Introduction to Clustering Techniques

Leo Wanner

IULA, July, 1st 2004

Introduction to Clustering Techniques

Definition 1 (Clustering) Clustering *is a division of data into groups of similar objects. Each group (= a cluster) consists of objects that are similar between themselves and dissimilar to objects of other groups.*
From the machine learning perspective, Clustering can be viewed as unsupervised learning of concepts*.*

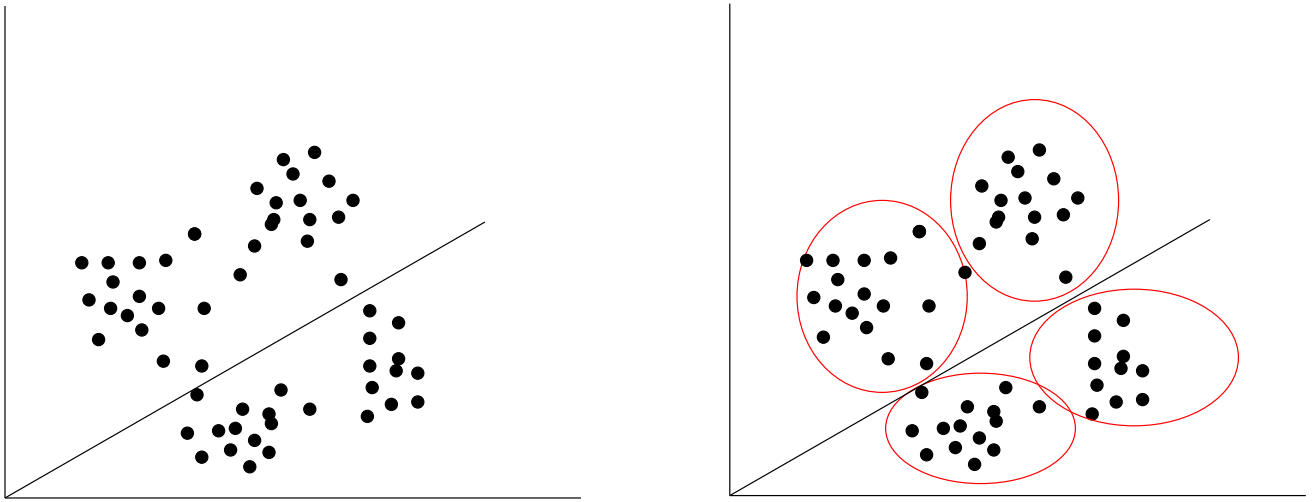
What can be clustered?

images (astronomical data), patterns (e.g., robot vision data), shopping items, feet (i.e., anatomical data), words, documents, . . .

Applications:

- Data Mining (DNA-Analysis, Marketing studies, Insurance Studies, . . .)
- Text Mining (Text Type Clustering)
- Information Retrieval (Document Clustering)
- Statistical Computational Linguistics (cluster-based n-gram models)
- Corpus-Based Computational Lexicography
- . . .

Problem of Clustering



We consider thus a data set X consisting of *data points* x_i ($1 \leq i \leq N$) (*objects instances*, *cases*, *patterns*, ...), where $x_i = \{a_{i_1}, a_{i_2}, \ldots, a_{i_d}\}$ and $a_{i_j} \in A$ is a numerical or nominal *attribute* from the *attribute space* A.

Why is Clustering Interesting for Us?

Document clustering:

- Improving precision and recall in Information Retrieval
- Browsing a collection of documents for purposes of Information Retrieval (scatter and gatherer strategy)
- Organizing the results provided by the search engine
- Generation of document taxonomies (cf. YAHOO!)

Linguistic Information Acquisition

- Induction of statistical language models
- Generation of subcategorization frame classes
- Generation of Ontologies
- Collocation Classification
- . . .

Clustering Algorithms

Requirements that should be satisfied by clustering algorithms:

- scalability
- dealing with different types of attributes
- discovering clusters of arbitrary shape
- ability to deal with noise and “outliers”
- high dimensionality
- insensitivity to the order of attributes
- interpretability and usability

Problems encountered with clustering algorithms:

- dealing with a large number of dimensions and a large number of objects can be prohibitive due to time complexity
- the effecteveness of an algorithm depends on the definition of similarity (distance)
- the outcomes of an algorithm can be interpreted in different ways

Classification of (Major) Clustering Algorithms (1)

I. Hierarchical Algorithms
- I.1 Agglomerative Algorithms
- I.2 Divisive Algorithms

II. Partitioning Algorithms
- II.1 K-medoids Algorithms
- II.2 K-means Algorithms
- II.3 Probabilistic Algorithms
- II.4 Density-Based Algorithms
  - II.4.1 Density-Based Connectivity Clustering
  - II.4.2 Density Functions Clustering

III. Grid-Based Algorithms

IV. Algorithms Based on Co-Occurrence of Categorial Data

V. Constraint-Based Clustering

VI. Evolutionary Algorithms

VII. Scalable Clustering Algorithms

VIII. Algorithms for High Dimensional Data
- VIII.1 Subspace Clustering
- VIII.2 Projection Algorithms, VIII.3 Co-Clustering

Classification of (Major) Clustering Algorithms (2)

More convenient classification:

- Hierarchical Clustering

  Establishing a cluster taxonomy

- Data Partitioning

  Establishing a set of FLAT partitions, i.e., NON-OVERLAPPING clusters

- Data Grouping

  Establishing a set of FLAT (possibly overlapping) clusters

Vector Space Model: The Root of Many Clustering Models

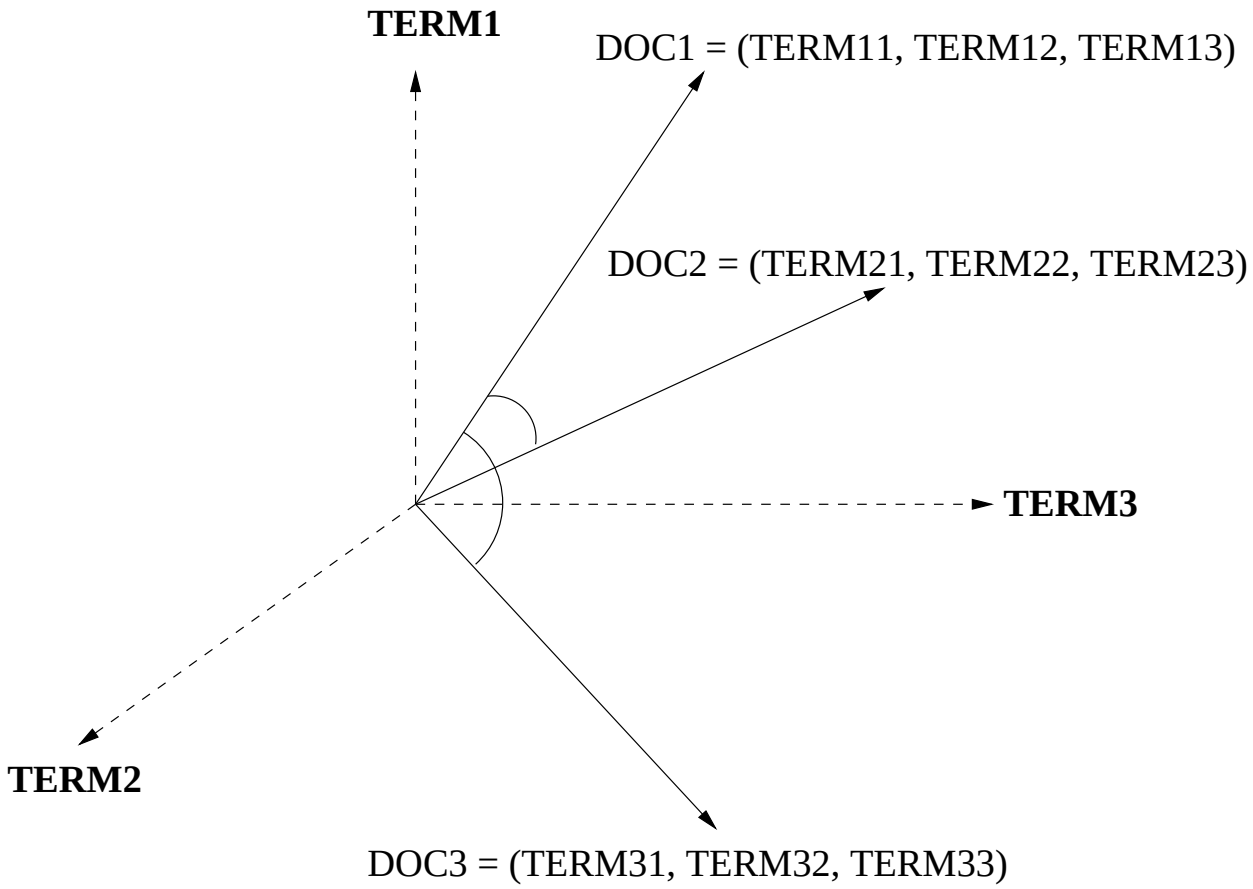
Basic idea:

Documents can be interpreted as points in a vector space, which is defined by all terms available in the document collection.

A document is thus represented by a term vector

$$< t_{i1}, t_{i2}, \ldots, t_{ij}, \ldots, t_{in} >$$

where t_{ij} is the weight of the term j in the document i



Vector Space Model: The Root of Many Clustering Models

Similarity measures between two term vectors; e.g.:

- **cosine**:

$$cos(doc_i, q_j) = \frac{\sum_{k=1}^{n} t_{ik} q_{jk}}{\sqrt{\sum_{k=1}^{n} {t_{ik}}^2 \sum_{k=1}^{n} {q_{jk}}^2}}$$

- **scalar product**:

$$< t_{i1}, t_{i2}, \ldots, t_{ik}, \ldots, t_{in} > \cdot < q_{j1}, q_{j2}, \ldots, q_{jk}, \ldots, q_{jn} >$$

Centroid: the “average” document in a given document subcollection:

$$C = \frac{1}{N} \Sigma_{i=1}^{N} doc_i$$

where N is the number of documents in the subcollection

Hierarchical Clustering

Hierarchical Clustering builds a cluster hierarchy (a tree of clusters).

1. ***a*gglomerative** (bottom-up) techniques
   starting with one point (singleton) clusters and recursively merging two or more most similar clusters to one "parent" cluster until the termination criterion is reached (e.g., k clusters have been built)

   vs.

2. ***d*ivisive** (top-down) techniques
   starting with one cluster of all objects and recursively splitting each cluster until the termination criterion is reached

Advantages:

- embedded flexibility regarding the level of granularity
- ease of handling any forms of similarity or distances
- applicability to any attribute types

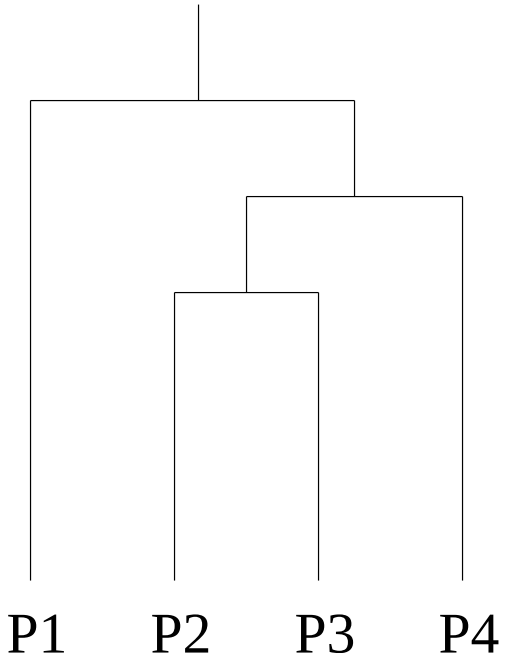
Disadvantages:

- vagueness of termination criteria
- most hierarchical algorithms do not revisit once-constructed (intermediate) clusters with the purpose of improvement

Hierarchical Clustering (cont.)

Simple Agglomerative Clustering Algorithm

1. Initialize the cluster set assuming each data point be a distinct cluster
2. Compute the similarity between all pairs of clusters, i.e., calculate the similarity matrix whose ijth entry gives the similarity between the ith and jth clusters.
3. Merge the most similar (closest) two clusters.
4. Updata the similarity matrix to reflect the pair-wise similarity between the new cluster and the original (remaining) clusters.
5. Repeat steps 3 and 4 until only a single cluster remains.



Hierarchical Clustering (cont.)

Linkage Metrics as Proximity Measures between Clusters

Let $D := N \times N$ be the dissimilarity matrix between individual objects (also called *connectivity matrix*), C_1, C_2 two clusters and $c_{1i} \in C_1$, $c_{2j} \in C_2$)

$d(C_1, C_2) = max(d(c_{1i}, c_{2j}) \in D)$ (*complete link*)

$d(C_1, C_2) = min(d(c_{1i}, c_{2j}) \in D)$ (*single link*)

$d(C_1, C_2) = \frac{1}{|C_1||C_2|} \Sigma_{i=1}^{|C_1|} \Sigma_{j=1}^{|C_2|} d(c_{1i}, c_{2j}) \in D$ (*average link*)

Other metrics are available: *centroid*, *median*, *minimum variance*, *cosine*, etc.

Restriction of linkage metrics based on Euclidean distance (as those above):

They naturally predispose to clusters of proper convex shapes. This is not always the case with real data!

Hierarchical Clustering, Example

Consider the following ten short texts:

1. No enchufe demasiado aparatos en el mismo enchufe ya que podría causar descargas eléctricas o incendio.
2. No abra la tapa posterior del aparato. En caso necesario, acuda al servicio técnico.
3. No obstruya ni cubra las ranuras de ventilación del aparato.
4. Como medida de seguridad, desenchufe el televisor antes de limpiarlo
5. Por razones de seguridad, se recomienda no dejar el aparato en modo de desconexion temporal cuando no se utilice. Apáguelo desenchufándolo cuando vaya a estar ausente.
6. Para su propia seguridad, durante una tormenta, no toque ninguna parte del apato.
7. Desenchufe el aparato tirando directamente de la clavija. Nunca tire del cable.
8. Para evitar riesgo de incendio o descargas eléctricas, no exponga el aparato a la lluvia ni a la humedad.
9. Para evitar riesgo de incendio, mantenga lejos del aparato los objetos inflamables.
10. No cubre las ranuras de ventilación del aparato con ningún objeto como cortinas, periodicos, etc.

Hierarchical Clustering, Example (cont.)

(abrir, acudir, apagar, aparato, ausente, causar, cable, clavija, cortina, cubrir, dejar, descarga, desconexion, desenchufar, directamente, eléctrica, enchufar, evitar, exponer, humedad, incendio, inflamable, lejos, limpiar, lluvia, mantener, medida, modo, necesario, objeto, obstruir, parte, periodico, posterior, ranura, razon, recomendar, riesgo, seguridad, servicio, tapa, técnico, televisor, temporal, tirar, tocar, tormenta, utilizar, ventilación, ver)

(0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 2, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)

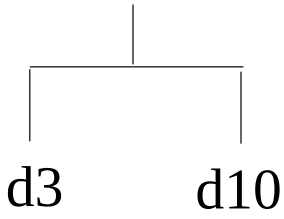
1. (0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 2, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)
2. (1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0)
3. (0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0)
4. (0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0)
5. (0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1)
6. (0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0)
7. (0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0)
8. (0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)
9. (0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)
10. (0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0)

Hierarchical Clustering, Example (cont.)

$$cos(d_i, d_j) = \frac{\sum_{k=1}^{n} t_k^i t_k^j}{\sqrt{\sum_{k=1}^{n} {t_k^i}^2 \sum_{k=1}^{n} {t_k^j}^2}}$$

	d_1	d_2	d_3	d_4	d_5	d_6	d_7	d_8	d_9	d_{10}
d_1		0.118	0.149	0.0	0.092	0.149	0.111	0.444	0.236	0.126
d_2			0.158	0.0	0.098	0.158	0.118	0.118	0.25	0.134
d_3				0.0	0.124	0.2	0.149	0.149	0.158	0.676
d_4					0.248	0.2	0.148	0.0	0.0	0.0
d_5						0.124	0.185	0.092	0.196	0.105
d_6							0.149	0.149	0.158	0.169
d_7								0.111	0.118	0.126
d_8									0.356	0.126
d_9										0.134
d_{10}										

Hierarchical Clustering, Example (cont.)



(0, 0, 0, 2, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0)

	d_1	d_2	$d_3 + d_{10}$	d_4	d_5	d_6	d_7	d_8	d_9
d_1									
d_2									
$d_3 + d_{10}$									
d_4									
d_5									
d_6									
d_7									
d_8									
d_9									

Alternative Hierarchical Clustering: CURE

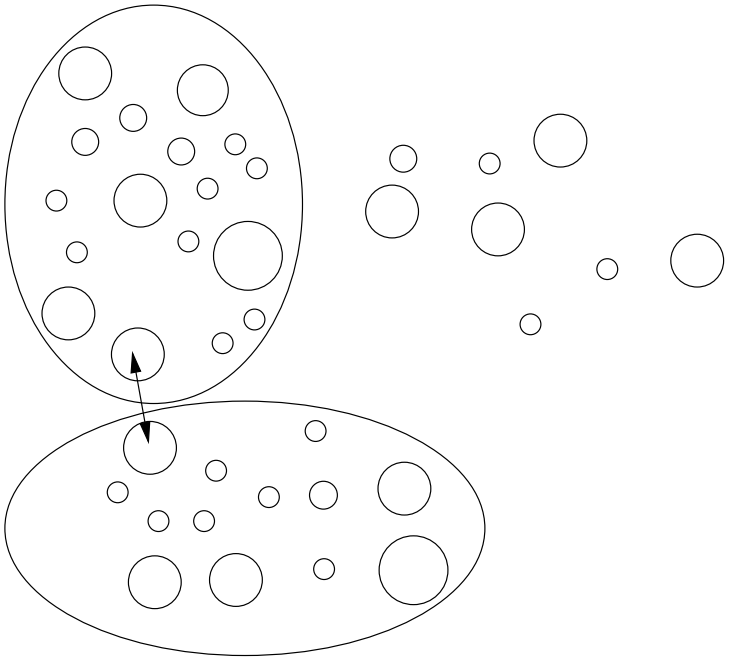
(to capture clusters of arbitrary shape)

1. Initialize the cluster set assuming each data point be a distinct cluster
2. For each cluster C_i, determine its closest cluster C_j (at this stage, the distance between C_i and C_j is simply the distance between the two points representing C_i and C_j, respectively).
3. Merge the closest pair of clusters C_k and C_l into the cluster C_m (as usual, C_m is the union of the data points (objects) in C_k and C_l).
4. For C_m, select c representative (*well scattered*) points:
   (a) select the point farthest from the mean
   (b) until c points have been selected:
   choose the point that is the farthest from the previously chosen one
5. Schrink the c points towards the mean by a fraction α ($p + \alpha * (mean - p)$)
6. Determine the cluster that is closest to C_m
7. Merge the closest pair of clusters
8. Repeat steps 3–7 until k clusters remain

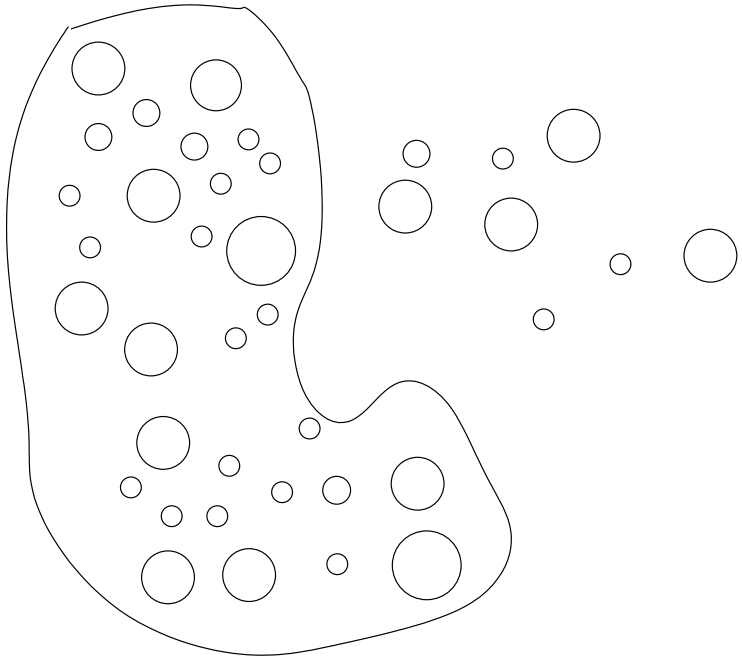
Alternative Hierarchical Clustering: CURE (cont.)

Result of CURE-clustering:

(a)



(b)



Partitioning Algorithms: K-Means

General characteristics of the approach:

— Each of the k clusters C_j is represented by the mean (or weighted average) c_j of its objects, the centroid.

— The clusters are iteratively recomputed to achieve stable centroids

A popular measure for the “intra-cluster variance” is the *square-error criterion*:

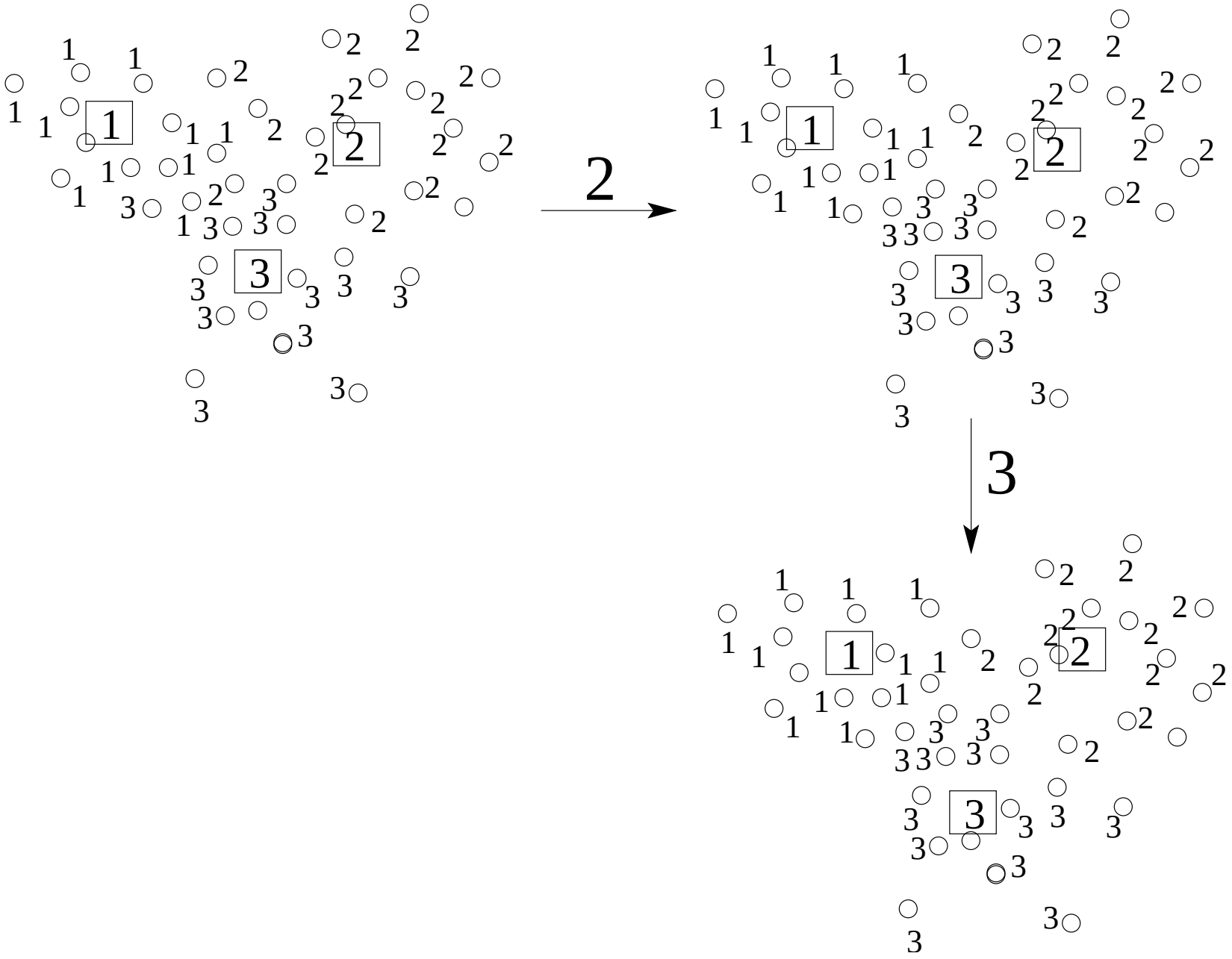
$$E = \Sigma_{i=1}^{k} \Sigma_{p \in C_i} ||p - m_i||^2$$

(with m_i as the mean of cluster C_i)

Basic K-Means Algorithm:

1. Select k data points as the initial centroids.
2. (Re)Assign all points to their closest centroids.
3. Recompute the centroid of each newly assembled cluster.
4. Repeat steps 2 and 3 until the centroids do not change.

Partitioning Algorithms: K-Means (Illustration)



Partitioning Algorithms: K-Means (cont.)

Advantages of the simple K-Means:

- It is easy to implement and works with any of the standard norms.
- It allows straightforward parallelization.
- It is insensitive with respect to data ordering

Problems with K-Means:

- The results strongly depend on the initial guess of centroids
- A local optimum (computed for a cluster) does not need to be a global optimum (overall clustering of a data set)
- It is not obvious what a good number K is in each case
- The process is sensitive with respect to outliers
- Resulting clusters can be unbalanced (even empty!!; cf. Forgy, 1965)
- . . .

Extension of K-Means (towards hierarchical clustering): Bisecting K-Means

Start with a single cluster of all documents.

1. Pick a cluster to split.
2. Find two subclusters using the basic K-means algorithm (bisecting step)
3. Repeat step 2 for n times and take the split that produces the clustering with the highest overall similarity.
4. Repeat steps 1, 2 and 3 until the desired number of clusters is reached.

Different ways to choose the cluster which is to be split: (a) largest cluster, (b) the most heterogeneous cluster, (c) the largest cluster which has a predetermined degree of heterogeneity, . . .

Partitioning Around K-Medoids

Given: a distance (or dissimilarity) matrix of he data points (= objects)

1. Determine a set of k *medoids*

   a 'medoid' is a representative object of a cluster whose average dissimilarity to all the objects in the cluster is minimal (cf. also *centrotypes* in the classification literature)

   $$M^* = argmin_M \Sigma_i min_k d(x_i, m_k)$$

   (a) Determine k "centrally" located objects $M = \{m_1, \ldots, m_k\}$ to be used as initial medoids.

   (b) If M^* can be reduced by interchanging (swapping) a selected object with an unselected object. Continue swapping until M^* can no longer be decreased.

2. Construct k clusters by assigning each object to its nearest medoid.

Partitioning Around K-Medoids, cont.

Two of the most difficult tasks in cluster analysis are:

- how to decide the appropriate number of clusters
- how to disitinguish a bad cluster from a good one

The k-Medoids algorithm family uses *silhouettes* to address these tasks.

> Each cluster is represented by one silhouette, showing which objects lie within the cluster, and which merely hold an intermediate position. The entire clustering is displayed by plotting all silhouettes into a single diagram, which thus illustrates the quality of the clusters.

Construction of a silhouette:

1. Consider any object i of the data set, and let A denote the cluster of i
2. Calculate the average distance a_i of i to all other objects in A

$$a_i = \frac{1}{N_A - 1} \Sigma_{j \in A, j \neq i} d(i, j)$$

Partitioning Around K-Medoids, cont.

3. Consider any cluster C different from A and define the average distance $d(i, C)$ of i to all objects of C

$$d(i, C) = \frac{1}{N_C} \Sigma_{c \in C} d(i, c)$$

4. Compute $d(i, C)$ for all clusters $C \neq A$, and then select the smallest of those: $b_i = min\ d(i, C), C \neq A$

Then, the *silhouette width* of i is defined as:

$$sil_i = \frac{b_i - a_i}{max(a_i, b_i)}$$

- Objects with large silhouette width are well-clustered, others tend to lie between clusters
- For a given number of clusters k, the overall *average silhouette width* for the clustering is the average over all objects i:

$$sil_{av} = \Sigma_i \frac{sil_i}{N}$$

- The number of clusters k can be determined by choosing a k that yields the highest average silhouette width.

Density-Based Partitioning

Basic Idea:

an open set of data points in the Euclidean space can be divided into a set of components, with each component containing connected points (i.e., points which are close to each other).

Density-Based Connectivity

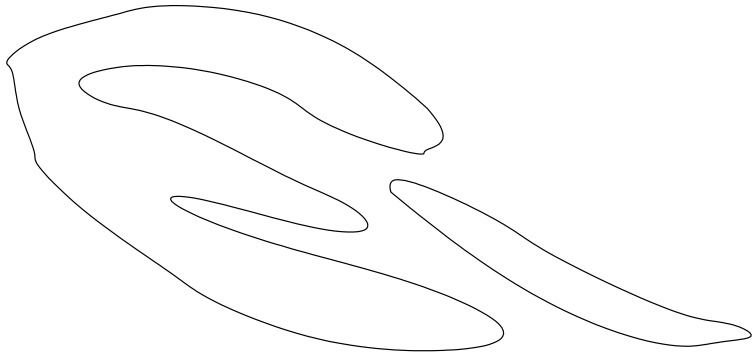
Important Notions:

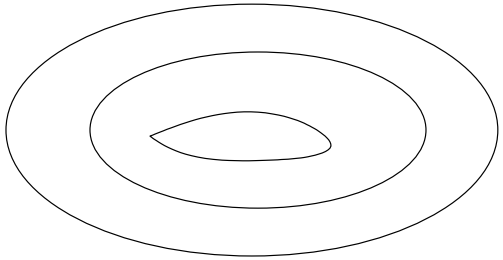
- ϵ-*neighborhood* $N_\epsilon(x) = \{y \in X | d(x, y) \leq \epsilon\}$.
- *Core object*: point with a neighborhood consisting of more then $MinPts$ points.
- A point y is *density-reachable* from a core object x if a finite sequence of core objects between x and y exists such that each next core object belongs to an ϵ-neighborhood of its predecessor.
- A *density connectivity* between two points x and y exists if both are density-reachable from a common core object.

Density-Based Partitioning, cont.

Basic algorithm:

1. Compute the ϵ-neighborhoods for all objects in the data space
2. Select a core object CO
3. For all objects $co \in CO$: add those objects y to CO which are "density connected" with co.
   Proceed until no further ys are encountered.
4. Repeat steps 2 and 3 until either all core objects have been processed or any of the non-processed core objects is connected to another previously already processed object.





Probabilistic Partitioning Clustering

Main Idea:

Use probabilities instead of distances!

Goal:

- find the most likely clusters given the data
- determine the probability with which an object belongs to a certain cluster

$$Pr(C|x) = \frac{Pr(x|C)Pr(C)}{Pr(x)} = \frac{Pr(x|C)Pr(C)}{\Sigma_C Pr(C)Pr(x|C)}$$

  where $Pr(C)$ is the probability that a randomly selected object belongs to cluster C, and $Pr(x|C)$ is the probability of observing the object x given the cluster C.

- Let us assume that we know that there are k clusters
- To learn the clusters, we need to determine their parameters (means and standard deviations)

Probabilistic Partitioning Clustering,cont.

The EM (estimation-maximization) algorithm

1. Calculate cluster probability (represented as object weights) for each object (estimation step).
2. Estimate distribution parameters based on the cluster probabilities (maximization step).
3. Procedure stops when log-likelihood saturates

   Log-likelihood: $\Sigma_i log(p_A Pr(x_i|A) + p_B Pr(x_i|B))$

Estimating parameters from weighted objects:

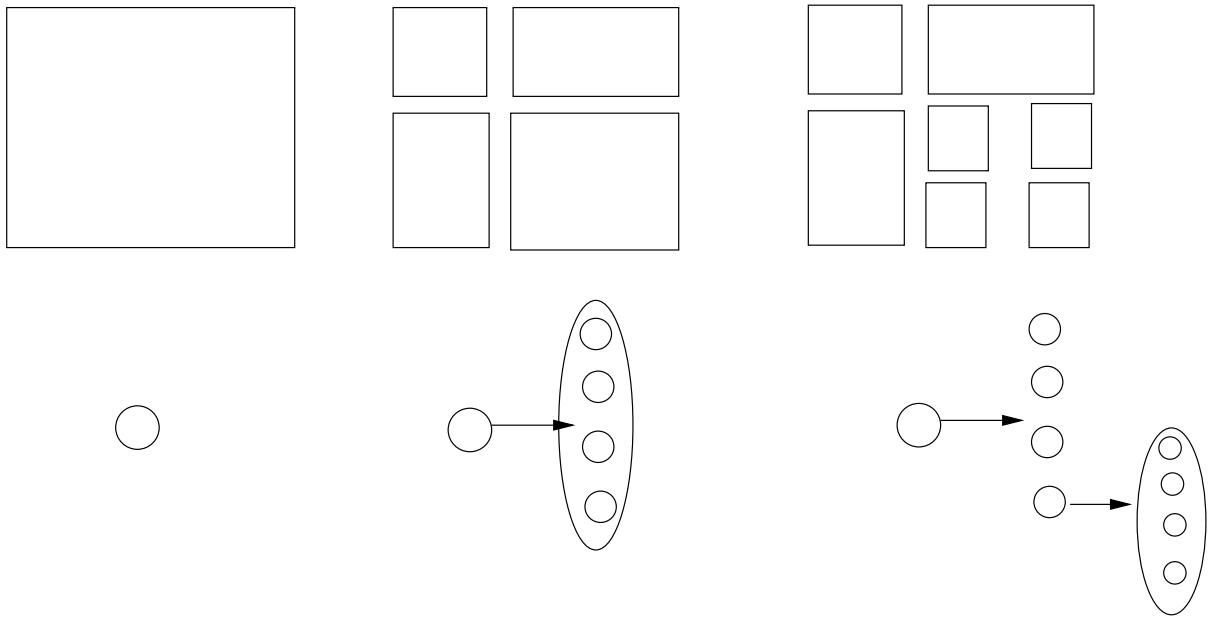
Mean of cluster A: $\mu_A = \frac{w_1 x_1 + w_2 x_2 + ... + w_n x_n}{w_1 + w_2 + ... + w_n}$

Standard deviation: $\sigma_A^2 = \frac{w_1 (x_1 - \mu)^2 + w_2 (x_1 2 - \mu)^2 + ... + w_n (x_n - \mu)^2}{w_1 + w_2 + ... + w_n}$

A Few Words on Other Types of Algorithms

- **Grid-based algorithms:**

  Attribute space partitioning: introduction of a grid; assigning of objects to grid cells.



- **Co-Occurrence of Categorial Data:**

  Data that belong to the same category are clustered depending on whether they occur in the same “transaction”.

  Often used for data mining: Cf. the transaction related to a shopping basket (with shopping items as items of the same category).

A Few Words on Other Types of Algorithms, cont.

- **Artificial Neural Networks**

  Towards “fuzzy” clustering (allowing an object to belong to several clusters

- **Evolutionary Algorithms**

  Example: **Genetic Algorithms**

  - A “population” of a set of “k-means” systems is given.
  - The population is improved through mutation and crossover.
  - Unlike in the standard k-means, clusters can have different size and elongation.

- **Constraint-Based Clustering**

  Allows for the specification of user constraints (exclusion of grids, restriction of the number of clusters, etc.)

  . . . Not really a new type of clustering techniques . . .