Clustering algorithms

Konstantinos Koutroumbas

Unit 9

- Minimum Spaning tree and SL algorithm
- Cophenetic matrix
- Divisive hierarchical clustering algorithms
- CURE algorithm

Agglomerative graph theory based Clustering Algorithms

Remarks:

- SL poses the weakest possible graph condition (connectivity) for the formation of a cluster, while CL poses the strongest possible graph condition (completeness) for the formation of a cluster.
- A variety of graph theory-based algorithms, that lie between these two extremes result for various choices of $h(k)$.
  - For k=1 all these algorithms **collapse** to the single link algorithm.
  - As k increases, the resulting subgraphs approach completeness.

Clustering algorithms based on the Minimum Spanning Tree (MST)

Definitions:

Spanning Tree: It is a connected graph (containing all the vertices of the graph), with no loops (only one path connects any two vertices).

Weight of a Spanning Tree: The sum of the weights of its edges (provided that they have been assigned with a weight).

Minimum Spanning Tree (MST): A spanning tree with the smallest weight among the spanning trees connecting all the vertices of the graph.

Agglomerative graph theory based Clustering Algorithms

Remarks:

- The MST has N-1 edges.
- When all the weights are different from each other, the MST is unique. Otherwise, it may not be unique.

➢ Employing the GTAS and substituting $g_{h(k)}(C_r, C_s)$ with

$$g(C_r, C_s) = min_{ij}\{w_{ij}: \boldsymbol{x}_i \in C_r, \boldsymbol{x}_j \in C_s\}$$

where $w_{ij} = d(\boldsymbol{x}_i, \boldsymbol{x}_j)$, we can determine the MST.

➢ **Alternatively,** a hierarchy of clusterings may be obtained by the MST as follows:

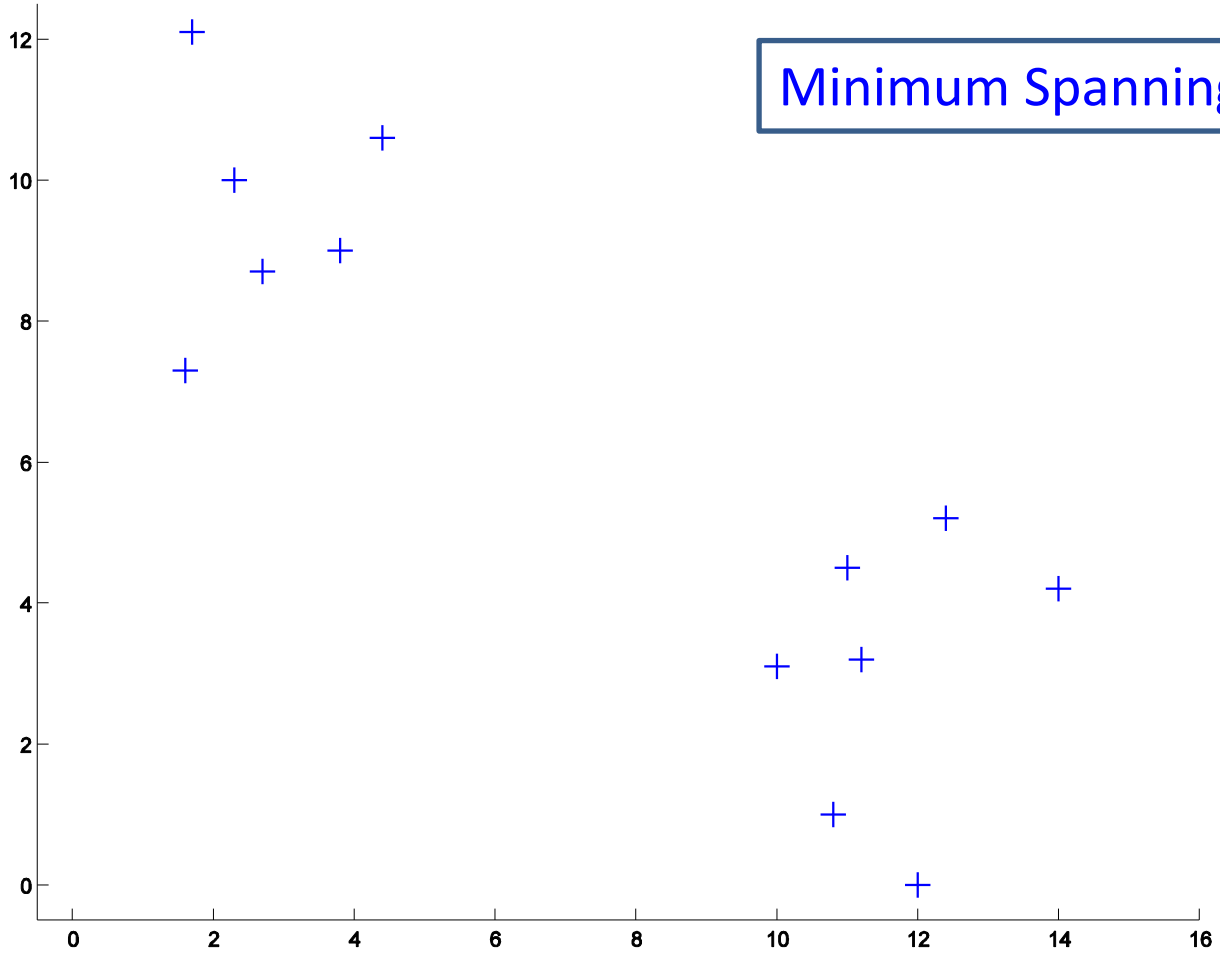
The clustering $\Re_t$ at the tth level is the set of connected components of the MST, when only its t smallest weights are considered.

Remark:

The hierarchy produced by MST is the same with that produced by the single link algorithm, at least when all w_{ij}´s are different from each other.

Agglomerative graph theory based Clustering Algorithms

Example:

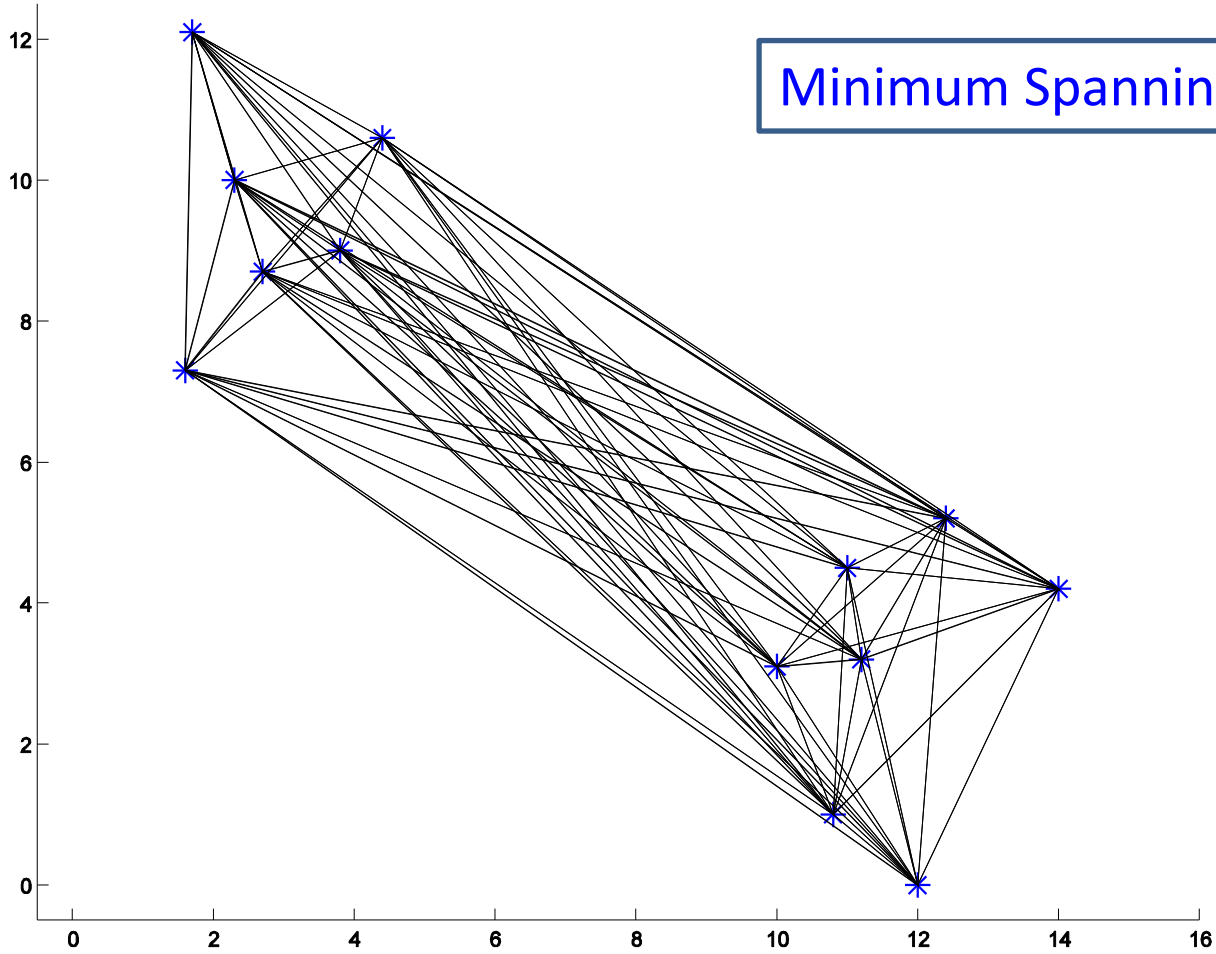


Minimum Spanning Tree (MST)

- **Define** a **complete** graph with vertices the data points and edges the segments connecting every pair of vertices.
- **Weight** each edge by the distance between its two end-points.
- **Define** the MST of the graph.

Agglomerative graph theory based Clustering Algorithms

Example:

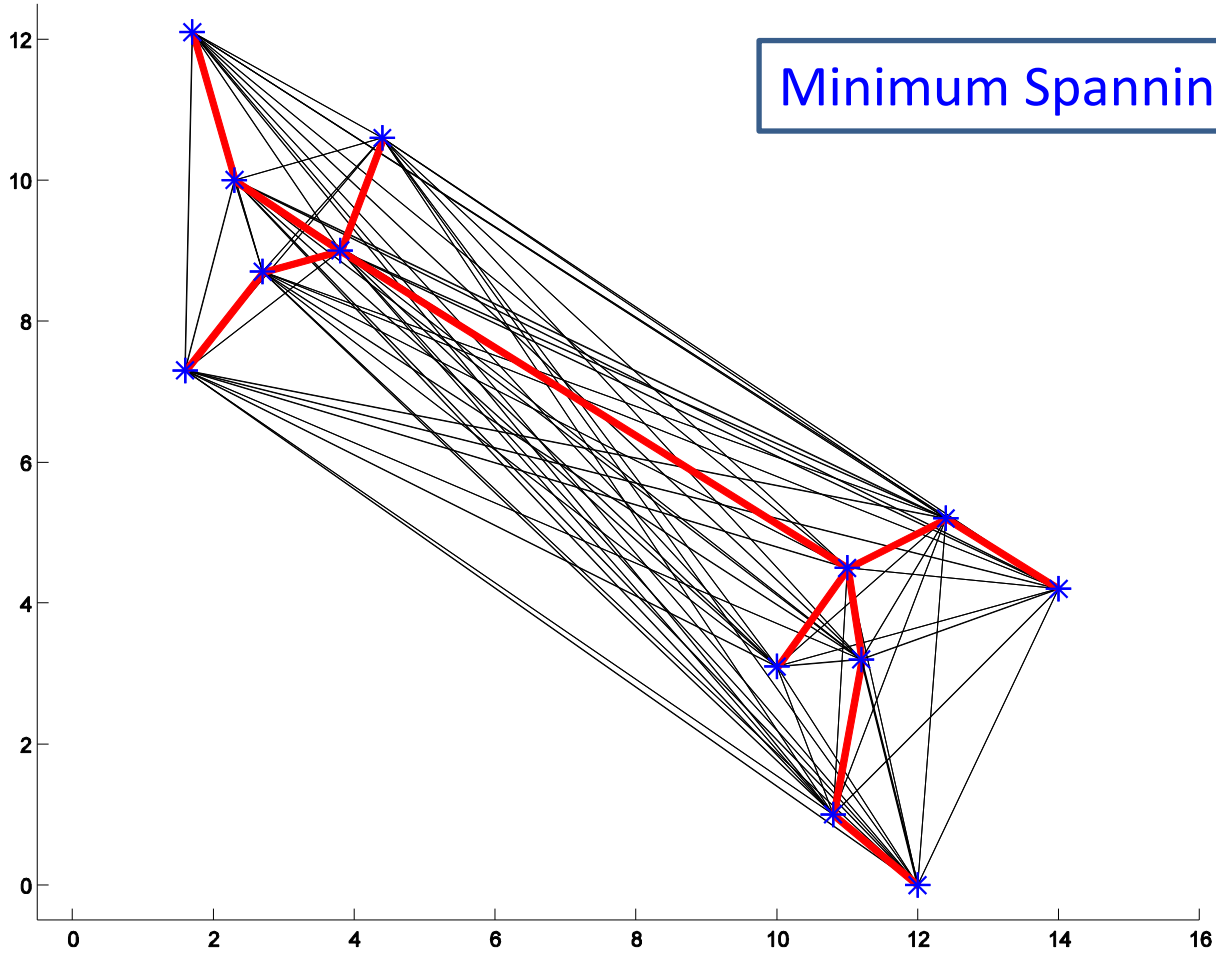


Minimum Spanning Tree (MST)

- **Define** a **complete** graph with vertices the data points and edges the segments connecting every pair of vertices.
- **Weight** each edge by the distance between its two end-points.
- **Define** the MST of the graph.

Agglomerative graph theory based Clustering Algorithms

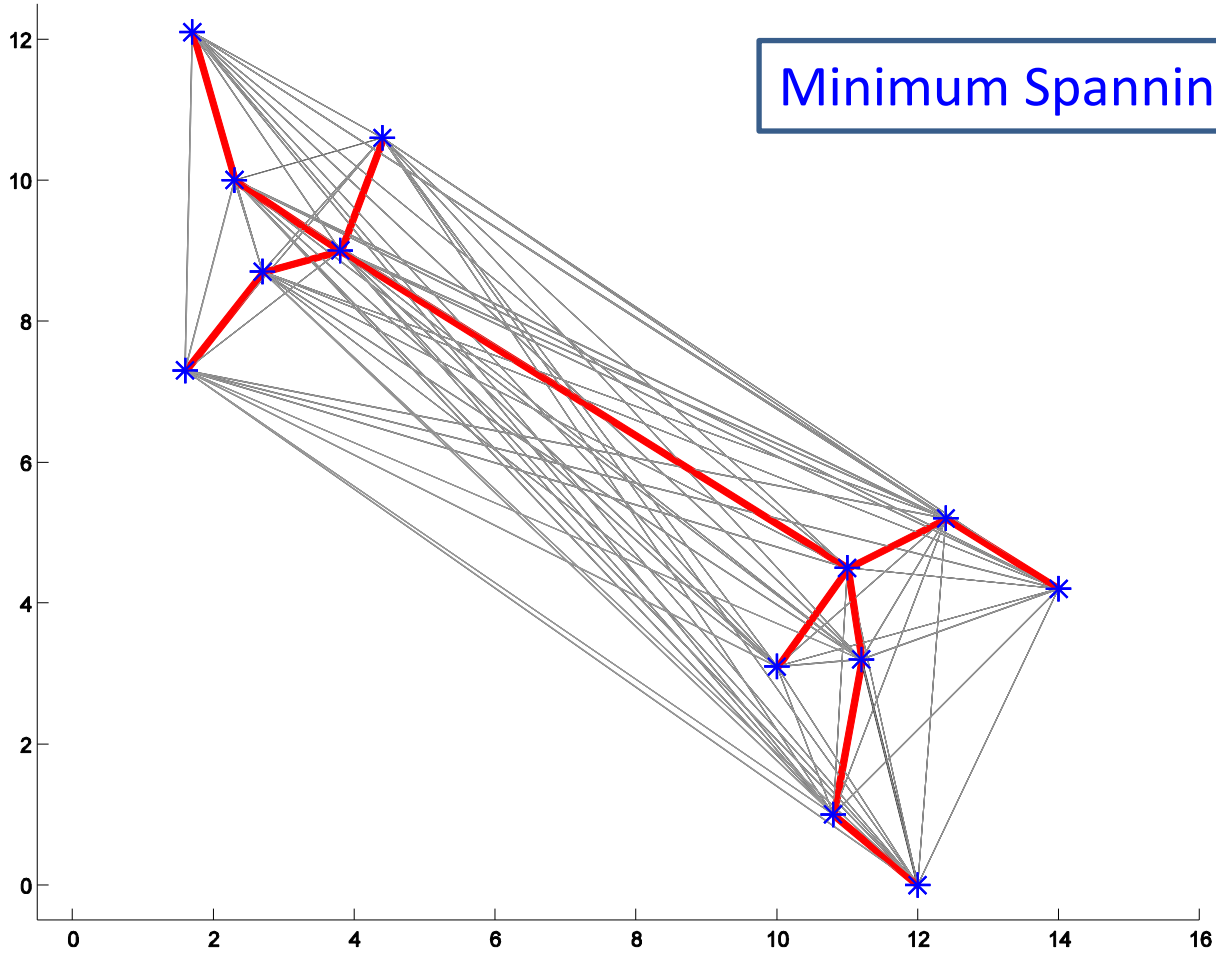
Example:



Minimum Spanning Tree (MST)

- **Define** a **complete** graph with vertices the data points and edges the segments connecting every pair of vertices.
- **Weight** each edge by the distance between its two end-points.
- **Define** the MST of the graph.

Agglomerative graph theory based Clustering Algorithms

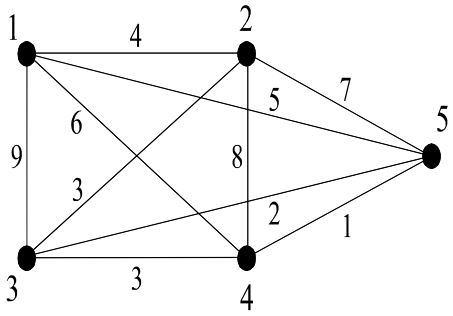


- **Define** a **complete** graph with vertices the data points and edges the segments connecting every pair of vertices.
- **Weight** each edge by the distance between its two end-points.
- **Define** the MST of the graph.

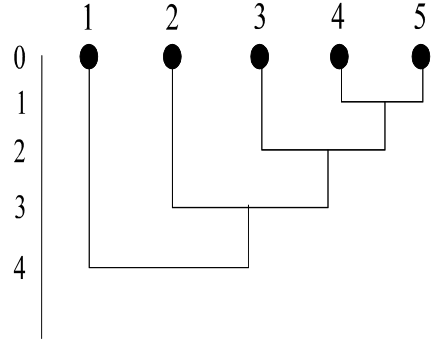
Agglomerative graph theory based Clustering Algorithms

- Ties in the proximity matrix
  - SL produces the same hierarchy of clusterings, independently of the order of consideration of edges with equal weights.
  - CL may produce different hierarchies, depending on the order of consideration of edges with equal weights.
  - The other graph theory-based algorithms behave as the CL.
  - The same trend appears in the matrix-based algorithms. In this case, *ties may appear at a later stage of the algorithm*.
- Example 6: Let

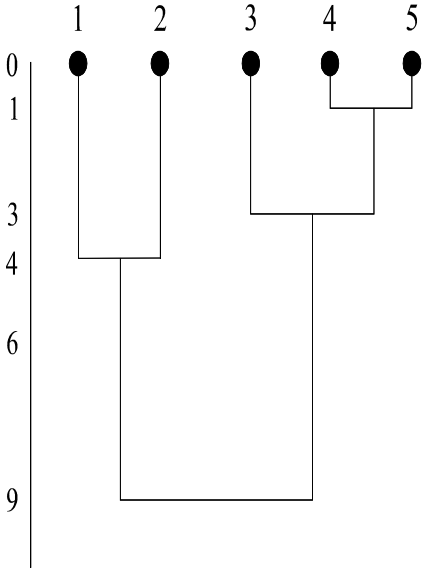
$$P = \begin{bmatrix} 0 & 4 & 9 & 6 & 5 \\ 4 & 0 & 3 & 8 & 7 \\ 9 & 3 & 0 & 3 & 2 \\ 6 & 8 & 3 & 0 & 1 \\ 5 & 7 & 2 & 1 & 0 \end{bmatrix}$$



(a)

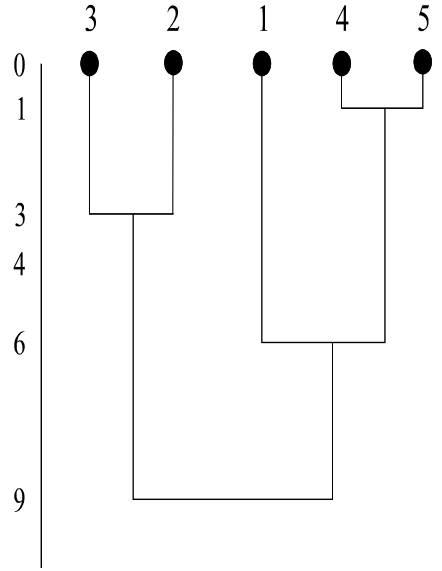


(b)



(c)

(CL(a))



(d)

(CL(b))

Note that $P(2,3)=P(3,4)$.

Agglomerative Clustering Algorithms: Cophenetic matrix

This is an alternative way to **represent** a hierarchical clustering.

Cophenetic distance between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$, $d_C(\boldsymbol{x}_i, \boldsymbol{x}_j)$: The proximity level, where $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ are found in the same cluster for the first time (distance metric).

Cophenetic matrix: An NxN matrix containing the **cophenetic distances** associated with all pairs of data vectors.

Example: Consider the following dissimilarity matrix (Euclidean distance)

$$P_0 = \begin{bmatrix} 0 & 1 & 2 & 26 & 37 \\ 1 & 0 & 3 & 25 & 36 \\ 2 & 3 & 0 & 16 & 25 \\ 26 & 25 & 16 & 0 & 1.5 \\ 37 & 36 & 25 & 1.5 & 0 \end{bmatrix}$$

The associated **cophenetic matrix** is

$$D_C = \begin{bmatrix} 0 & 1 & 2 & 16 & 16 \\ 1 & 0 & 2 & 16 & 16 \\ 2 & 2 & 0 & 16 & 16 \\ 16 & 16 & 16 & 0 & 1.5 \\ 16 & 16 & 16 & 1.5 & 0 \end{bmatrix}$$

The results of the single link algorithm are (in parenthesis the proximity level where the associated clustering has been formed):

$\mathcal{R}_0 = \{\{\underline{x}_1\}, \{\underline{x}_2\}, \{\underline{x}_3\}, \{\underline{x}_4\}, \{\underline{x}_5\}\}$, (**0**)

$\mathcal{R}_1 = \{\{\underline{x}_1, \underline{x}_2\}, \{\underline{x}_3\}, \{\underline{x}_4\}, \{\underline{x}_5\}\}$, (**1**)

$\mathcal{R}_2 = \{\{\underline{x}_1, \underline{x}_2\}, \{\underline{x}_3\}, \{\underline{x}_4, \underline{x}_5\}\}$, (**1.5**)

$\mathcal{R}_3 = \{\{\underline{x}_1, \underline{x}_2, \underline{x}_3\}, \{\underline{x}_4, \underline{x}_5\}\}$, (**2**)

$\mathcal{R}_4 = \{\{\underline{x}_1, \underline{x}_2, \underline{x}_3, \underline{x}_4, \underline{x}_5\}\}$, (**16**)

Divisive Clustering Algorithms

- Let $g(C_i, C_j)$ be a dissimilarity function between two clusters.
- Let C_{tj} denote the j-th cluster of the t-th clustering $\Re_t$, $t = 0, \ldots, N-1$, $j = 1, \ldots, t+1$.

Generalized Divisive Scheme (GDS)

- Initialization
  - Choose $\Re_0 = \{X\}$ as the initial clustering.
  - $t = 0$
- **Repeat**
  - $t = t + 1$
  - For $i = 1$ to t
    - **Among** all possible pairs of clusters (C_r, C_s) that form a partition of $C_{t-1,i}$, **find** the pair $(C^1{}_{t-1,i}, C^2{}_{t-1,i})$ that gives the max. value for g.
  - End for
  - From the t pairs defined in the previous step, choose the one that maximizes g. Suppose that this is $(C^1{}_{t-1,j}, C^2{}_{t-1,j})$.
  - The new clustering is:

$$\Re_t = (\Re_{t-1} - \{C_{t-1,j}\}) \cup \{C^1{}_{t-1,j}, C^2{}_{t-1,j}\}$$

  - **Relabel** the clusters of $\Re_t$.
- **Until** each vector lies in a single cluster.

Divisive Clustering Algorithms

Remarks:

- Different choices of g give rise to different algorithms.
- The GDS is computationally very demanding even for small N.
- Algorithms that rule out many partitions as not "reasonable", under a pre-specified criterion, have also been proposed.
- Algorithms where the splitting of the clusters is based on all features of the feature vectors are called polythetic algorithms. Otherwise, if the splitting is based on a single feature at each step, the algorithms are called monothetic algorithms.

Choice of the best number of clusters

A major issue associated with hierarchical algorithms is:

"How the clustering that best fits the data is extracted from a hierarchy of clusterings?"

Some **approaches**:

- Search in the proximity dendrogram for clusters that have a large lifetime (the difference between the proximity level at which a cluster is created and the proximity level at which it is absorbed into a larger cluster (however, this method involves human subjectivity)).

- Define a function $h(C)$ that measures the dissimilarity between the vectors of the same cluster C ("self-similarity"). Then, we have two alternatives:
  - Let ϑ be an appropriate threshold for $h(C)$. Then *$\mathcal{R}_t$ is the final clustering if there exists a cluster C in $\mathcal{R}_{t+1}$ with dissimilarity between its vectors ($h(C)$) greater than ϑ* (extrinsic method).
  - The final clustering $\mathcal{R}_t$ must satisfy the following condition:

$$d^{ss}{}_{min}(C_i,C_j) > max\{h(C_i),h(C_j)\}, \quad \forall C_i,C_j \in \mathcal{R}_t$$

    In words, in the final clustering, the dissimilarity between every pair of clusters is larger than the "self-similarity" of each one of them (intrinsic method).

Hierarchical Algorithms for large data sets

Remark:

Since the number of operations required by GAS is greater than $O(N^2)$, algorithms resulting by GAS are prohibitive for very large data sets encountered, for example, in web mining and bioinformatics. To overcome this drawback, various hierarchical algorithms of special type have been developed that are tailored to handle large data sets.

Typical examples are:

- The CURE algorithm.
- The ROCK algorithm.
- The Chameleon algorithm.

The CURE (Clustering Using Representatives) algorithm

In CURE:

- Each cluster C is represented by a set, R_C, of $k > 1$ representatives.
- These representatives try to "**capture**" the "shape " of the cluster.
- They are **chosen** at the "border" of the cluster and then, they are **pushed** toward its mean, in order to discard the irregularities of the border.

- **Determination** of R_C:
  - Select $\boldsymbol{x} \in C$, with the maximum distance from the mean $\boldsymbol{m}_C$ of C and set $R_C = \{\boldsymbol{x}\}$
  - For $i=2$ to $\min\{k, n_C\}$ (n_C is the number of points in C)
    - Determine $\boldsymbol{y} \in C - R_C$ that lies farthest from the points of R_C and set $R_C = R_C \cup \{\boldsymbol{y}\}$.
  - Shrink the points $\underline{x} \in R_C$ toward the mean $\boldsymbol{m}_C$ in C by a factor a. That is $\boldsymbol{x} = (1-a)\,\boldsymbol{x} + a\,\boldsymbol{m}_C$, $\forall \boldsymbol{x} \in R_C$.

CURE is a **special case** of GAS (single link) where the distance between two clusters is defined as: $d(C_i, C_j) = min_{\boldsymbol{x} \in R_{C_i}, \boldsymbol{y} \in R_{C_j}} d(\boldsymbol{x}, \boldsymbol{y})$

The CURE (Clustering Using Representatives) algorithm

Clustering Using REpresentatives ($CURE(X)$)

- Initialization
  - **Choose** $\Re_0 = \{\{\boldsymbol{x}_1\}, \ldots, \{\boldsymbol{x}_N\}\}$
  - $t = 0$
- **Repeat**
  - $t = t + 1$
  - **Choose** (C_i, C_j) in $\Re_{t-1}$ such that

$$d(C_i, C_j) = min_{\boldsymbol{x} \in R_{C_i}, \boldsymbol{y} \in R_{C_j}} d(\boldsymbol{x}, \boldsymbol{y})$$

  - **Define** $C_q = C_i \cup C_j$ and produce $\Re_t = \left(\Re_{t-1} - \{C_i, C_j\}\right) \cup \{C_q\}$
  - **Determine** R_{C_q} (*)
- **Until** all vectors lie in a single cluster.

(*) The **determination** of R_{C_q} may be conducted:

(i) Either by performing the procedure of the previous slide taking into account **all the data points** of C_q (more accurate approach).

(ii) Or by performing the procedure of the previous slide taking into account the **data points** in $R_{C_i} \cup R_{C_j}$ (the union of the representatives of the clusters that constitute C_q) (faster approach).

The CURE (Clustering Using Representatives) algorithm

- Worst case time complexity for CURE: $O(N^2 log_2 N)$.
- This is prohibitive for very large data sets.
- Solution: Adoption of the random sampling technique.
  The size N' of a sample data set X', created from X, via random sampling, should be sufficiently large in order to ensure that the probability of missing a cluster due to sampling is low.

CURE utilizing random sampling

- Identification of clusters
  - Partition randomly X into $p=N/N'$ sample data sets.
  - For each one of the p sample data sets.
    - Apply the original version of CURE, until N'/q clusters (at the most) are formed (q is user-defined).
  - Consider all the above $p * (N'/q)$ clusters (at the most) and apply the original CURE until the required number of clusters, m, is formed.
- Assignment of points to clusters
  - For each of the m clusters select a random sample of k representative points.
  - Assign each point $\boldsymbol{x}$ that is not cluster representative to the cluster that contains the representative closest to it.

The CURE (Clustering Using Representatives) algorithm

Clustering Using Representatives- Random Sampling (*CURE-RS*(X))

Identification of clusters

- **Partition** randomly X into $p=N/N'$ sample data sets, $X_1, X_2, \ldots, X_p$.
- For $i=1$ to p
  - **Run** *CURE-RS*(X_i) and **return** the ${\Re_k}^i$ clustering with N'/q clusters (at the most (q is user-defined).
- End – For
- **Set** $X' = {\Re_k}^1 \cup {\Re_k}^2 \cup \cdots \cup {\Re_k}^p$
- **Run** *CURE*(X') and determine the most appropriate clustering ${\Re_m}'$.

The algorithm starts from the $\Re'_{p*(\frac{N'}{q})}$ and ends with the ${\Re_m}'$ clustering ($m<p*k$)

Assignment of points to clusters

- For each of the m clusters of ${\Re_m}'$ **select** a random sample of k representative points.
- **Assign** each point x that is not cluster representative to the cluster that contains the representative closest to it.

The CURE (Clustering Using Representatives) algorithm

Remarks:

- CURE is sensitive to the parameters k, N', a. Specifically:
  - k must be large enough to capture the geometry of each cluster.
  - N' must be higher than a certain percentage of N (typically $N' \geq 2.5\%\ N$)
  - For small a CURE **behaves** like the single-link algorithm, while for large a it resembles the algorithms that use a single point representative for each cluster.
- Worst case time complexity for CURE using random sampling: $O(N'^2 log_2 N')$
- The algorithm exhibits low sensitivity with respect to outliers within the clusters.
- A few stages before its termination, CURE **checks** for clusters containing very few data points and removes them (since they are likely to be outliers).
- If N'/q is sufficiently large, compared to m, it is expected that the partition of X will not affect significantly the final clustering obtained by CURE.
- CURE can, in principle, **reveal** clusters of non-spherical or elongated shapes, as well as clusters of wide variance in size.
- CURE can be implemented efficiently using the *heap* and the *k-d tree* data structures.