14. Hierarchical Clustering
2019년 가을 학기

2019년 08월 21일
https://www.github.com/KU-BIG/KUBIG_2019_Autumn

이 글은 백준걸 교수님의 데이터 마이닝 강의의 강의안을 많이 참고하였다.

비지도학습인 clustering은 크게 거리를 기반으로 하거나 밀도를 기반으로 하는 방법으로 나뉜다. 거리 기반 방법은 Hierarchical clustering과 Partitioning clustering으로 나뉜다. 이번 강의에서는 계층을 이용하는 Hierarchical clustering에 대해서 살펴보겠다.

1. Introduction

Hierarchical Clustering, 계층적 군집화는 중심점을 기준으로 clustering하는 K-means와 다르게 계층적으로, 순차적으로 군집을 묶어가거나 분리하는 방법이다. 하나의 cluster에서 세부 cluster로 분리하는 방법인 Divisive(top-down) 방법이 있다. 반대로 모든 point가 하나의 cluster인 형태에서 점차 하나의 cluster로 묶여지는 Agglomerative(bottom-up) 방식이 있다. 좀 더 이해하기 쉽게 아래의 동물들을 각각의 방식으로 군집화해보자.

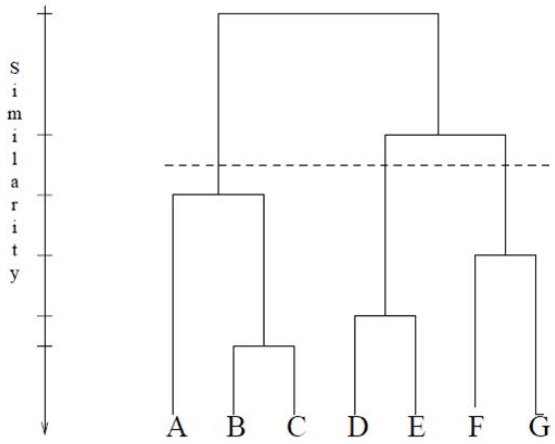
진돗개, 셰퍼드, 요크셔테리어, 푸들, 물소, 젖소

우선은 divisive 방법을 사용해보자. **(진돗개, 셰퍼드, 요크셔테리어, 푸들, 물소, 젖소)**로 하나의 큰 clustering에서 쪼개져 나가야 한다. 이 하나의 cluster를 개와 소로 분리시킨다. (**[(진돗개, 셰퍼드, 요크셔테리어, 푸들), (물소, 젖소)]**) 그 다음 개 군집에서 중형견과 소형견을 분리시키면 최종적으로 **[(진돗개, 셰퍼드), (요크셔테리어, 푸들), (물소, 젖소)]**로 세 군집이 생성된다.

다음은 agglomerative 방법을 이용해보자. 진돗개, 셰퍼드, 요크셔테리어, 푸들, 물소, 젖소는 각각 하나의 cluster이다. 이들을 중형견, 소형견, 소끼리 묶으면 **[(진돗개, 셰퍼드), (요크셔테리어, 푸들), (물소, 젖소)]**로 세 군집이 형성된다. 중형견과 소형견을 개로 묶자. (**[(진돗개, 셰퍼드, 요크셔테리어, 푸들), (물소, 젖소)]**) 마지막으로 나머지 두 군집을 동물이라는 하나의 카테고리로 묶으면 **(진돗개, 셰퍼드, 요크셔테리어, 푸들, 물소, 젖소)** 하나의 cluster만 남게 된다.

실제 알고리즘은 다소 다르나 전체적인 맥락은 이와 같다. 이처럼 각 cluster가 다른 cluster에 속하기에, 즉 nested 구조를 띠기 때문에 이와 같은 방법을 계층적 군집화라고 부른다.

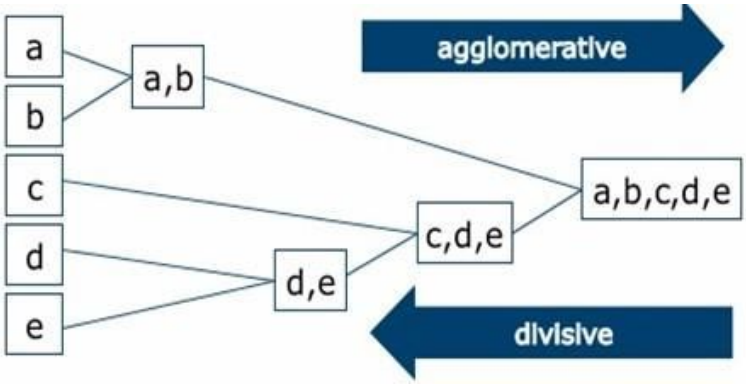
Hierarchical clustering이 nested된 방식으로 cluster를 생성하면서 갖는 이점이 있다. K-means와는 달리 학습하기 전에 따로 cluster개수를 지정하지 않아도 된다. Hierarchical clustering을 하면서 파생되는 dendrogram을 보고 분석자가 적절하다고 판단되는 지점에서 클러스터간의 연결 고리를 끊어주면 된다. 다시 말해 결과를 보고 클러스터 개수를 정할 수 있는 이점이 있다.



https://commons.wikimedia.org/wiki/File:Agglomerative_clustering_dendogram.png

위의 그림은 dendrogram으로 총 3개의 clustering을 생성한 것을 볼 수 있다. Dendrogram은 proximity matrix를 시각적으로 보여주는 것으로 막대기의 세로 길이가 길수록 그 군집/개체들의 거리가 멀다고 생각할 수 있다. (B, C)를 묶는 막대의 길이가 (F, G)를 묶는 막대의 길이보다 짧은 것을 보아 B,C간의 거리가 F,G간의 거리보다 가까운 것을 알 수 있다.

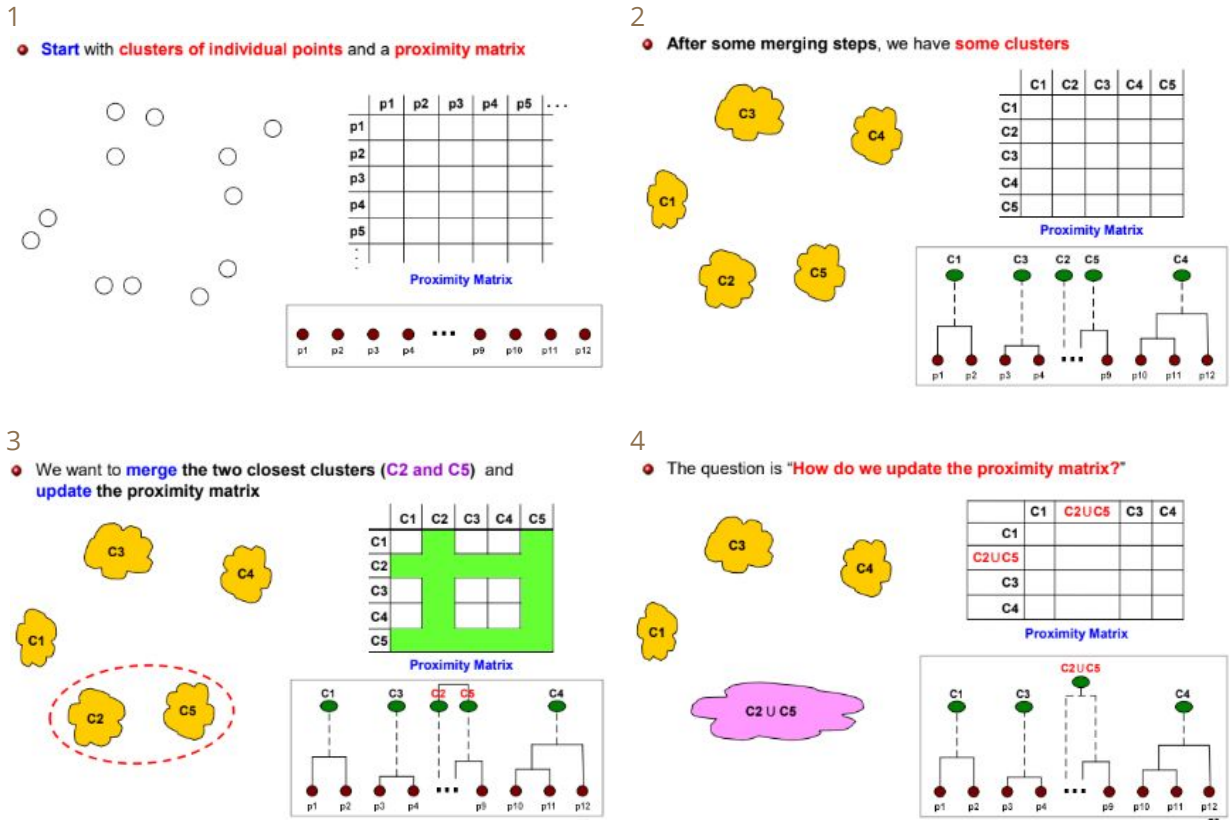
이번 강의에서는 일반적으로 주로 사용되는 Bottom-up 방식에 대해서만 서술하겠다.



2. Agglomerative Clustering Algorithm

Bottom-up방식의 hierarchical clustering의 알고리즘은 매우 간단하다. Introduction에 소개한 그 방식대로 하면 된다. 사실 이 방법은 알고리즘 그 자체보다도 cluster간 거리를 어떻게 계산하는지가 더욱 중요하다. 그래도 pseudo 알고리즘으로 간단하게 방식을 짚고 넘어가겠다.

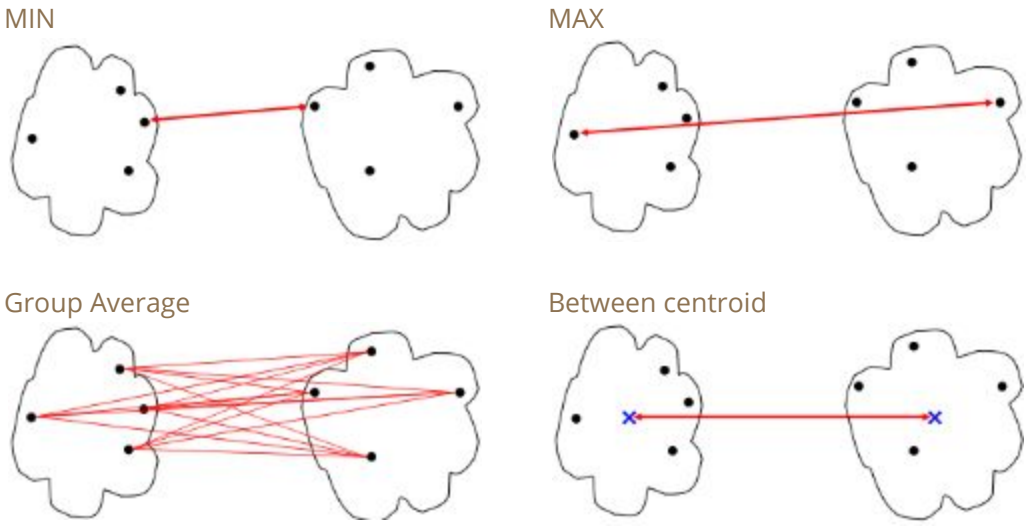
1. Proximity matrix를 계산한다. (Similarity matrix, Dissimilar(distance) matrix)
2. 각 data point를 하나의 cluster로 둔다.
3. 아래의 과정을 반복한다.
    a. 가장 가까운 두 cluster들을 묶는다.
    b. Proximity matrix를 업데이트 한다.
    c. 하나의 cluster가 남을 때까지 반복한다.



위의 슬라이드 자료를 보면 이해가 더 잘 될 것이다. 가까운 cluster끼리 묶고 묶어서 하나의 전체적인 cluster을 만드는 것이 바로 agglomerative clustering이다. 하지만 어떻게 cluster간의 거리를 측정하고 계산하는지가 모호할 것이다. 기준이 매우 다양하기 때문이다.

3. Cluster distance

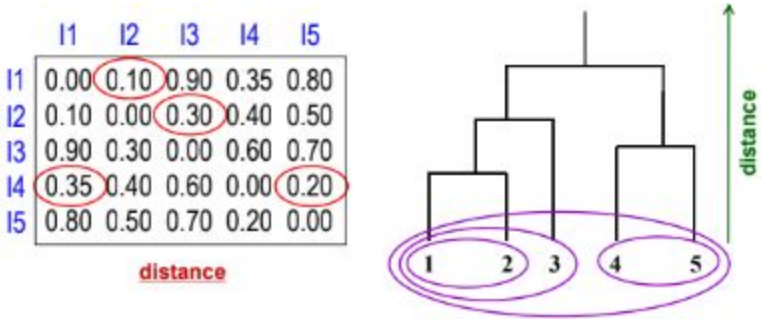
Agglomerative clustering에서의 핵심은 어떠한 기준으로 군집 간의 거리를 측정할 것인지이다. Cluster간의 거리 측정의 방식은 다양하며 각 방식마다 장단점이 다르다. 이에 각 거리간의 장단점을 제대로 파악하여 분석하고자 하는 data에 알맞은 기준을 사용하는 것이 중요하다.

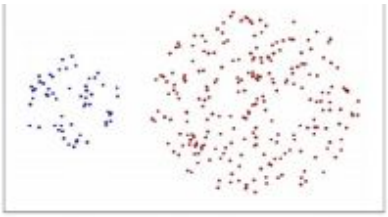


위의 그림에서 보는 것과 같이 이번 장에서는 Min / Max / Group average / Between centroid distance에 추가로 Wards' Method도 살펴볼 것이다.

3.1 Min distance / Single linkage

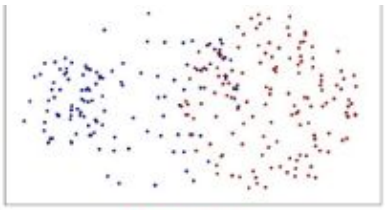
클러스터 u의 모든 데이터 i와 클러스터 v의 모든 데이터 j의 모든 조합에 대해 측정된 거리 중 최솟값을 이용한다. 즉, 두 군집에서 가장 가까운 두 지점의 거리를 기반으로 한다. Single linkage는 다른 크기의 cluster들이 있거나 cluster의 형태가 구형이 아니더라도 잘 적합하다는 장점이 있다. 하지만 noise와 outlier에 취약하며, chaining 현상이 일어난다.





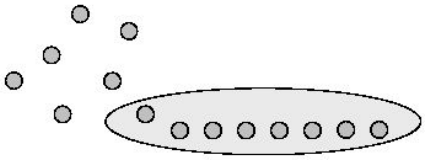
Two Clusters

Six Clusters



Two Clusters

The chaining problem:

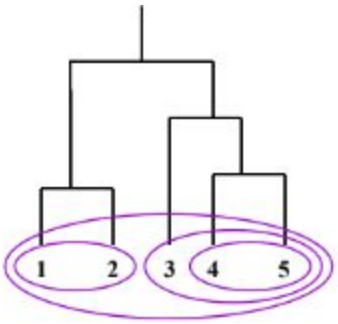


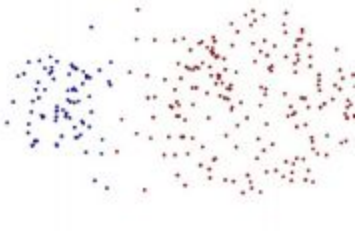
3.2 Max distance / Complete linkage

클러스터 u의 모든 데이터 i와 클러스터 v의 모든 데이터 j의 모든 조합에 대해 측정된 거리 중 최댓값을 사용한다. 즉, 두 군집에서 가장 먼 두 지점의 거리를 기반으로 한다. Complete linkage는 noise와 outlier에 덜 민감하지만 군집의 크기를 서로 비슷하게 맞추려는 경향이 있으며, 군집의 형태를 구형으로 생성하려는 경향이 있다.

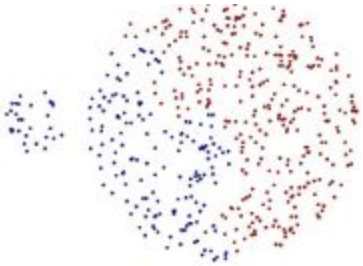
	I1	I2	I3	I4	I5
I1	0.00	0.10	0.90	0.35	0.80
I2	0.10	0.00	0.30	0.40	0.50
I3	0.90	0.30	0.00	0.60	0.70
I4	0.35	0.40	0.60	0.00	0.20
I5	0.80	0.50	0.70	0.20	0.00

distance





Two Clusters



Two Clusters

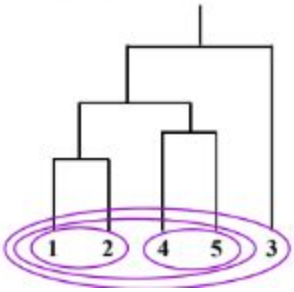
3.3 Group Average distance / Average linkage

클러스터 u의 모든 데이터 i와 클러스터 v의 모든 데이터 j의 모든 조합에 대해 측정한 거리들의 평균값을 사용한다. 즉, 두 군집의 point 쌍 별 거리의 평균을 기반으로 한다. Average linkage는 noise와 outlier에 비교적 덜 민감하지만 군집의 형태를 구형으로 생성하려는 경향이 있다. Single link와 complete link의 중간자적인 성격을 지닌다.

$$\text{proximity}(\text{Cluster}_i, \text{Cluster}_j) = \frac{\sum_{\substack{p_i \in \text{Cluster}_i \\ p_j \in \text{Cluster}_j}} \text{proximity}(p_i, p_j)}{|\text{Cluster}_i| \times |\text{Cluster}_j|}$$

	I1	I2	I3	I4	I5
I1	0.00	0.10	0.90	0.35	0.80
I2	0.10	0.00	0.30	0.40	0.50
I3	0.90	0.30	0.00	0.60	0.70
I4	0.35	0.40	0.60	0.00	0.20
I5	0.80	0.50	0.70	0.20	0.00

distance

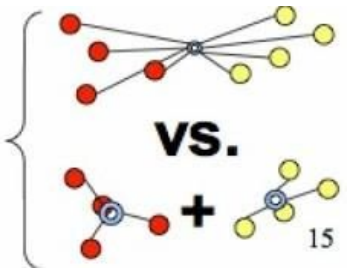


3.4 Between centroid distance

각 클러스터의 중심점(centroid)를 정의한 다음, 두 중심점의 거리를 클러스터 간의 거리로 정의한다. Between centroid distance는 단순하고 이해하기 쉬우며 다른 방법들에 비해 noise와 outlier에 덜 민감하지만 average linkage나 ward's method에 비해 수행능력이 떨어진다.

3.5 Ward's Method

위 4개의 거리와는 살짝 다른 개념으로, 모든 변수들에 대하여 두 군집의 ANOVA sum of square를 더한 값을 이용한다. 두 군집만 합쳐졌을 때 그 오차 제곱합의 증가분에 기반해서 측정한다. 즉, distance matrix를 구할 때 오차 제곱합의 증분(increase of ESS)을 두 군집 사이의 거리로 측정하게 된다.



유사성 측도 : ESS(error sum of squares) 의 증분.

$$d(i+j,k) = \frac{\left\| \mu_{i+j} - \mu_k \right\|^2}{\frac{1}{n_1} + \frac{1}{n_2}}$$

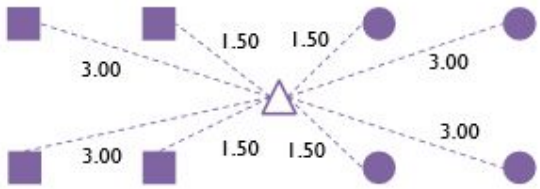
$$ESS = \sum_{k=1}^{K} \sum_{x_i \in C_k} \sum_{j=1}^{n} (x_{ij} - \bar{x}_{kj})^2$$

* ESS : error sum of squares
** k : number of clusters (1 ~ K)
x_i : elements of cluster C_k
j : number of variables (1 ~ n)

,

- SSE before merge: $1^2 + 1^2 + 1^2 + 1^2 + 1^2 + 1^2 + 1^2 + 1^2 = 8$

- SSE after merge: $4 \times 1.5^2 + 4 \times 3^2 = 45$



- Ward distance: **45-8 = 37**

Ward's method는 noise나 outlier에 덜 민감하지만 비슷한 크기의 군집으로, 구형의 군집으로 묶어주는 경향이 있다. Average linkage와 Ward's method의 유사성 측정 수식은 비슷하지만, average linkage의 유사성 측도 대비 Ward linkage에는 가중값이 추가되었다는 점이 다르다.

3.6 Hierarchical Cluster 한계

다음장으로 넘어가기 전에 hierarchical cluster의 한계를 짚고 넘어가겠다. Hierarchical cluster은 한번 두 cluster을 묶으면 그 결정을 되돌릴 수 없다. 또한 직접적으로 최소화를 시킬 objective function이 없기에 항상 local optimal에 빠질 위험이 존재한다. 마지막으로 어떠한 거리 기준을 사용하느냐에 따라서 한계가 다르다.

- noise와 outlier에 민감함 : MIN
- 다른 크기, 다른 형태의 cluster들을 다루기 어려움 : MAX, Average, Ward
- Cluster size를 맞추려는 경향 때문에 큰 cluster를 쪼갬 : MAX

4. Cluster 개수 정하기

위에서 언급한 것처럼 hierarchical clustering에서는 사용자가 나중에 클러스터의 개수를 정하게 됨. 그렇다면 적절한 클러스터의 개수(k)를 어떻게 정할 수 있을까?

Ward's Method를 제외하고, 기본적으로 클러스터링이 잘 되었다면 1) 같은 클러스터 안의 data끼리는 거리가 가까워야하며, 2) 다른 클러스터 안의 data끼리는 거리가 멀어야한다. 따라서 우리는 Within-cluster sum of squares(WWS)와 Between-cluster sum of squares(BSS)를 측정해야 하며, 계산 방법은 아래에 나와있다.

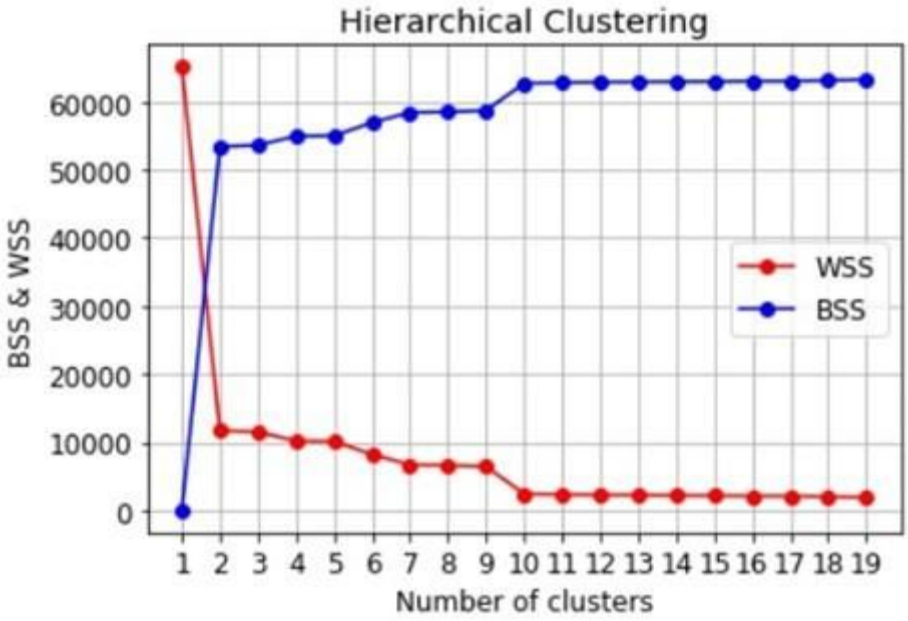
$$\mathrm{WSS}(C) = \sum_{i=1}^{k} \sum_{x_j \in C_i} d(x_j, \mu_i)^2$$

$$\mathrm{BSS}(C) = \sum_{i=1}^{k} |C_i| d(\mu, \mu_i)^2$$

μ_i는 cluster C의 centroid를 나타내며, μ는 전체 data set의 centroid이다.

위에서는 Euclidean 방식을 이용해 거리를 계산했다.

Cluster의 개수에 따른 WSS와 BSS를 모두 계산해 표로 살펴본 뒤, 가장 적합한 k를 구하면 된다.



k=2일 때나, 그 이상일 때나 WSS와 BSS 값이 크게 변하지 않는다.

따라서 k=2로 클러스터 개수를 정할 수 있다.

이 방법의 단점은 다음과 같다. 1) 각 군집들이 어떤 특성을 가지고 있는지 알려주지 못한다. 2) 거리계산 방법이 달라질 때마다 추천 군집의 수가 달라진다. 3) 군집으로 한 번 설정하면 군집 이동이 불가능하다. 4) 계산시간이 오래 걸리기 때문에 관측치가 150개 이하일 때 주로 사용된다.

Distance based clustering은 우리가 배웠던 K-means와 agglomerative clustering 외에도 divisive clustering이 있다. 이 방법은 주로 사용되지는 않지만 agglomerative clustering과 달리 한번 군집으로 나뉘어져도 그 과정을 다시 번복할 수 있다. 여기서 divisive clustering에 대해서 소개하지는 않지만 이 방법에 관심이 있는 사람은 divisive clustering중에서도 기초적인 방법인 Minimum Spanning Tree(MST)에 대해서 찾아보기 바란다.

Algorithm 7.5 MST Divisive Hierarchical Clustering Algorithm

1: Compute a minimum spanning tree for the proximity graph.
2: **repeat**
3: Create a new cluster by breaking the link corresponding to the largest distance (smallest similarity).
4: **until** Only singleton clusters remain