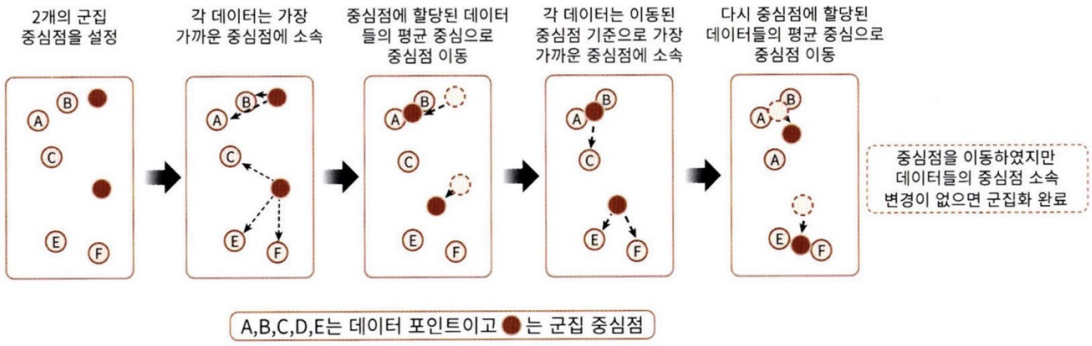
CH7

K-평균 알고리즘

- 군집화 알고리즘
- 군집 중심점(centroid)이라는 특정한 임의의 지점을 선택해, 해당 중심에 가장 가까운 포인트를 선택하는 군집화 기법



[장점]

- 알고리즘 쉽고 간결

[단점]

- 거리 기반 알고리즘 → 속성 많으면 정확도 감소
- 반복 횟수 많을 수록 시간 느려짐
- 몇 개의 군집을 선택할지 가이드 어려움

사이킷런 KMeans 클래스

- n_clusters : 군집화할 개수, 군집중심점 개수
- init : 초기 군집 중심점 좌표 설정 방식 → 일반적으로 k-means++
- max_iter : 최대 반복 횟수
- labels_ : 각 데이터 포인트가 속한 군집 중심점 레이블
- cluster_centers_ : 각 군집 중심점 좌표

```
kmeans = KMeans(n_clusters=3, init='k-means++', max_iter=300,random_state=0)
kmeans.fit(irisDF)
```

** cluster_std가 작을수록 군집 중심에 데이터가 모여 있음

** 평균 거리 중심으로 중심 이동하면서 군집화 수행하므로 원형으로 흩어진 데이터에 효과적

군집 평가

- 군집화는 데이터 내 숨어있는 별도의 그룹을 찾거나 동일한 분류 내에서 더 세분화된 군집화 추구 혹은 다른 분류값 데이터를 군집화 레벨화 하는 등 분류와 다름

1) 실루엣 분석

- 각 군집 간 거리가 얼마나 효율적으로 분리되어 있는지 나타냄
  → 동일 군집끼리 가깝게, 다른 군집과는 멀리
- **실루엣 계수** 기반으로 분석
  : 개별 데이터가 가지는 실루엣 계수는 해당 데이터가 같은 군집 내의 데이터와 얼마나 가깝게 군집화돼 있고, 다른 군집에 있는 데이터와는 얼마나 멀리 분리되어 있는지를 나타냄

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

$$s(i) = \begin{cases} 1 - a(i)/b(i), & \text{if } a(i) < b(i) \\ 0, & \text{if } a(i) = b(i) \\ b(i)/a(i) - 1, & \text{if } a(i) > b(i) \end{cases}$$

- a(i) : 해당 데이터 포인트와 같은 군집 내의 데이터 포인트까지 평균 거리
- b(i) : 해당 데이터 포인트가 속하지 않은 군집 중 가장 가까운 군집과의 평균 거리
- 실루엣 계수는 -1~1 사이 값, 1에 가까울수록 근처 군집과 멀리 떨어져 있다는 뜻 (0은 반대), 음수는 아예 다른 군집에 포인트가 할당됨

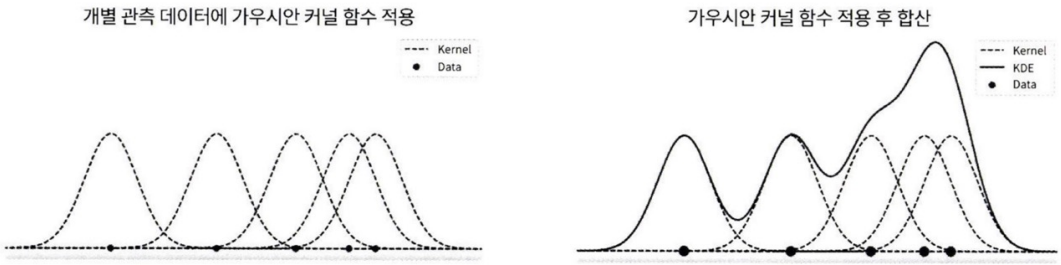
```
from sklearn.metrics import silhouette_samples, silhouette_score
score_samples = silhouette_samples(iris.data, irisDF['cluster'])
average_score = silhouette_score(iris.data, irisDF['cluster'])
```

** silhouette_score 1에 가까울 수록 좋음!

** 다만 전체 실루엣 계수 평균값은 높지만, 특정 군집의 계수만 높은 것이라면 좋은 군집화 조건 아님! → 시각화를 통해 확인하자

평균 이동(Mean Shift)

- k평균과 유사하나 중심을 데이터가 모여있는 밀도 가장 높은 곳으로 이동
- 데이터 분포도 이용 → pdf
- 거리값을 KDE(Kernel Density Estimation) 이용해서 업데이트
  → 개별 관측 데이터에 커널 함수 적용 후 적용 값을 모두 더해 개별 관측 데이터 건수로 나눠 확률 밀도 함수를 추정하며, 커널함수는 대표적으로 가우시안 사용



KDE는 다음과 같은 커널 함수식으로 표현됩니다. 다음 식에서 K는 커널 함수, x는 확률 변숫값, xi는 관측값, h는 대역폭(bandwidth)입니다.

$$\mathrm{KDE} = \frac{1}{n}\sum_{i=1}^{n} K_h(x - x_i) = \frac{1}{nh}\sum_{i=1}^{n} K\left(\frac{x - x_i}{h}\right)$$

- h값(대역폭)을 작을수록 좁고 뾰족한 KDE → 변동성 큼, 오버피팅 위험
- h값 매우 크면 평활화 → 단순, 과소적합, 적은 수의 군집 중심점 보유

```
from sklearn.cluster import MeanShift
meanshift= MeanShift(bandwidth=1)
cluster_labels = meanshift.fit_predict(X)

from sklearn.cluster import estimate_bandwidth
bandwidth = estimate_bandwidth(X) # 최적 대역폭 계산
```

[장점]

- 데이터 형태나 모델 가정 X → 유연한 군집화
- 이상치 영향력 적고 군집 개수 정할 필요 없음

[단점]

- 수행 시간 길고 대역폭 크기 영향 큼

⇒ 컴퓨터 비전 잘 사용됨!

GMM (Gaussian Mixture Model)

- 데이터가 여러 개의 가우시안 분포를 가진 데이터 집합들이 섞여서 생성된 것이라는 가정 하에 군집화를 수행하는 방식
- 여러 개의 정규 분포 곡선을 추출하고 개별 데이터가 이 중 어떤 정규 분포에 속하는지 결정한다.

⇒ **모수 추정** : 아래의 두 가지를 추정

1) 개별 정규 분포의 평균과 분산

2) 각 데이터가 어떤 정규 분포에 해당되는지의 확률

```
from sklearn.mixture import GaussianMixture
gmm = GaussianMixture(n_components=3, random_state=0).fit(iris.data)
gmm_cluster_labels = gmm.predict(iris.data)
```

- n_components : 모델의 총 개수 → 중요 파라미터!!

DBSCAN (Density Based Spatial Clustering Applications with Noise)

- 밀도 차이 기반 대표 군집화
- 간단한 알고리즘이지만 기하학적으로 복잡한 데이터셋에도 효과적

[중요 파라미터]

1) epsilon : 입실론 반경을 가지는 원형 영역 → 보통 1이하로

2) minpoints : 입실론 주변 영역에 포함되는 타 데이터 개수

- 특정 핵심 포인트에서 직접 접근이 가능한 다른 핵심 포인트를 연결하면서 군집화 구성

[데이터 포인트 종류]

1) 핵심 포인트 : 주변 영역 내 최소 데이터 개수 이상 가지고 있을 때

2) 이웃 포인트 : 주변 영역 내 위치한 타 데이터

3) 경계 포인트 : 주변 영역 내 최소 데이터 개수 이하 + 핵심 포인트를 이웃 포인트로 가질 때

4) 잡음 포인트 : 최소 데이터 개수 이하 +핵심 포인트도 이웃 포인트로 가지고 있지 않을 때

```
from sklearn.cluster import DBSCAN
dbscan = DBSCAN(eps=0.6, min_samples=8, metric='euclidean')
dbscan_labels = dbscan.fit_predict(iris.data)
```

** 군집 레이블이 -1 : 노이즈에 속하는 군집임을 의미

** 군집 개수 지정 무의미함

** eps 커지면 반경 커져 노이즈 개수 감소, min_samples 커지면 노이즈 개수 증가