과정에서 계산한
중심점 사이의 **거리 비례 확률**에 따라 선정
=> 데이터 사이의 거리에서 **최대한 먼 곳**에
다음 중심점을 생성한다는 의미

K-means 단점 개선

초기 중심점 설정 방법론

K-means++

앞선 과정을 K번 반복하여 총 K개의 중심점 생성

한 번에 K개의 중심점을 생성하는 것이 아니라,
중심점 사이의 거리를 최대한 멀리 위치시키는 방향으로
1개씩 총 K번 반복하여 K개의 중심점을 만들어내는 것

K-means.py

```
from sklearn.cluster import KMeans

kmeans = KMeans(init='random', n_cluster=k)
```

K-means++.py

```
from sklearn.cluster import KMeans

kmeans = KMeans(init='k-means++', n_cluster=k)
```

sklearn KMeans 클래스 init: default='k-means++'

미니배치 K-means

K-means는 중심위치와 모든 데이터 사이의 거리를 계산 **데이터 개수 ↑** **계산량 ↑**

데이터의 수가 너무 많을 때는 미니배치 K-means 군집화 방법을 사용하면 계산량을 줄일 수 있음

미니배치 K-means

데이터를 미니배치 크기만큼 무작위로 분리하여 K-means 군집화

모든 데이터를 한번에 썼을 때와 군집 결과가 다를 수는 있지만 큰 차이 X

MiniBatch_K-means.py

```
from sklearn.cluster import MiniBatchKMeans
```
sklearn.cluster 모듈에서 MiniBatchKMeans 클래스 제공

```
from sklearn.datasets import make_blobs
X, y = make_blobs(n_samples=150000, cluster_std=[1.0, 2.5, 0.5], random_state=170)
```
150,000개 데이터를 이용하여 실행 시간 비교

K-means.py

```
%%time
model1 = KMeans(n_clusters=3).fit(X)
```

```
CPU times: user 1.48 s, sys: 3.32 s, total: 4.81 s
Wall time: 10 s
```

MiniBatch_K-means.py

```
%%time
model2 = MiniBatchKMeans(n_clusters=3, batch_size=1000,
                         compute_labels=True).fit(X)
```

```
CPU times: user 340 ms, sys: 1 s, total: 1.34 s
Wall time: 2.9 s
```

속도 ↑

01_K-means 개념

02_K-means 과정

03_K-means 장단점

04_K-means&PCA

04

K-means와 PCA

K-means와 PCA

K-means는 군집 알고리즘이지만, PCA 분해 알고리즘과 유사한 점을 가짐

PCA

데이터에서 분산이 가장 큰 방향을 찾음

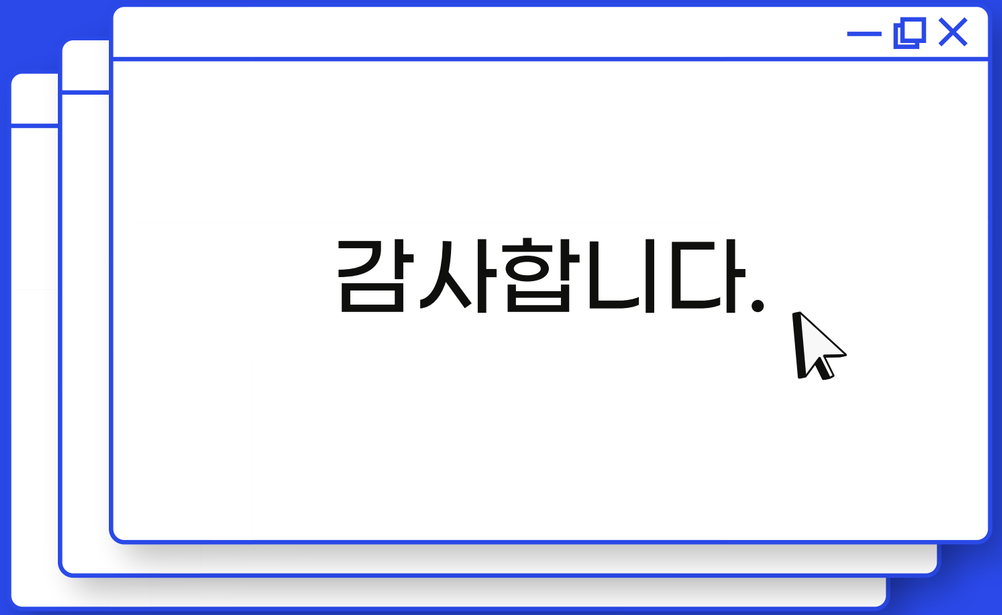
데이터 포인트를 성분의 합으로 표현

K-means

군집 중심으로 각 데이터 포인트를 표현

이를 각 데이터 포인트가 군집 중심,
즉 하나의 성분으로 표현된다고 볼 수 있음

K-means를 각 포인트가
하나의 성분으로 분해되는 관점으로 보는 것을
벡터 양자화(Vector Quantization)


감사합니다.