Unsupervised Learning

Hierarchical Clustering

Richard Röttger

University of Southern Denmark

February 13, 2017

Objectives for Today

- Introduction into hierarchical clustering
- Discussion of the most common linking functions

- We will elaborate on:
  - Strengths/Weaknesses of different linking functions
  - Strengths/Weaknesses of agglomerative or divisive
  - Telling apart a good or bad hierarchical clustering

- Hierarchical clustering for large-scale datasets

- Numerous slides are based on the lecture "Cluster Analysis" of Prof. Enza Messina

Hierarchical Clustering

- In a hierarchical classification the data are not partitioned into a particular number of classes or clusters at a single step
- Instead the classification consists of a series of partitions
  - They run from a single cluster containing all individuals ...
  - ... to singletons, clusters containing a single object
- Remember our definition of the hierarchical clustering:

Definition

Hierarchical clustering builds a nested structural partition $C = \{C_1, \ldots, C_K\}$ of V such that $\bigcup_{i=1}^{K} C_i = V$ and $C_i \neq \emptyset$ for $i = 1, \ldots, K$. Furthermore, for each pair C_i, C_j of clusters with $i, j = 1, \ldots, K$ and $i \neq j$, exactly one of the following conditions holds:

- $C_i \cap C_j = \emptyset$
- $C_i \subset C_j$
- $C_j \subset C_i$

Hierarchical Clustering

- In a hierarchical classification the data are not partitioned into a particular number of classes or clusters at a single step
- Instead the classification consists of a series of partitions
  - They run from a single cluster containing all individuals ...
  - ... to singletons, clusters containing a single object
- Remember our definition of the hierarchical clustering:

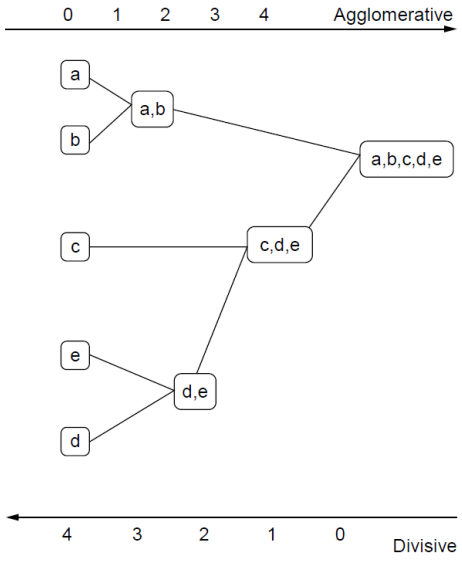
Definition

Hierarchical clustering builds a nested structural partition $C = \{C_1, \ldots, C_K\}$ of V such that $\bigcup_{i=1}^{K} C_i = V$ and $C_i \neq \emptyset$ for $i = 1, \ldots, K$. Furthermore, for each pair C_i, C_j of clusters with $i, j = 1, \ldots, K$ and $i \neq j$, exactly one of the following conditions holds:

- $C_i \cap C_j = \emptyset$
- $C_i \subset C_j$
- $C_j \subset C_i$

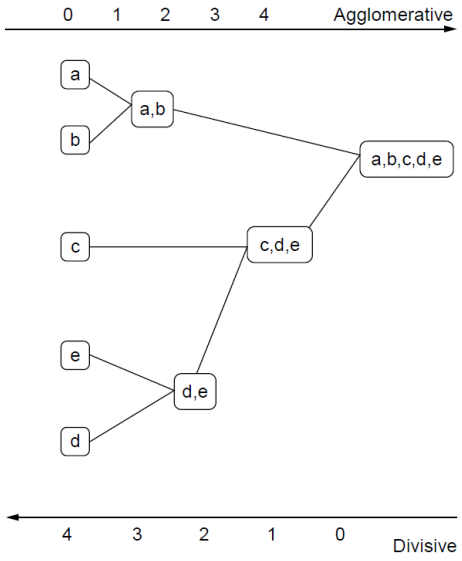
Agglomerative vs. Divisive

- Two different kinds of methods:
  - Agglomerative
  - Divisive
- In order to receive a partitional clustering, a cut-off at a certain level has to be set
- Decisions of the clustering are irrevocable; e.g., a splitting cannot be undone



Agglomerative vs. Divisive

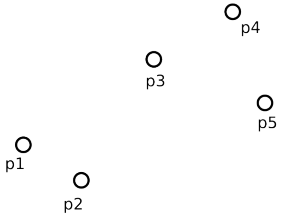
- Two different kinds of methods:
  - Agglomerative
  - Divisive
- In order to receive a partitional clustering, a cut-off at a certain level has to be set
- Decisions of the clustering are irrevocable; e.g., a splitting cannot be undone



Hierarchical Clustering

Agglomerative Methods

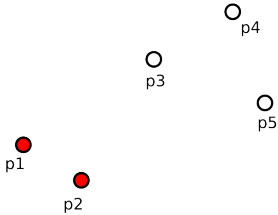
Small Example



$$D = \begin{array}{c} \\ p_1 \\ p_2 \\ p_3 \\ p_4 \\ p_5 \end{array} \begin{array}{c} \begin{array}{ccccc} p_1 & p_2 & p_3 & p_4 & p_5 \end{array} \\ \begin{pmatrix} 0.0 & & & & \\ 2.0 & 0.0 & & & \\ 6.0 & 5.0 & 0.0 & & \\ 10.0 & 9.0 & 4.0 & 0.0 & \\ 9.0 & 8.0 & 5.0 & 3.0 & 0.0 \end{pmatrix} \end{array}$$

- According to the distance matrix D, p_1 and p_2 should be joined
- To do that, we now need to define the distances from the cluster (p_1, p_2)
- The simplest method is to define the difference as the minimal difference:
  - $d((p_1, p_2), p_3) = \min\{d(p_1, p_3), d(p_2, p_3)\} = 5.0$
  - $d((p_1, p_2), p_3) = \min\{d(p_1, p_4), d(p_2, p_4)\} = 9.0$
  - $d((p_1, p_2), p_3) = \min\{d(p_1, p_5), d(p_2, p_5)\} = 8.0$

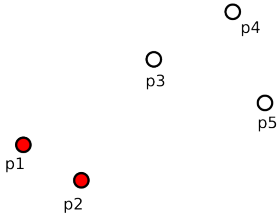
Small Example



$$D = \begin{array}{c} \\ p_1 \\ p_2 \\ p_3 \\ p_4 \\ p_5 \end{array} \begin{array}{c} \begin{array}{ccccc} p_1 & p_2 & p_3 & p_4 & p_5 \end{array} \\ \begin{pmatrix} 0.0 & & & & \\ 2.0 & 0.0 & & & \\ 6.0 & 5.0 & 0.0 & & \\ 10.0 & 9.0 & 4.0 & 0.0 & \\ 9.0 & 8.0 & 5.0 & 3.0 & 0.0 \end{pmatrix} \end{array}$$

- According to the distance matrix D, p_1 and p_2 should be joined
- To do that, we now need to define the distances from the cluster (p_1, p_2)
- The simplest method is to define the difference as the minimal difference:
  - $d((p_1, p_2), p_3) = \min\{d(p_1, p_3), d(p_2, p_3)\} = 5.0$
  - $d((p_1, p_2), p_3) = \min\{d(p_1, p_4), d(p_2, p_4)\} = 9.0$
  - $d((p_1, p_2), p_3) = \min\{d(p_1, p_5), d(p_2, p_5)\} = 8.0$

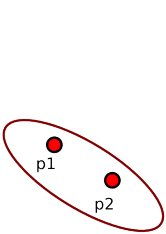
Small Example



$$D = \begin{array}{c} \\ p_1 \\ p_2 \\ p_3 \\ p_4 \\ p_5 \end{array} \begin{array}{c} \begin{array}{ccccc} p_1 & p_2 & p_3 & p_4 & p_5 \end{array} \\ \begin{pmatrix} 0.0 & & & & \\ 2.0 & 0.0 & & & \\ 6.0 & 5.0 & 0.0 & & \\ 10.0 & 9.0 & 4.0 & 0.0 & \\ 9.0 & 8.0 & 5.0 & 3.0 & 0.0 \end{pmatrix} \end{array}$$

- According to the distance matrix D, p_1 and p_2 should be joined
- To do that, we now need to define the distances from the cluster (p_1, p_2)
- The simplest method is to define the difference as the minimal difference:
  - $d((p_1, p_2), p_3) = \min\{d(p_1, p_3), d(p_2, p_3)\} = 5.0$
  - $d((p_1, p_2), p_3) = \min\{d(p_1, p_4), d(p_2, p_4)\} = 9.0$
  - $d((p_1, p_2), p_3) = \min\{d(p_1, p_5), d(p_2, p_5)\} = 8.0$

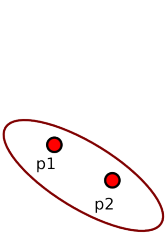
Small Example



$$D = \begin{array}{c} \\ (p_1,p_2) \\ p_3 \\ p_4 \\ p_5 \end{array} \begin{array}{c} \begin{array}{cccc} (p_1,p_2) & p_3 & p_4 & p_5 \end{array} \\ \begin{pmatrix} 0.0 & & & \\ 5.0 & 0.0 & & \\ 9.0 & 4.0 & 0.0 & \\ 8.0 & 5.0 & 3.0 & 0.0 \end{pmatrix} \end{array}$$

- According to the distance matrix D, p_1 and p_2 should be joined
- To do that, we now need to define the distances from the cluster (p_1, p_2)
- The simplest method is to define the difference as the minimal difference:
  - $d((p_1,p_2),p_3) = \min\{d(p_1,p_3), d(p_2,p_3)\} = 5.0$
  - $d((p_1,p_2),p_3) = \min\{d(p_1,p_4), d(p_2,p_4)\} = 9.0$
  - $d((p_1,p_2),p_3) = \min\{d(p_1,p_5), d(p_2,p_5)\} = 8.0$

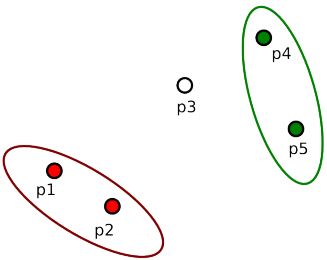
Small Example



$$D = \begin{array}{c} \\ (p_1,p_2) \\ p_3 \\ p_4 \\ p_5 \end{array} \begin{array}{c} \begin{array}{cccc} (p_1,p_2) & p_3 & p_4 & p_5 \end{array} \\ \begin{pmatrix} 0.0 & & & \\ 5.0 & 0.0 & & \\ 9.0 & 4.0 & 0.0 & \\ 8.0 & 5.0 & 3.0 & 0.0 \end{pmatrix} \end{array}$$

- Now, clearly p_4 and p_5 should be joined.
- After analyzing the new, (p_4, p_5) should be joined with p_3
- Now, we have only two clusters left, which are joined in the last step
- The structure can best be displayed as a dendrogram

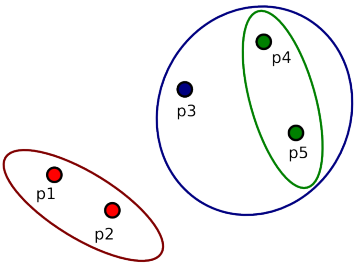
Small Example



$$D = \begin{array}{c} \\ (p_1, p_2) \\ p_3 \\ (p_4, p_5) \end{array} \begin{array}{c} \begin{array}{ccc} (p_1, p_2) & p_3 & (p_4, p_5) \end{array} \\ \begin{pmatrix} 0.0 & & \\ 5.0 & 0.0 & \\ 8.0 & 4.0 & 0.0 \end{pmatrix} \end{array}$$

- Now, clearly p_4 and p_5 should be joined.
- After analyzing the new, (p_4, p_5) should be joined with p_3
- Now, we have only two clusters left, which are joined in the last step
- The structure can best be displayed as a dendrogram

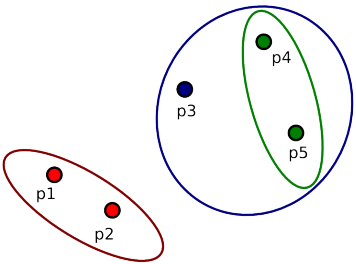
Small Example



$$D = \begin{array}{c} \\ (p_1, p_2) \\ (p_3, (p_4, p_5)) \end{array} \begin{array}{c} \begin{array}{cc} (p_1, p_2) & (p_3, (p_4, p_5)) \end{array} \\ \begin{pmatrix} 0.0 & \\ 5.0 & 0.0 \end{pmatrix} \end{array}$$

- Now, clearly p_4 and p_5 should be joined.
- After analyzing the new, (p_4, p_5) should be joined with p_3
- Now, we have only two clusters left, which are joined in the last step
- The structure can best be displayed as a dendrogram

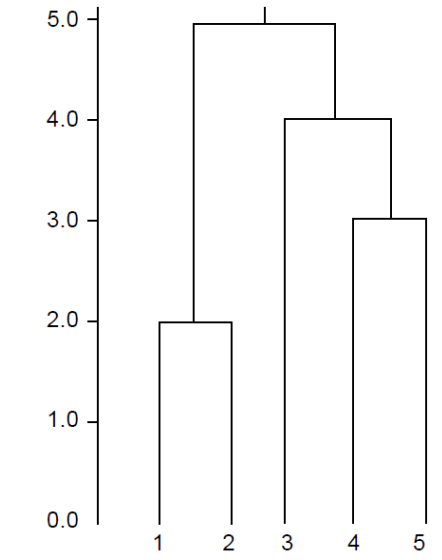
Small Example



$$D = \begin{array}{c} \\ (p_1, p_2) \\ (p_3, (p_4, p_5)) \end{array} \begin{array}{c} \begin{array}{cc} (p_1, p_2) & (p_3, (p_4, p_5)) \end{array} \\ \begin{pmatrix} 0.0 & \\ 5.0 & 0.0 \end{pmatrix} \end{array}$$

- Now, clearly p_4 and p_5 should be joined.
- After analyzing the new, (p_4, p_5) should be joined with p_3
- Now, we have only two clusters left, which are joined in the last step
- The structure can best be displayed as a dendrogram

Small Example



Partition	Members
P5	[1 2 3 4 5]
P4	[1 2], [3 4 5]
P3	[1 2], [3], [4 5]
P2	[1 2], [3], [4], [5]
P1	[1], [2], [3], [4], [5]

Small Example: Recap

- Properties of a dendrogram:
  - The hight of the dendrogram represents the distance at which each fusion was made
  - We will learn more about these later
- The method we have applied is the so-called single-linkage clustering
  - Very easy to calculate (just look for the min)
  - Only the distance matrix is required
  - But how about outliers, oddly shaped clusters, etc?
- There a plenty of different methods in order to decide where to split
- The following slides are taken from `http://www.cse.buffalo.edu/~jing/cse601/fa12/`

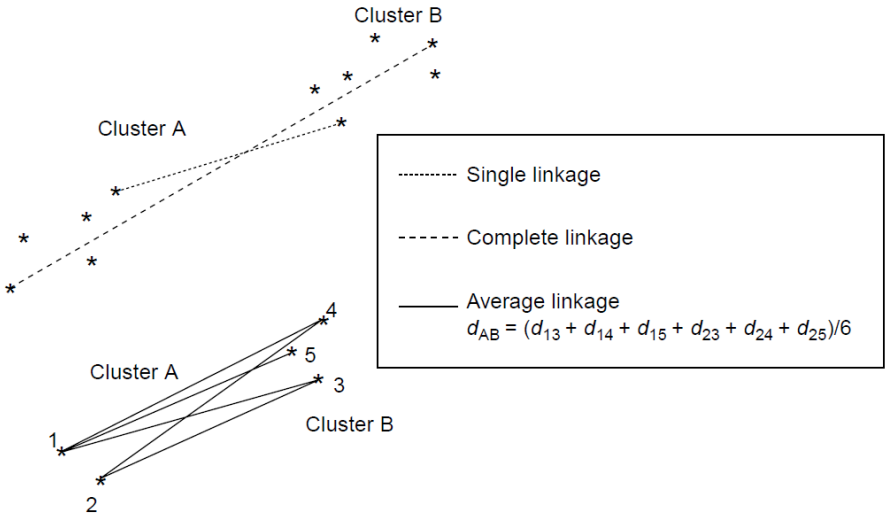
Small Example: Recap

- Properties of a dendrogram:
  - The hight of the dendrogram represents the distance at which each fusion was made
  - We will learn more about these later
- The method we have applied is the so-called single-linkage clustering
  - Very easy to calculate (just look for the min)
  - Only the distance matrix is required
  - But how about outliers, oddly shaped clusters, etc?
- There a plenty of different methods in order to decide where to split
- The following slides are taken from `http://www.cse.buffalo.edu/~jing/cse601/fa12/`

Small Example: Recap

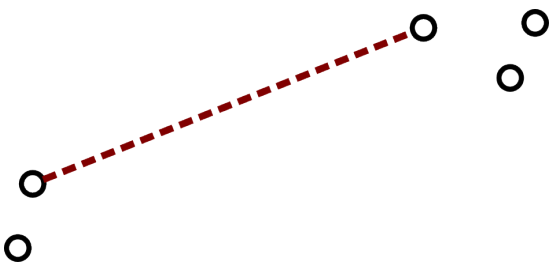
- Properties of a dendrogram:
  - The hight of the dendrogram represents the distance at which each fusion was made
  - We will learn more about these later
- The method we have applied is the so-called single-linkage clustering
  - Very easy to calculate (just look for the min)
  - Only the distance matrix is required
  - But how about outliers, oddly shaped clusters, etc?
- There a plenty of different methods in order to decide where to split
- The following slides are taken from `http://www.cse.buffalo.edu/~jing/cse601/fa12/`

Most commonly used linkage functions



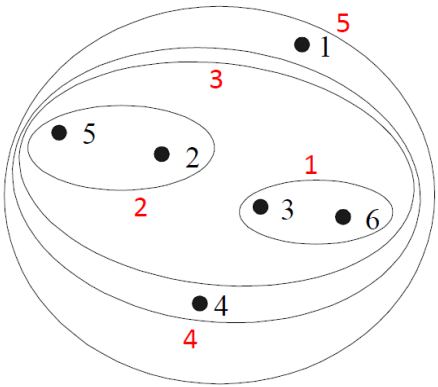
Single Linkage

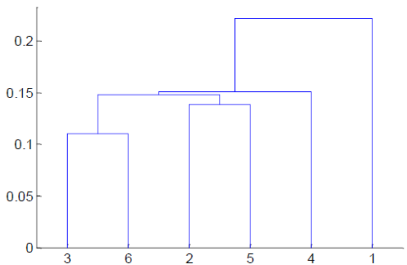
- The distance between two clusters is represented by the distance of the closest pair of data objects belonging to different clusters.
- Determined by one pair of points, i.e., by one link in the proximity graph



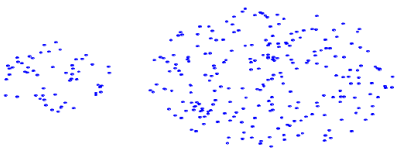
$$d(C_i, C_j) = \min_{p \in C_i, q \in C_j} d(p, q)$$

Small Example

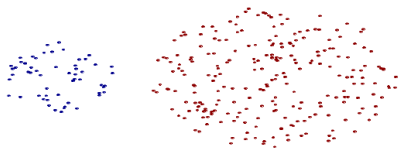




Strength of Single Linkage Clustering



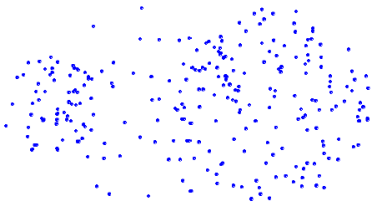
Original Points



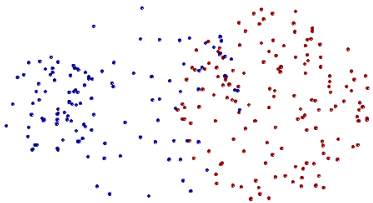
Two Clusters

- Can handle non-spherical shaped clusters

Limitations of Single Linkage Clustering



Original Points

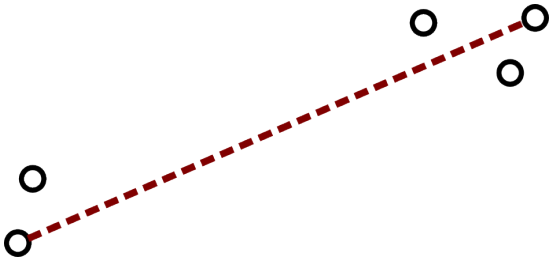


Two Clusters

- Sensitive to outliers and noise

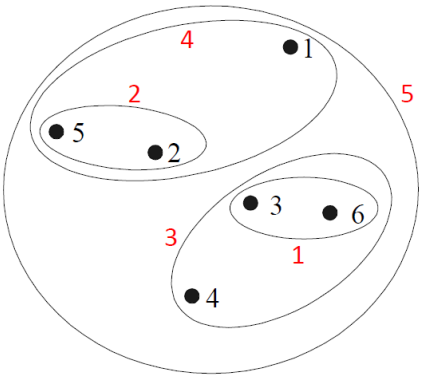
Complete Linkage

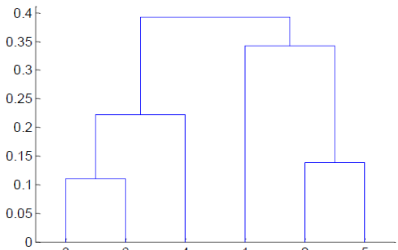
- The distance between two clusters is represented by the distance of the farthest pair of data objects belonging to different clusters



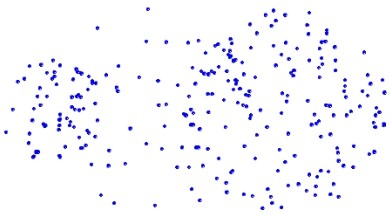
$$d(C_i, C_j) = \max_{p \in C_i, q \in C_j} d(p, q)$$

Small Example

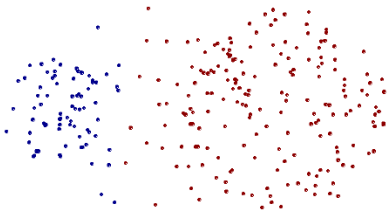




Strength of Complete Linkage Clustering



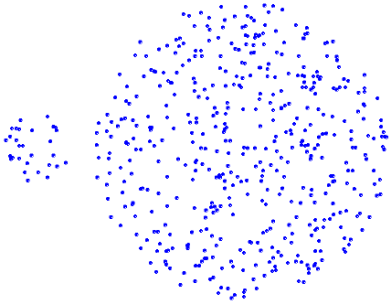
Original Points

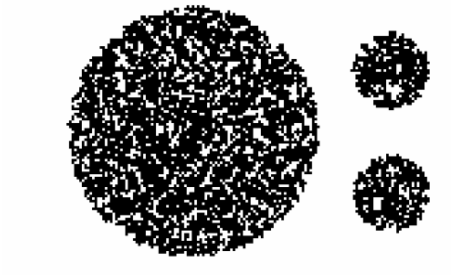


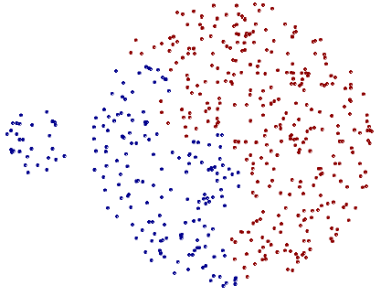
Two Clusters

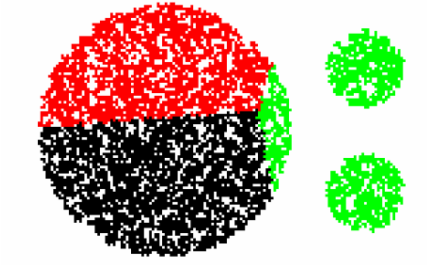
- Less susceptible to noise and outliers

Limitations of Complete Linkage Clustering

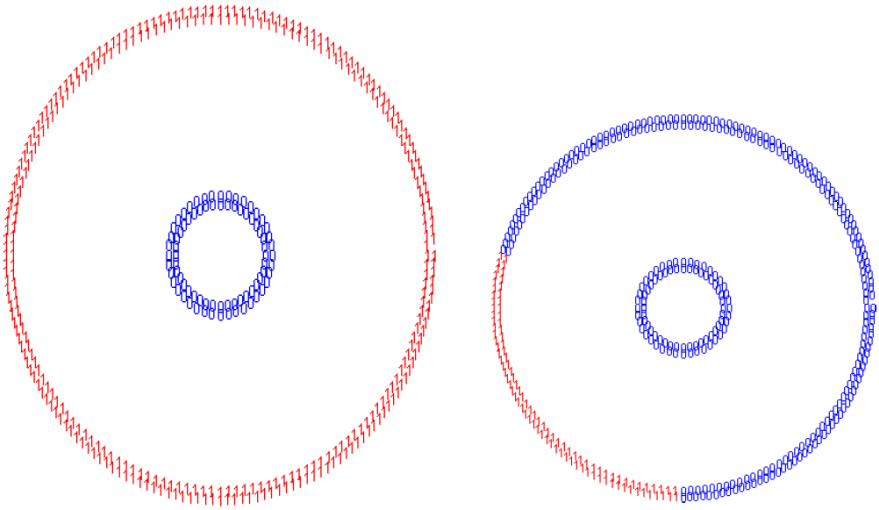






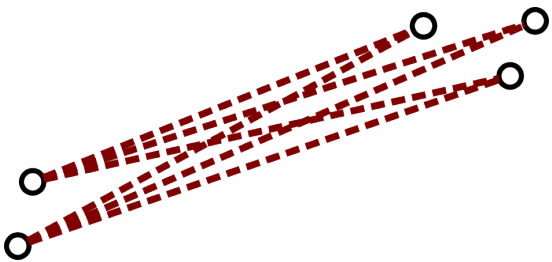


Single Linkage vs. Complete Linkage



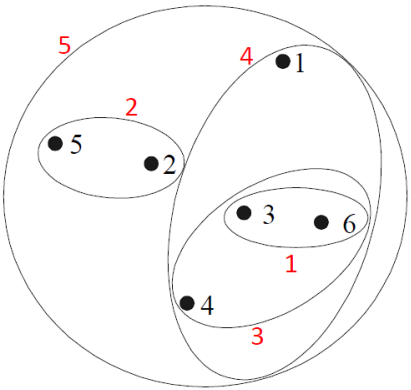
Average Linkage

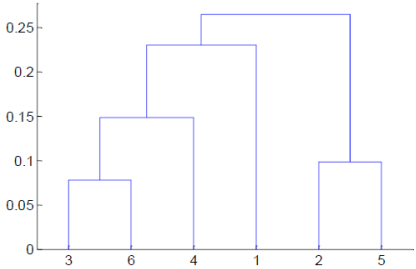
- The distance between two clusters is represented by the average distance of all pairs of data objects belonging to different clusters
- Determined by all pairs of points in the two clusters
- Sometimes also called unweighted pair-group method using the average approach (UPGMA)



$$d(C_i, C_j) = \frac{1}{|C_i||C_j|} \sum_{p \in C_i, q \in C_j} d(p, q)$$

Small Example





Average Linkage Clustering

Average Linkage Clustering

- Compromise between Single and Complete Link
- Strengths
  - Less susceptible to noise and outliers
- Limitations
  - Biased towards spherical clusters

Weighted Average Linkage Clustering

- This method is also known as the weighted pair group method average (WPGMA)
- The difference is that the distances between the newly formed cluster and the rest are weighted based on the number of data points in each cluster

Average Linkage Clustering

Average Linkage Clustering

- Compromise between Single and Complete Link
- Strengths
  - Less susceptible to noise and outliers
- Limitations
  - Biased towards spherical clusters

Weighted Average Linkage Clustering

- This method is also known as the weighted pair group method average (WPGMA)
- The difference is that the distances between the newly formed cluster and the rest are weighted based on the number of data points in each cluster

Other Approaches

Centroid Clustering

- Also known as the unweighted pair-group method using the centroid approach (UPGMC)
- Merges clusters with the most similar mean vectors
- Requires raw-data

Median Linkage

- Also known as the weighted pair-group method using the centroid approach (WPGMC)
- Similar to centroid clustering
- The centroids of the constituent clusters are weighted equally to produce the new centroid of the merged cluster
- Avoids that the centroid is dominated by the larger cluster

Other Approaches

Centroid Clustering

- Also known as the unweighted pair-group method using the centroid approach (UPGMC)
- Merges clusters with the most similar mean vectors
- Requires raw-data

Median Linkage

- Also known as the weighted pair-group method using the centroid approach (WPGMC)
- Similar to centroid clustering
- The centroids of the constituent clusters are weighted equally to produce the new centroid of the merged cluster
- Avoids that the centroid is dominated by the larger cluster

Ward's Methods

Ward's Methods

- Utilizes a sum-of-squares criterion
- Those two clusters are merged which produce the least overall error sum

$$E = \sum_{C_i \in C} E_{C_i}$$

where

$$E_{C_i} = \sum_{x \in C_i} \sum_{k=1}^{d} (x_k - \overline{x}_{ik})$$

with $\overline{x}_{ik}$ being the average value of dimension k of all objects $x \in C_i$

- Very similar to the Centroid, except for the weighting of the centroid

Ward's Methods

Ward's Methods

- Utilizes a sum-of-squares criterion
- Those two clusters are merged which produce the least overall error sum

$$E = \sum_{C_i \in C} E_{C_i}$$

where

$$E_{C_i} = \sum_{x \in C_i} \sum_{k=1}^{d} (x_k - \overline{x}_{ik})$$

with $\overline{x}_{ik}$ being the average value of dimension k of all objects $x \in C_i$

- Very similar to the Centroid, except for the weighting of the centroid

Recurrence Formula for Agglomerative Methods

- It would be quite inefficient, if we would completely recalculate all distances after each merge step
- Therefore, a unified "iterative" method is used in order to calculated the above mentioned distances
- Also called the "Lance and Williams Formula"

- When two clusters C_i and C_j are merged, we need to update the distances to all third clusters C_k by

$$d_{k(ij)} = \alpha_i d_{ki} + \alpha_j d_{kj} + \beta d_{ij} + \gamma |d_{ki} - d_{kj}|$$

- How does this help us?

Recurrence Formula for Agglomerative Methods

- It would be quite inefficient, if we would completely recalculate all distances after each merge step
- Therefore, a unified "iterative" method is used in order to calculated the above mentioned distances
- Also called the "Lance and Williams Formula"

- When two clusters C_i and C_j are merged, we need to update the distances to all third clusters C_k by

$$d_{k(ij)} = \alpha_i d_{ki} + \alpha_j d_{kj} + \beta d_{ij} + \gamma |d_{ki} - d_{kj}|$$

- How does this help us?

Parameters For the Different Methods

	Parameters		
Method	α_i	β	γ
Single linkage	$\frac{1}{2}$	0	$-\frac{1}{2}$
Complete linkage	$\frac{1}{2}$	0	$\frac{1}{2}$
Average linkage	$\frac{n_i}{n_i+n_j}$	0	0
Centroid linkage	$\frac{n_i}{n_i+n_j}$	$-\frac{n_i n_j}{(n_i+n_j)^2}$	0
Median linkage	$\frac{1}{2}$	$\frac{1}{4}$	0
Ward's method	$\frac{n_k+n_i}{n_k+n_i+n_j}$	$-\frac{n_k}{n_k+n_i+n_j}$	0

Summary of the "Lance and Williams Formula"

- If we just look again on the formula

$$d_{k(ij)} = \alpha_i d_{ki} + \alpha_j d_{kj} + \beta d_{ij} + \gamma |d_{ki} - d_{kj}|$$

- We only need the pairwise distances
- When joining cluster A and B to a new cluster R, the distances of this new cluster to all other clusters Q are a liniear combination of the distances of A and B to Q

Consequences:

- Once we have a distance matrix, joining two clusters is relatively cheap
- Especially there is no need for re-calculating cluster means, etc.

Summary of the "Lance and Williams Formula"

- If we just look again on the formula

$$d_{k(ij)} = \alpha_i d_{ki} + \alpha_j d_{kj} + \beta d_{ij} + \gamma |d_{ki} - d_{kj}|$$

- We only need the pairwise distances
- When joining cluster A and B to a new cluster R, the distances of this new cluster to all other clusters Q are a liniear combination of the distances of A and B to Q

Consequences:

- Once we have a distance matrix, joining two clusters is relatively cheap
- Especially there is no need for re-calculating cluster means, etc.

Summary

Single linkage (or Nearest Neighbor)

- Minimum distance between pair of objects from different clusters
- Tends to produce unbalanced and straggly clusters
- Does not take account of cluster structure

Complete linkage (or Furthest Neighbor)

- Maximum distance between pair of objects from different clusters
- Tends to find compact clusters with equal diameters (maximum distance between objects)
- Does not take account of cluster structure

Summary

Single linkage (or Nearest Neighbor)

- Minimum distance between pair of objects from different clusters
- Tends to produce unbalanced and straggly clusters
- Does not take account of cluster structure

Complete linkage (or Furthest Neighbor)

- Maximum distance between pair of objects from different clusters
- Tends to find compact clusters with equal diameters (maximum distance between objects)
- Does not take account of cluster structure

Summary

(Group) Average linkage (UPGMA)

- Average distance between pair of objects different clusters
- Relatively robust, tends to join clusters with small variances
- Intermediate between single and complete linkage
- Takes account of cluster structure

Weighted average linkage (WPGMA)

- Average distance between pair of objects from different clusters
- Compared to average linkage points in small clusters weighted more highly than points in large clusters
- Useful if cluster sizes are likely to be uneven

Summary

(Group) Average linkage (UPGMA)

- Average distance between pair of objects different clusters
- Relatively robust, tends to join clusters with small variances
- Intermediate between single and complete linkage
- Takes account of cluster structure

Weighted average linkage (WPGMA)

- Average distance between pair of objects from different clusters
- Compared to average linkage points in small clusters weighted more highly than points in large clusters
- Useful if cluster sizes are likely to be uneven

Summary

Centroid linkage (UPGMC)

- Squared Euclidean distance between mean vectors
- Requires Raw-data
- Assumes points can be represented in Euclidean space
- The more numerous of the two groups clustered dominates the merged cluster

Median linkage (WPGMC)

- Squared Euclidean distance between weighted centroids
- Requires Raw-data
- Assumes points can be represented in Euclidean space
- New group is intermediate in position between merged groups

Summary

Centroid linkage (UPGMC)

- Squared Euclidean distance between mean vectors
- Requires Raw-data
- Assumes points can be represented in Euclidean space
- The more numerous of the two groups clustered dominates the merged cluster

Median linkage (WPGMC)

- Squared Euclidean distance between weighted centroids
- Requires Raw-data
- Assumes points can be represented in Euclidean space
- New group is intermediate in position between merged groups

Summary

Ward's method (Minimum sum of squares)

- Increase in sum of squares within clusters, after fusion, summed over all variables
- Assumes points can be represented in Euclidean space for geometrical interpretation
- Tends to find same-size, spherical clusters
- Sensitive to outliers

Conclusion

Pros:

- Do not have to assume any particular number of clusters
- Any desired number of clusters can be obtained by 'cutting' the dendrogram at the proper level
- The dendrograms might deliver more information than a partitional clustering
  - E.g., meaningful taxonomies

Cons:

- Once a decision is made to combine two clusters, it cannot be undone
- No objective function is directly minimized
- Different schemes have problems with one or more of the following:
  - Sensitivity to noise and outliers
  - Difficulty handling different sized clusters and irregular shapes
  - Breaking large clusters

Conclusion

Pros:

- Do not have to assume any particular number of clusters
- Any desired number of clusters can be obtained by 'cutting' the dendrogram at the proper level
- The dendrograms might deliver more information than a partitional clustering
  - E.g., meaningful taxonomies

Cons:

- Once a decision is made to combine two clusters, it cannot be undone
- No objective function is directly minimized
- Different schemes have problems with one or more of the following:
  - Sensitivity to noise and outliers
  - Difficulty handling different sized clusters and irregular shapes
  - Breaking large clusters

Hierarchical Clustering

Divisive Approach

Some Initial Thoughts

- Compared to agglomerative hierarchical clustering, divisive clustering proceeds in the opposite way
- In the beginning, the entire data set belongs to a cluster, and a procedure successively divides it until all clusters are singletons
- For a data set with N objects you would need to consider all non-empty partitions into two clusters
  - That would be $2^{N-1} - 1$ possibilities
  - That is computational too expensive, even for small instances
- Therefore, divisive clustering is not a common choice in practice

Some Initial Thoughts

- However, the divisive clustering algorithms do provide clearer insights of the main structure of the data
- The larger clusters are generated at the early stage of the clustering process
- It is less likely to suffer from the accumulated erroneous decisions, which cannot be corrected by the successive process
- Given the computational burden of this approach, we rely on heuristic methods

DIANA

- DIANA (Divisive Analysis) considers only a part of all the possible divisions
- DIANA consists of a series of iterative steps in order to move the closer objects into the splinter group, which is seeded with the object that is farthest from the others in the cluster to be divided
- The cluster with the largest diameter, defined as the largest distance between any pair of objects, is selected for further division

DIANA

Let's assume that Cluster C_l is going to be split into clusters C_i and C_j

1. Start with $C_i = C_l$ and C_j is empty
2. For each data object x_m in C_i:
   - For the first iteration, compute its average distance to all the other objects

$$d(x_m, C_i \setminus \{x_m\}) = \frac{1}{|C_i| - 1} \sum_{x_p \in C_i \setminus \{x_m\}} d(x_m, x_p)$$

   - For the remaining iterations, compute the difference between the average distance to C_i and the average distance to C_j:

$$d(x_m, C_i \setminus \{x_m\}) - d(x_m, C_j) = \frac{1}{|C_i| - 1} \sum_{x_p \in C_i \setminus \{x_m\}} d(x_m, x_p) - \frac{1}{|C_j|} \sum_{x_q \in C_j} d(x_m, x_q)$$

DIANA

3. Decide whether to move element x_m to cluster C_j or keep it in cluster C_i

   - For the first iteration, move the object with the maximum value to C_j (that is, the element farther from every other element is separated from the others)

   - For the remaining iterations, if the maximum difference value is greater than zero, move the data object with the maximum difference into C_j, then repeat steps 2b and 3b. If the maximum value is less than zero, stop.

Monothetic vs. Polythetic

- During each division of DIANA, all features are used; hence the divisive algorithm is called polythetic
- As mentioned before, this is only a heuristic, as not all possibilities can be tested

- However, for data consisting of p binary variables, relatively simple and computationally efficient methods are available
- They are called monothetic divisive methods
- These generally divide clusters according to the presence or absence of each of the p variables, so that at each stage clusters contain members with certain attributes either all present or all absent
- In other words: In each step, a variable is selected on which the data is separated in terms of presence/absence

Monothetic vs. Polythetic

- During each division of DIANA, all features are used; hence the divisive algorithm is called polythetic
- As mentioned before, this is only a heuristic, as not all possibilities can be tested

- However, for data consisting of p binary variables, relatively simple and computationally efficient methods are available
- They are called monothetic divisive methods
- These generally divide clusters according to the presence or absence of each of the p variables, so that at each stage clusters contain members with certain attributes either all present or all absent
- In other words: In each step, a variable is selected on which the data is separated in terms of presence/absence

Dendrograms

Creating Dendrograms

Some Remarks

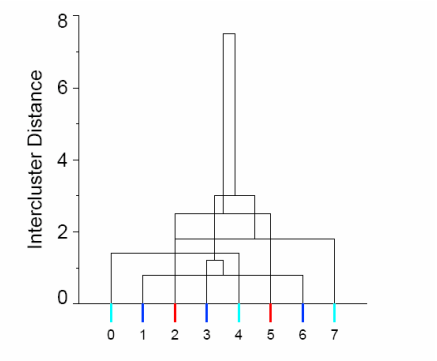
- The dendrogram is a mathematical and pictorial representation of the complete clustering procedure
- The nodes of the dendrogram represent clusters
- The lengths of the stems (heights) represent the distances at which clusters are joined
- The stems may be drawn so that they do not extend to the zero line of the diagram, in order to indicate the order in which objects first join clusters
- Dendrograms which do not have numerical information attached to the stems are termed unweighted or ranked
- Most dendrograms have two edges emanating from each node (binary trees)

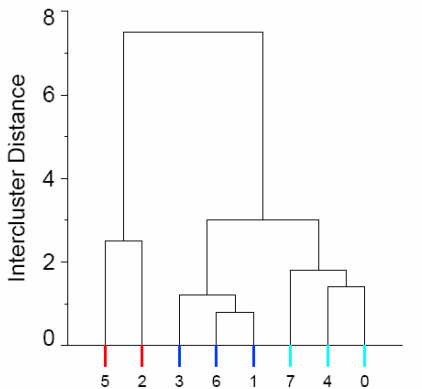
Steps in Order to Create a Dendrogram

1. Perform a agglomerative clustering
2. For each cluster join, draw vertical line from top of each joined cluster up to height corresponding to the distance between them, connect with horizontal line
3. To avoid crossed lines, must have ordered the points so that at each step, joined clusters are next to each other (get unique dendrogram if specify rule for left-right order at each join)

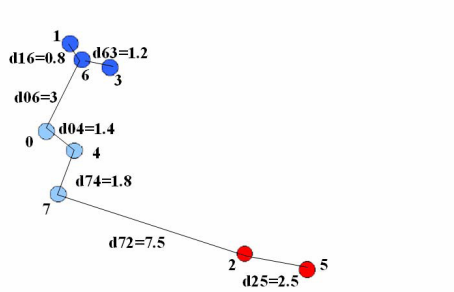
- Note, the same data and clustering method can give rise to 2^{n-1} dendrograms with different appearances, depending on the order in which the nodes are displayed
- But the ordering of the data is important

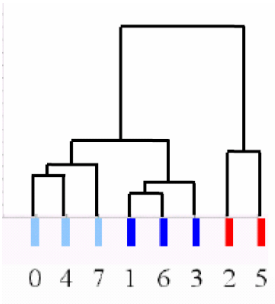
Bad vs. Good Ordering





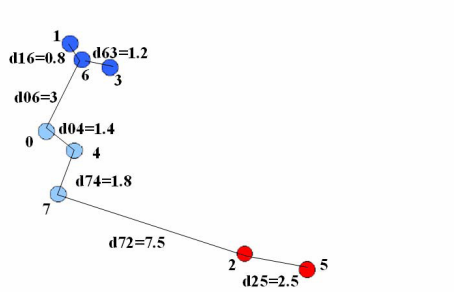
Finding an Ordering by Left Sliding

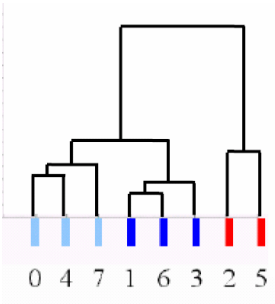




- We start with the ordering: 0 1 2 3 4 5 6 7
- First merge: (1,6): 0 16 2 3 4 5 7
- Next merge: ((1,6),3): 0 163 2 4 5 7
- Next merge: (0,4): 04 163 2 5 7
- Next merge: ((0,4),7): 047 163 2 5
- Next merge: (2,5): 047 163 25
- ... 04716325

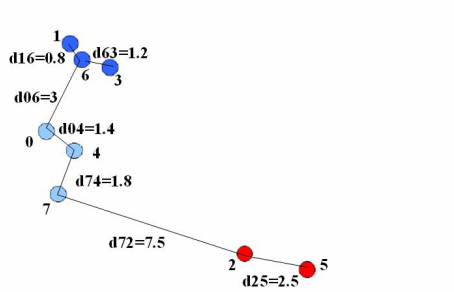
Finding an Ordering by Left Sliding

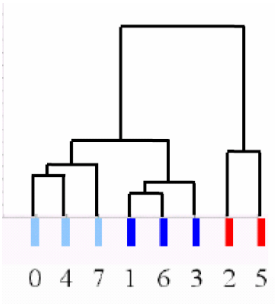




- We start with the ordering: 0 1 2 3 4 5 6 7
- First merge: (1,6): 0 16 2 3 4 5 7
- Next merge: ((1,6),3): 0 163 2 4 5 7
- Next merge: (0,4): 04 163 2 5 7
- Next merge: ((0,4),7): 047 163 2 5
- Next merge: (2,5): 047 163 25
- ... 04716325

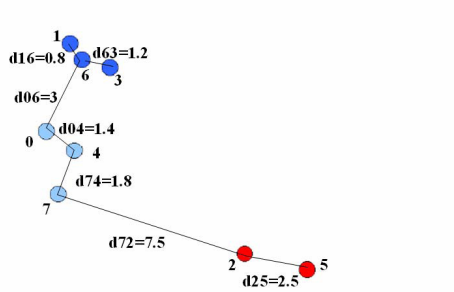
Finding an Ordering by Left Sliding

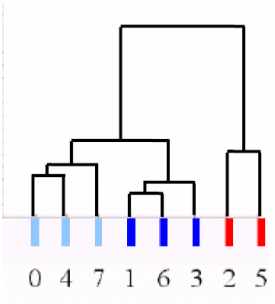




- We start with the ordering: 0 1 2 3 4 5 6 7
- First merge: (1,6): 0 16 2 3 4 5 7
- Next merge: ((1,6),3): 0 163 2 4 5 7
- Next merge: (0,4): 04 163 2 5 7
- Next merge: ((0,4),7): 047 163 2 5
- Next merge: (2,5): 047 163 25
- ... 04716325

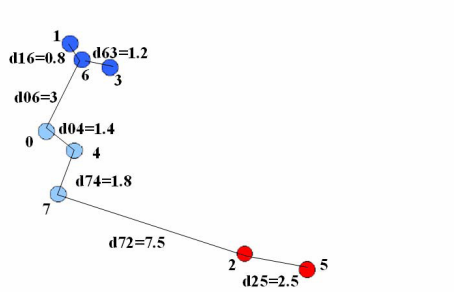
Finding an Ordering by Left Sliding

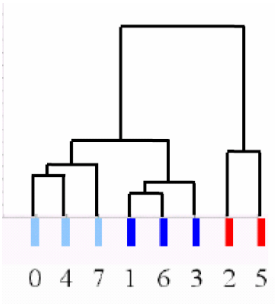




- We start with the ordering: 0 1 2 3 4 5 6 7
- First merge: (1,6): 0 16 2 3 4 5 7
- Next merge: ((1,6),3): 0 163 2 4 5 7
- Next merge: (0,4): 04 163 2 5 7
- Next merge: ((0,4),7): 047 163 2 5
- Next merge: (2,5): 047 163 25
- ... 04716325

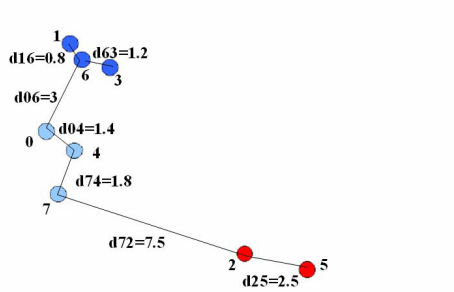
Finding an Ordering by Left Sliding

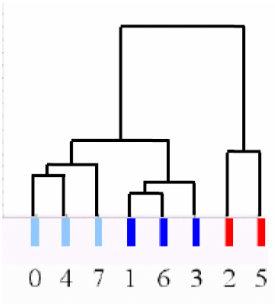




- We start with the ordering: 0 1 2 3 4 5 6 7
- First merge: (1,6): 0 16 2 3 4 5 7
- Next merge: ((1,6),3): 0 163 2 4 5 7
- Next merge: (0,4): 04 163 2 5 7
- Next merge: ((0,4),7): 047 163 2 5
- Next merge: (2,5): 047 163 25
- ... 04716325

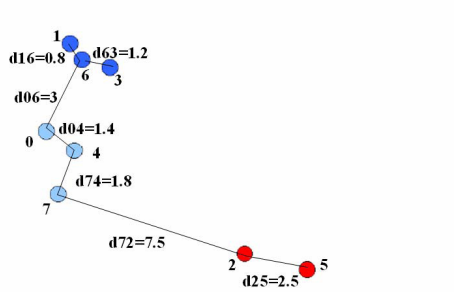
Finding an Ordering by Left Sliding

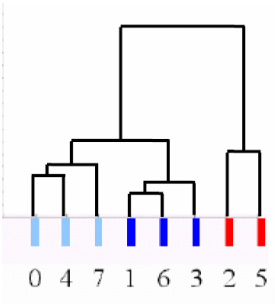




- We start with the ordering: 0 1 2 3 4 5 6 7
- First merge: (1,6): 0 16 2 3 4 5 7
- Next merge: ((1,6),3): 0 163 2 4 5 7
- Next merge: (0,4): 04 163 2 5 7
- Next merge: ((0,4),7): 047 163 2 5
- Next merge: (2,5): 047 163 25
- ... 04716325

Finding an Ordering by Left Sliding





- We start with the ordering: 0 1 2 3 4 5 6 7
- First merge: (1,6): 0 16 2 3 4 5 7
- Next merge: ((1,6),3): 0 163 2 4 5 7
- Next merge: (0,4): 04 163 2 5 7
- Next merge: ((0,4),7): 047 163 2 5
- Next merge: (2,5): 047 163 25
- ... 04716325

Extensions to Dendrograms

- This method gives a unique ordering for a given input sequence
- But still depends on the initial ordering of the data-points
- Nevertheless, we should look at options to compare the dendrograms

Extensions to Dendrograms

- Espaliers encode in the length of the horizontal line about the homogeneity and separation of the clusters
- The pyramid is a specialized dendrogram for displaying overlapping clustering
- The additive tree is a generalization of the dendrogram in which the lengths of the paths between the nodes represent proximities between objects

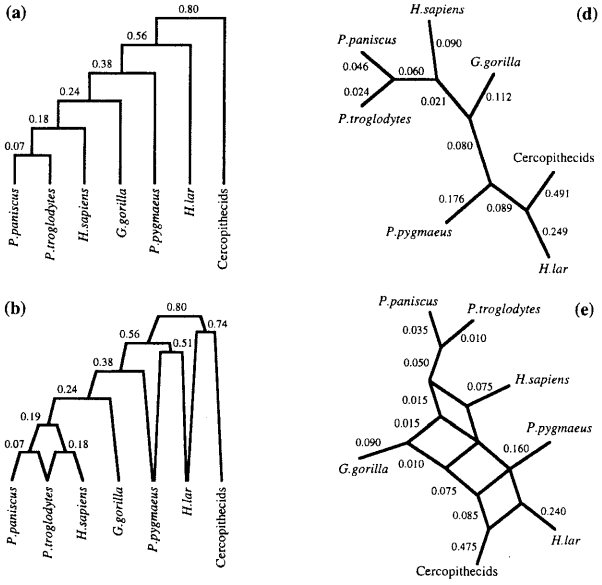
Extensions to Dendrograms

- This method gives a unique ordering for a given input sequence
- But still depends on the initial ordering of the data-points
- Nevertheless, we should look at options to compare the dendrograms

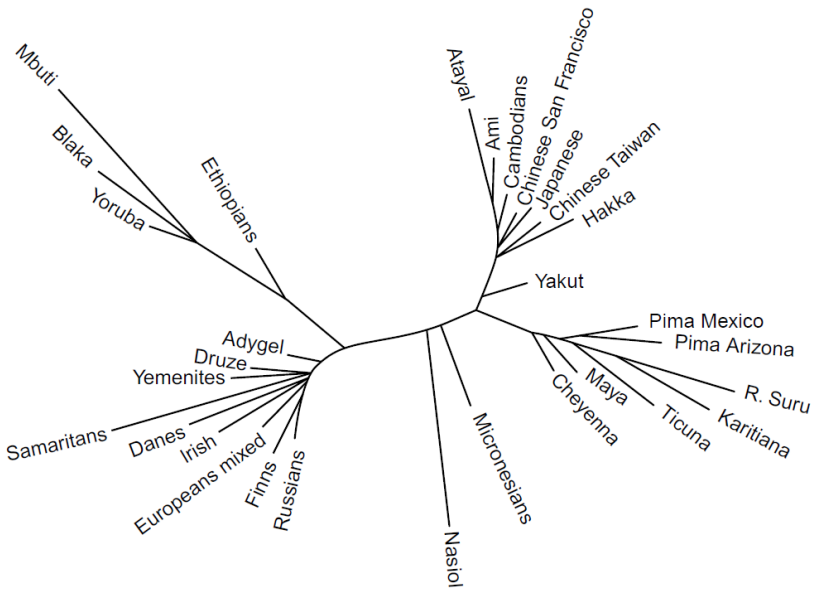
Extensions to Dendrograms

- Espaliers encode in the length of the horizontal line about the homogeneity and separation of the clusters
- The pyramid is a specialized dendrogram for displaying overlapping clustering
- The additive tree is a generalization of the dendrogram in which the lengths of the paths between the nodes represent proximities between objects

Example for a Pyramid



Example for an Additive Tree



Comparing Dendrograms

- As we have seen, dendrograms can look quite differently, even though when they use the same algorithm
- It might come handy to compare dendrograms

- Furthermore, hierarchical clustering techniques impose a hierarchical structure on data
  - Is this structure is acceptable?
  - Introduces this unacceptable distortion?
- We should compare a dendrogram with a proximity matrix

- For that we need to find measures invariant to the order in the Dendrogram

Comparing Dendrograms

- As we have seen, dendrograms can look quite differently, even though when they use the same algorithm
- It might come handy to compare dendrograms

- Furthermore, hierarchical clustering techniques impose a hierarchical structure on data
  - Is this structure is acceptable?
  - Introduces this unacceptable distortion?
- We should compare a dendrogram with a proximity matrix

- For that we need to find measures invariant to the order in the Dendrogram

Comparing Dendrograms

- As we have seen, dendrograms can look quite differently, even though when they use the same algorithm
- It might come handy to compare dendrograms

- Furthermore, hierarchical clustering techniques impose a hierarchical structure on data
  - Is this structure is acceptable?
  - Introduces this unacceptable distortion?
- We should compare a dendrogram with a proximity matrix

- For that we need to find measures invariant to the order in the Dendrogram

Cophenetic Matrix

- A standard Method is based on so-called cophenetic matrix
- The elements of this matrix are the heights, h_{ij}, where two objects become members of the same cluster in the dendrogram
- It is unaffected by the indeterminacy of the appearance of the dendrogram

- In order to compare two cophenetic matrices or a cophenetic matrices with a proximity matrix, the cophenetic correlation is used
- The cophenetic correlation is the product moment correlation between the $\frac{n(n-1)}{2}$ entries h_{ij} in the appropriate cophenetic matrices (excluding those on the diagonals)

Cophenetic Matrix

- A standard Method is based on so-called cophenetic matrix
- The elements of this matrix are the heights, h_{ij}, where two objects become members of the same cluster in the dendrogram
- It is unaffected by the indeterminacy of the appearance of the dendrogram

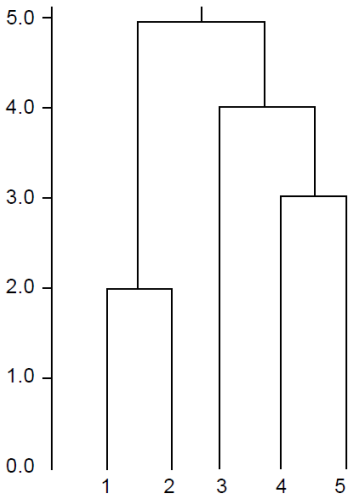
- In order to compare two cophenetic matrices or a cophenetic matrices with a proximity matrix, the cophenetic correlation is used
- The cophenetic correlation is the product moment correlation between the $\frac{n(n-1)}{2}$ entries h_{ij} in the appropriate cophenetic matrices (excluding those on the diagonals)

Example

$$D = \begin{array}{c} \\ p_1 \\ p_2 \\ p_3 \\ p_4 \\ p_5 \end{array} \begin{array}{c} \begin{array}{ccccc} p_1 & p_2 & p_3 & p_4 & p_5 \end{array} \\ \begin{pmatrix} 0.0 & & & & \\ 2.0 & 0.0 & & & \\ 6.0 & 5.0 & 0.0 & & \\ 10.0 & 9.0 & 4.0 & 0.0 & \\ 9.0 & 8.0 & 5.0 & 3.0 & 0.0 \end{pmatrix} \end{array}$$

The corresponding cophenetic matrix:

$$H = \begin{array}{c} \\ p_1 \\ p_2 \\ p_3 \\ p_4 \\ p_5 \end{array} \begin{array}{c} \begin{array}{ccccc} p_1 & p_2 & p_3 & p_4 & p_5 \end{array} \\ \begin{pmatrix} 0.0 & & & & \\ 2.0 & 0.0 & & & \\ 5.0 & 5.0 & 0.0 & & \\ 5.0 & 5.0 & 4.0 & 0.0 & \\ 5.0 & 5.0 & 4.0 & 3.0 & 0.0 \end{pmatrix} \end{array}$$

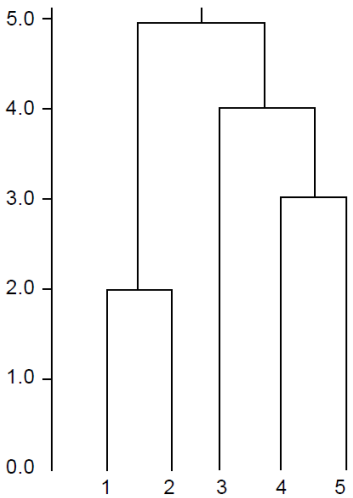


Example

$$D = \begin{array}{c} \\ p_1 \\ p_2 \\ p_3 \\ p_4 \\ p_5 \end{array} \begin{array}{c} \begin{array}{ccccc} p_1 & p_2 & p_3 & p_4 & p_5 \end{array} \\ \begin{pmatrix} 0.0 & & & & \\ 2.0 & 0.0 & & & \\ 6.0 & 5.0 & 0.0 & & \\ 10.0 & 9.0 & 4.0 & 0.0 & \\ 9.0 & 8.0 & 5.0 & 3.0 & 0.0 \end{pmatrix} \end{array}$$

The corresponding cophenetic matrix:

$$H = \begin{array}{c} \\ p_1 \\ p_2 \\ p_3 \\ p_4 \\ p_5 \end{array} \begin{array}{c} \begin{array}{ccccc} p_1 & p_2 & p_3 & p_4 & p_5 \end{array} \\ \begin{pmatrix} 0.0 & & & & \\ 2.0 & 0.0 & & & \\ 5.0 & 5.0 & 0.0 & & \\ 5.0 & 5.0 & 4.0 & 0.0 & \\ 5.0 & 5.0 & 4.0 & 3.0 & 0.0 \end{pmatrix} \end{array}$$



Calculating the Cophenetic Correlation

- In order to compare the two matrices, they are arranged as a vector, e.g. column-wise:

$$\mathbf{H} = (2, 5, 5, 5, 5, 5, 5, 4, 4, 3)$$

$$\mathbf{D} = (2, 6, 10, 9, 5, 9, 8, 4, 5, 3)$$

- This allows us to calculate the normal correlation between these data points
- In this example, that would be 0.82

Modern Hierarchical Clustering

Coping with Large-Scale Datasets

BIRCH

Balanced Iterative Reducing and Clustering using Hierarchies

Designed for very large data sets

- Time and memory are limited
- Incremental and dynamic clustering of incoming objects
- Only one scan of data is necessary
- Does not need the whole data set in advance
- Assumes raw-data with euclidean coordinates

Two Main Phases

- Scans the database to build an in-memory tree
- Applies clustering algorithm to cluster the leaf nodes

Preliminaries

Given a cluster $C_k = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ of N objects. With that we can define:

Centroid

$$\overline{\mathbf{x}} = \frac{1}{N} \sum_{\mathbf{x}_i \in C_k} \mathbf{x}_i$$

Radius

average distance from member points to centroid

$$R = \left(\frac{1}{N} \sum_{\mathbf{x}_i \in C_k} (\mathbf{x}_i - \overline{\mathbf{x}})^2 \right)^{\frac{1}{2}}$$

Diameter

average pair-wise distance within a cluster

$$D = \left(\frac{1}{N(N-1)} \sum_{\mathbf{x}_i \in C_k} \sum_{\mathbf{x}_j \in C_k} (\mathbf{x}_i - \mathbf{x}_j)^2 \right)^{\frac{1}{2}}$$

Preliminaries

Furthermore, BIRCH supports the following "linkage" functions between two clusters C_k and C_l:

Centroid Euclidean/Manhattan Distance

$$d_0 = d_{\text{Euclidean}}(\overline{\mathbf{x}}_k, \overline{\mathbf{x}}_l) \qquad d_1 = d_{\text{Manhattan}}(\overline{\mathbf{x}}_k, \overline{\mathbf{x}}_l)$$

Average Inter Cluster Distance

$$d_2 = \left(\frac{1}{|C_k||C_l|} \sum_{x_i \in C_k} \sum_{x_j \in C_l} (\mathbf{x}_i - \mathbf{x}_j)^2 \right)^{\frac{1}{2}}$$

Average Intra Cluster Distance

$$d_3 = \left(\frac{1}{|C_{kl}|(|C_{kl}| - 1)} \sum_{x_i \in C_{kl}} \sum_{x_j \in C_{kl}} (\mathbf{x}_i - \mathbf{x}_j)^2 \right)^{\frac{1}{2}}$$

Variance Increase

$$d_4 = \text{Var}[C_{kl}] - \text{Var}[C_k] - \text{Var}[C_l]$$

BIRCH: Central Data Structure

- The Birch algorithm builds a dendrogram called clustering feature tree (CF tree) while scanning the data set

- Each entry in the CF tree represents a cluster of objects and is characterized by a 3-tuple: (N_i, LS_i, SS_i), where N_i is the number of objects in the i-th cluster and

$$LS_i = \sum_{j=1}^{N_i} \mathbf{x}_j \qquad \text{Linear Sum of the data points}$$

$$SS_i = \sum_{j=1}^{N_i} \mathbf{x}_j^2 \qquad \text{Squared Sum of the data points}$$

Why these Clustering Features?

- CF entries are compact
  - Compared to storing each data point
  - They store information for an entire sub-cluster
- A CF has sufficient information to calculate d_0-d_4
- Additivity of the CFs allows us to merge CFs incrementally and consistently

$$CF_1 + CF_2 = (N_1 + N_2, LS_1 + LS_2, SS_1 + SS_2)$$

Why these Clustering Features?

- CF entries are compact
  - Compared to storing each data point
  - They store information for an entire sub-cluster
- A CF has sufficient information to calculate d_0-d_4
- Additivity of the CFs allows us to merge CFs incrementally and consistently

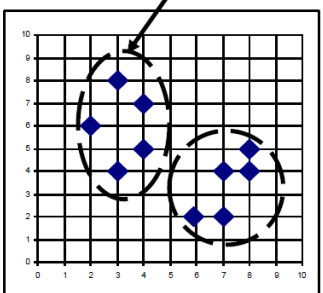
$$CF_1 + CF_2 = (N_1 + N_2, LS_1 + LS_2, SS_1 + SS_2)$$

Why these Clustering Features?

- CF entries are compact
  - Compared to storing each data point
  - They store information for an entire sub-cluster
- A CF has sufficient information to calculate d_0-d_4
- Additivity of the CFs allows us to merge CFs incrementally and consistently

$$CF_1 + CF_2 = (N_1 + N_2, LS_1 + LS_2, SS_1 + SS_2)$$

CF Example

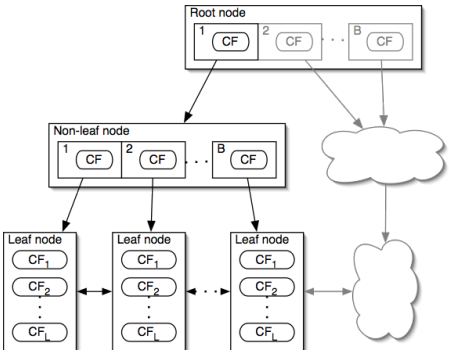


CF = (5, (16,30),(54,190))

(3,4)
(2,6)
(4,5)
(4,7)
(3,8)

The CF Tree

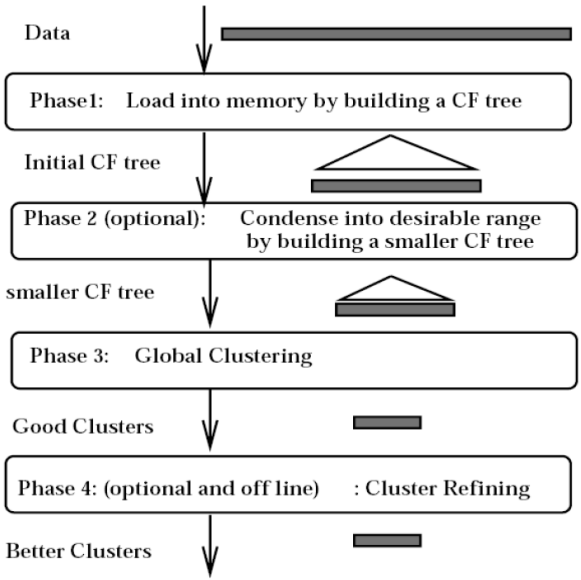
- Each non-leaf node has at most B entries (Branching Factor)
- Each leaf node has at most L CF entries, each of which satisfies threshold T (e.g., not exceeding a given radius R or diameter D
- Node size is determined by dimensionality of data space and input parameter P (page size)



The CF Tree

- The tree size is a function of T (the larger the T is, the smaller the tree is)
- We require a node to fit in a page of size of P
- B and L are determined by P
- Very compact representation of the dataset because each entry in a leaf node is not a single data point but a subcluster

BIRCH Algorithm



Building the CF Tree (Phase 1)

- CF of a single data point $(3, 4)$ is $(1, (3, 4), (9, 16))$
- Insert a point to the tree
  - Find the path from the root to the best fitting leaf (based on d_0 - d_4 between CF of children in a non-leaf node)
  - When the best-fitting leaf is found:
    - Find best fitting CF (based on d_0 - d_4)
    - Check if the CF can absorb the new data point
  - Modify the path to the leaf (update the corresponding CFs)
  - Split if leaf node is full

Compactifying the tree (Phase 2)

- The threshold defines that a certain limit of similarity has to be guaranteed in the leaf nodes
- The algorithm scans all the leaf entries in order to:
  - Removing outliers
  - Grouping crowded subclusters into larger ones.
- In case of memory problems:
  - Increase threshold
  - The larger threshold will allow to merge CFs together
  - Rebuilding the tree costs at most h additional nodes (height of the tree)
- Idea is to find a balance which uses the available resources as good as possible while being as exact as possible

Global Clustering (Phase 3)

- Consider CF entries in leaf nodes only
- Use centroid as the representative of a cluster
- Perform traditional clustering (e.g. agglomerative hierarchy (complete link $==$ D_2) or K-means or ...)
- Cluster CF instead of data points

Cluster Refining (Phase 4)

- Require scan of dataset one more time
- Use clusters found in phase 3 as seeds
- Redistribute data points to their closest seeds and form new clusters
- Removal of outliers
- Acquisition of membership information

Conclusions

- A CF tree is a height-balanced tree that stores the clustering features for a hierarchical clustering
- Given a limited amount of main memory, BIRCH can minimize the time required for I/O
- BIRCH is a scalable clustering algorithm with respect to the number of objects, and good quality of clustering of the data
- A clustering algorithm taking consideration of I/O costs, memory limitation
- Utilize local information (each clustering decision is made without scanning all data points)
- Not every data point is equally important for clustering purpose

Take Away Message

- Very common and well studies clustering technique
- It provides the user with additional hierarchical information
- Only requires a parameters to derive a partitional clustering

- Agglomerative
  - Fast, simple
  - Decisions once made cannot be undone
  - Errors might propagate and reinforce

- Divisive
  - Large clusters are formed at the beginning, thus limited risk of propagated errors
  - Computational very expensive $\rightarrow$ We need to use heuristics

- General
  - What linking function to use?
  - Is there one better than the other?
  - Can we see if we accumulated an error?