

# 9장 비지도학습 1부

# 주요 내용

- 군집/군집화
- k-평균
- DBSCAN
- 가우스 혼합

## 비지도 학습이란?

- 레이블이 없는 데이터 학습
- 용도
  - 군집화(clustering)
  - 이상치 탐지
  - 데이터 밀도 추정

## 군집화

- 비슷한 샘플끼리 군집 형성하기
- 활용 예제
  - 고객 분류
  - 추천 시스템
  - 데이터 분석
  - 차원 축소
  - 특성 공학
  - 준지도 학습
  - 검색 엔진
  - 이미지 분할

## 이상치 탐지

- 정상 데이터 이상치 구분
- 활용 예제
  - 제조라인에서 결함 제품 탐지
  - 시계열데이터에서 새로운 트렌드 찾기

## 데이터 밀도 추정

- 데이터셋의 확률 밀도 추정
- 활용 예제:
  - 이상치 분류: 밀도가 낮은 지역에 위치한 샘플
  - 데이터 시각화
  - 데이터 분석

## 9.1. 분류 대 군집화

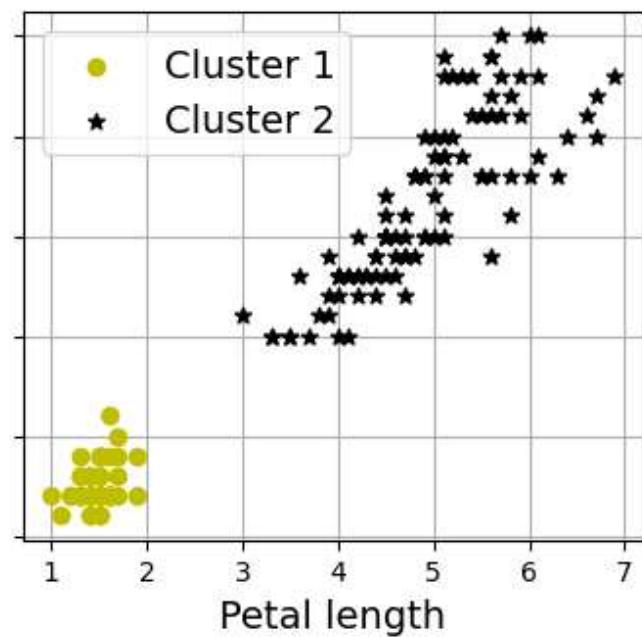
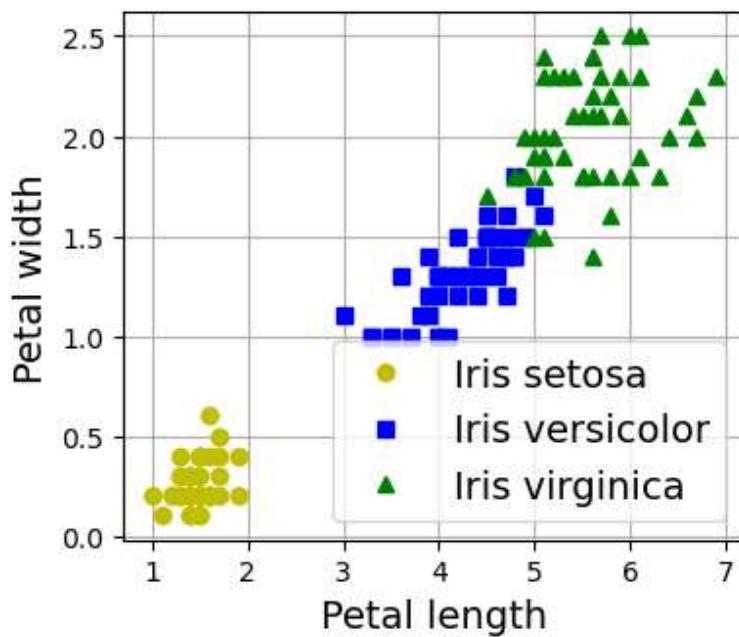
## 군집과 군집화

- 군집  $\text{cluster}$ : 유사한 대상들의 모음
  - 산이나 공원에서 만나는 이름 모르는 동일 품종의 꽃으로 이뤄진 군집
  - 신발/의류/스마트폰 쇼핑몰 고객들의 군집
- 군집화  $\text{clustering}$ : 특정 기준으로 여러 개의 군집으로 나누는 과정

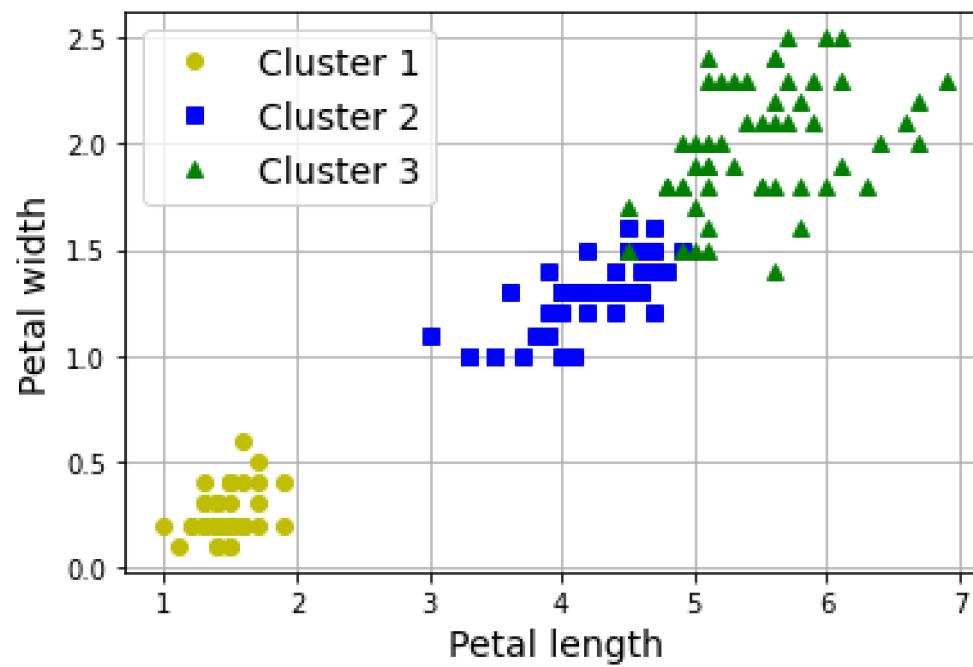
## 분류 대 군집화

- 유사점: 각 샘플에 하나의 그룹 할당
- 차이점
  - 분류: 미리 지정된 레이블(타깃)을 최대한 정확하게 예측하는 과정
  - 군집화: 미리 지정된 레이블(타깃)이 없음에도 불구하고 예측기 스스로 찾아낸 특정 기준을 이용해서 여러 개의 군집으로 나누는 과정
- 군집화에 사용되는 알고리즘에 따라 군집의 특성과 모양 다름
  - k-평균: 센트로이드(중심)라는 특정 샘플을 중심으로 모인 샘플들의 집합
  - DBSCAN: 밀집된 샘플들의 연속으로 이루어진 집합
  - 가우스 혼합: 특정 가우스 분포를 따르는 샘플들의 집합

- 예제: 붓꽃 데이터셋 분류 대 군집화



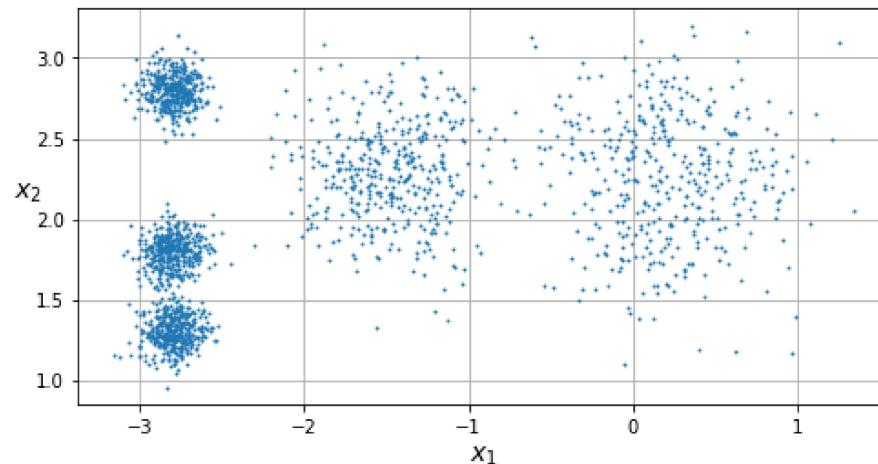
- 가우스 혼합 모델을 적용하면 매우 정확한 군집화 가능. 단, 꽃잎의 너비/길이, 꽃받침의 너비/길이 모두 특성으로 사용해야 함.



## 9.2. k-평균

- 군집의 중심인 센트로이드centroid 몇 개를 찾은 다음 각 센트로이드에 가깝게 위치한 샘플들로 구성된 군집을 형성하는 기법
- 군집 개수 먼저 지정 필요

## 예제: 샘플 덩어리 다섯 개로 이루어진 데이터셋



```
from sklearn.cluster import KMeans  
  
k = 5  
kmeans = KMeans(n_clusters=k, random_state=42)  
y_pred = kmeans.fit_predict(X)
```

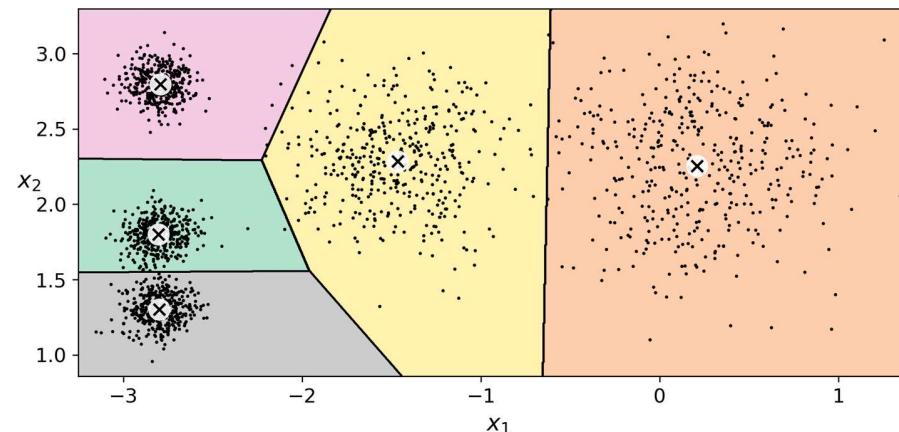
## 예측값

- `predict()` 함수의 반환값: 임의로 지정된 군집의 인덱스를 가리키는 0, 1, 2, 3, 4 등  
의 정수 사용

```
>>> y_pred  
array([4, 0, 1, ..., 2, 1, 0])
```

## 보로노이 다이어그램

- 평면을 특정 점까지의 거리가 가장 가까운 점들의 집합으로 분할한 그림
- 군집이 잘 구성되었는지 쉽게 확인 가능
- 여기서는 경계 부분 일부 데이터 샘플이 잘못 구분됨. 중앙에 위치한 군집의 직경이 상대적으로 너무 커서 경계에 위치한 일부 데이터가 다른 센트로이드에 거리상 보다 가깝기 때문임.



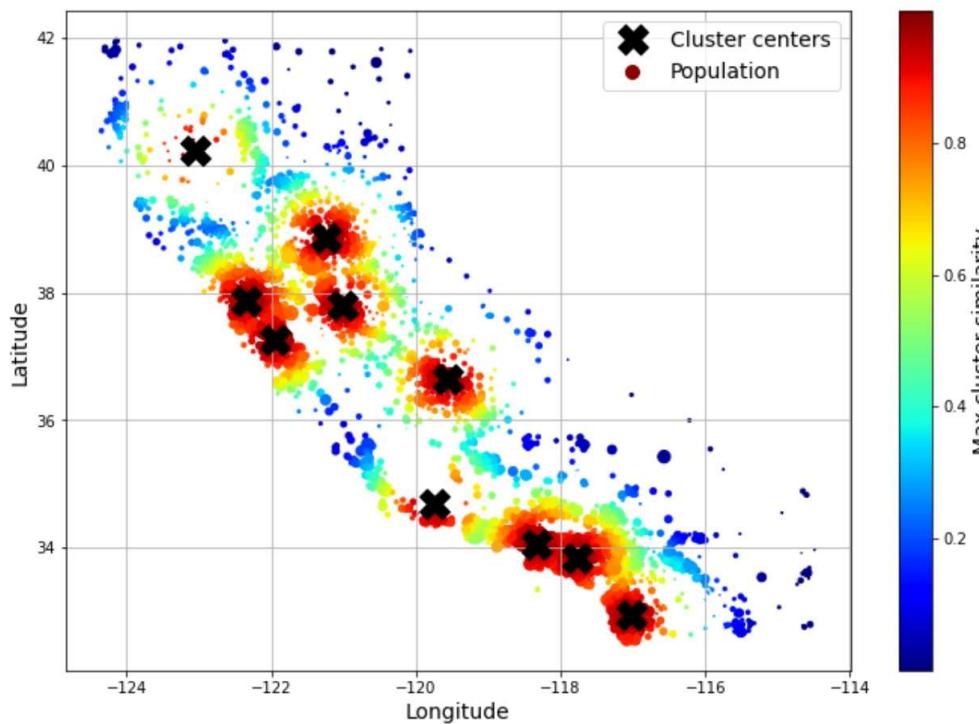
## 하드 군집화 대 소프트 군집화

- 하드 군집화: 각 샘플에 대해 가장 가까운 군집 선택
- 소프트 군집화: 샘플과 각 군집 사이의 관계를 점수로 부여
  - 예제: 각 군집과 샘플사이의 거리
  - 예제: 가우스 방사기저 함수를 이용한 유사도 점수
- k-평균 모델의 transform() 메서드: 샘플과 각 센트로이드 사이의 (유클리드) 거리를 점수로 계산

```
>>> kmeans.transform(X_new).round(2)
array([[2.81, 0.33, 2.9 , 1.49, 2.89],
       [5.81, 2.8 , 5.85, 4.48, 5.84],
       [1.21, 3.29, 0.29, 1.69, 1.71],
       [0.73, 3.22, 0.36, 1.55, 1.22]])
```

## ClusterSimilarity 모델

- 2장에서 캘리포니아의 구역을 10개의 군집으로 쪼갤 때 사용한 모델
- transform() 메서드: 각 샘플에 대해 10개의 센트로이드와의 유사도 점수 계산
- 계산된 유사도를 새로운 특성으로 추가해서 모델 훈련에 사용함.



```
class ClusterSimilarity(BaseEstimator, TransformerMixin):
    def __init__(self, n_clusters=10, gamma=1.0, random_state=None):
        self.n_clusters = n_clusters
        self.gamma = gamma
        self.random_state = random_state

    def fit(self, X, y=None, sample_weight=None):
        self.kmeans_ = KMeans(self.n_clusters, n_init=10,
random_state=self.random_state)
        self.kmeans_.fit(X, sample_weight=sample_weight)
        return self

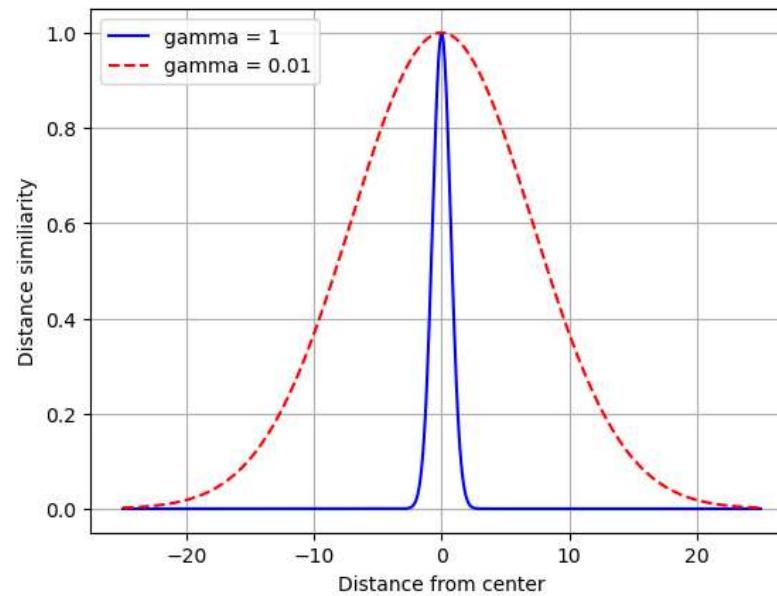
    def transform(self, X):
        return rbf_kernel(X, self.kmeans_.cluster_centers_, gamma=self.gamma)

cluster_simil = ClusterSimilarity(n_clusters=10, gamma=1., random_state=42)

housing_labels = strat_train_set["median_house_value"].copy()
similarities = cluster_simil.fit_transform(housing[["latitude", "longitude"]],
sample_weight=housing_labels)
```

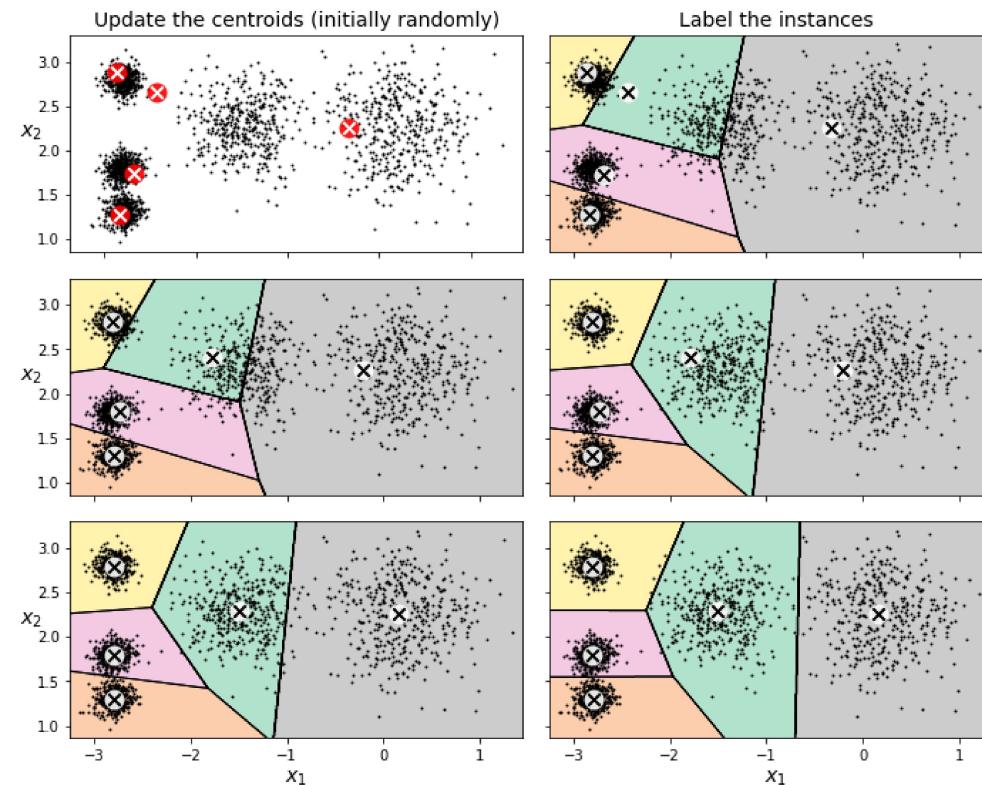
## 가우스 방사 기저 함수

$$\phi(\mathbf{x}, \mathbf{m}) = \exp(-\gamma \|\mathbf{x} - \mathbf{m}\|^2)$$



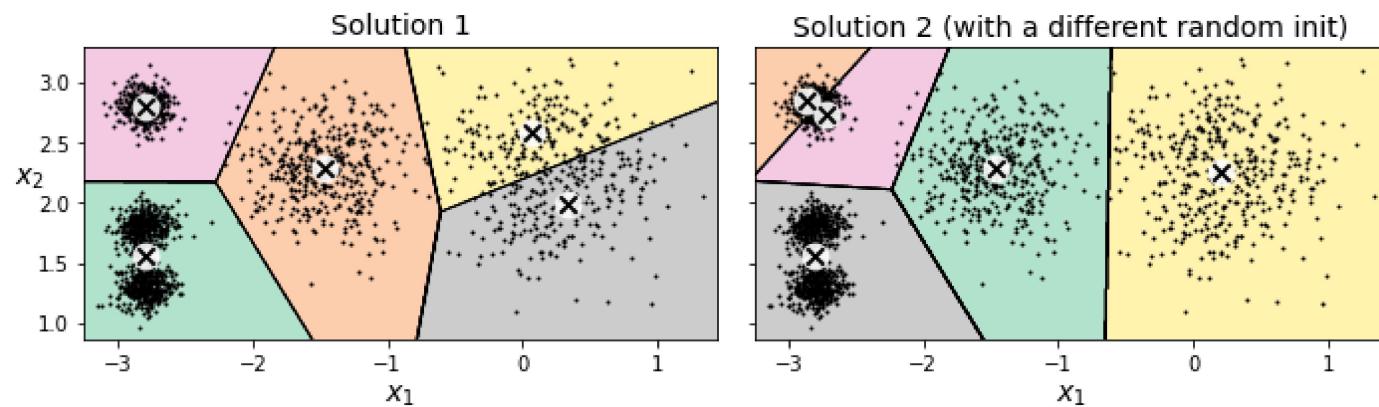
# k-평균 알고리즘

- 먼저 k 개의 센트로이드를 무작위로 선택한 후 수렴할 때까지 다음 과정 반복
  - 각 샘플을 가장 가까운 센트로이드에 할당
  - 군집별로 샘플의 평균을 계산하여 새로운 센트로이드 지정



## 무작위 초기화 문제

- 임의로 선택된 초기 센트로이드에 따라 매우 다른 군집화 발생 가능



## 관성 inertia

- k-평균 모델의 성능 평가 방법
- 샘플과 가장 가까운 센트로이드와의 거리의 제곱의 합
- 각 군집이 센트로이드에 얼마나 가까이 모여있는가를 측정
- KMeans 모델의 `score()` 메서드가 관성의 음수값을 반환. 점수(score)는 높을 수록 좋은 모델이어야 하기 때문.
- 다양한 초기화 과정을 실험한 후에 가장 좋은 것 선택 `n_init = 10`이 기본값으로 사용됨. 즉, 10번 학습 후 가장 낮은 관성을 갖는 모델 선택.

## k-평균++ 초기화 알고리즘

- 센트로이드를 무작위로 초기화하는 대신 특정 확률분포를 이용하여 선택하여 센트로이드들 사이의 거리를 크게 할 가능성이 높아짐.
  - 임의로 하나의 센트로이드  $c_1$  선택 후  $k$  개의 센트로이드를 지정할 때까지 아래 과정을 반복한다.
  - $c_1, \dots, c_{i-1}$ 이 이미 선택되었가고 가정했을 대, 각 샘플  $\mathbf{x}_j$ 가 아래의 확률로 새로운 센트로이드  $c_i$ 로 선택되도록 한다.

$$\frac{D(\mathbf{x}_j)^2}{\sum_{j=1}^m D(\mathbf{x}_j)^2}$$

단,  $m$ 은 훈련셋의 크기를,  $D(\mathbf{x}_j)$ 는  $\mathbf{x}_j$ 와 이미 선택된  $c_1, \dots, c_{i-1}$  중에서 가장 가까운 센트로이드 사이의 거리를 가리킨다.

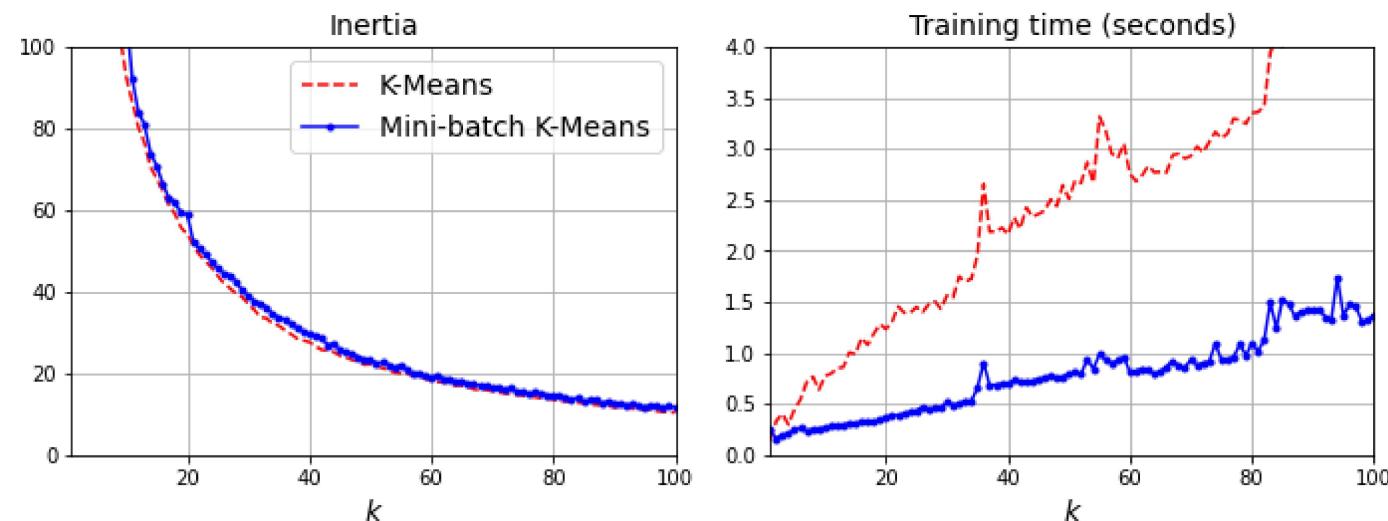
## 미니배치 k-평균

- 미니배치를 사용해서 센트로이드를 조금씩 이동하는 k-평균 알고리즘
- 사이킷런의 `MiniBatchMeans` 모델이 지원.

```
from sklearn.cluster import MiniBatchKMeans  
  
minibatch_kmeans = MiniBatchKMeans(n_clusters=5, random_state=42)  
minibatch_kmeans.fit(X)
```

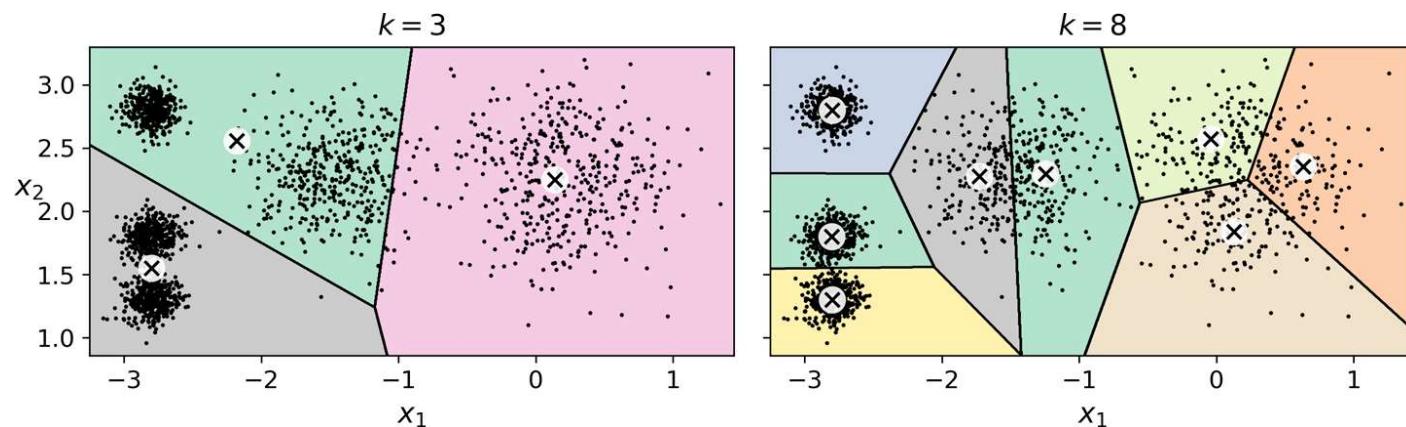
## 미니배치 k-평균의 특징

- 성능은 기본 k-평균 모델보다 조금 낫음.
- 군집수가 커질 수록 k-평균보다 훨씬 빠르게 훈련됨.
- 군집수가 커져도 성능 차이가 유지됨.



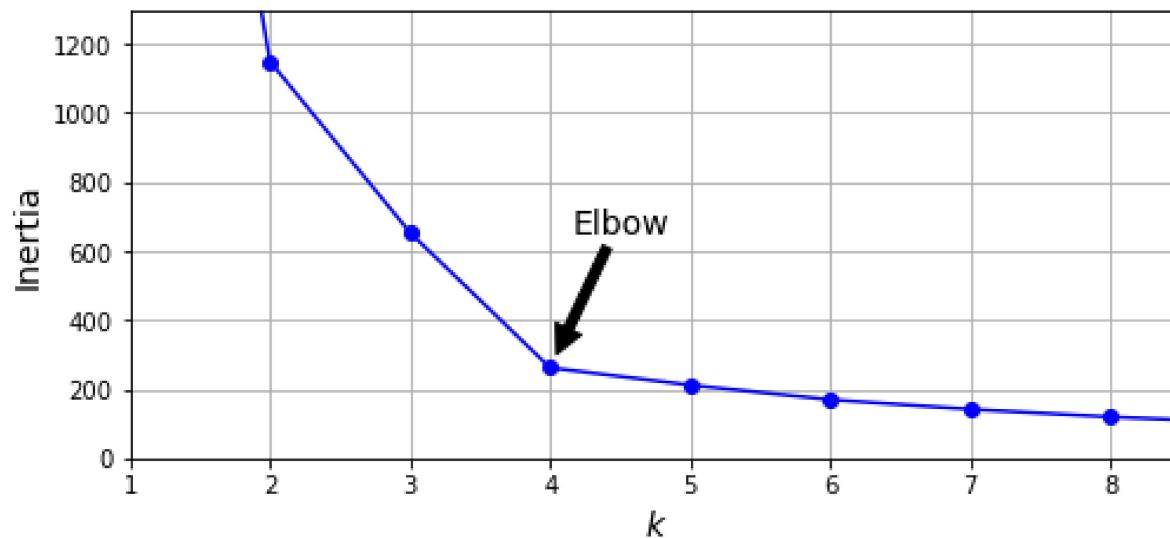
## 최적의 군집수 찾기

- 군집수가 적절하지 않으면 좋지 않은 모델로 수렴할 수 있음.

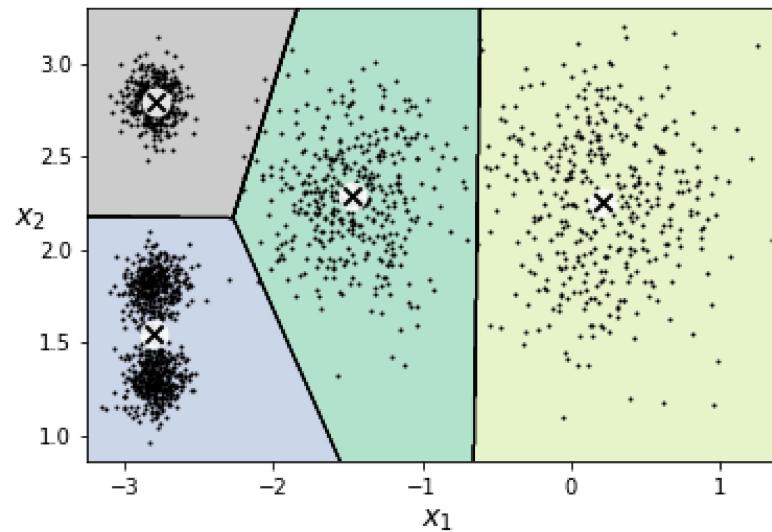


## 방법 1: 관성과 군집수

- 군집수  $k$ 가 증가할 수록 관성은 기본적으로 줄어듬. 따라서 관성만으로 모델을 평가 할 없음.
- 관성이 더 이상 획기적으로 줄어들지 않는 지점을 선택할 수 있음.



- 하지만 아래 그림에서 보듯이 반드시 좋은 모델이라 평가하기 어려움.



## 방법 2: 실루엣 점수와 군집수

- 샘플별 실루엣 계수

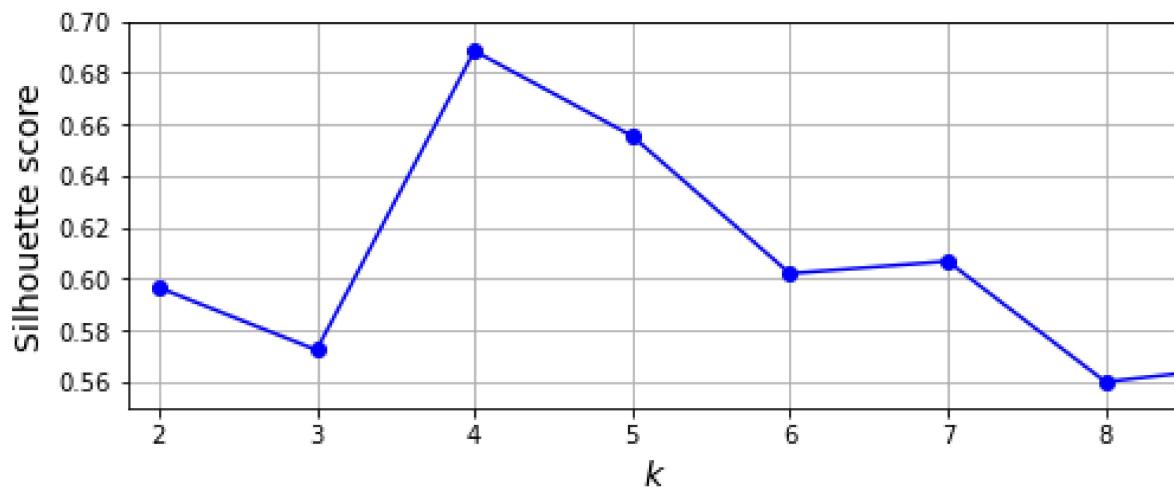
$$\frac{b - a}{\max(a, b)}$$

- $a$ : 동일 군집 내의 다른 샘플과의 거리의 평균값
- $b$ : 가장 가까운 타 군집 샘플과의 거리의 평균값

- 실루엣 계수는 -1과 1사이의 값임.

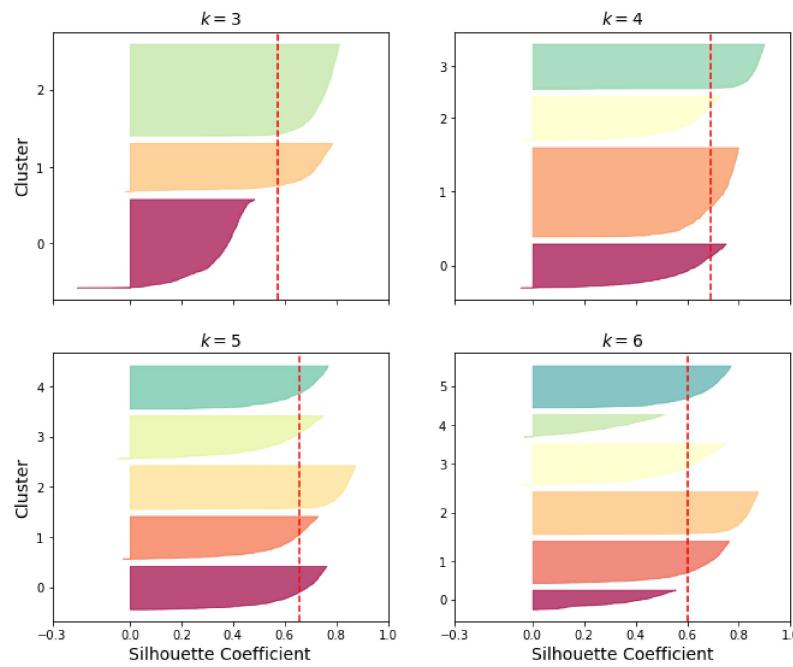
- 1에 가까운 값: 적절한 군집에 포함됨. 이유는  $b$ 가  $a$ 보다 월등이 크기에  $\frac{b-a}{\max(a,b)}$  가  $\frac{b}{b}$ 에 가까워짐.
- 0에 가까운 값: 군집 경계에 위치. 이유는  $a \simeq b$  이기에  $\frac{b-a}{\max(a,b)}$  가 0에 가까워짐.
- 1에 가까운 값: 잘못된 군집에 포함됨. 이유는  $a$ 가  $b$ 보다 월등이 크기에  $\frac{b-a}{\max(a,b)}$  가  $\frac{-a}{a}$ 에 가까워짐.

- 실루엣 점수: 실루엣 계수의 평균값.
- 실루엣 점수가 높은 모델을 선택할 수 있음. 아래 그림에 의해  $k=5$  도 좋은 선택이 될 수 있지만 확실하지는 않음.



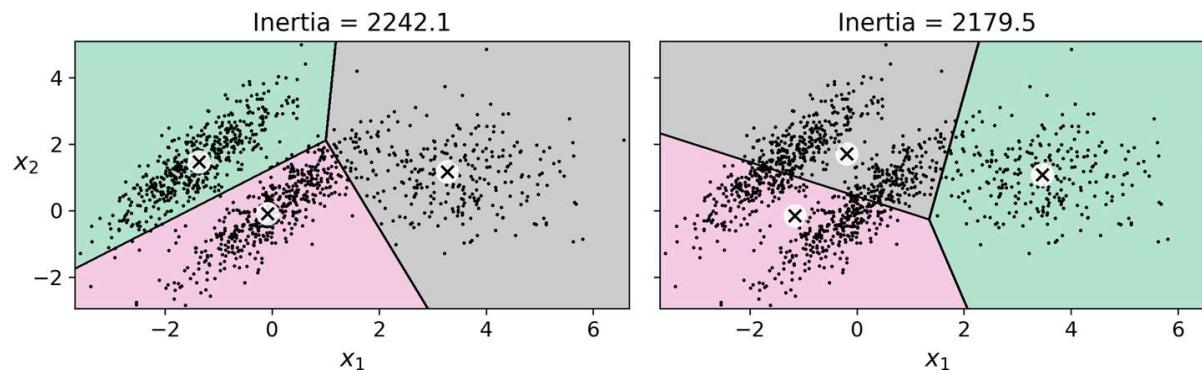
## 방법 3: 실루엣 다이어그램과 군집수

- 실루엣 다이어그램: 군집별 실루엣 계수들의 모음.
- 군집별로 칼날 모양 형성.
- 칼날의 두께가 서로 비슷해야, 즉, 군집별 크기가 비슷해야 좋은 모델임.
- $k=5$  가 보다 좋은 모델임.



## k-평균의 한계

- 최적의 모델을 구하기 위해 여러 번 학습해야 함.
- 군집수를 미리 지정해야 함.
- 군집의 크기나 밀집도가 다르거나, 원형이 아닐 경우 잘 작동하지 않음 (아래 그림 참고)



### 9.3. 군집화 활용

## 이미지 분할

이미지 분할은 보통 다음 세 가지 중에 하나를 가리킨다.

- 시맨틱 분할
- 인스턴스 분할
- 색상 분할

# 시맨틱 분할과 인스턴스 분할

- 시맨틱 분할 semantic segmentation: 사진에 들어 있는 사물들을 클래스별로 분할
- 인스턴스 분할 instance segmentation: 클래스 뿐만 아니라 객체 끼리도 서로 분할



## 이미지 색상 분할

- 색상 분할: 유사 색상으로 이루어진 군집으로 분할.
- 예제: 무당벌레 이미지 색상 분할



## 준지도 학습

- 레이블이 있는 데이터가 적고, 레이블이 없는 데이터가 많을 때 활용
- 예제: 미니 MNist
  - 미니 MNist 데이터셋: 1,797 개의 8x8 크기의 손글씨 이미지로 구성됨
  - 예를 들어, 50개의 군집으로 나눈 후 50개 군집별로 센트로이드에 가장 가까운 샘플을 **대표 이미지**로 선정.
  - 선정된 50개 샘플만을 이용하여 훈련해도 84.9% 정도의 정확도가 달성됨.

## 레이블 전파

- 대표 이미지의 레이블을 해당 군집의 모든 샘플로 전파하는 기법
- 미니 MNIST 데이터셋의 50개 군집의 대표 이미지의 레이블을 각 군집의 전체 샘플에 전파한 다음에 전체 훈련셋을 대상으로 분류 모델을 훈련하면 정확도가 89% 이상으로 좋아짐.
- 센트로이드에 가장 멀리 떨어진 1%의 데이터를 이상치로 취급하여 각 군집에서 제외시킨 다음에 레이블 전파된 훈련셋을 이용하면 분류 모델의 성능이 조금 더 향상됨.
- `sklearn.semi_supervised` 패키지는 다양한 레이블 전파 클래스를 제공한다.
  - `LabelSpreading`
  - `LabelPropagation`
  - `SelfTrainingClassifier`

## 준지도학습과 능동학습

- 모델의 성능을 보다 높이기 위한 다음 단계로 **능동 학습** Active Learning 기법을 적용 가능
- **불확실성 샘플링** uncertainty sampling 전략 추천하며, 훈련 성능이 더 이상 개선되지 않을 때까지 아래 과정 반복.
  1. 기존에 정리된 훈련셋을 이용하여 모델을 학습시킨다.
  2. 훈련된 모델이 예측에 대해 가장 불확실해 하는 샘플들을 대상으로 사람이 직접 라벨을 확인한다.