In statistics, **single-linkage clustering** is one of several methods of hierarchical clustering. It is based on grouping clusters in bottom-up fashion (agglomerative clustering), at each step combining two clusters that contain the closest pair of elements not yet belonging to the same cluster as each other.

A drawback of this method is that it tends to produce long thin clusters in which nearby elements of the same cluster have small distances, but elements at opposite ends of a cluster may be much farther from each other than two elements of other clusters. This may lead to difficulties in defining classes that could usefully subdivide the data.[1]

Overview of agglomerative clustering methods

In the beginning of the agglomerative clustering process, each element is in a cluster of its own. The clusters are then sequentially combined into larger clusters, until all elements end up being in the same cluster. At each step, the two clusters separated by the shortest distance are combined. The definition of 'shortest distance' is what differentiates between the different agglomerative clustering methods.

In single-linkage clustering, the distance between two clusters is determined by a single element pair, namely those two elements (one in each cluster) that are closest to each other. The shortest of these links that remains at any step causes the fusion of the two clusters whose elements are involved. The method is also known as **nearest neighbour clustering**. The result of the clustering can be visualized as a dendrogram, which shows the sequence of cluster fusion and the distance at which each fusion took place.[2]

Mathematically, the linkage function – the distance $D(X,Y)$ between clusters X and Y – is described by the expression

$$D(X,Y) = \min_{x \in X, y \in Y} d(x,y),$$

where X and Y are any two sets of elements considered as clusters, and $d(x,y)$ denotes the distance between the two elements x and y.

Naive algorithm

The following algorithm is an agglomerative scheme that erases rows and columns in a proximity matrix as old clusters are merged into new ones. The $N \times N$ proximity matrix D contains all distances $d(i,j)$. The clusterings are assigned sequence numbers 0,1,....., $(n-1)$ and $L(k)$ is the level of the kth clustering. A cluster with sequence number m is denoted (m) and the proximity between clusters (r) and (s) is denoted $d[(r),(s)]$.

The algorithm is composed of the following steps:

1. Begin with the disjoint clustering having level $L(0) = 0$ and sequence number m = 0.

2. Find the most similar pair of clusters in the current clustering, say pair (r), (s), according to $d[(r),(s)] = \min d[(i),(j)]$ where the minimum is over all pairs of clusters in the current clustering.

3. Increment the sequence number: $m = m + 1$. Merge clusters (r) and (s) into a single cluster to form the next clustering m. Set the level of this clustering to $L(m) = d[(r),(s)]$

4. Update the proximity matrix, *D*, by deleting the rows and columns corresponding to clusters (*r*) and (*s*) and adding a row and column corresponding to the newly formed cluster. The proximity between the new cluster, denoted (*r*,*s*) and old cluster (*k*) is defined as *d*[(*k*), (*r*,*s*)] = **min *d*[(*k*),(*r*)], *d*[(*k*),(*s*)]**.

5. If all objects are in one cluster, stop. Else, go to step 2.

Working example

This working example is based on a JC69 genetic distance matrix computed from the 5S ribosomal RNA sequence alignment of five bacteria: *Bacillus subtilis* ( ), *Bacillus stearothermophilus* ( ), *Lactobacillus viridescens* ( ), *Acholeplasma modicum* ( ), and *Micrococcus luteus* ( ).[3][4]

First step

- **First clustering**

Let us assume that we have five elements and the following matrix of pairwise distances between them:

	a	b	c	d	e
a	0	17	21	31	23
b	17	0	30	34	21
c	21	30	0	28	39
d	31	34	28	0	43
e	23	21	39	43	0

In this example, is the lowest value of , so we cluster elements and .

- **First branch length estimation**

Let denote the node to which and are now connected. Setting ensures that elements and are equidistant from . This corresponds to the expectation of the ultrametricity hypothesis. The branches joining and to then have lengths (*see the final dendrogram*)

- **First distance matrix update**

We then proceed to update the initial proximity matrix into a new proximity matrix (see below), reduced in size by one row and one column because of the clustering of with . Bold values in correspond to the new distances, calculated by retaining the **minimum distance** between each element of the first cluster and each of the remaining elements:

Italicized values in are not affected by the matrix update as they correspond to distances between elements not involved in the first cluster.

Second step

- **Second clustering**

We now reiterate the three previous actions, starting from the new distance matrix :

	(a,b)	c	d	e
(a,b)	0	**21**	**31**	**21**
c	**21**	0	*28*	*39*
d	**31**	*28*	0	*43*
e	**21**	*39*	*43*	0

Here, and are the lowest values of , so we join cluster with element and with element .

- **Second branch length estimation**

Let denote the node to which , and are now connected. Because of the ultrametricity constraint, the branches joining or to , and to , and also to are equal and have the following total length:

We deduce the missing branch length:
(*see the final dendrogram*)

- **Second distance matrix update**

We then proceed to update the matrix into a new distance matrix (see below), reduced in size by two rows and two columns because of the clustering of with and with :

Final step

The final matrix is:

	((a,b),c,e)	d
((a,b),c,e)	0	**28**
d	**28**	0

So we join clusters and .

Let denote the (root) node to which and are now connected. The branches joining and to then have lengths:

We deduce the remaining branch length:

The single-linkage dendrogram

The dendrogram is now complete. It is ultrametric because all tips (, , , , and) are equidistant from :

The dendrogram is therefore rooted by , its deepest node.

Other linkages

The naive algorithm for single linkage clustering is essentially the same as Kruskal's algorithm for minimum spanning trees. However, in single linkage clustering, the order in which clusters are formed is important, while for minimum spanning trees what matters is the set of pairs of points that form distances chosen by the algorithm.

Alternative linkage schemes include complete linkage clustering, average linkage clustering (UPGMA and WPGMA), and Ward's method. In the naive algorithm for agglomerative clustering, implementing a different linkage scheme may be accomplished simply by using a different formula to calculate inter-cluster distances in the algorithm. The formula that should be adjusted has been highlighted using bold text in the above algorithm description. However, more efficient algorithms such as the one described below do not generalize to all linkage schemes in the same way.

Faster algorithms

The naive algorithm for single-linkage clustering is easy to understand but slow, with time complexity .[5] In 1973, R. Sibson proposed an algorithm with time complexity and space complexity (both optimal) known as SLINK. The slink algorithm represents a clustering on a set of numbered items by two functions. These functions are both determined by finding the smallest cluster that contains both item and at least one larger-numbered item. The first function, , maps item to the largest-numbered item in cluster . The second function, , maps item to the distance associated with the creation of cluster . Storing these functions in two arrays that map each item number to its function value takes space , and this information is sufficient to determine the clustering itself. As Sibson shows, when a new item is added to the set of items, the updated functions representing the new single-linkage clustering for the augmented set, represented in the same way, can be constructed from the old clustering in time . The SLINK algorithm then loops over the items, one by one, adding them to the representation of the clustering.[6][7]

An alternative algorithm, running in the same optimal time and space bounds, is based on the equivalence between the naive algorithm and Kruskal's algorithm for minimum spanning trees. Instead of using Kruskal's algorithm, one can use Prim's algorithm, in a variation without binary heaps that takes time and space to construct the minimum spanning tree (but not the clustering) of the given items and distances. Then, applying Kruskal's algorithm to the sparse graph formed by the edges of the minimum spanning tree produces the clustering itself in an additional time and space .[8]

See also

- Cluster analysis
- Complete-linkage clustering
- Hierarchical clustering
- Molecular clock
- Neighbor-joining
- UPGMA
- WPGMA

References

1. Everitt B (2011). *Cluster analysis*. Chichester, West Sussex, U.K: Wiley. ISBN 9780470749913.

2. Legendre P, Legendre L (1998). *Numerical Ecology*. Developments in Environmental Modelling. **20** (Second English ed.). Amsterdam: Elsevier.

3. Erdmann VA, Wolters J (1986). "Collection of published 5S, 5.8S and 4.5S ribosomal RNA sequences" . *Nucleic Acids Research*. 14 Suppl (Suppl): r1–59. PMC 341310 . PMID 2422630 .

4. Olsen GJ (1988). "Phylogenetic analysis using ribosomal RNA". *Methods in Enzymology*. **164**: 793–812. PMID 3241556 .

5. Murtagh F, Contreras P (2012). "Algorithms for hierarchical clustering: an overview". *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*. Wiley Online Library. **2** (1): 86–97. doi:10.1002/widm.53 .

6. Sibson R (1973). "SLINK: an optimally efficient algorithm for the single-link cluster method" (PDF). *The Computer Journal*. British Computer Society. **16** (1): 30–34. doi:10.1093/comjnl/16.1.30 .

7. Gan G (2007). *Data clustering : theory, algorithms, and applications*. Philadelphia, Pa. Alexandria, Va: SIAM, Society for Industrial and Applied Mathematics American Statistical Association. ISBN 9780898716238.

8. Gower JC, Ross GJ (1969). "Minimum spanning trees and single linkage cluster analysis". *Journal of the Royal Statistical Society, Series C*. **18** (1): 54–64. JSTOR 2346439 . MR 0242315 ..

External links

- Linkages used in Matlab

Last edited 2 months ago by Boghog