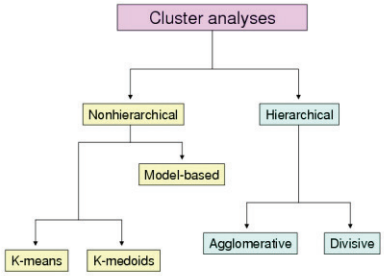
Cluster Analysis

Summer School on Geocomputation

27 June 2011 – 2 July 2011
Vysoké Pole

Lecture delivered by:
doc. Mgr. Radoslav Harman, PhD.
Faculty of Mathematics, Physics and Informatics
Comenius University, Bratislava, Slovakia

Structure of cluster analyses



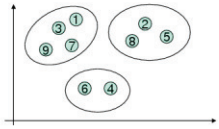
Nonhierarchical clustering

Finds a decomposition of objects 1,...,n into k disjoint clusters $C_1,...,C_k$ of „similar" objects:

$$C_1 \cup ... \cup C_k = \{1,...,n\},\ i \neq j \Rightarrow C_i \cap C_j = \emptyset$$

The objects are (mostly) characterized by „vectors of features" $x_1,...,x_n \in \Re^p$

p=2
k=3
n=9



$C_1 = \{1,3,7,9\}$ $|C_1| = 4$
$C_2 = \{2,5,8\}$ $|C_2| = 3$
$C_3 = \{4,6\}$ $|C_3| = 2$

How do we understand „decomposition into clusters of similar objects"?

How is this decomposition calculated?

Many different approaches: **k-means**, **k-medoids**, **model-based clustering**,...

K-means clustering

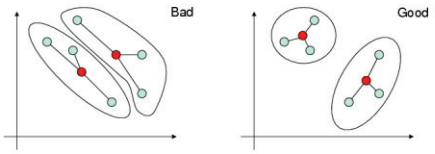
The target function to be minimized with respect to the selection of clusters:

$\sum_{i=1}^{k} \sum_{r \in C_i} \rho^2(x_r, c_i)$ where $c_i = \frac{1}{|C_i|} \sum_{r \in C_i} x_r$ is the centroid of C_i.

ρ is the Euclidean distance:

$$\rho(x, y) = \sqrt{\sum_{t=1}^{p} (x_{(t)} - y_{(t)})^2}$$

where $x_{(t)}, y_{(t)}$ are the t^{th} components of the vectors x, y.



K-means clustering

It is a difficult problem to find the clustering that minimizes the target function of the k-means problem. There are many efficient heuristics that find a „good", although not always optimal solution. Example:

Lloyd's Algorithm

- Create a random initial clustering $C_1,..., C_k$.
- Until a maximum prescribed number of iterations is reached, or no reassignment of objects occurs do:
  - Calculate the centroids $c_1,..., c_k$ of clusters.
  - For every i=1,...,k :
    - Form the new cluster C_i from all the points that are closer to c_i than to any other centroid.

Illustration of the k-means algorithm

Choose an initial clustering

p=2
k=3
n=11

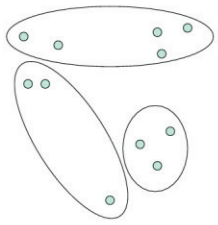


Illustration of the k-means algorithm

Calculate the centroids of clusters

p=2
k=3
n=11

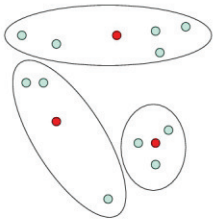


Illustration of the k-means algorithm

Assign the points to the closest centroids

p=2
k=3
n=11

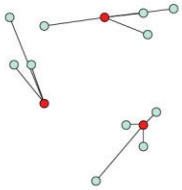


Illustration of the k-means algorithm

Create the new clustering

p=2
k=3
n=11

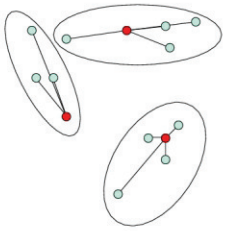


Illustration of the k-means algorithm

Create the new clustering

p=2
k=3
n=11

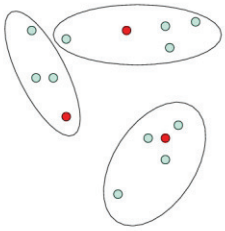


Illustration of the k-means algorithm

Calculate the new centroids of clusters

p=2
k=3
n=11

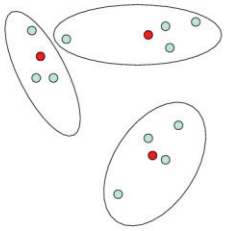


Illustration of the k-means algorithm

Assign the points to the closest centroids

p=2
k=3
n=11

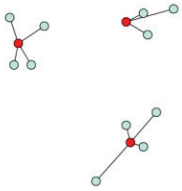


Illustration of the k-means algorithm

Create the new clustering

p=2
k=3
n=11

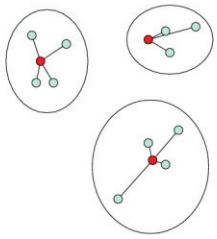


Illustration of the k-means algorithm

Create the new clustering

p=2
k=3
n=11

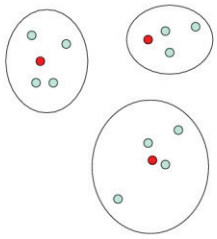


Illustration of the k-means algorithm

Calculate the new centroids of clusters

p=2
k=3
n=11

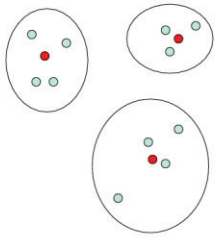


Illustration of the k-means algorithm

Assign the points to the closest centroids

p=2
k=3
n=11

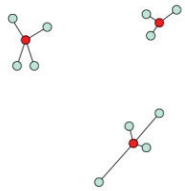


Illustration of the k-means algorithm

Create the new clustering

p=2
k=3
n=11

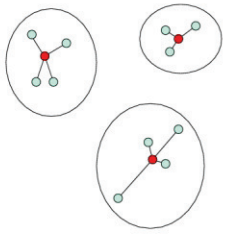
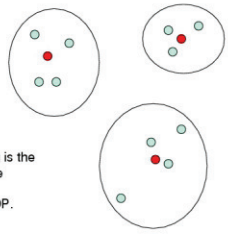


Illustration of the k-means algorithm

Create the new clustering

p=2
k=3
n=11



The clustering is the same as in the previous step, therefore STOP.

Properties of the k-means algorithm

Advantages:
- Simple to understand and implement.
- Fast and convergent in a finite number of steps.

Disadvantages:

- Different initial clusterings can lead to different final clusterings. It is thus advisable to run the procedure several times with different (random) initial clusterings.

- The resulting clustering depends on the units of measurement. If the variables are of different nature or are very different with respect to their magnitude, then it is advisable to standardize them.

- Not suitable for finding clusters with nonconvex shapes.

- The variables must be euclidean (real) vectors, so that we can calculate centroids and measure the distance from centroids; it is not enough to have only the matrix of pairwise distances or "dissimilarities".

Computational issues of k-means

Complexity: Linear in the number of objects, provided that we bound the number of iterations.

In R (library `stats`):

```
kmeans(x, centers, iter.max, nstart, algorithm)
```

- `x`: Dataframe of real vectors of features
- `centers`: The number of clusters
- `iter.max`: Maximum number of iterations
- `nstart`: Number of restarts
- `algorithm`: The method used (`"Hartigan-Wong"`, `"Lloyd"`, `"Forgy"`, `"MacQueen"`)

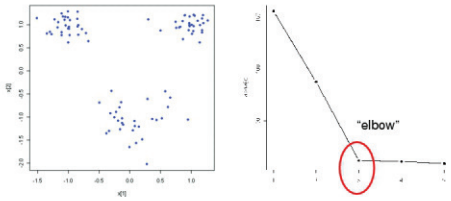
The "elbow" method

Selecting k – the number of clusters – is frequently a problem.

Often done by graphical heuristics, such as the elbow method.

$\alpha(k) = \sum_{i=1}^{k} \sum_{r \in C_i^{(k)}} \rho^2\left(x_r - c_i^{(k)}\right)$ $C_1^{(k)},\ldots,C_k^{(k)}$ … optimal clustering obtained by assuming k clusters

$c_1^{(k)},\ldots,c_k^{(k)}$ … corresponding centroids



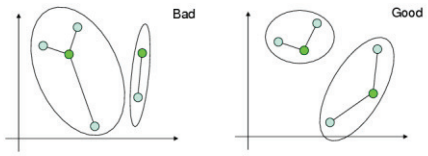
K-medoids clustering

Instead of centroids uses „**medoids**" – the most central objects (the „best representatives") of each cluster.

This allows using only „**dissimilarities**" $d(r,s)$ of all pairs (r,s) of the objects.

The aim is to find the clusters $C_1,\ldots,C_k$ that minimize the target function:

$\sum_{i=1}^{k} \sum_{r \in C_i} d(r, m_i)$ where for each i the medoid m_i minimizes $\sum_{r \in C_i} d(r, m_i)$



K-medoids algorithm

Similarly as for k-means, it is a difficult problem to find the clustering that minimizes the target function of the k-medoids problem. There are many efficient heuristics that find a „good", although not always optimal solution. Example:

Algorithm „Partitioning around medoids" (PAM)

- Randomly select k objects $m_1,\ldots,m_k$ as initial medoids.
- Until the maximum number of iterations is reached or no improvement of the target function has been found do:
  - Calculate the clustering based on $m_1,\ldots,m_k$ by associating each point to the nearest medoid and calculate the value of the target function.
  - For all pairs (m_i, x_s), where x_s is a non-medoid point, try to improve the target function by taking x_s to be a new medoid point and m_i to be a non-medoid point.

Properties of the k-medoids algorithm

Advantages:
- Simple to understand and implement.
- Fast and convergent in a finite number of steps.
- Usually less sensitive to outliers than k-means.
- Allows using general dissimilarities of objects.

Disadvantages:

- Different initial sets of medoids can lead to different final clusterings. It is thus advisable to run the procedure several times with different initial sets of medoids.

- The resulting clustering depends on the units of measurement. If the variables are of different nature or are very different with respect to their magnitude, then it is advisable to standardize them.

Computational issues of k-medoids

Complexity: At least quadratic, depending on the actual implementation.

In R (library `cluster`):

```
pam(x, k, diss, metric, medoids, stand,...)
```

- Dataframe of real vectors of features or a matrix of dissimilarities
- The number of clusters
- Metrics used (`euclidean`, `manhattan`)
- Standardize data? (`TRUE`, `FALSE`)
- Is `x` a dissimilarity matrix? (`TRUE`, `FALSE`)
- Vector of initial medoids

The silhouette

$a(r)$ … the average dissimilarity of the object r and the objects of the same cluster

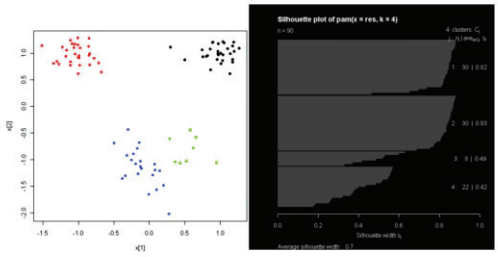
$b(r)$ … the average dissimilarity of the object r and the objects of the "neighboring" cluster

"Silhouette" of the object r … the measure of "how well" is r "clustered"

$$s(r) = \frac{b(r) - a(r)}{\max(b(r), a(r))} \in [-1,1]$$

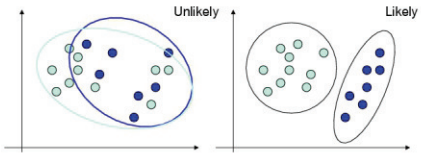
$s(r)$ close to 1 … the object r is well clustered
close to 0 … the object r is at the boundary of clusters
less than 0 … the object r is probably placed in a wrong cluster

The silhouette



Model-based clustering

- We assume that the vectors of features of objects from the j-th cluster follow a multivariate normal distribution $N_p(\mu_j, \Sigma_j)$.
- The method of calculating the clustering is based on maximization of a (mathematically complicated) likelihood function.
- Idea: Find the „most probable" (most „likely") assignment of objects to clusters (and, simultaneously, the most likely positions of the centers μ_j of the clusters and their covariance matrices Σ_j representing the „shape" of the clusters).



Model-based clustering

Disadvantages compared to k-means and k-medoids

- More difficult to understand properly.
- Computationally more complex to solve.
- Cannot use only dissimilarities (disadvantage compared to k-medoids).

Advantages over k-means and k-medoids

- Can find elliptic clusters with very high eccentricity, while k-means and k-medoids tend to form spherical clusters.
- The result is not dependent on the scale of variables (no standardization is necessary).
- Can find „hidden clusters" inside other more dispersed clusters.
- Allows a formal testing of the most appropriate number of clusters.

Computational issues of the model based clustering

Complexity: Computationally a very hard problem, solved iteratively. We can use the so-called **EM-algorithm**, or algorithms of stochastic optimization.

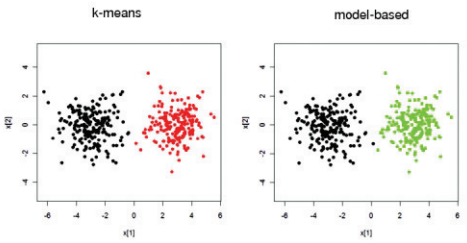
Modern computers can deal with problems with hundreds of variables and thousands of objects in a reasonable time.

In R (library `mclust`):

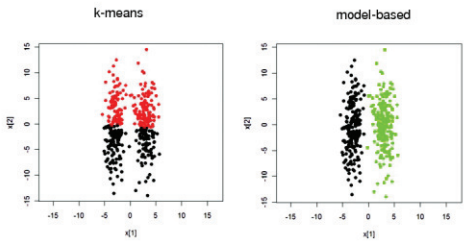
```
Mclust(data, modelNames,...)
```

- Dataframe of real vectors of features
- Model used (`EII, VII, EEI, VEI, EVI, VVI, EEE, EEV, VEV, VVV`)

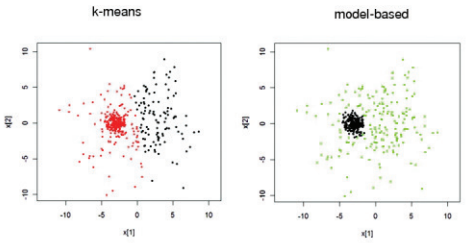
Comparison of nonhierarchical clustering methods on artificial 2D data



Comparison of nonhierarchical clustering methods on artificial 2D data

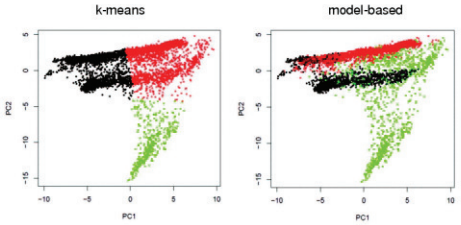


Comparison of nonhierarchical clustering methods on artificial 2D data



Comparison of nonhierarchical clustering methods on the Landsat data

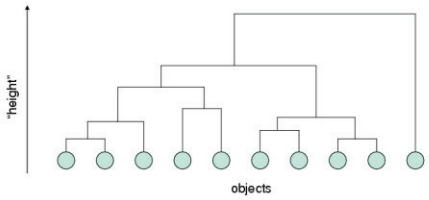
p=36 dimensional measurements of color intensity of n=4435 areas



Hierarchical clustering

- Creates a hierarchy of objects represented by a „tree of similarities" called **dendrogram**.
- Most appropriate to cluster "objects" that were formed by a process of "merging", "splitting", or "varying", such as countries, animals, commercial products, languages, fields of science etc.
- Advantages:
  - For most methods, it is enough to have the dissimilarity matrix D between objects: $D_{ij}=d(r,s)$ is the dissimilarity between objects r and s.
  - Does not require the knowledge of the number of clusters.
- Disadvantages:
  - Depends on the scale of data.
  - Computationally complex for large datasets.
  - Different methods sometimes lead to very different dendrograms.

Example of a dendrogram



The dendrogram is created either:
- „bottom-up" (**aglomerative**, or ascending, clustering), or
- „top-down" (**divisive**, or descending, clustering).

Agglomerative clustering

Algorithm:

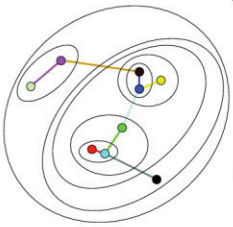
- Create the set of clusters formed by individual objects (each object forms an individual cluster).
- While there are more than one top-level clusters do:
  - Find the two top level clusters with the smallest mutual distance and join them into a new top level cluster.

Different measures of distance between clusters provide different variants: Single linkage, Complete linkage, Average linkage, Ward's distance

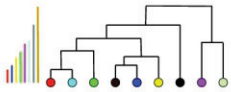
Single linkage in agglomerative clustering

- The distance of two clusters is the dissimilarity of the least dissimilar objects of the clusters:

$$D_S(C_i, C_j) = \min_{r \in C_i, s \in C_j} d(r,s)$$



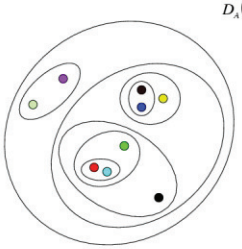
Dendrogram



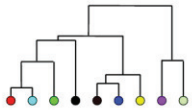
Average linkage in agglomerative clustering

- The distance of two clusters is the average of mutual dissimilarities of the objects in the clusters:

$$D_A(C_i, C_j) = \frac{1}{|C_i||C_j|} \sum_{r \in C_i} \sum_{s \in C_j} d(r,s)$$



Dendrogram



Other methods of measuring a distance of clusters in agglomerative clustering

- **Complete linkage**: the distance of clusters is the dissimilarity of the most dissimilar objects:

$$D_C(C_i, C_j) = \max_{r \in C_i, s \in C_j} d(r,s)$$

- **Ward's distance**: Requires that for each object r we have the real vector of features x_r. (The matrix of dissimilarities is not enough.) It is the difference between "an extension" of the two clusters combined and the sum of the "extensions" of the two individual clusters.

$$D_W(C_i, C_j) = \sum_{m \in C_i \cup C_j} \rho^2(x_m, c_{ij}) - \left[\sum_{r \in C_i} \rho^2(x_r, c_i) + \sum_{s \in C_j} \rho^2(x_s, c_j) \right]$$

$c_{ij}, c_i, c_j \ldots$ the centroids of $C_i \cup C_j, C_i, C_j$

$\rho \ldots$ the distance between vectors

Computational issues of agglomerative clustering

- **Complexity:** At least quadratic complexity with respect to the number of objects (depending on implementation).

In R (library `cluster`):

```
agnes(x, diss, metric, stand, method, ...)
```

- Dataframe of real vectors of features or a matrix of dissimilarities
- Is `x` a dissimilarity matrix? (`TRUE`, `FALSE`)
- Standardize data? (`TRUE`, `FALSE`)
- Metrics used (`euclidean`, `manhattan`)
- Method of measuring the distance of clusters (`single`, `average`, `complete`, `Ward`)

Divisive clustering

Algorithm:

- Form a single cluster consisting of all objects.
- For each "bottom level" cluster containing at least two objects:
  - Find the "most eccentric" object that initiates a "splinter group". (The object that has maximal average dissimilarity to other objects.)
  - Find all objects in the cluster that are more similar to the "most eccentric" object than to the rest of the objects. (For instance, the objects that have higher average dissimilarity to the eccentric object than to the rest of the objects.)
  - Divide the cluster into two subclusters accordingly.
- Continue until all "bottom level" clusters consist of a single object.

Illustration of the divisive clustering

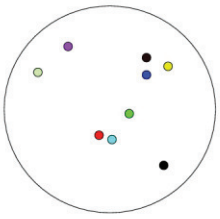


Illustration of the divisive clustering

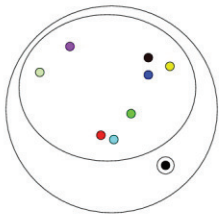


Illustration of the divisive clustering

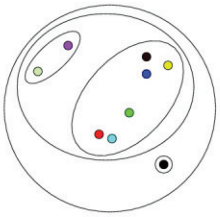


Illustration of the divisive clustering

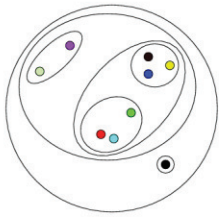


Illustration of the divisive clustering

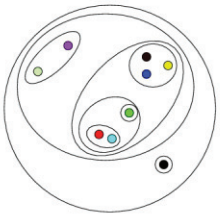


Illustration of the divisive clustering

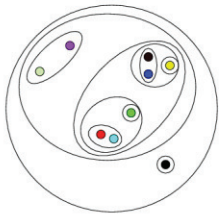


Illustration of the divisive clustering

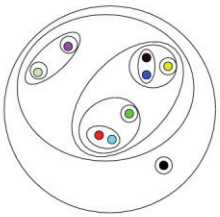


Illustration of the divisive clustering

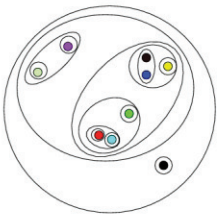
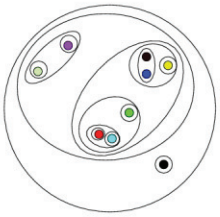
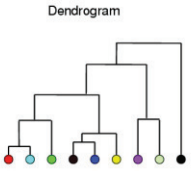


Illustration of the divisive clustering





Computational issues of divisive clustering

- **Complexity:** At least linear with respect to the number of objects (depending on implementation and a on the kind of the „splitting subroutine").

In R (library `cluster`):

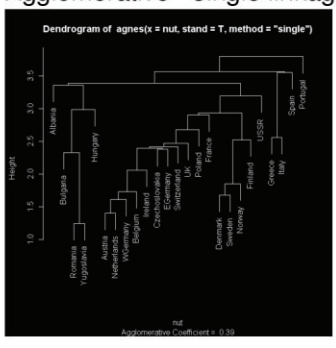
`diana(x, diss, metric, stand, …)`

Dataframe of real vectors of features or a matrix of dissimilarities	Is `x` a dissimilarity matrix? (`TRUE`, `FALSE`)	Metrics used (`euclidean`, `manhattan`)	Standardize data? (`TRUE`, `FALSE`)

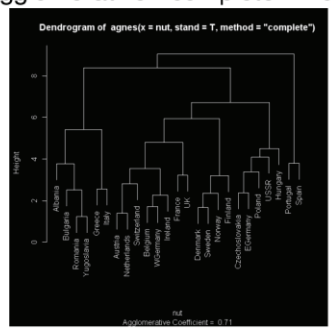
Comparison of hierarchical clustering methods

- n=25 objects - European countries (`Albania, Austria, Belgium, Bulgaria, Czechoslovakia, Denmark, EGermany, Finland, France, Greece, Hungary, Ireland, Italy, Netherlands, Norway, Poland, Portugal, Romania, Spain, Sweden, Switzerland, UK, USSR, WGermany, Yugoslavia`)
- p=9 dimensional vectors of features - consumption of various kinds of food (`Red Meat, White Meat, Eggs, Milk, Fish, Cereals, Starchy foods, Nuts, Fruits/Vegetables`)

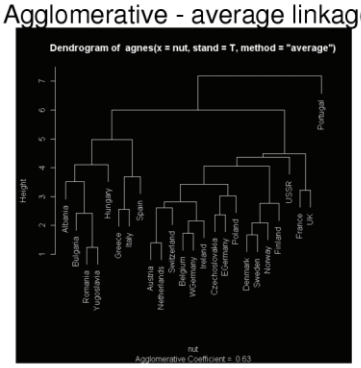
Agglomerative - single linkage



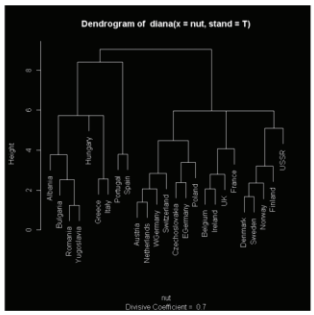
Agglomerative - complete linkage



Agglomerative - average linkage



Divisive clustering



Thank you for attention