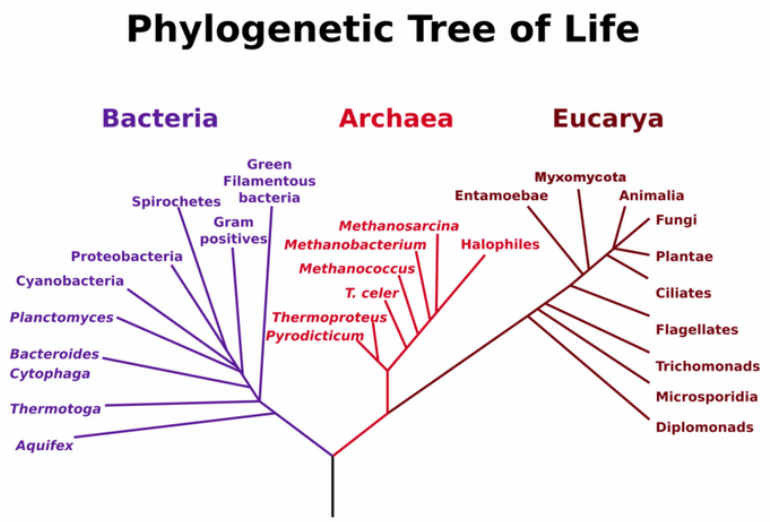
Clustering
Finding groups in data

Stefano Rovetta

Introduction

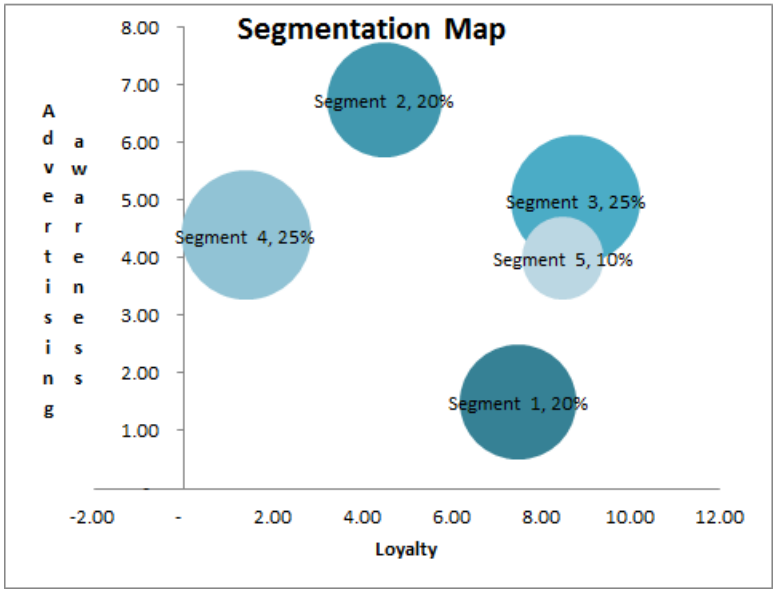
Motivational examples

In biology: Living beings (taxonomy, “systematics”)



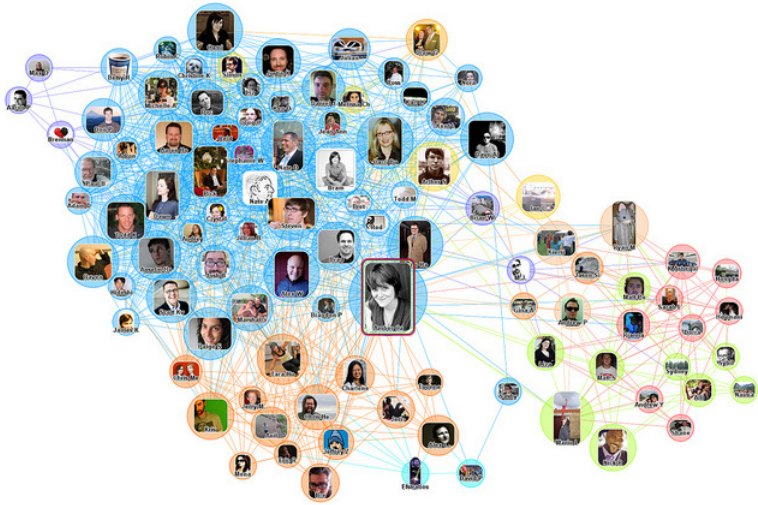
Motivational examples

In marketing: Customers



Motivational examples

In the social sciences: People



What is clustering?

CLUSTERING = Discovering “natural” groupings in your data

- Different from classification:
  CATEGORIES ARE NOT GIVEN, must be discovered
- Different from template matching:
  GROUP DEFINITION IS NOT GIVEN, must be discovered

Ingredients of clustering

Ingredients

1. Data
   how to represent them
2. Proximities
   how to measure closeness of two data objects
3. Grouping criterion
   how to decide which data objects should go into a given group
   and which ones should go in other groups

Different types of data

- Data objects only have an identifier but no name:
  `object_1, object_2, object_3`
- Data objects are described by a set of simple attributes:
  `[colour = blue, height = 175, weight = 70 ]`
  Attributes also termed “features”
- Data Objects are described by complex attributes:
  - A string of symbols, possibly with variable length
    A text string, a DNA sequence, a set of tags
  - A hierarchical description
    XML data, an ontology subtree
  - Something with **complex semantics** that requires additional, external knowledge to be interpreted
    A natural-language text, an image, other multimedia data

Proximities

Catch-all term for:

- Measure of similarity
- Measure of dissimilarity
- Metric – special dissimilarities such that:
  - $d(a, b) \geq 0$ (non-negativity)
  - $d(a, b) = 0 \iff a \equiv b$ (identity of "indiscernibles")
  - $d(a, b) = d(b, a)$ (symmetry)
  - $d(a, b) \leq d(a, c) + d(c, b)$ (triangle inequality)

Proximities

Depend on type of data representation

- Data objects only have an identifier but no attributes:
  Proximities must be given!
- Data objects are described by a set of simple attributes:
  The most common case, more on this in few seconds
- Data Objects are described by complex attributes:
  Two options:
  1. Use ad-hoc measures
  2. Reduce to the previous case

Data described by collections of simple attributes

(1) Categorical

Proximity can be computed only based on identity (a particular feature is/is not equal)
Example: Hamming distance

$$d(\mathbf{x}_1, \mathbf{x}_2) = \sum_{i=1}^{d} [x_{1,i} = x_{2,i}]$$

(2) Numerical

Several measures of proximity can be computed
Example: Euclidean distance

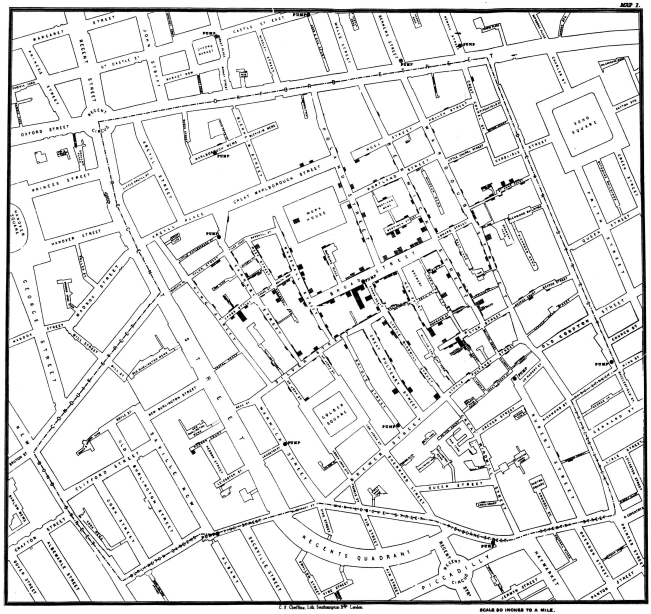
$$d(\mathbf{x}_1, \mathbf{x}_2) = \sqrt{\frac{1}{d} \sum_{i=1}^{d} (x_{1,i} - x_{2,i})^2}$$

Example: Cosine similarity

$$d(\mathbf{x}_1, \mathbf{x}_2) = \frac{1}{d} \sum_{i=1}^{d} x_{1,i} x_{2,i}$$

Data with numerical features and with a proximity that is a metric
can be treated as vectors in Euclidean space $\rightarrow$ often very handy!

Clustering metric data



1854

Broad Street cholera outbreak
Soho district, London

The cholera bacterium (Vibrio choleræ) was not known

Two theories:

1. Miasma theory
   – Propagation through air
2. Germ theory
   – Propagation through water

Clustering metric data

1854

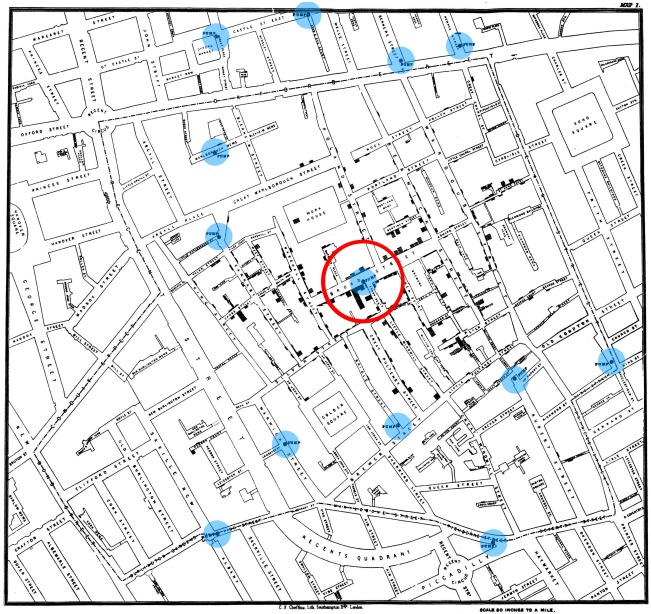
Broad Street cholera outbreak
Soho district, London

The cholera bacterium (Vibrio choleræ) was not known

Two theories:

1. Miasma theory
   – Propagation through air
2. Germ theory
   – Propagation through water

Clustering metric data



1854

Broad Street cholera outbreak
Soho district, London

The cholera bacterium (Vibrio choleræ) was not known

Two theories:

1. Miasma theory
   – Propagation through air
2. Germ theory
   – Propagation through water

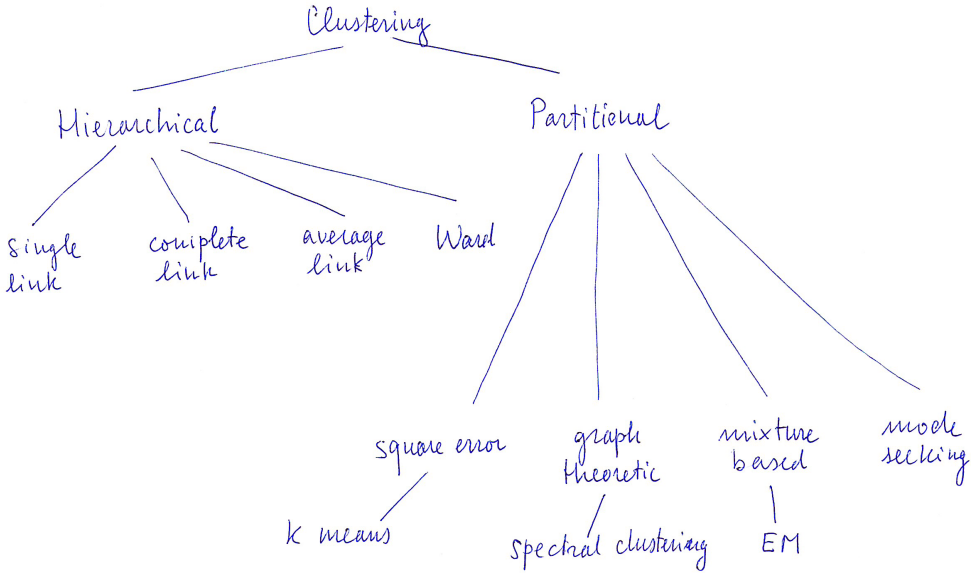
Clustering network data

A network can be described in various ways depending on the type of information given:

- Explicit graph, as an adjacency matrix A
  A can be directly used as a proximity
  Alternatively, we can compute statistics using A: degree, centrality, betweenness, assortativity…
- Explicit graph, as a weighted adjacency matrix W
  As above, but more choices
- Special case: When A or W are symmetrical, we can use them as proximities and employ spectral graph theory
- Objects described by properties (features)
  We are free to compute proximities as we like
- Objects described by properties, PLUS adjacency info
  We can do anything!
  E.g., combine adjacency and features to obtain a content-rich proximity

Categorizing clustering methods

One possible taxonomy



Cross-cutting issues

- Agglomerative vs divisive
- Deterministic vs stochastic
- Crisp (or hard) vs fuzzy (or soft)
- With vs without outliers

Clustering methods and clustering algorithms

- A clustering method is defined by the 3 ingredients: data/proximity/grouping criterion
- A clustering algorithm is defined by a sequence of steps and implements a clustering methods
- Usually, given a method, there is an optimal algorithm with a given computational complexity. This is why we can speak of the complexity of a method
- **The optimal clustering task is a high-complexity one**, so usually algorithms are suboptimal / heuristic/ approximate

Clustering methods

Methods (especially w.r.t. the clustering criterion) can be:

- Heuristic
  when they are based on an intuition that mostly works
- Optimization-based
  when they seek the optimum of some objective function

Some clustering methods

- k means
- Hierarchical clustering
- Spectral clustering

k means clustering

- Metric data (vectors, Euclidean distance)
- Partitional method

k means

- Clusters are represented by **prototypes** or **codevectors** or **centroids**= representative points
- Each centroid has the “centroid property”: it is the mean of all points in its cluster (barycenter)
- The number k of clusters (and therefore of centroids) must be specified in advance

Hence the name: k means.

One possible k means algorithm

Lloyd, Stuart P. (1982), "Least squares quantization in PCM", IEEE Transactions on Information Theory, 28 (2): 129-137

- Step 0: centroids are initialized
- Step 1: Each data point is put in the cluster **whose centroid is nearest**
- Step 2: Centroids are adjusted to actually be in the **barycenter (mean) of the cluster they represent**
- Repeat steps 1 and 2 until nothing changes anymore.

k means algorithm

Input: Training set X, number of clusters k

0 Initialize k centroids $\mathbf{y}_1, \ldots, \mathbf{y}_k$, for instance randomly. Create an indicator vector $\mathbf{u}_l$ for each data point.

1 For each data point $\mathbf{x}_l$:

1.1 Compute the distance $d_{lj} = \|\mathbf{x}_l - \mathbf{y}_j\|$ to each centroid $\mathbf{y}_j$

1.2 Select $j^* = \arg\min_j d_{lj}$

1.3 Set

$$\mathbf{u}_l = \begin{bmatrix} 0 & 0 & \ldots & 1 & \ldots & 0 & 0 \\ 1 & 2 & \ldots & j^* & \ldots & k-1 & k \end{bmatrix}$$

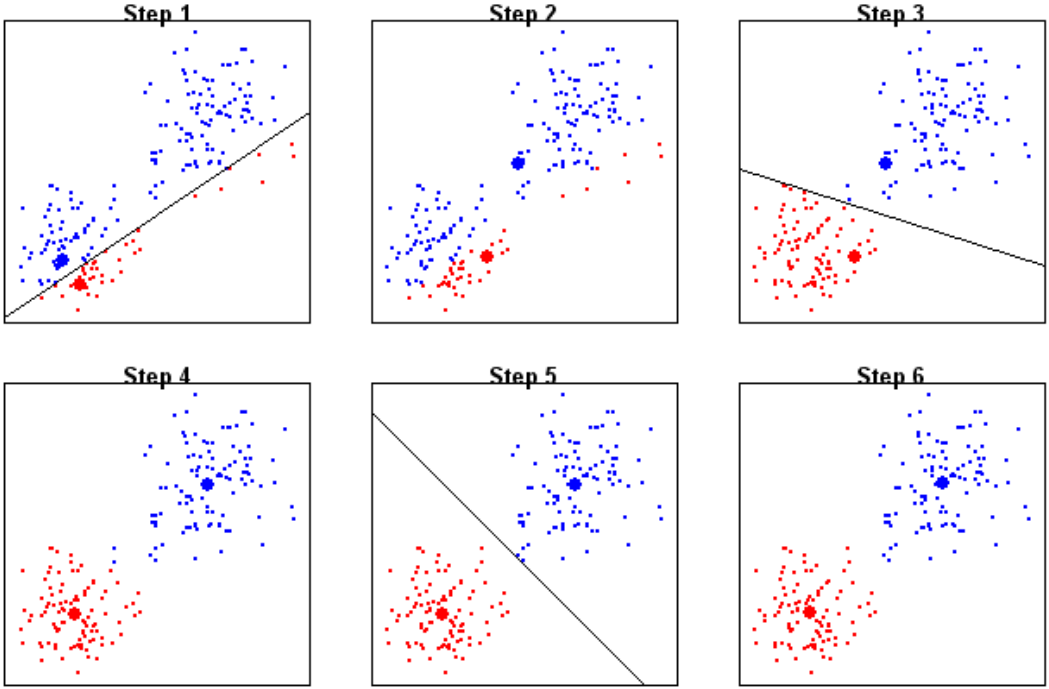
2 For each centroid $\mathbf{y}_j$:

2.1 Select all data points that have $\mathbf{u}_{l,j} = 1$

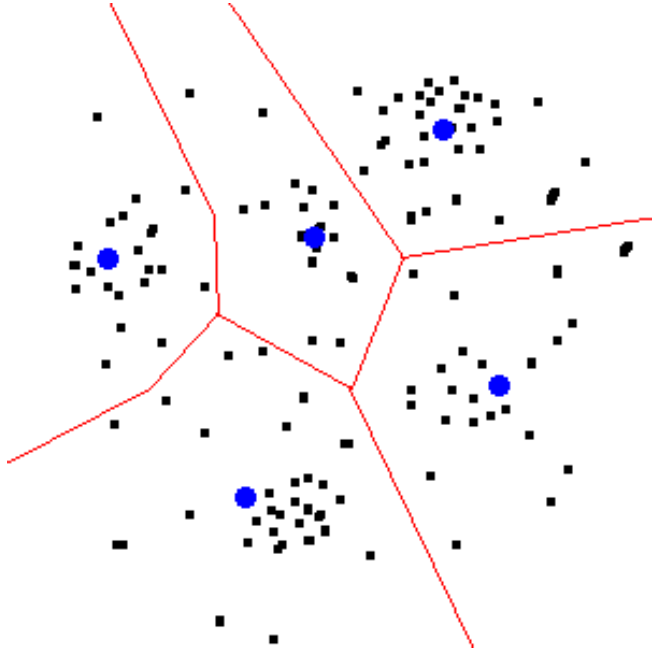
2.2 Move centroid $\mathbf{y}_j$ to their geometric mean

3 If centroids have changed w.r.t. last iteration, goto 1

k means example



Voronoi tessellation



k means objective

$$J = \sum_{l=1}^{n} \|\mathbf{x}_l - \mathbf{y}_{j^*}\|^2$$

where $\mathbf{y}_{j^*}$ is the **nearest centroid** to $\mathbf{x}_l$.

Using indicator vectors

$$J = \sum_{l=1}^{n} \sum_{j=1}^{k} u_{lj} \|\mathbf{x}_l - \mathbf{y}_j\|^2$$

and (in step 2) the centroids can be computed as:

$$\mathbf{y}_j = \frac{\sum_{l=1}^{n} u_{lj} \mathbf{x}_l}{\sum_{i=1}^{n} u_{ij}}$$

Pros/cons of k means

Pros:

- Very simple
- Guaranteed not to diverge; usually it converges quickly
- Works with high-dimensional data

Cons:

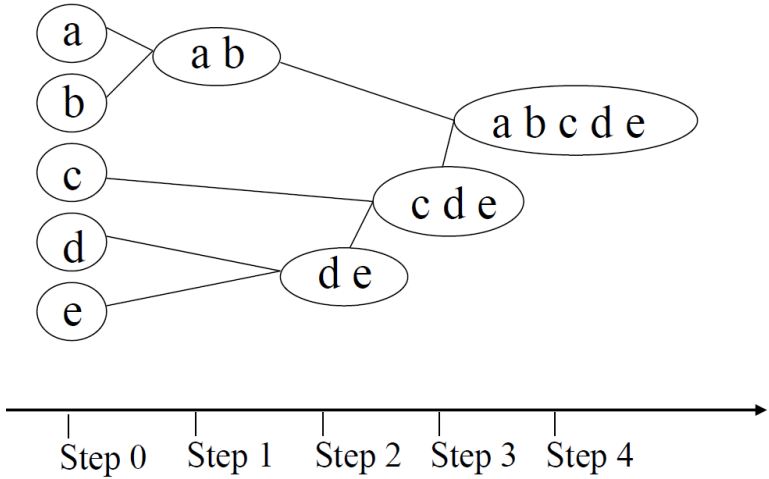
- Stochastic - Depends on initialization
- Local optimization only: J is not convex, many local minima
- Always finds k clusters (even if they are not there)

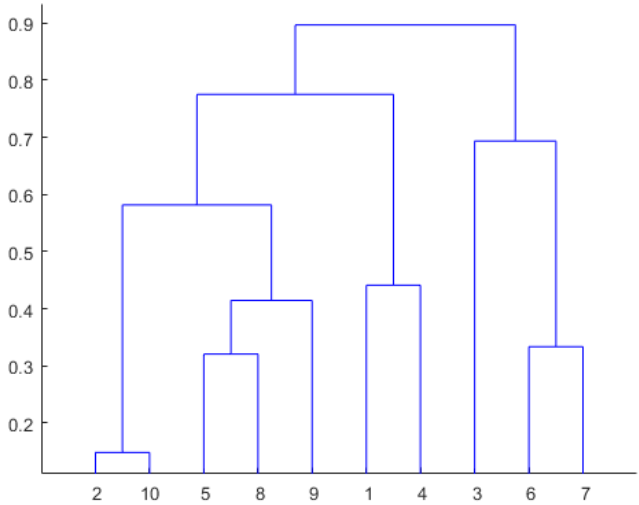
Hierarchical clustering

- Produces a sequence of nested partitions
- At the top: the whole data set
- At the bottom: each data object by itself (singleton clusters)
- Result: a dendrogram
- Can be agglomerative or divisive

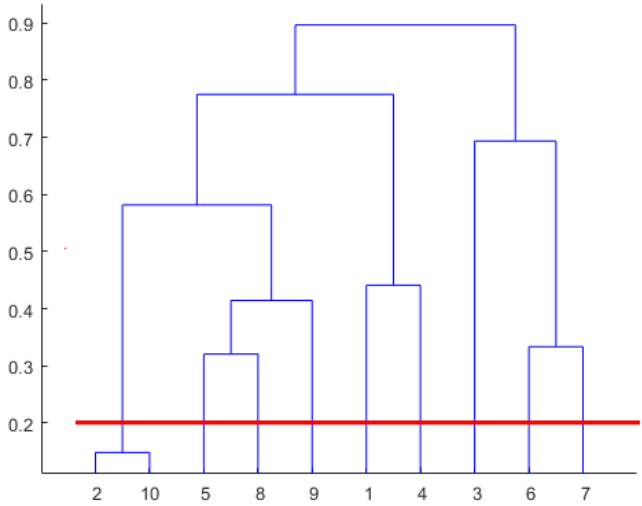
- Agglomerative: at each step, merge two objects or clusters
- Divisive: at each step, split a cluster in two

Agglomerative hierarchical clustering (HAC)
is the most popular.

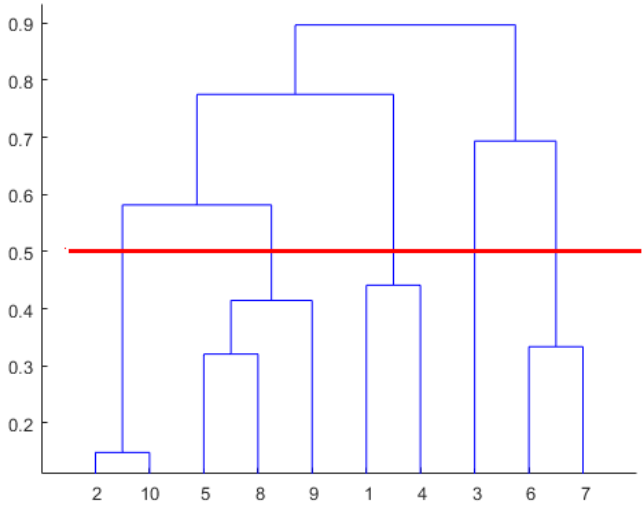

a
b
c
d
e
a b
c d e
d e
a b c d e
Step 0
Step 1
Step 2
Step 3
Step 4



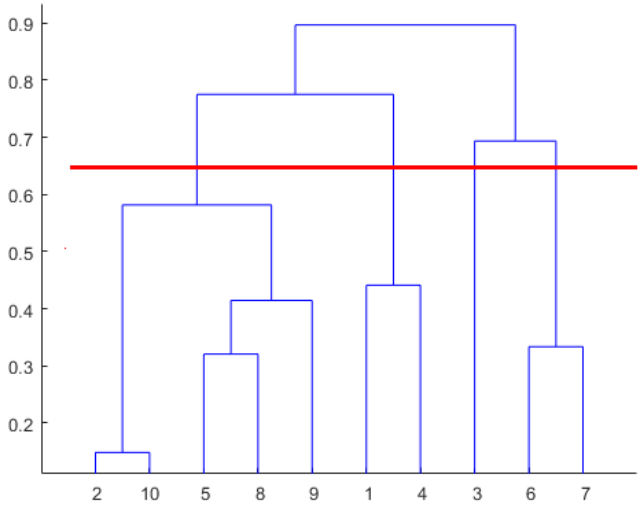
Actual clustering consists in deciding at what proximity level to cut



Actual clustering consists in deciding at what proximity level to cut



Actual clustering consists in deciding at what proximity level to cut



Actual clustering consists in deciding at what proximity level to cut

Agglomerative hierarchical clustering – the basic algorithm

1. Compute the proximity matrix.
2. **repeat**
3. Merge the closest two clusters.
4. Update the proximity matrix to reflect the proximity between the new cluster and the original clusters.
5. **until** only one cluster remains

Data

Data don't need to be metric

We just need to be able to compute a proximity

Some variants work even if proximities are given

Proximity

Any proximity which makes sense for the data at hand

- Example: DNA sequence similarity
  →molecular taxonomy of living beings
- Example: similarity of items bought in an online shop (nature, quantity)
  →market basket analysis
- Example: similarity of interests and personal links on social networks
  →recommendation systems

Grouping criterion

Based on the linkage between groups

Suppose the proximity is given by a dissimilarity $d(x, y)$ between objects x and y. Then:

- Single linkage

$$\text{Linkage}(C_1, C_2) = \min_{x \in C_1, y \in C_2} d(x, y)$$

- Complete linkage

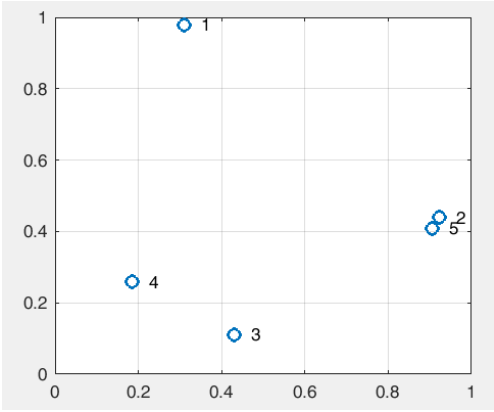
$$\text{Linkage}(C_1, C_2) = \max_{x \in C_1, y \in C_2} d(x, y)$$

- Average linkage

$$\text{Linkage}(C_1, C_2) = \frac{1}{|C_1||C_2|} \sum_{x \in C_1, y \in C_2} d(x, y)$$

- Other approach: Ward's method
- ...more criteria are possible...

HAC example 1



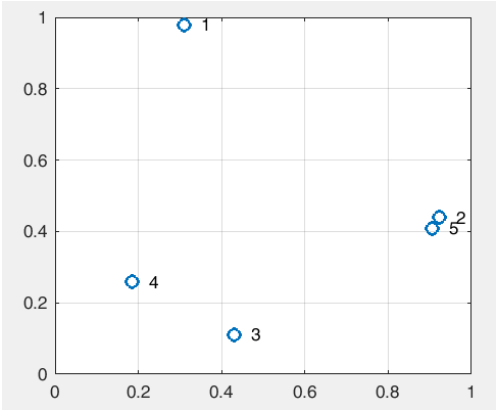
Data:

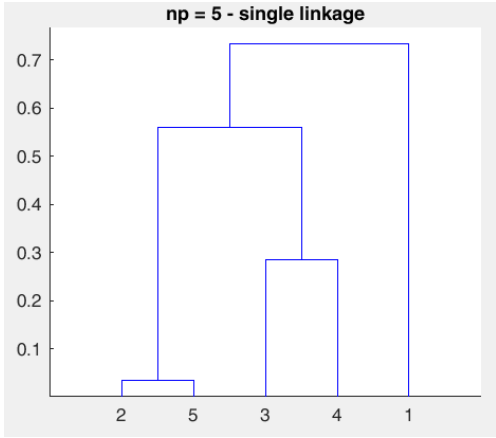
$$\begin{pmatrix} 0.3111 & 0.9797 \\ 0.9234 & 0.4389 \\ 0.4302 & 0.1111 \\ 0.1848 & 0.2581 \\ 0.9049 & 0.4087 \end{pmatrix}$$

Distance matrix:

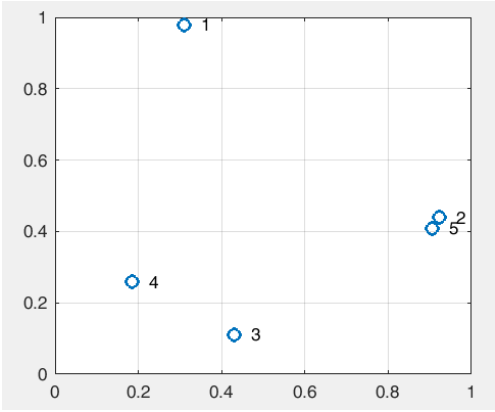
$$\begin{pmatrix} 0 & & & & \\ 0.6674 & 0 & & & \\ 0.7687 & 0.3506 & 0 & & \\ 0.5368 & 0.5782 & 0.0818 & 0 & \\ 0.6786 & 0.0013 & 0.3139 & 0.5412 & 0 \end{pmatrix}$$

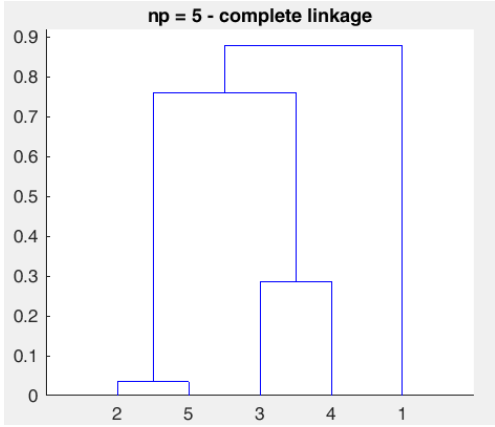
Example 1 - Single linkage



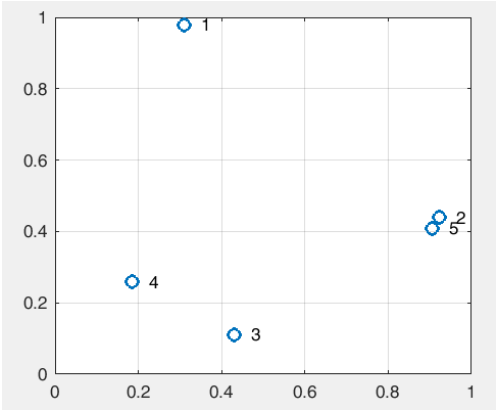


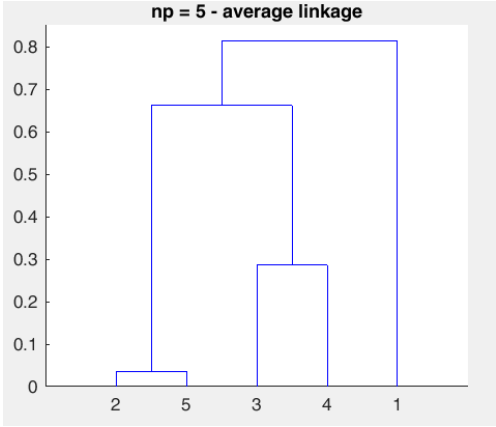
Example 1 - Complete linkage



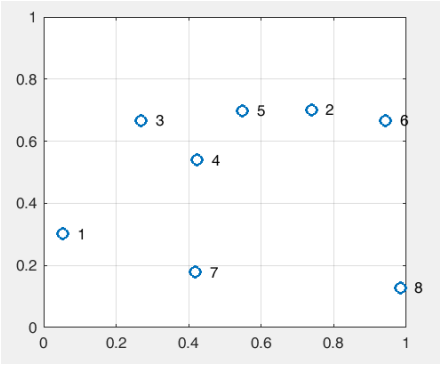


Example 1 - Average linkage





HAC example 2



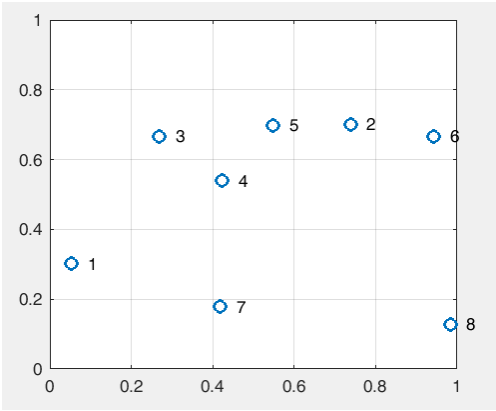
Data:

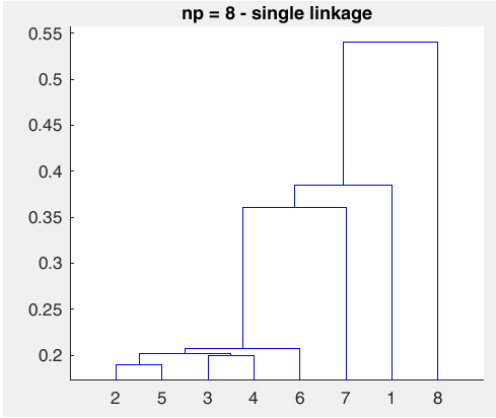
$$\begin{pmatrix} 0.0527 & 0.3015 \\ 0.7379 & 0.7011 \\ 0.2691 & 0.6663 \\ 0.4228 & 0.5391 \\ 0.5479 & 0.6981 \\ 0.9427 & 0.6665 \\ 0.4177 & 0.1781 \\ 0.9831 & 0.1280 \end{pmatrix}$$

Distance matrix:

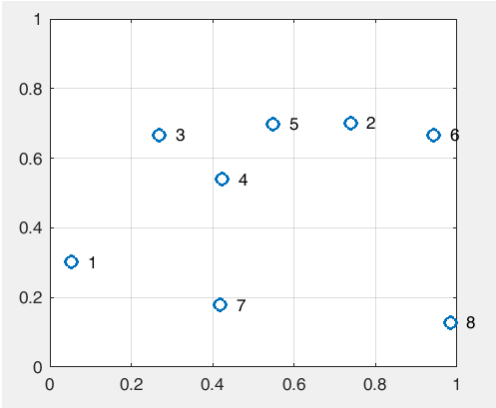
$$\begin{pmatrix} 0 & & & & & & \\ 0.6292 & 0 & & & & & \\ 0.1800 & 0.2209 & 0 & & & & \\ 0.1935 & 0.1255 & 0.0398 & 0 & & & \\ 0.4025 & 0.0361 & 0.0787 & 0.0409 & 0 & & \\ 0.9255 & 0.0432 & 0.4538 & 0.2865 & 0.1569 & 0 & \\ 0.1485 & 0.3760 & 0.2604 & 0.1303 & 0.2873 & 0.5141 & 0 \\ 0.8957 & 0.3885 & 0.7995 & 0.4829 & 0.5144 & 0.2916 & 0.3221 \end{pmatrix}$$

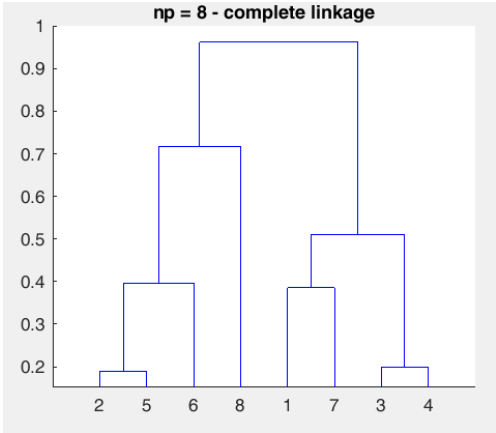
Example 2 - Single linkage



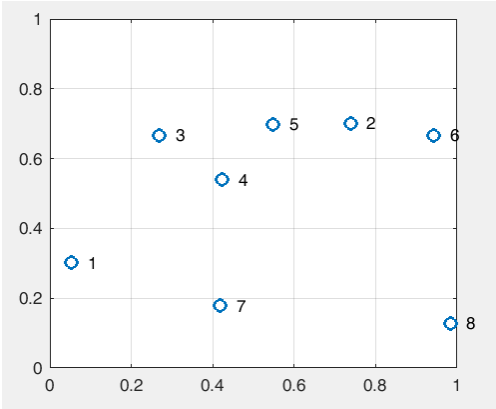


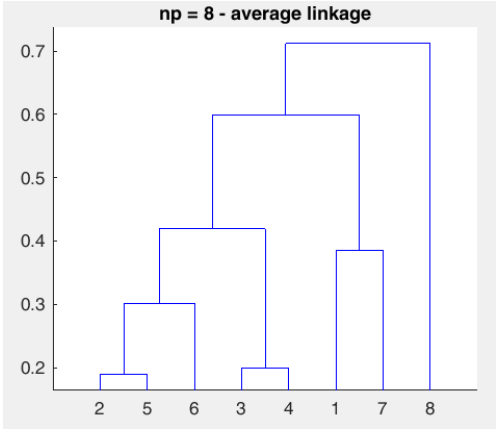
Example 2 - Complete linkage





Example 2 - Average linkage





Ward's method

Ward, J. H., Jr. (1963), "Hierarchical Grouping to Optimize an Objective Function", Journal of the American Statistical Association, 58, 236-244

- Clustering according to an objective function rather than a local linkage evaluation
- **Suitable for metric data**, so $d(x, y) =$ squared Euclidean distance
  and linkage = mean squared distance, or "variance" σ^2
- For each cluster C_i

$$\sigma^2(C_i) = \frac{2}{n(n-1)} \sum_{x,y \in C_i} d(x, y)$$

- Grouping criterion:
  merge together clusters C_i and C_j if
  the increase from $\frac{1}{2}\left(\sigma^2(C_i) + \sigma^2(C_j)\right)$ **to** $\sigma^2(C_i \cup C_j)$ **is minimum**

Spectral clustering

Filippone, M., Camastra, F., Masulli, F., Rovetta, S. (2008). A survey of kernel and spectral methods for clustering. Pattern recognition, 41(1), 176-190.

- Based on spectral graph theory
- Needs a symmetric, weighted adjacency matrix W
- Not necessarily for graphs; W can be a pairwise proximity matrix
- Uses the eigenvectors of a matrix related to W

A spectral clustering algorithm

1. Compute the degree matrix

$$D = \text{diag}(\text{sum}(W)) \qquad \text{or} \qquad D_{ii} = \sum_j W_{ij}$$

2. Compute the graph Laplacian matrix

$$L = D - W$$

3. Optional (but advised): Compute the normalized Laplacian as either

$$L_{sym} = D^{-1/2} L D^{-1/2} \qquad \text{or } L_{rw} = D^{-1} L$$

4. Compute the eigenvectors of L or L_{sym} or L_{rw}
obtaining $\hat{V}$ = matrix containing eigenvectors as columns

5. Discard the first column of $\hat{V}$ (it is constant, so uninformative)

6. Keep k columns from the remaining ones, obtaining V

7. Optional (but advised): normalize each row of V to have norm 1

8. Cluster the rows of V with k means

Pros/cons of spectral clustering

Pros:

- Very simple, one-step, no optimization
- Few parameters: k, plus those possibly in the proximity function
- Works with high-dimensional data
- Finds clusters of complex shapes

Cons:

- Proximity-based, requires n^2 space (not good for large data sets)
- Needs good separation between clusters
- Sensitive to the choice of proximity and its parameters

finis

More on graph clustering:

Schaeffer, Satu Elisa. "Graph clustering." Computer science review 1.1 (2007): 27-64.