19기 정규세션

ToBig's 18기 강의자
강효은

Clustering

군집화

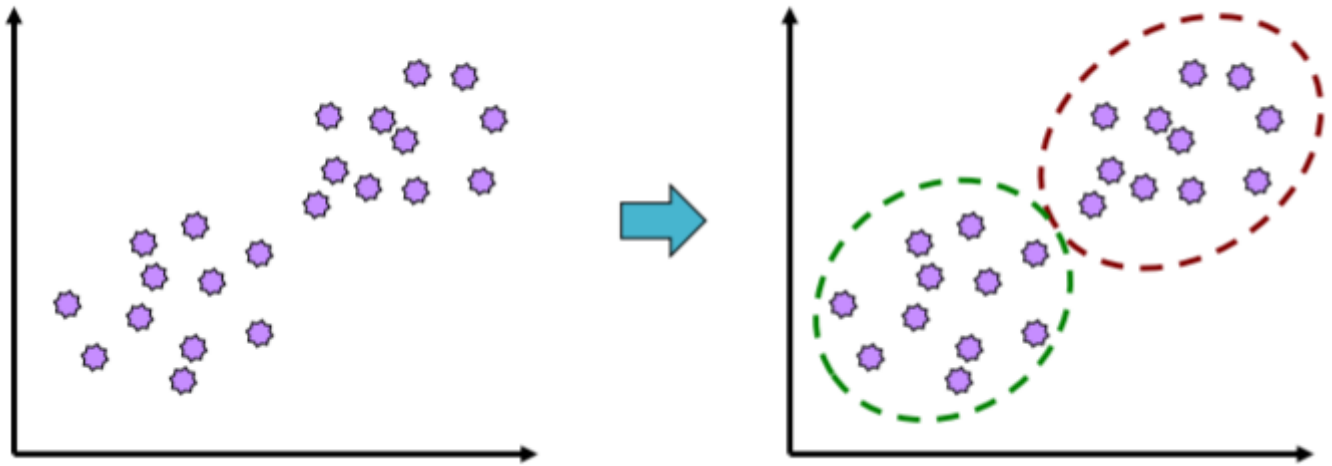
Contents

Unit 01 | Clustering

Clustering(군집화)

- 비지도 학습의 경우 일반적으로 데이터를 적절하게 scaling 해야함 !
- Categorical의 경우 One-Hot Encoding 실행

- 유사한 속성을 갖는 데이터를 묶어 몇 개의 군집(그룹)으로 나누는 것

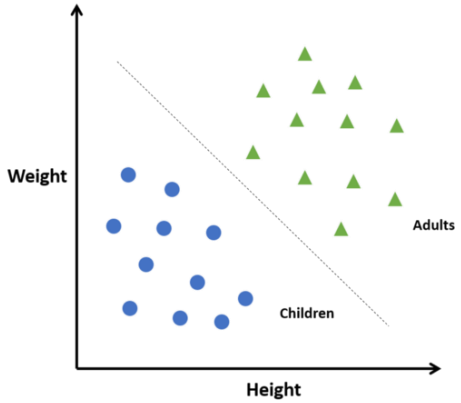


Unit 01 | Clustering

Classification VS Clustering

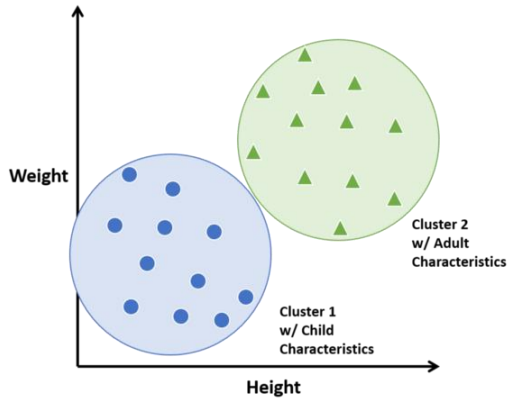
Classification (Supervised)

- 사전 정의된 범주가 있는 (labeled) 데이터로부터 비슷한 집단으로 묶는 방법
- 지도학습의 한 종류



Clustering (Unsupervised)

- 사전 정의된 범주가 없는 (unlabeled) 데이터로부터 최적의 그룹을 찾아나가는 문제
- 비지도학습의 한 종류

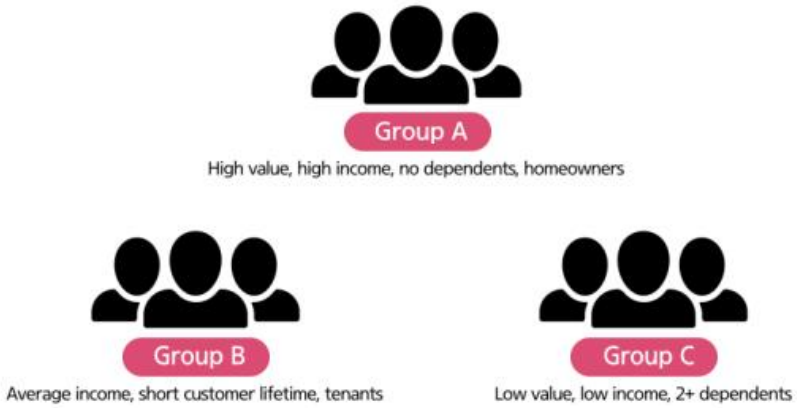


Unit 01 | Clustering

Clustering의 적용

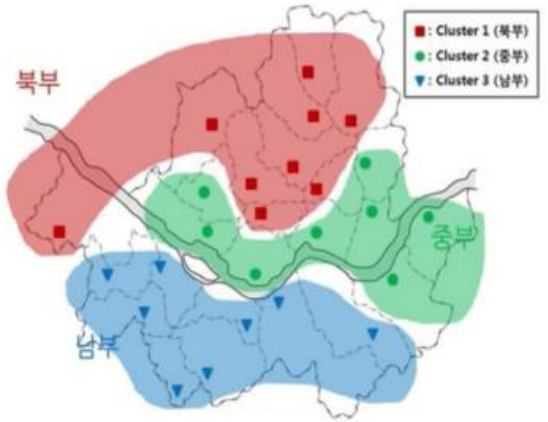
- 데이터들로부터 유의미한 그룹들을 찾음
- 주로 데이터의 경향성을 파악

1. 고객 세분화



2. 유사 문서 군집화

3. 서울시 오존농도 패턴 군집화



Unit 01 | Clustering

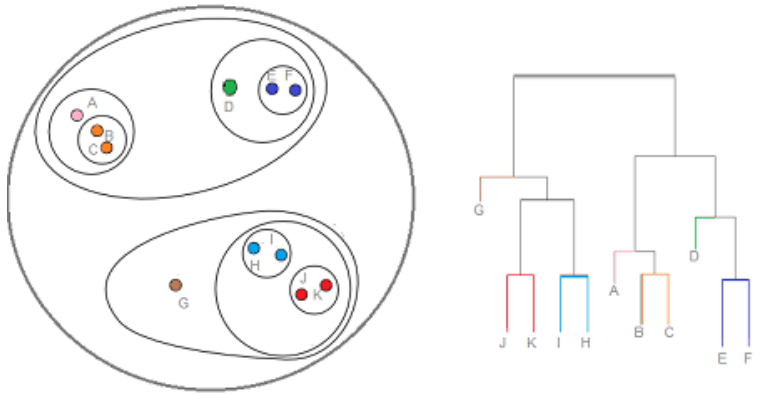
Clustering 종류

어떤 클러스터링 알고리즘을 사용할 것인가?

1 Hierarchical Clustering

- 계층적 군집화
- 개체들을 가까운 집단부터 차근차근 묶어나가는 방식
- 군집화 결과 뿐만 아니라 유사한 개체들이 결합되는 dendrogram 생성

2 Partitioning Clustering



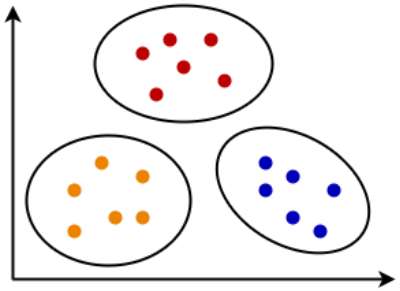
Unit 01 | Clustering

Clustering 종류

어떤 클러스터링 알고리즘을 사용할 것인가?

1 Hierarchical Clustering

2 Partitioning Clustering



- 전체 데이터의 영역을 특정 기준에 의해 동시에 구분
- 각 개체들은 사전에 정의된 개수의 군집 중 하나에 속하게 됨

Contents

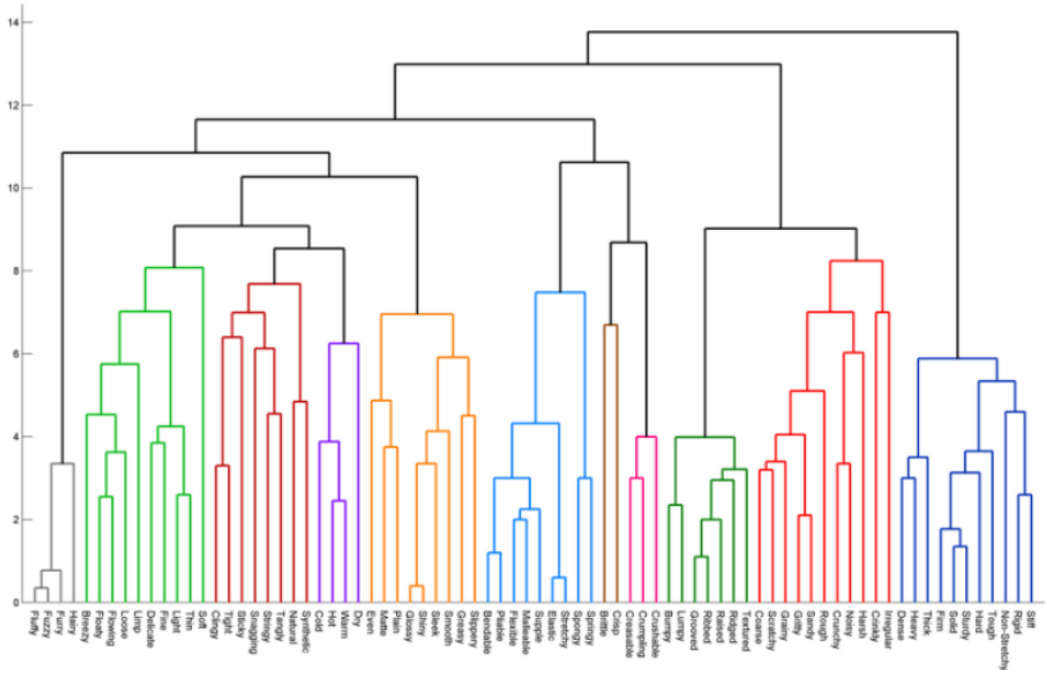
Unit 02 | Hierarchical Clustering

Hierarchical Clustering (계층적 군집화)

- 개체들을 가까운 집단부터 **차근차근** 묶어가는 방식
  - 계층적 트리모형을 이용하여 개별 개체들을 순차적/계층적으로 유사한 개체/군집과 통합한다.

- 클러스터 수를 미리 정해주지 않아도 되는 장점

- **덴드로그램**을 통해 시각화가 가능 !
  - 덴드로그램 : 개체들이 결합되는 순서를 나타내는 트리 형태의 구조



Unit 02 | Hierarchical Clustering

Proximity Matrix (근접 행렬)

: A square matrix in which the entry in cell (j,k) is some measure of the similarity (or distance) between the items to which row j and column k correspond

클러스터들이 있을 때,

두 클러스터가 유사한지
알아보기 위해

	C1	C2	C3	C4	C5
C1					
C2					
C3					
C4					
C5					

근접 행렬

클러스터 간의 거리를 나타낸 행렬

Hierarchical Clustering 수행 예시

1 모든 개체들 사이의 거리에 대한 유사도 행렬 계산

	A	B	C	D
A		20	7	2
B			10	25
C				3
D				

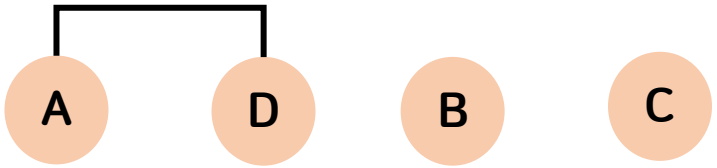
A D B C

Unit 02 | Hierarchical Clustering

Hierarchical Clustering 수행 예시

1. 모든 개체들 사이의 거리에 대한 유사도 행렬 계산
2. 거리가 인접한 관측치끼리 군집 형성

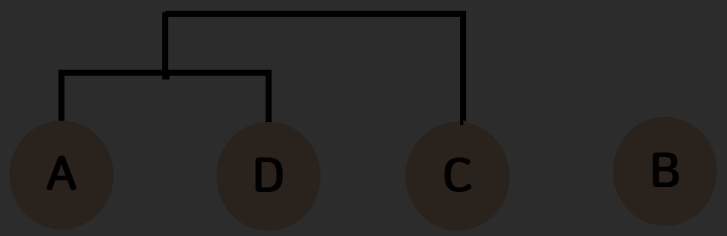
	A	B	C	D
A		20	7	2
B			10	25
C				3
D				



Unit 02 | Hierarchical Clustering

Hierarchical Clustering 수행 예시

1. 모든 개체들 사이의 거리에 대한 유사도 행렬 계산
2. 거리가 인접한 관측치끼리 군집 형성
3. 유사도 행렬 업데이트



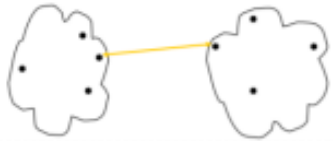
	A,D	B	C	
A,D		20	3	
B			10	
C				

A-D 군집과 나머지 개체 사이의 거리는 어떻게 결정하지 ?!

Unit 02 | Hierarchical Clustering

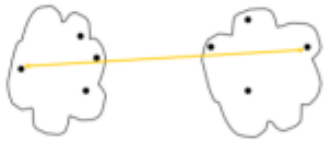
군집 간 거리 측정 두 군집 사이의 유사성 / 거리 측정

1. MIN (Single link)



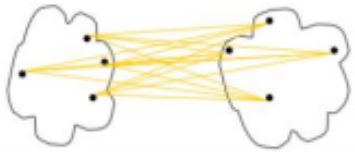
$$d(A,B) = \min_{x \in A, y \in B} d(x-y)$$

2. MAX (Complete link)



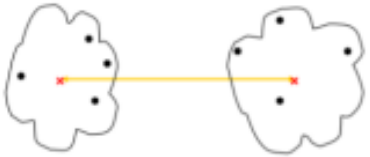
$$d(A,B) = \max_{x \in A, y \in B} d(x-y)$$

3. Group Average



$$d(A,B) = \sum_{x \in A, y \in B} d(x-y)/n_A n_B$$

4. Centroid

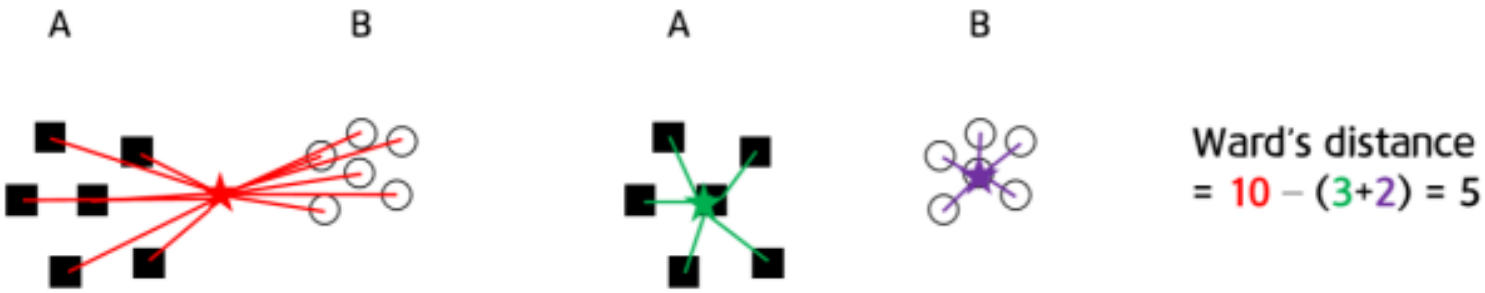


Unit 02 | Hierarchical Clustering

군집 간 거리 측정 Ward's

$$Ward\ Distance = \sum_{i \in A \cup B} \|x_i - m_{A \cup B}\|^2 - \left\{ \sum_{i \in A} \|x_i - m_A\|^2 + \sum_{i \in B} \|x_i - m_B\|^2 \right\}$$

m_A is the center of cluster A.

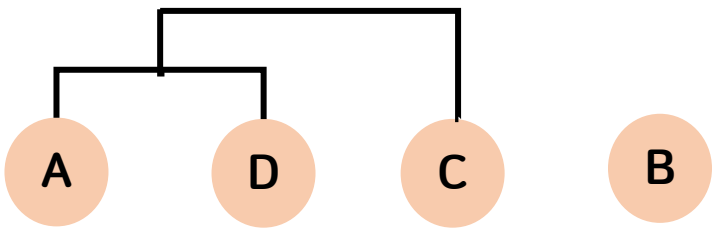


데이터들을 하나의 군집으로 묶음으로써 생기는 정보의 손실을 측정

Unit 02 | Hierarchical Clustering

Hierarchical Clustering 수행 예시

1. 모든 개체들 사이의 거리에 대한 유사도 행렬 계산
2. 거리가 인접한 관측치끼리 군집 형성
3. 유사도 행렬 업데이트

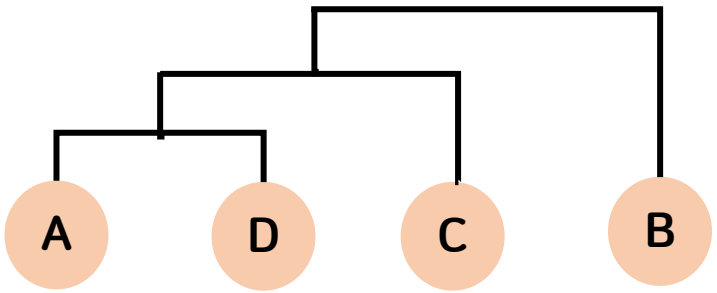


	A,D	B	C	
A,D		20	3	
B			10	
C				

Unit 02 | Hierarchical Clustering

Hierarchical Clustering 수행 예시

1. 모든 개체들 사이의 거리에 대한 유사도 행렬 계산
2. 거리가 인접한 관측치끼리 군집 형성
3. 유사도 행렬 업데이트
4. 위 과정 반복



	A,D,C	B		
A,D,C		10		
B				

Unit 02 | Hierarchical Clustering

Hierarchical Clustering 정리

1) 단일 데이터 간 거리를 정의

Euclidean dist. / Manhattan dist. / …

2) 군집 – 군집 or 군집 – 개체 간 거리 정의

MIN, MAX, Ward's method

3) 반복

Contents

Unit 03 | Partitioning Clustering

K-Means Clustering (K-평균 군집화)

- 대표적인 분리형 군집화 알고리즘
- 각 군집은 하나의 중심을 가진다.
- 사전에 군집의 수 K가 정해져야 알고리즘을 실행

- 각 개체는 가장 가까운 중심에 할당되며,
  같은 중심에 할당된 개체들이 모여 하나의 군집을 형성

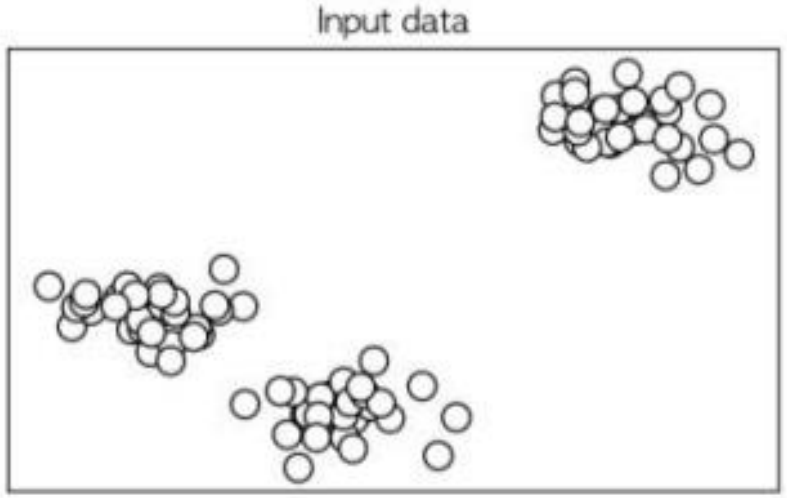
클러스터 간의 중첩X

$$X = C_1 \cup C_2 \cdots \cup C_k, \; C_i \cap C_j = \emptyset, \; i \neq j$$

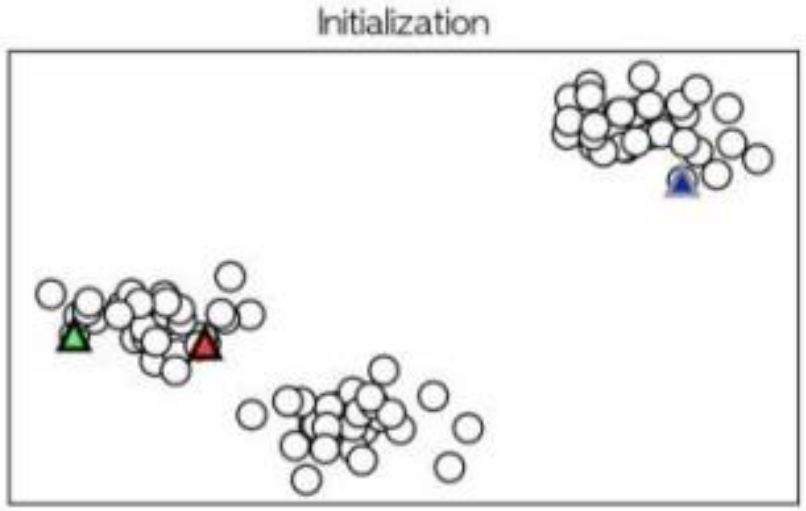
$$\underset{c}{\operatorname{argmin}} \sum_{i=1}^{K} \sum_{x_j \in C_i} \|x_j - c_i\|^2$$

각 클러스터(집합)별 중심점~클러스터 내 점들간 거리의 제곱합을 최소로 하는 집합 C를 찾는 것!

K-Means Clustering 수행 예시

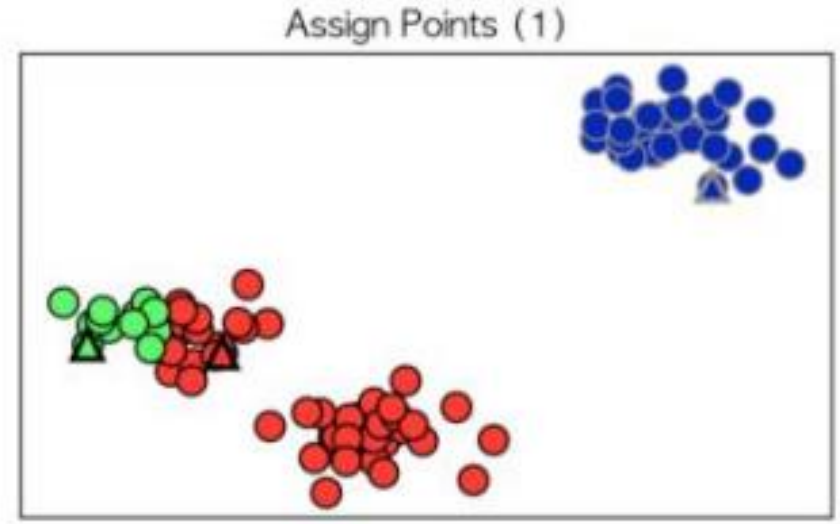


K-Means Clustering 수행 예시



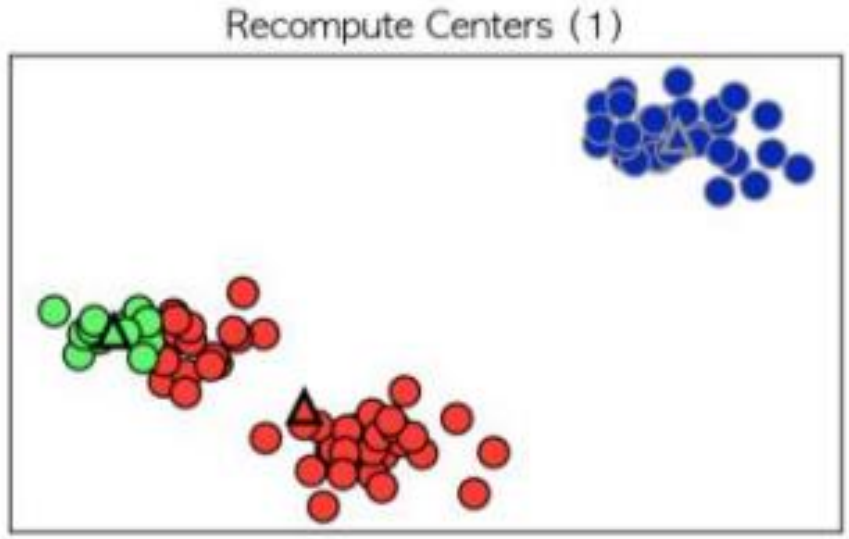
1 초기 중심(centroid)로 할 K개의 데이터를 임의로 선택

K-Means Clustering 수행 예시



2 각 데이터를 가장 가까운 군집 중심(centroid)에 할당

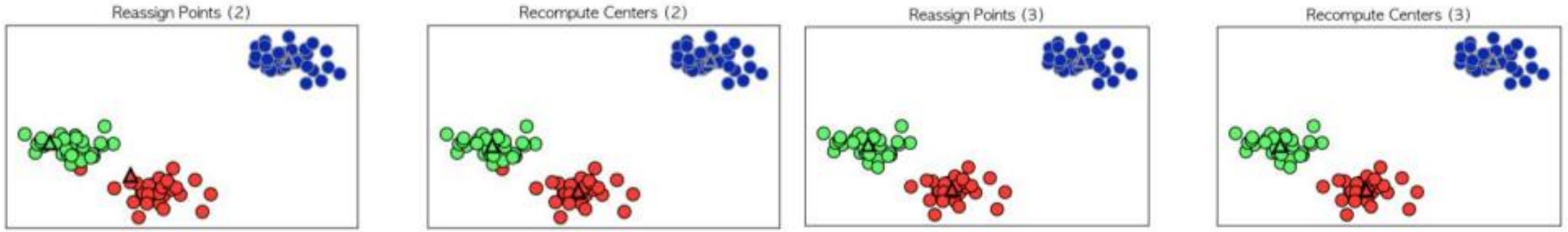
K-Means Clustering 수행 예시



3 각 군집 내의 데이터들의 평균을 계산하여 군집 중심 update

Unit 03 | Partitioning Clustering

K-Means Clustering 수행 예시



4 군집 중심의 변화가 없을 때까지 2, 3 의 과정 반복

K-Means Clustering 수행 절차

1. 초기 중심 K개를 임의로 생성
2. 개별 관측치로부터 각 중심까지의 거리를 계산 후 가장 가까운 중심이 이루는 군집에 관측치 할당
3. 각 군집이 중심을 다시 계산
4. 중심이 변하지 않을 때까지 2,3의 과정을 반복

K-Means Clustering 수행 절차

1. 초기 중심 K개를 임의로 생성

➡

1. K(군집의 수)는 몇 개로 지정하지 ?
2. 초기 중심(centroid)으로 할 데이터를 선택하는 기준은 ?

2. 개별 관측치로부터 각 중심까지의 거리를 계산 후 가장 가까운 중심이 이루는 군집에 관측치 할당
3. 각 군집이 중심을 다시 계산
4. 중심이 변하지 않을 때까지 2,3의 과정을 반복

Unit 03 | Partitioning Clustering

1 초기 중심 K개를 임의로 생성 ➔ **K(군집의 수)는 몇 개로 지정 ?!**

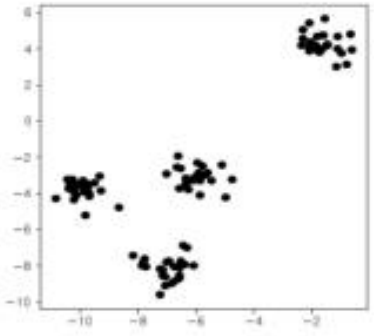
1. Elbow Method

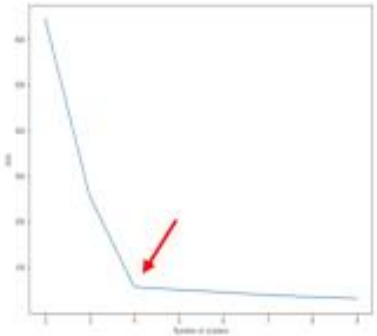
- 군집 내 편차제곱합(WSS)이 최소가 되도록 군집의 중심을 결정해 나가는 방법
- WSS의 총합이 급격하게 감소하기 시작하는 K를 선택
- 그래프가 꺾이는 모양이 팔꿈치 같아 Elbow Method라고 불림

$$WSS = \sum_{x=C}(x - \mu_C)^2$$

x : 데이터

μ_C : 군집의 중심. K-Means의 경우 군집 안 데이터들의 '평균'이 됨.





Unit 03 | Partitioning Clustering

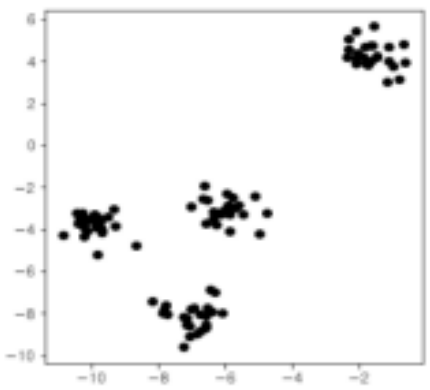
1 초기 중심 K개를 임의로 생성 ➔ **K(군집의 수)는 몇 개로 지정 ?!**

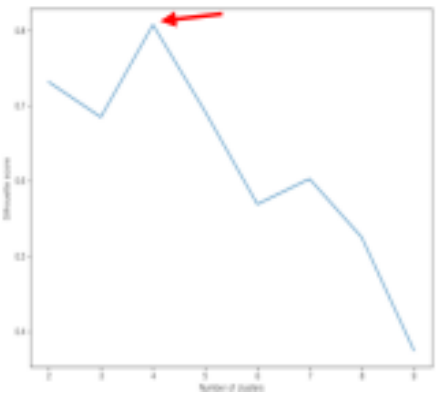
2. Silhouette Coefficient

- 실루엣 계수는 군집 안의 데이터가 자신이 속한 군집 안의 다른 데이터와 얼마나 유사하며, 다른 군집에 속한 데이터와 얼마나 차이가 나는지 측정
- (-1,1)의 값을 가지며 1에 가까울수록 적절한 군집화가 되었다고 판단

a : 데이터 x와 동일한 군집 내의 나머지 데이터들과의 평균 거리 - **군집 내 응집도 (cohesion)**
b : 데이터 x와 가장 가까운 군집 내의 모든 데이터들 간의 평균 거리 - **군집 간 분리도 (separation)**

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





Unit 03 | Partitioning Clustering

1 초기 중심 K개를 임의로 생성 ➔ **초기 중심으로 할 데이터를 선택하는 기준**

많은 경우 초기 중심 설정이 최종 결과에 큰 영향을 미치지는 않음! ➔ 가장 기본적인 방법 : 랜덤초기화

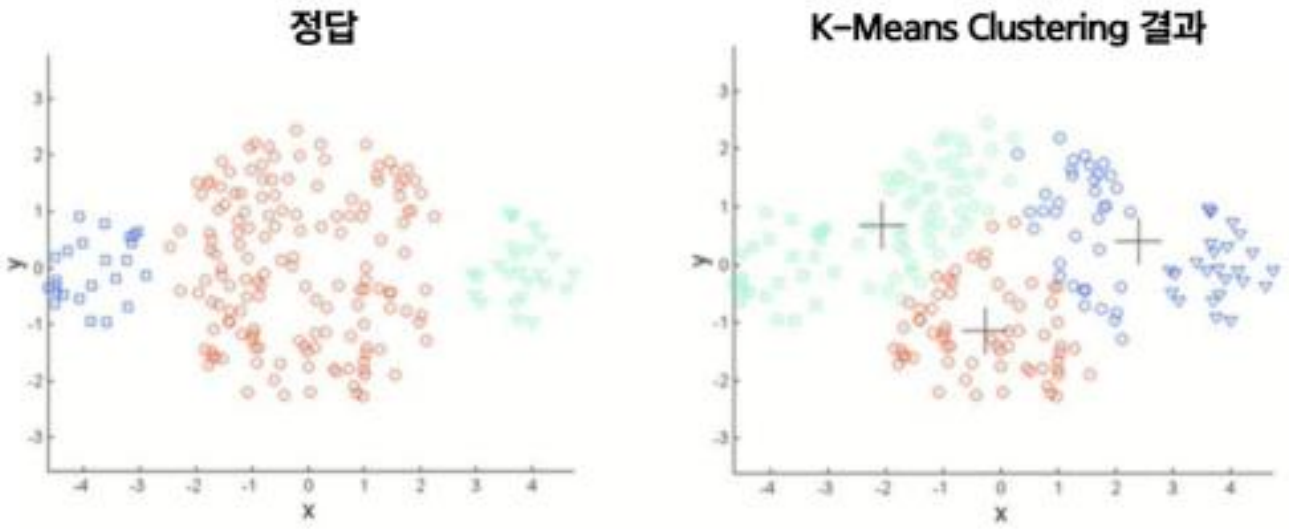
랜덤 초기 중심 설정의 위험을 피하고자 다양한 연구 존재

- 반복적으로 수행하여 가장 여러 번 나타나는 군집 사용
- 전체 데이터 중 일부만 샘플링하여 계층적 군집화를 수행한 뒤 초기 군집 중심 설정
- 데이터 분포의 정보를 사용하여 초기 중심 설정

Unit 03 | Partitioning Clustering

K-Means Clustering 문제점

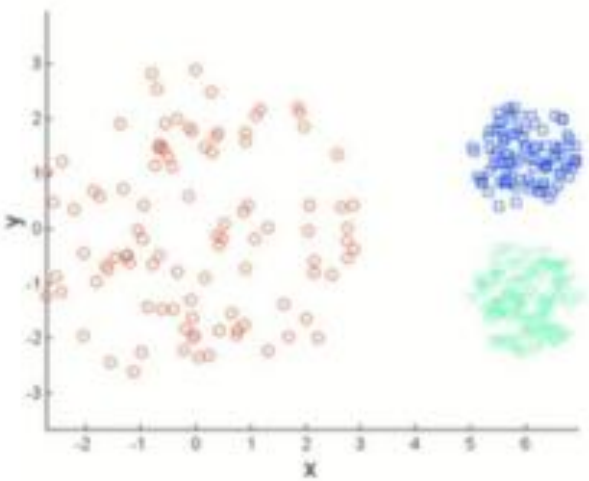
1 서로 다른 크기의 군집을 잘 찾아내지 못한다



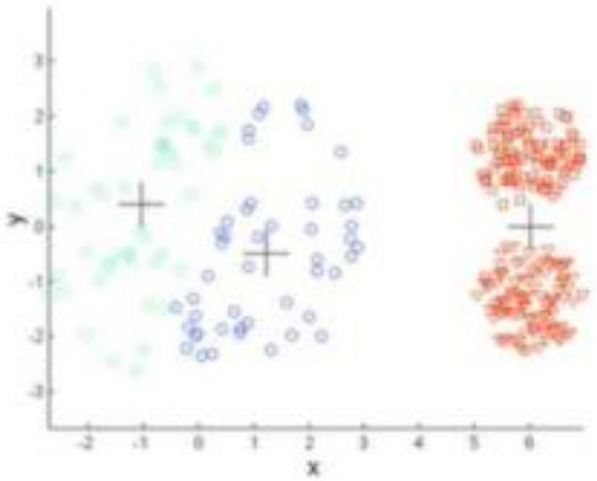
K-Means Clustering 문제점

2 서로 다른 밀도의 군집을 잘 찾아내지 못한다

정답

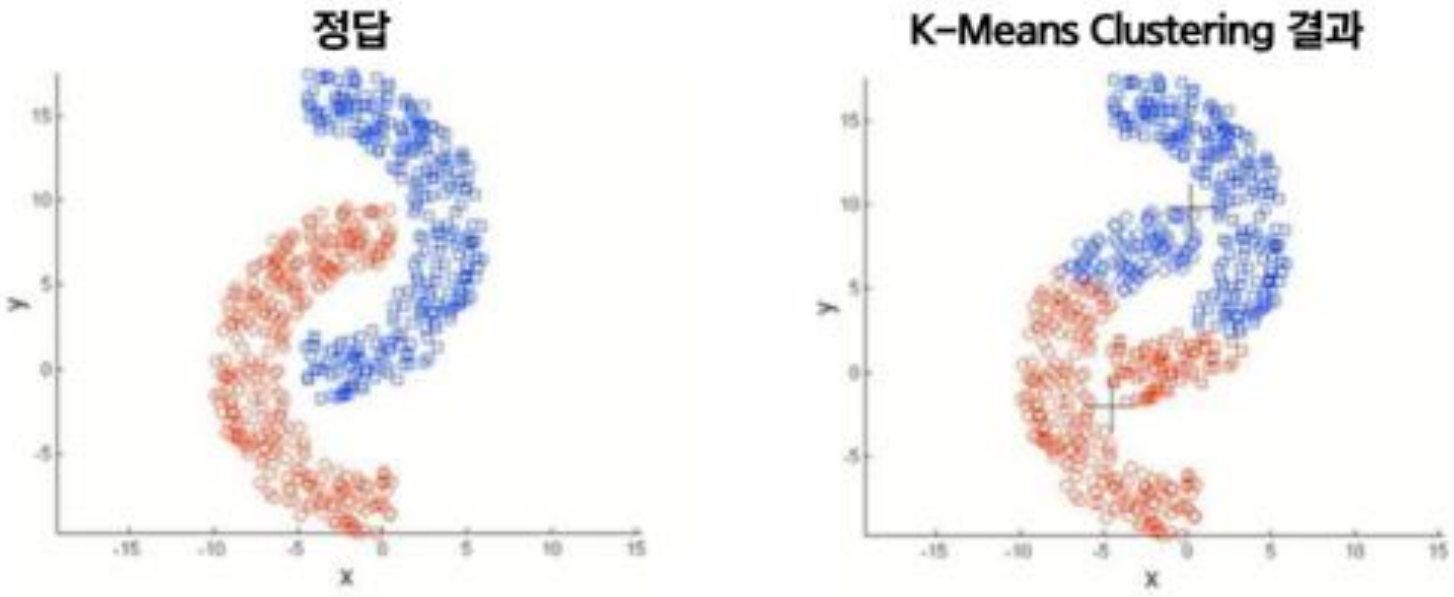


K-Means Clustering 결과



K-Means Clustering 문제점

3 지역적 패턴이 존재하는 군집을 판별하기 어렵다.



Unit 03 | Partitioning Clustering

DBSCAN Density-Based Spatial Clustering of Application with Noise

공간상에 높은 밀도를 가지고 모여있는 관측치들을 하나의 그룹으로 간주하고, 낮은 밀도를 가지고 홀로 있는 관측치는 이상치 또는 잡음으로 분류하는 밀도 기반 군집화 알고리즘

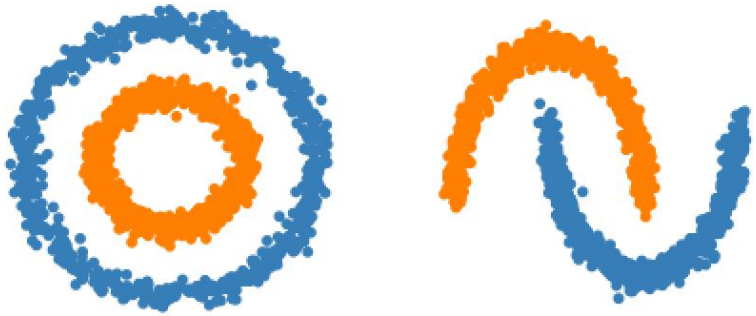
[특징]

- 클러스터의 개수를 미리 지정할 필요 X
- 병합 군집이나 k-평균보다는 다소 느리지만 비교적 큰 데이터셋에도 적용 가능
- 복잡한 형상도 찾을 수 있으며, 어떤 클래스에도 속하지 않는 데이터를 구분 가능
- 클러스터링을 수행하는 동시에 노이즈 데이터도 분류할 수 있기 때문에, 이상치에 의해 클러스터링 성능이 하락하는 현상 완화 가능

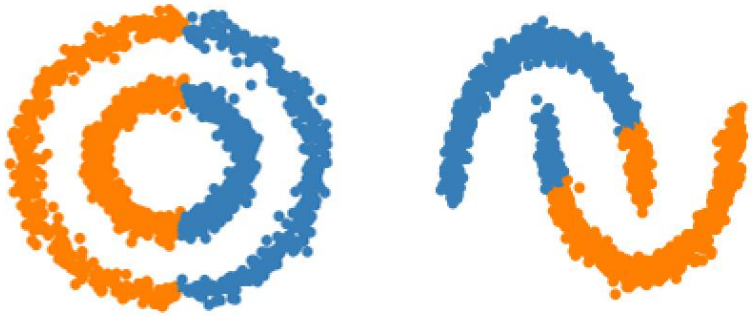
Unit 03 | Partitioning Clustering

DBSCAN Density-Based Spatial Clustering of Application with Noise

DBSCAN



k-means



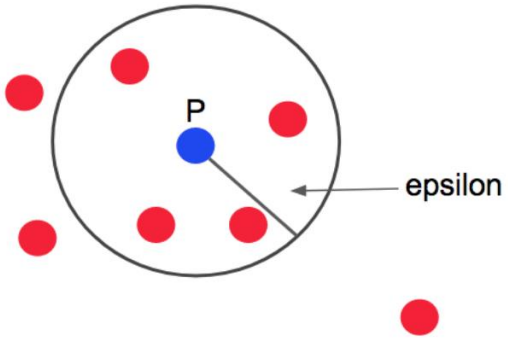
Unit 03 | Partitioning Clustering

DBSCAN Density-Based Spatial Clustering of Application with Noise

[매개변수]

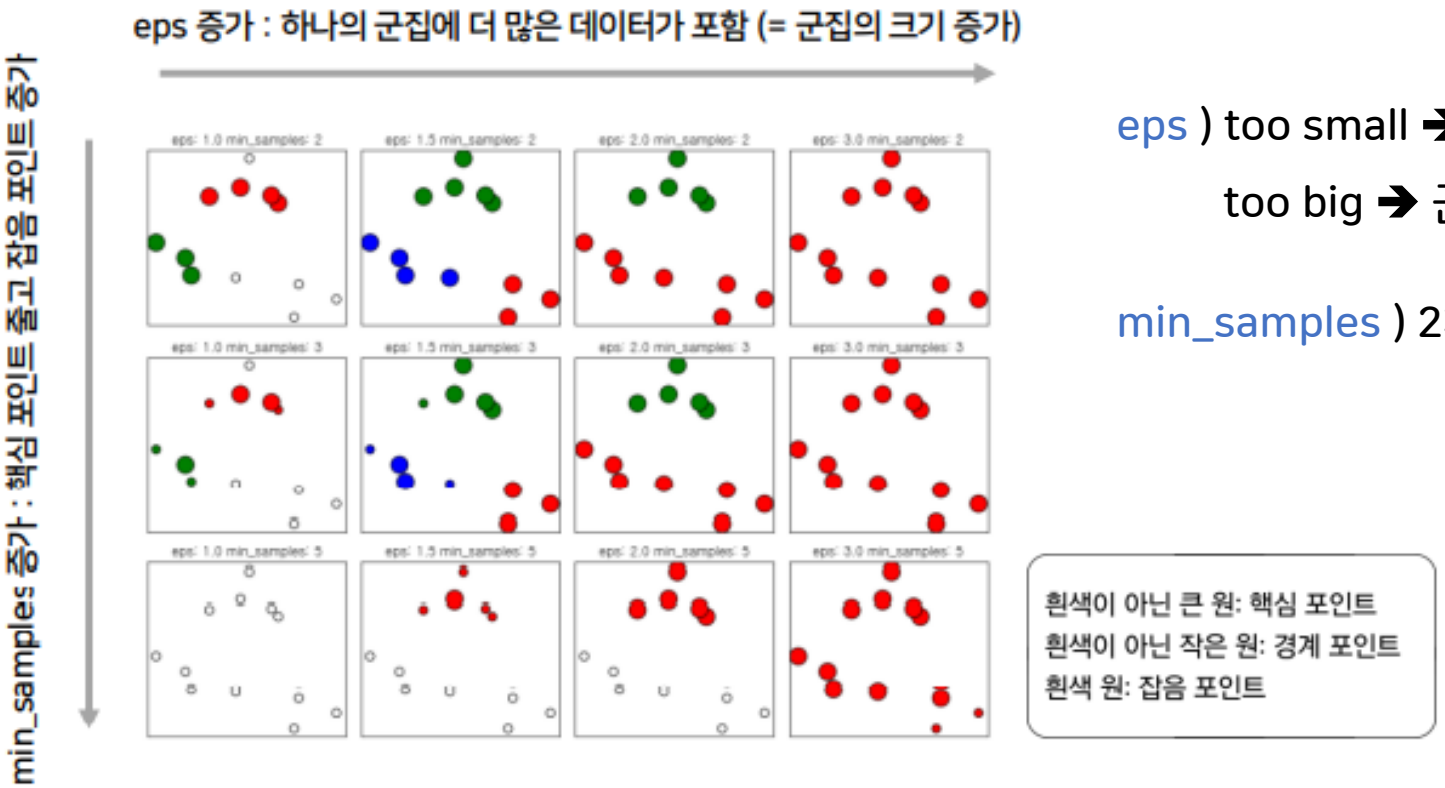
eps(epsilon)	한 데이터가 주변으로부터 얼마만큼 떨어진 거리를 같은 군집이라고 할 것인가? (주변 거리)
min_samples	적어도 한 군집에 몇 개의 데이터가 있어야 군집이라고 할 것인가? (최소 데이터 수)

➔ 거리 eps 내에 데이터가 min_samples개 이상 있으면 하나의 cluster라고 인식



Unit 03 | Partitioning Clustering

DBSCAN Density-Based Spatial Clustering of Application with Noise



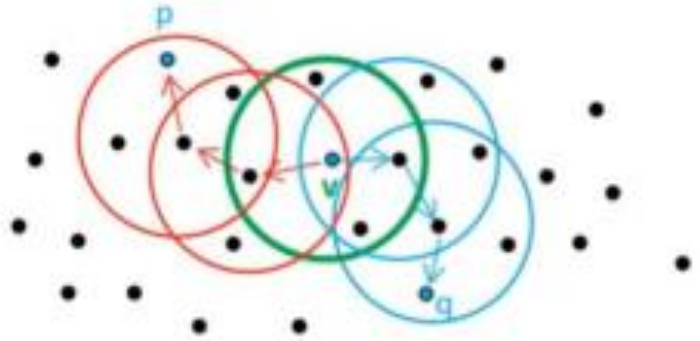
흰색이 아닌 큰 원: 핵심 포인트
흰색이 아닌 작은 원: 경계 포인트
흰색 원: 잡음 포인트

eps) too small ➔ 많은 관측치가 잡음 자료로 분류
too big ➔ 군집의 개수가 너무 작아짐

min_samples) 2차원 데이터의 경우 통상적으로 4를 사용

Unit 03 | Partitioning Clustering

DBSCAN Density-Based Spatial Clustering of Application with Noise



작동 절차

1. 랜덤으로 데이터 포인트로 선택
2. 그 포인트에서 eps 거리 안의 모든 포인트를 찾음
   - 2-1. N(eps 거리안에 있는 데이터) < min_samples ➔ 어떤 클래스에도 속하지 않는 잡음 포인트로 레이블
   - 2-2. N(eps 거리안에 있는 데이터) > min_samples ➔ 핵심 포인트로 레이블하고 새로운 클러스터 레이블 할당
3. 2-2의 핵심포인트의 eps 거리 안의 모든 이웃을 살핌
   - 3-1. 만약 어떤 클러스터에도 아직 할당되지 않았다면 바로 전에 만든 클러스터 레이블 할당
   - 3-2. 만약 핵심 샘플이면 그 포인트의 이웃을 차례로 방문
4. Eps 거리 안에 더 이상 핵심포인트가 없을 때까지 진행

Unit 03 | Partitioning Clustering

DBSCAN Density-Based Spatial Clustering of Application with Noise

한계

1. 사전에 데이터에 대한 충분한 이해도가 없다면 eps와 min_samples를 정하기 어려움
2. 연산량이 많아 K-Means에 비해 속도가 느림
3. 차원의 저주 문제
   - 차원의 수가 낮은 데이터는 문제가 되지 않지만, 고차원 데이터일수록 학습 데이터 양이 급증해 많은 연산 필요
   - 유클리드 거리 사용하는 모든 모델의 공통적인 단점

Contents

Unit 04 | 모델 평가

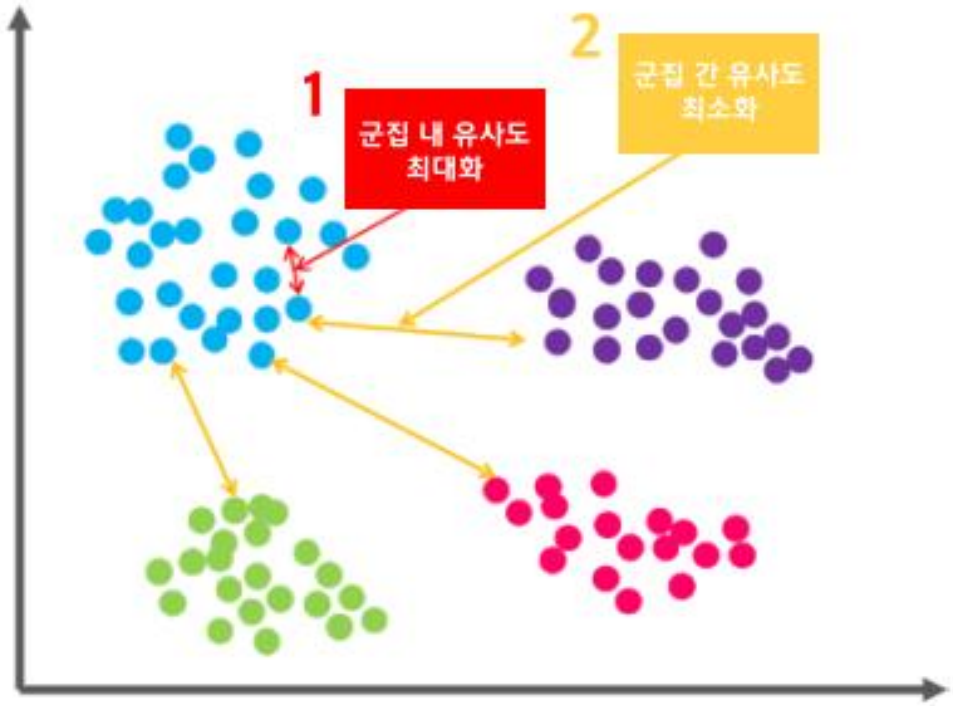
Good Clustering !

1. Maximizes the similarity within a group

: 군집 내 응집도 (cohesion) 최대화

2. Maximizes the difference between groups

: 분리도 (separation) 최대화



Unit 04 | 모델 평가

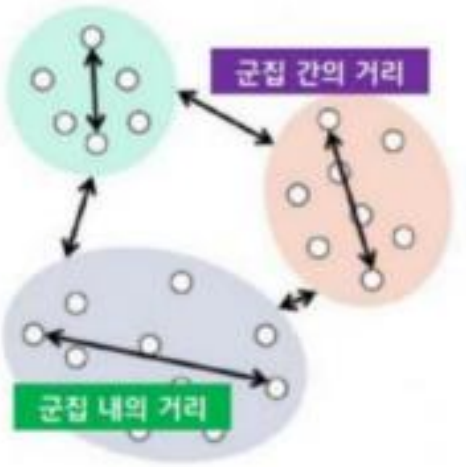
“ 분류 알고리즘 처럼 모든 상황에 적용가능한 평가 지표 부재 “

1. 내부 평가 지표 : Dunn Index, Silhouette, Sum of Squared Error …
2. 외부 평가 지표 : Rank Index, Jaccard Coefficient, …

Unit 04 | 모델 평가

1. Dunn's Index

$$DI = \frac{\text{군집과 군집 사이의 거리 중 최솟값}}{\text{군집 내 객체 간 거리 중 최대값}}$$



군집과 군집 사이의 거리가 클수록, 군집 내 객체가 거리가 작을수록 좋은 모델 ➔ DI가 큰 모델이 좋은 모델

Unit 04 | 모델 평가

2. Silhouette Coefficient

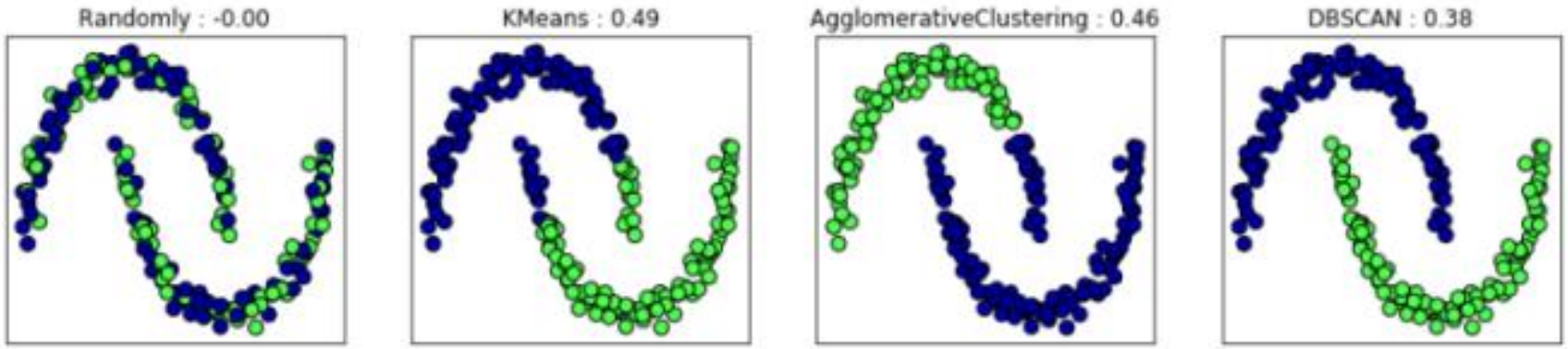
- 실루엣 계수는 군집 안의 데이터가 자신이 속한 군집 안의 다른 데이터와 얼마나 유사하며, 다른 군집에 속한 데이터와 얼마나 차이가 나는지 측정
- [-1,1]의 값을 가지며 1에 가까울수록 적절한 군집화가 되었다고 판단
- 일반적으로 S의 값이 0.5보다 크면 군집 결과가 타당하다고 볼 수 있음

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

a : 데이터 x와 동일한 군집 내의 나머지 데이터들과의 평균 거리 - **군집 내 응집도 (cohesion)**
b : 데이터 x와 가장 가까운 군집 내의 모든 데이터들 간의 평균 거리 - **군집 간 분리도 (separation)**

Unit 04 | 모델 평가

2. Silhouette Coefficient



눈으로 보기엔 DBSCAN의 결과가 제일 좋지만, K-Means의 실루엣 계수가 가장 높음을 볼 수 있음

➔모양이 복잡할 때는 평가가 잘 들어맞지 않음

Contents

Unit 05 | Clustering 실습 - titanic 생존자 예측

```
#데이터 EDA
train_data.describe()
```

	Pclass	Age	SibSp	Parch
count	891.000000	714.000000	891.000000	891.000000
mean	2.308642	29.699118	0.523008	0.381594
std	0.836071	14.526497	1.102743	0.806057
min	1.000000	0.420000	0.000000	0.000000
25%	2.000000	20.125000	0.000000	0.000000
50%	3.000000	28.000000	0.000000	0.000000
75%	3.000000	38.000000	1.000000	0.000000
max	3.000000	80.000000	8.000000	6.000000

```
[ ] train_data.isna().sum()
```

```
Pclass      0
Sex         0
Age       177
SibSp       0
Parch       0
dtype: int64
```

Age의 경우 다른 변수에 비해 결측치가 많은 것을 확인할 수 있다.

K-means를 활용

> 클러스터링 작업을 통해 같은 군집에 속하는 데이터들의 평균으로 결측치를 채워넣을 수 있음

```
[ ] k_train_data = train_data.copy()
    k_test_data = test.copy()
```

```
[ ] #결측치가 존재하는 Age column을 제외하고 클러스터링을 진행할 것

    from sklearn.preprocessing import StandardScaler

    scaler = StandardScaler()
    scaler.fit(k_train_data.drop(["Age"], axis=1)) #Age를 제외한 변수들 스케일링 (정규화)

    train_fill_data = scaler.transform(k_train_data.drop(["Age"], axis=1))
    test_fill_data = scaler.transform(k_test_data.drop(["Age"], axis=1))
```

Unit 05 | Clustering 실습 - titanic 생존자 예측

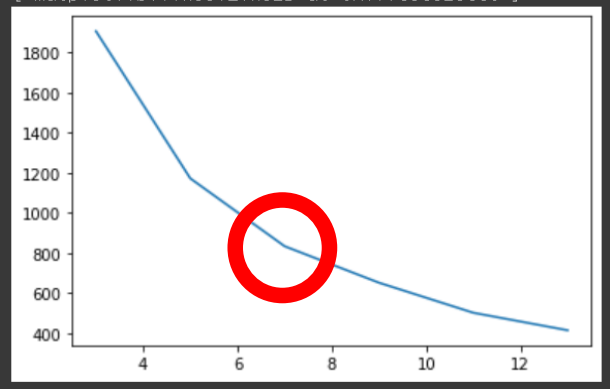
Elbow method를 이용해 k값 결정 (SSE 찍어보기)

```
[ ]  from sklearn.cluster import KMeans

     n_cluster = []
     sse = []
     for n in range(3, 15, 2):
         kmeans = KMeans(n_clusters=n)
         kmeans.fit(train_fill_data)
         n_cluster += [n]
         sse += [kmeans.inertia_]
```

```
[ ]  plt.plot(n_cluster, sse)
```

```
[<matplotlib.lines.Line2D at 0x7ff5e5c2e550>]
```




```
[ ]  #elbow point를 7으로 결정
     n_clusters = 7
```

```
[ ]  kmeans = KMeans(n_clusters=n_clusters)
     kmeans.fit(train_fill_data)
```

```
KMeans(n_clusters=7)
```

```
[ ]  train_fill_data
```

```
array([[ 0.82737724,  0.43279337, -0.47367361, -0.73769513,  0.73769513],
       [-1.56610693,  0.43279337, -0.47367361,  1.35557354, -1.35557354],
       [ 0.82737724, -0.4745452 , -0.47367361,  1.35557354, -1.35557354],
       ...,
       [ 0.82737724,  0.43279337,  2.00893337,  1.35557354, -1.35557354],
       [-1.56610693, -0.4745452 , -0.47367361, -0.73769513,  0.73769513],
       [ 0.82737724, -0.4745452 , -0.47367361, -0.73769513,  0.73769513]])
```

```
[ ]  clustered_train = kmeans.predict(train_fill_data)
     clustered_test = kmeans.predict(test_fill_data)
```

```
[ ]  print(len(clustered_train))
     clustered_train
```

```
891
array([4, 0, 5, 0, 4, 4, 6, 2, 3, 0, 5, 0, 4, 3, 5, 0, 2, 1, 5, 5, 1, 1,
       5, 6, 5, 3, 4, 2, 5, 4, 6, 0, 5, 1, 6, 6, 4, 4, 5, 5, 5, 0, 4, 3,
       5, 4, 4, 5, 4, 5, 2, 4, 0, 0, 6, 6, 0, 4, 3, 2, 4, 0, 6, 2, 6, 4,
       0, 4, 2, 4, 1, 2, 1, 4, 4, 4, 4, 4, 1, 5, 4, 4, 5, 6, 0, 5, 3, 4,
       3, 4, 4, 4, 6, 4, 4, 4, 6, 6, 0, 1, 5, 4, 6, 4, 4, 4, 5, 4, 4, 5,
       6, 5, 4, 5, 5, 4, 4, 1, 6, 2, 1, 4, 1, 0, 6, 4, 4, 4, 5, 4, 4, 4,
       5, 0, 1, 1, 3, 6, 4, 6, 3, 5, 5, 4, 1, 1, 4, 3, 1, 1, 1, 0, 4, 4,
       4, 6, 5, 4, 4, 2, 4, 0, 4, 4, 2, 4, 0, 3, 6, 4, 6, 2, 5, 4, 6, 4,
       2, 0, 1, 4, 2, 1, 2, 1, 3, 6, 5, 6, 4, 4, 0, 1, 5, 1, 0, 0, 4, 4,
       5, 0, 4, 2, 4, 4, 4, 5, 4, 4, 5, 6, 4, 0, 4, 1, 4, 0, 5, 1, 0, 1,
       4, 1, 4, 4, 6, 4, 1, 4, 1, 5, 0, 4, 1, 2, 1, 5, 1, 3, 1, 1, 5, 5,
       1, 4, 4, 6, 5, 3, 6, 1, 4, 5, 6, 4, 3, 3, 0, 0, 0, 0, 4, 2, 6, 6,
       5, 1, 2, 4, 0, 0, 6, 4, 0, 6, 5, 0, 5, 1, 2, 5, 4, 4, 4, 4, 6, 4,
       4, 4, 1, 5, 0, 0, 1, 5, 4, 6, 4, 3, 6, 0, 5, 4, 4, 0, 4, 6, 0, 0,
       1, 0, 0, 3, 0, 4, 1, 5, 0, 1, 3, 0, 4, 4, 0, 0, 2, 0, 4, 0, 5, 0,
       5, 6, 6, 4, 0, 4, 6, 0, 4, 6, 1, 3, 1, 1, 1, 0, 0, 5, 4, 4, 4, 6,
```

모든 데이터의 군집이 결정된 것을 확인 가능

- 클러스터링이 잘 되었는지 확인하기 위해 실루엣 계수 확인

```
[ ]  from sklearn.metrics import silhouette_score

     mean_score = silhouette_score(train_fill_data, clustered_train)
     print(mean_score)
```

```
0.563016053595228
```

실루엣 계수의 값이 0.5보다 크므로 타당하다고 판단 가능

```
[ ]  #각 군집 별 Age 평균값 찾기
     cluster_fill_value = {}
     for i in range(n_clusters):
         class_mean = k_train_data.loc[clustered_train == i, "Age"].dropna().mean()
         cluster_fill_value[i] = class_mean

     cluster_fill_value
```

```
{0: 33.650375939849624,
 1: 30.74070707070707,
 2: 7.888888888888889,
 3: 25.48,
 4: 28.066753246753247,
 5: 21.726666666666667,
 6: 41.2769696969697}
```

Unit 05 | Clustering 실습 - titanic 생존자 예측

```
[ ]  #정확도 비교

     mean_test_accuracy = accuracy_score(test_label, mean_test_pred)
     cluster_test_accuracy = accuracy_score(test_label, cluster_test_pred)

     print(f"Test Accuracy for mean data is {mean_test_accuracy:.4f}")
     print(f"Test Accuracy for cluster data is {cluster_test_accuracy:.4f}")
```

```
Test Accuracy for mean data is 0.8086
Test Accuracy for cluster data is 0.8349
```

k-means를 통해 결측치를 채웠을 때의 결과가 더 높은 정확성을 가짐을 확인 가능하다.

과제

Clustering 해보기

주어진 데이터로 주석과 함께 자유롭게 과제를 진행해주세요 ☺

- Preprocessing / EDA
- Clustering (수업시간에 배운 세 가지 방법 + α)
- Evaluation

데이터 : https://www.kaggle.com/datasets/vjchoudhary7/customer-segmentation-tutorial-in-python

Reference

참고자료

- 투빅스 17기 이지수님 강의자료
- 투빅스 16기 박한나님 강의자료
- 투빅스 15기 김현지님 강의자료
- 고려대학교 김성범 교수님의 핵심 머신러닝 군집분석 강의 : https://www.youtube.com/watch?v=8zB-_LrAraw
- Clustering Evaluation and assessment (tistory.com)

들어주셔서 감사합니다.