군집분석(Cluster Analysis)

개체(자료)들을 서로간 거리를 이용하여 유사한 특성을 갖는 몇 개의 군집(cluster)으로 집단화하는 다변량기법
(군집의 수는 사전에 알려져 있지 않음)~ 주어진 자료를 원하는 개수의 군집으로 나누기.

① Hierarchical clustering(계층적 군집방법) :
개체 간의 거리의 의하여 나무모양의 계층구조를 형성해가는 방법.
agglomerative method(start with n clusters) & divisive method(start with one cluster)

②Non-hierarchical clustering(비계층적 군집방법):
군집의 수를 정한 상태에서 설정된 군집의 중심에 가장 가까운 개체를 하나씩 포함해 가는 방식으로 군집을 형성해가는 방법. 주로, k-means군집방법이 사용.

1. 거리의 척도

자료행렬 $X = \begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1p} \\ x_{21} & x_{22} & \cdots & x_{2p} \\ & & \cdots & \\ x_{n1} & x_{n2} & \cdots & x_{np} \end{pmatrix}$ n×p 행렬 (n개의 관측벡터(자료, 개체), p개의 변수)

두 자료 $\boldsymbol{x}_i$와 $\boldsymbol{x}_j$ 사이의 거리 : $d_{ij} = d(\boldsymbol{x}_i, \boldsymbol{x}_j)$

Cityblock distance (Minkowski distance m=1): $\sum_{k=1}^{p} |\boldsymbol{x}_{ik} - \boldsymbol{x}_{jk}|$

Euclidean distance (Minkowski distance m=2): $\left[\sum_{k=1}^{p} |\boldsymbol{x}_{ik} - \boldsymbol{x}_{jk}|^2 \right]^{1/2} = \sqrt{(\boldsymbol{x}_i - \boldsymbol{x}_j)'(\boldsymbol{x}_i - \boldsymbol{x}_j)}$

Mahalanobis distance : 변수간의 상관관계 고려
$\sqrt{(\boldsymbol{x}_i - \boldsymbol{x}_j)' S^{-1} (\boldsymbol{x}_i - \boldsymbol{x}_j)}$, S: 표본 공분산행렬

거리행렬(Distance matrix)

$\boldsymbol{D} = \begin{pmatrix} d_{11} & d_{12} & \cdots & d_{1n} \\ d_{21} & d_{22} & \cdots & d_{2n} \\ & & \cdots & \\ d_{n1} & d_{n2} & \cdots & d_{nn} \end{pmatrix}$, * $d_{ii} = 0$, $d_{ij} > 0$ $(i \neq j)$, $d_{ij} < d_{ik} + d_{jk}$

2. 계층적군집방법(hierarchical clustering)

agglomerative method(start with n clusters) & divisive method(start with one cluster)

군집연결법:

ⓐ최단연결법(single linkage method)
군집(uv)와 임의의 다른 개체 w와의 거리를$d_{(uv)w} = \min\{d_{uw}, d_{vw}$ 로 정의

ⓑ최장연결법(complete linkage method):
군집(uv)와 임의의 다른 개체 w와의 거리를$d_{(uv)w} = \max d_{uw}, d_{vw}$ 로 정의.

ⓒ평균연결법(average linkage method):
군집(uv)와 임의의 다른 개체 w와의 거리를$d_{(uv)w} = d_{uw} + d_{vw}/2$ 로 정의

ⓓ중심연결법(centroid linkage method)
군집(uv)와 임의의 다른 개체 w와의 거리를 $d_{(uv)w} = d_{c(u,v)w}$,$c(uv) = (\boldsymbol{x}_u + \boldsymbol{x}_v)/2$로 정의.

ⓓ ward linkage method: 두 군집이 합쳤을 때의 오차 제곱합의 증가분에 기반해서 계산.

eg) $X = \begin{bmatrix} 0.3 & 0.9 \\ -1.3 & 1.4 \\ 0.7 & -1.6 \\ 1.6 & -1.4 \\ -0.7 & 0.6 \end{bmatrix}$, $D = \begin{bmatrix} 0 & 1.6763 & 2.5318 & 2.6420 & 1.0440 \\ 1.6763 & 0 & 3.6056 & 4.0311 & 1.0000 \\ 2.5318 & 3.6056 & 0 & 0.9220 & 2.6077 \\ 2.6420 & 4.0311 & 0.9220 & 0 & 3.0480 \\ 1.0440 & 1.0000 & 2.6077 & 3.0480 & 0 \end{bmatrix}$ (euclidean 거리이용)

$d_{12} = \sqrt{(0.3 - (-1.3))^2 + (0.9 - 1.4)^2} = 1.6763$, $d_{13} = \sqrt{(0.3 - 0.7)^2 + (0.9 - (-1.6))^2} = 2.5318$, $\cdots$

eg0) 거리행렬 구하기

```
import numpy as np
import matplotlib.pyplot as plt
X=np.array([[ 0.3, 0.9], [-1.3,1.4],[0.7,-1.6],[1.6,-1.4],[-0.7, 0.6]])
'''
다음과 같은 방법으로 표현가능
X=[ 0.3, 0.9, -1.3,1.4, 0.7,-1.6, 1.6,-1.4, -0.7, 0.6]
X=np.array(X, dtype=np.float32).reshape((5,2))
'''
plt.figure(0)
plt.plot(X[:,0],X[:,1],'k*')
plt.xlabel('X[:,0]');plt.ylabel('X[:,1]')

from scipy.spatial.distance import cdist
D=cdist(X,X,'euclidean') #'cityblock', 'mahalanobis'
print(D)
```

군집연결법(최단연결법):

거리행렬(D)의 최단 거리=0.9220 = d_{34} → 3과 4 연결 (3,4)

다음 최단 거리=1.00=d_{25} → 2와 5 연결 (2,5) ~ 남은 개체=1

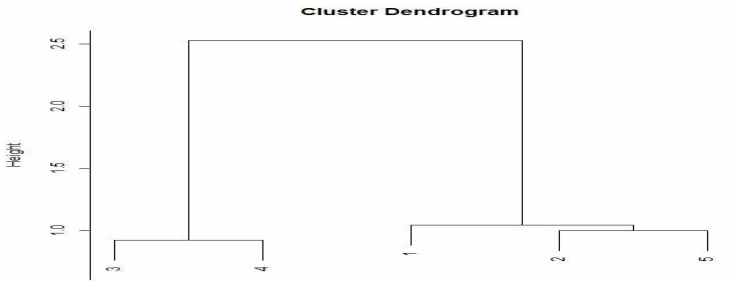
(3,4)와 1 최단거리 = min (d_{31}, d_{41})=min (2.53, 2.64)=2.53

(2,5)와 1 최단거리 = min (d_{21}, d_{51})=min (1.67, 1.04)=1.04

(2,5)와 1 최단거리 < (34)와 1 최단거리 → (2,5)와 1 연결 (2,5,1): $d_{25,1} = 1.04$

(3,4)와 (2,5,1) 최단거리= $\min(d_{34,25}, d_{34,1}) = \min(2.60, 2.53) = 2.53$ ~모든 관측값이므로 연결: $d_{34,251} = 2.53$

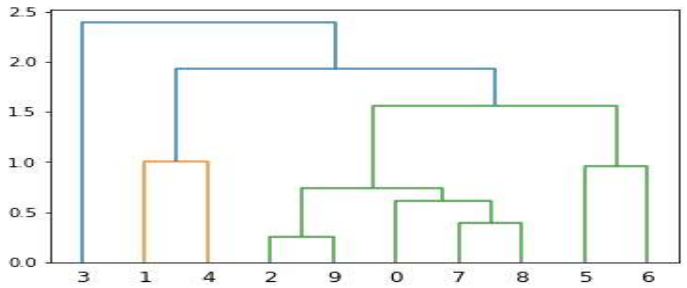
$d_{34,25} = \min(d_{3,25}, d_{4,25}) = 2.60$, $d_{3,25} = \min(d_{32}, d_{35}) = 2.60$, $d_{4,25} = \min(d_{42}, d_{45}) = 3.04$



자료=(1,2,3,4,5) ~ 군집수=5, (3,4),1,2,5 ~ 군집수=4

(3,4), (2,5) ,1 ~ 군집수=3, (3,4), (2,5,1) ~ 군집수=2, (3,4,2,5,1) ~ 군집수=1

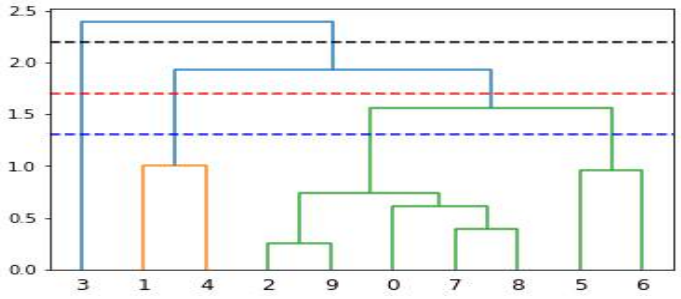
자료수가 적은 경우: dendrogram로 각군집에 속하는 자료 파악 가능



2군집=[3], [0,1,4,2,9,0,7,8,5,6] (군집에 속하는 자료번호로 표현)

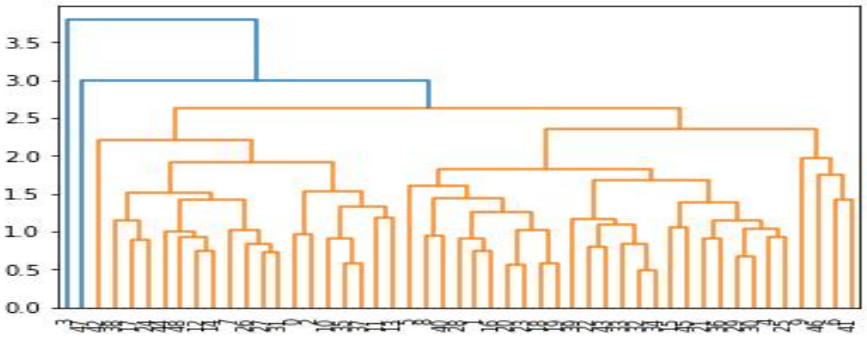
3군집=[3], [1,4], [2,9,0,7,8,5,6], 4군집=[3], [1,4], [2,9,0,7,8], [5,6]

* 수평으로 선을 그어서 만나는 수직선 갯수가 군집 수



검은 점선 ⇒ 2군집=[3], [1,4,2,9,0,7,8,5,6]
빨간 점선 ⇒ 3군집=[3], [1,4], [2,9,0,7,8,5,6]
파란 점선 ⇒ 4군집=[3], [1,4], [2,9,0,7,8], [5,6]

자료수가 많은 경우: dendrogram 읽기 어려움 ⇒ AgglomerativeClustering 이용



3. k-평균군집방법(k-means clustering)

(i) 각 개체를 초기에 설정된 k개의 군집에 각각 할당, CL_1 , CL_2 , ⋯, CL_k

j번째 군집의 중심: $\boldsymbol{c}_j = \frac{1}{n_j} \sum_{i \in CL_i} \boldsymbol{x}_i,\ j = 1,\cdots,k$

(ii) 개체들과 각군집의 중심과의 거리계산, $d(i, \boldsymbol{c}_j)$

- 각 개체를 군집의 중심에 가까운 군집으로 재할당

(iii) 각군집이 변하지 않을 때까지 반복

*사용이 편하고 속도가 빠른 알고리즘.
중심점을 구할때 처음에 랜덤으로 중심점의 위치를 찾기 때문에, 잘못하면, 중심점과 다른 자료의 거리가 전역 최소값이 아니고 지역 최소값이 구해질 수 있다는 약점.

eg1) 계층적 군집방법(자료수 적은 경우)

```
import numpy as np
import matplotlib.pyplot as plt
X = [-0.4326,  -0.1867,  -1.6656,  0.7258,  0.1253,  -0.5883,  0.2877,  2.1832, \
   -1.1465,  -0.1364,  1.1909,  0.1139,  1.1892,  1.0668,  -0.0376,  0.0593,\
    0.3273,  -0.0956,  0.1746,  -0.8323]
X=np.array(X, dtype=np.float32).reshape((10,2))  자료수(개체수)=10, 입력변수수=2
plt.figure(1)
plt.plot(X[:,0],X[:,1],'k*')
plt.xlabel('X[:,0]');plt.ylabel('X[:,1]')

# 계층적 군집방법
from scipy.cluster.hierarchy import dendrogram, linkage
```

```
clusters = linkage(X, method='average', metric='euclidean')
# method(군집연결법)='single(default)', 'complete', 'average', 'centroid', 'ward',..
# metric(거리척도) ='euclidean'(default),'cityblock',..
plt.figure(1)
dendrogram(clusters)
plt.axhline(y=2.2, color='k', linestyle='--')
plt.axhline(y=1.7, color='r', linestyle='--')
plt.axhline(y=1.3, color='b', linestyle='--')
plt.show()
print('\n')
```

eg2) 계층적 군집방법(자료수 많은 경우)

```
import numpy as np
import matplotlib.pyplot as plt
# 자료(water_treatment data 에서 결측치 제거한 자료) 불러오기
x0=pd.read_csv('d:/Python2021_Lecture/water_treat_short.csv')
x0=x0.values # 자료수=49, 입력변수수=37

from sklearn.cluster import AgglomerativeClustering
cL = AgglomerativeClustering(n_clusters=4, affinity='euclidean', linkage='average')
clusterA=cL.fit_predict(x0)
# 군집0,1,2,3에 속하는 자료 번호(0~48)
v0=np.where(clusterA==0)[0]
v1=np.where(clusterA==1)[0]
v2=np.where(clusterA==2)[0]
v3=np.where(clusterA==3)[0]
print('4 clusters by H-clustering=',v0,v1,v2,v3)
```

eg3) k-평균군집방법

```
from sklearn.cluster import KMeans
km=KMeans(n_clusters=3).fit(X) # 군집수=3
clusterK= km.labels_
# 군집0,1,2에 속하는 자료 번호(0~9)
vk0=np.where(clusterK==0)[0]
vk1=np.where(clusterK==1)[0]
vk2=np.where(clusterK==2)[0]
center=km.cluster_centers_ # 각군집의 중심 위치
print('3 clusters by KMeans=',vk0,vk1,vk2)
plt.figure(3)
plt.plot(X[vk0,0],X[vk0,1],'k*')
plt.plot(X[vk1,0],X[vk1,1],'r*')
plt.plot(X[vk2,0],X[vk2,1],'b*');
plt.xlabel('x1'); plt.ylabel('x2')
```