계층적 군집 (1)

계층적 군집분석은 `데이터와 데이터`, `군집과 데이터`, `군집과 군집` 간의 **거리**에 대한 유사도로 군집을 형성

동작하는 방식

두 점 사이를 가까운 것끼리 합침. 이 과정을 반복하여 전체가 하나의 클러스터에 속할 때까지 반복.

> 이 과정에 대한 시각화 결과물이 덴드로그램 형태로 나타남

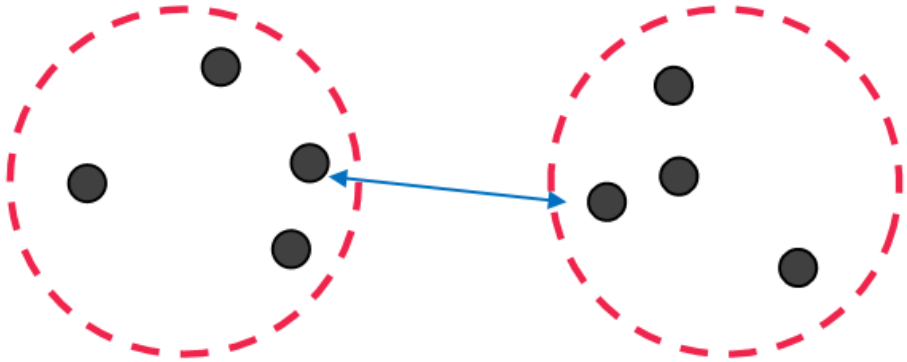
어떻게 두 점 사이의 거리를 결정할지에 대한 측정 방식이 필요함

> 유클리디안 거리 측정(직선거리), 맨하탄 거리 측정 방식(블록거리)

군집간의 거리를 구하는 방법

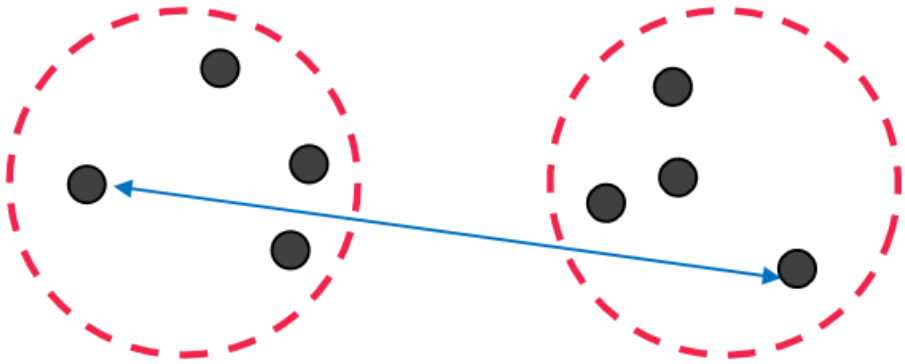
1. Min(Single Link)

군집과 군집의 거리를 구할 때 가장 최소거리인것을 유사도로 측정하는 방식



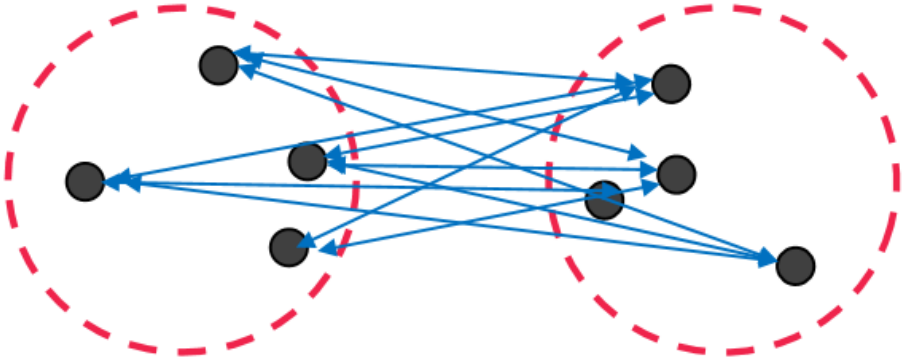
2. Max(Complete Link)

군집과 군집의 거리를 구할 때 가장 최대거리인것을 유사도로 측정하는 방식



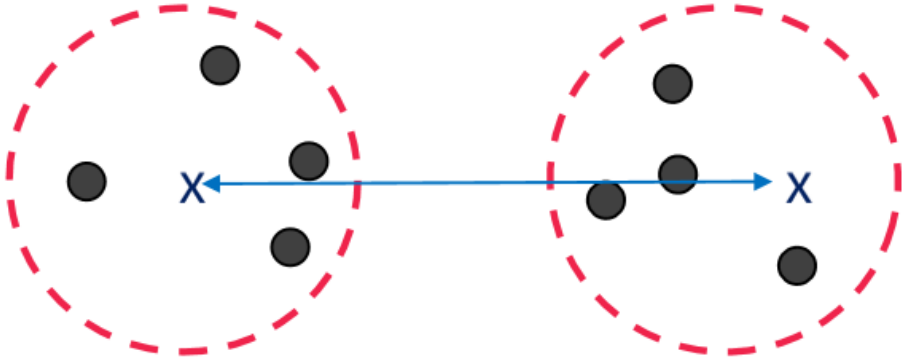
3. Average Link

군집과 군집의 거리를 구할 때 거리의 평균을 구해 유사도로 측정하는 방식



4. Centroids

군집과 군집의 거리를 구할 때 데이터의 중심점 거리를 유사도로 측정하는 방식

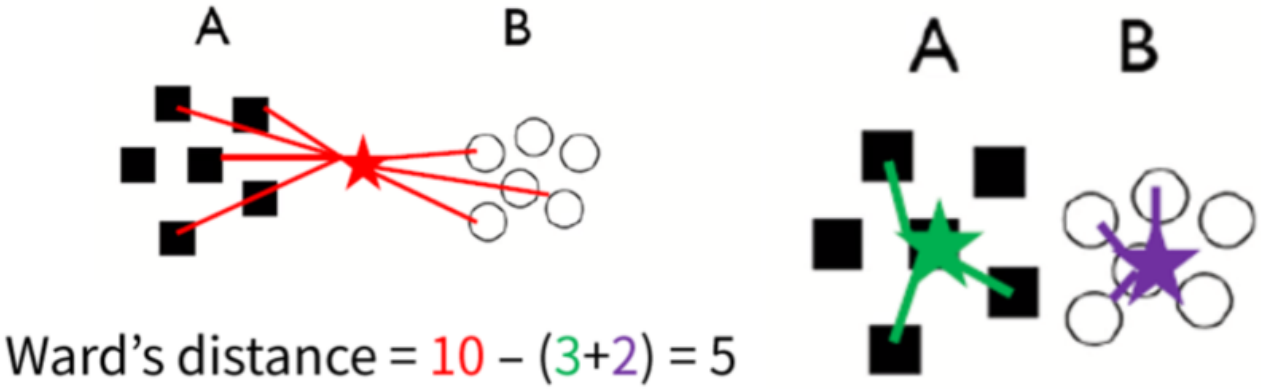


5. Ward's method

두개의 군집이 병합되었을 때 증가하는 변동성의 양으로 유사도를 측정하는 방식

변동성

두 군집의 중앙값과 두 군집에 있는 모든 데이터와의 거리의 합에서 군집간에 형성되는 거리를 뺀 값



#01. 패키지 참조

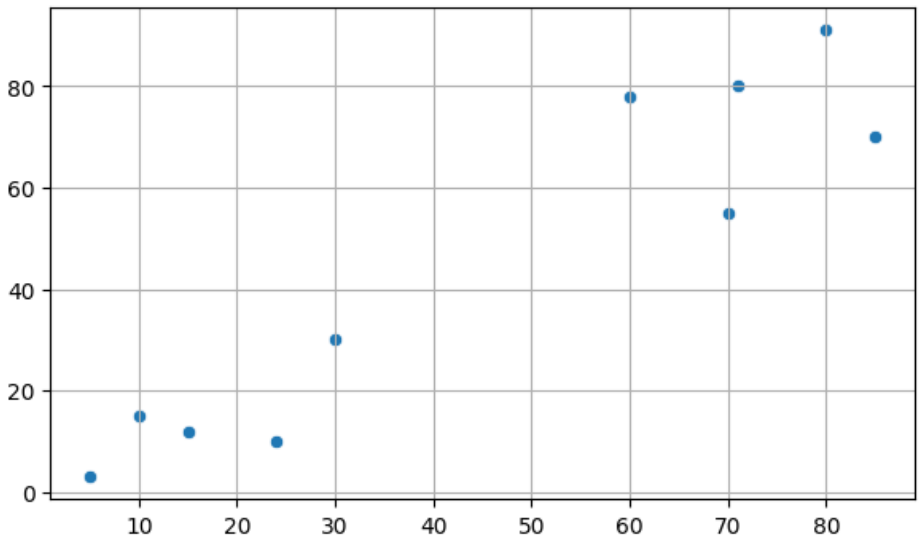
```
import seaborn as sb
import numpy as np
from matplotlib import pyplot as plt
from scipy.cluster.hierarchy import dendrogram, linkage

# sklearn은 AgglomerativeClustering() 함수를 제공한다.
```

#02. 실습 데이터 준비

```
x = np.array([[5,3],
    [10,15],
    [15,12],
    [24,10],
    [30,30],
    [85,70],
    [71,80],
    [60,78],
    [70,55],
    [80,91],])
x
```

```
array([[ 5,  3],
       [10, 15],
       [15, 12],
       [24, 10],
       [30, 30],
       [85, 70],
       [71, 80],
       [60, 78],
       [70, 55],
       [80, 91]])
```

```
plt.figure(figsize=(7, 4))
sb.scatterplot(x=x[:,0], y=x[:,1])
plt.grid()
plt.show()
plt.close()
```



#03. 계층 군집 수행

method 파라미터

군집간의 거리를 구하는 방법

`single`, `complete`, `average`, `weighted`, `centroid` 중 선택

> https://docs.scipy.org/doc/scipy/reference/generated/scipy.cluster.hierarchy.linkage.html

metric 파라미터

> scipy.spatial.distance.pdist 클래스의 인스턴스

- `euclidean` : 점 사이의 직선거리 측정
- `cityblock` : 점 사이의 맨하탄 거리 측정(블록)
- `seuclidean` : 표준화 된 유클리디안
- `sqeuclidean` : 제곱된 유클리디안
- `cosine` : 코사인 거리 계산
- 콜백함수 지정 가능함

> https://docs.scipy.org/doc/scipy/reference/generated/scipy.spatial.distance.pdist.html#scipy-spatial-distance-pdist

```
lnk = linkage(x, method='single', metric='euclidean')
lnk
```

```
array([[ 1.        ,  2.        ,  5.83095189,  2.        ],
       [ 3.        , 10.        ,  9.21954446,  3.        ],
       [ 6.        ,  7.        , 11.18033989,  2.        ],
       [ 0.        , 11.        , 13.        ,  4.        ],
       [ 9.        , 12.        , 14.2126704 ,  3.        ],
       [ 5.        , 14.        , 17.20465053,  4.        ],
       [ 4.        , 13.        , 20.88061302,  5.        ],
       [ 8.        , 15.        , 21.21320344,  5.        ],
       [16.        , 17.        , 47.16990566, 10.        ]])
```

군집 결과 시각화

```
plt.figure(figsize=(10, 6))

dendrogram(lnk,
           orientation='top',   # 방향: top(기본값), bottom, left, right
           labels=range(1, 11), # 라벨 인덱스

           # 각 노드 n에 대해 두 하위 링크가 표시되는 순서
           # False: 아무것도 안함(기본값)
           # 'ascending': 클러스터에 원본 개체수가 가장 적은 하위 객체가 먼저 출력
           # `descending`: 클러스터에 원본 개체수가 가장 많은 하위 객체가 먼저 출력
           count_sort='ascending',
           # 각 노드 n에 대해 두 하위 링크가 표시되는 순서
           # False: 아무것도 안함(기본값)
           # 'ascending': 직계 자손의 사이의 거리가 최소인 하위 항목이 먼저 표시됨
           # `descending`: 직계 자손의 사이의 거리가 최대인 하위 항목이 먼저 표시됨
           distance_sort='ascending',
           show_leaf_counts=True, # True: 맨 아래에 노드에 속한 개체 수 표시(k>1 경우만...)
    )
```

```
plt.show()
plt.close()
```