Clustering:
A survey

R. Capaldo F. Collovà

raffaele.capaldo@gmail.com francesco.collova@gmail.com

http://uroutes.blogspot.com/

Index

Definition of clustering

- **Clustering**

  Clustering is the process of grouping a set of objects into classes of *similar* objects.

  A **cluster** is a collection of data objects that are *similar* to one another within the same cluster and are *dissimilar* to the objects in other clusters.

  Clustering is regarded as a specific branch of the *Data Mining field* [1] .

- **Typical applications**
  - Image Processing and Pattern Recognition
  - Spatial Data Analysis

    Create thematic maps in GIS by clustering feature space

    Detect spatial clusters or for other spatial mining tasks
  - Economic Science (especially market research)
  - Document classification and clustering on the World Wide Web
  - Community Scouting and Sociometry data analysis

Definition of clustering

- **Requirements of a good clustering in Data Mining [(1)]**

  - Scalability: *The ability of the algorithm to perform well with a large number of data object (tuples).*
  - Ability to deal with different types of attributes: *The ability to analyze dataset with mixture of attribute types.*
  - Discovery of clusters with arbitrary shape: *It is important to develop algorithms which can detect clusters of arbitrary shape (in graphic data).*
  - Minimal requirements for domain knowledge to determine input parameters: *Many clustering algorithms require users to input certain parameters* (*such as k-number of desired clusters*). *Many parameters are hard to determine, because of the clustering results are often quite sensitive to Input parameters.*
  - Able to deal with noise and outliers: *Clustering algorithms should be able to handle deviations or outliers, in order to improve cluster quality.*

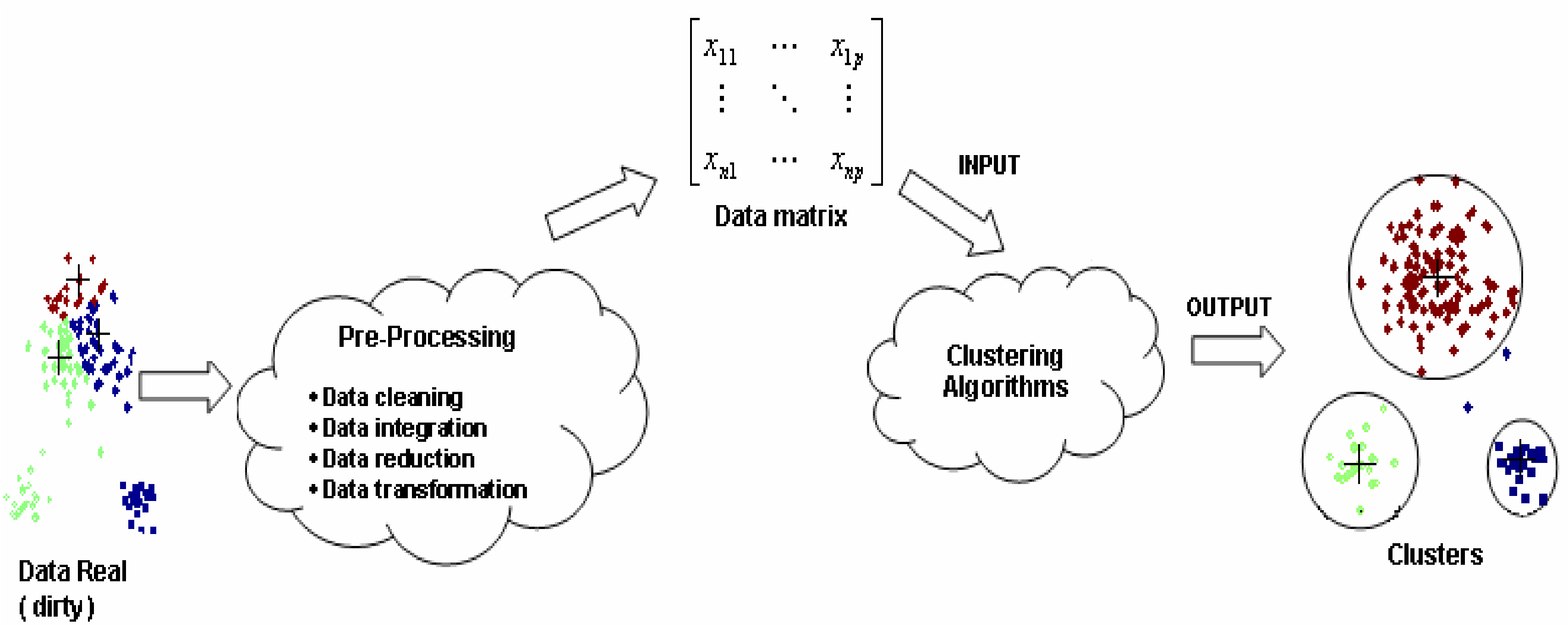
Definition of clustering

- Insensitive to order of input records: *The same data set, when presented to certain algorithms in different orders, may lead to dramatically different clustering. Thus it is important that algorithms be insensitive to the order of input.*
- High dimensionality: *The number of attributes or dimensions in many data sets is large, and many clustering algorithms can produce meaningful results only when the number of dimensions is small. Thus it is important that algorithms can produce results even if number of dimensions is high.*
- Incorporation of user-specified constraints: *Real applications may need to perform clustering under various kinds of constraints. Thus a good algorithm have to produce results even if data satisfying various constraints.*
- Interpretability and usability: *The clustering results should be interpretable, comprehensible and usable.*

(1) *Ref. J. Han and M. Kamber. Data Mining: Concepts and Techniques. Morgan Kaufmann Publishers, August 2000.*

Definition of clustering

- **Clustering process**



Definition of clustering

- **Data Pre-processing**

  - **Data cleaning**: *Fill in missing values, smooth noisy data, identify or remove outliers, and resolve inconsistencies.*
  - **Data integration**: *Integration of multiple databases, data cubes, or files.*
  - **Data transformation**: *Normalization and aggregation.*
  - **Data reduction**: *Obtains reduced representation in volume but produces the same or similar analytical results.*

Mathematical elements

- **Partition of a set (data set)**

• A **partition** P_j of a set ***X*** , with ***n*** objects (size(***X***) = ***n***) and *j = 1,2,…,**s*** , where $1 \leq s \leq n$, is a non empty subset of ***X*** such that every element ***x*** in ***X*** is in exactly one of these subsets P_j . Apply the following two equations:

1) The <u>union</u> of the elements of P_j is equal to ***X***. (We say the elements of P_j *cover* ***X***).

$$\boxed{\bigcup_{j=1}^{s} P_j = X}$$

2) The <u>intersection</u> of any two subsets P_j is empty. (We say the subsets P_j are <u>pairwise disjoint</u>)

$$\boxed{P_j \bigcap_{j \neq i} P_i = O} \qquad i, j = 1,2,\ldots,\boldsymbol{s}$$

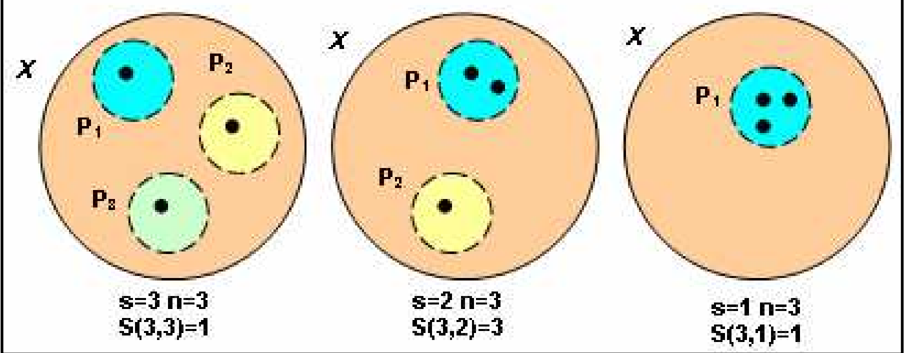
Mathematical elements

- **Example of partition**

For example, the set X={a, b, c } with ***n***=3 objects can be partitioned so:

- into three subsets $\{P_j\}_{j=1}^{3}$ in one way : { {a}, {b}, {c} }, where P_1={a}, P_2={b}, P_3={c}, and ***s***=**3**;
- into two subsets $\{P_j\}_{j=1}^{2}$ in three ways : { {a, b}, {c} }, { {a, c}, {b} },{ {a}, {b, c} }, where P_1={a,b} or {a,c} or {b,c}, P_2={c} or {b} or {a}, and ***s***=**2**;
- into one subset $\{P_j\}_{j=1}^{1}$ in one way : { {a, b, c} }, where P_1={a, b, c}, and ***s***=**1**.

FIG. 1 S(n,s) Stirling's numbers of II kind



The number of all possible partitions of data set **X** of **n** elements in to **s** non empty **sets** are Stirling's numbers of the second kind [16] :

$$\boxed{S(n,s)=\frac{1}{s!}\sum_{i=1}^{s}(-1)^{s-i}\binom{s}{i}i^{n}}$$

Mathematical elements

- **Geometric objects**
  - Let ***X***= ***R***^{n} *x* ***R***^{p} the **vector space** of **matrices** *n x p:* $(x_{ir})_{i,r=1}^{n,p}$

where *i* is row index: $1 \le i \le n$

and *r* is column index: $1 \le r \le p$

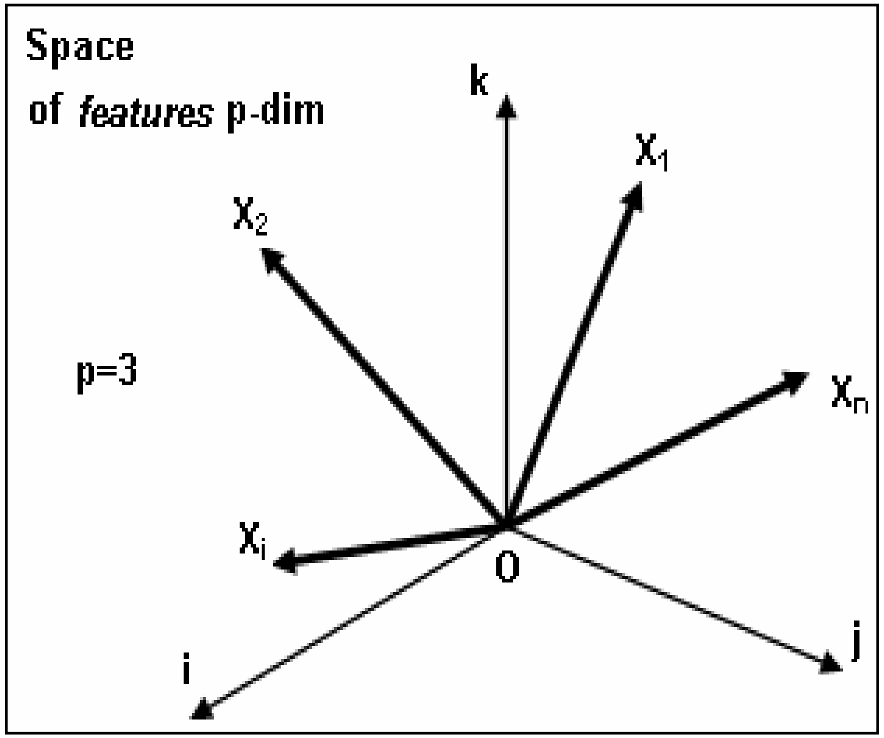
(*n* is number of ***objects*** and *p* number of ***variables*** or ***features*** or ***attributes***).

A generic element of ***X*** is

$$\boxed{\vec{x}_i = (x_{iq})_{q=1}^{p} = (x_{i1}, \ldots, x_{ip})}$$

and so we denot

x$= \{\vec{x}_1, \ldots, \vec{x}_n\} = \{\vec{x}_i\}_{i=1}^{n}$.



Mathematical elements

- **Geometric objects**

A ***dissimilarity*** (or ***distance measure***) between object $\vec{x}_i$ and $\vec{y}_j$ with *i,j =1,2,…,**n*** is matrix-function ***n**n***

$$\boxed{d(\vec{x}_i,\vec{y}_j): \quad X \times X \quad \rightarrow R}$$

which satisfied the following conditions:

$$\boxed{\begin{cases} d(\vec{x}_i,\vec{x}_j)=0 \\ d(\vec{x}_i,\vec{y}_j)\geq 0 \\ d(\vec{x}_i,\vec{y}_j)=d(\vec{y}_i,\vec{x}_j) \end{cases}}$$

For distance we require ***triangle inequality*** to satisfy, i.e.for any objects $\vec{x}_i$, $\vec{y}_j$, and $\vec{z}_k$

$$\boxed{d(\vec{x}_i,\vec{z}_k)\leq d(\vec{x}_i,\vec{y}_j)+d(\vec{y}_j,\vec{z}_k)}$$

A ***similarity*** between object $\vec{x}_i$ and $\vec{y}_j$ with *i,j =1,2,…,**n*** is matrix-function ***n**n***

$$\boxed{s(\vec{x}_i,\vec{y}_j): \quad X \times X \quad \rightarrow R}$$

which satisfied the following conditions:

$$\boxed{\begin{cases} s(\vec{x}_i,\vec{x}_j)=1 \\ s(\vec{x}_i,\vec{y}_j)\geq 0 \\ s(\vec{x}_i,\vec{y}_j)=s(\vec{y}_i,\vec{x}_j) \end{cases}}$$

Mathematical elements

- **Geometric objects**

Here are some ***distance*** used most frequently between object $\vec{x}_i$ and $\vec{x}_j$ with *i,j =1,2,…,**n***

- ***Minkowski*** (with q ≥1)

$$\boxed{d(\vec{x}_i,\vec{x}_j)_{\text{Minkowski}} = \sqrt[q]{\sum_{r=1}^{p} | x_{ir} - x_{jr} |^q}}$$

- ***Euclidean*** (with q =2)

$$\boxed{d(\vec{x}_i,\vec{x}_j)_{\text{Euclidean}} = \sqrt{\sum_{r=1}^{p} | \vec{x}_{ir} - \vec{x}_{jr} |^2}}$$

- ***Manhattan*** (with q =1)

$$\boxed{d(\vec{x}_i,\vec{x}_j)_{\text{Manhattan}} = \sum_{r=1}^{p} | \vec{x}_{ir} - \vec{x}_{jr} |}$$

Some ***similarity*** used most frequently between object $\vec{x}_i$ and $\vec{x}_j$ with *i,j =1,2,…,**n***

- ***Pearson product correlation***

$$\boxed{s(\vec{x}_i,\vec{x}_j)_{\text{Pearson}} = \frac{\sum_{r=1}^{p}(x_{ir}-\bar{x}_r)(x_{jr}-\bar{x}_r)}{\sqrt{\sum_{r=1}^{p}(x_{ir}-\bar{x}_r)^2}\sqrt{\sum_{r=1}^{p}(x_{jr}-\bar{x}_r)^2}}}$$

- ***Jaccard coefficient***

$$\boxed{s(\vec{x}_i,\vec{x}_j)_{\text{Jaccard}} = \frac{\sum_{r=1}^{p} x_{ir}x_{jr}}{\sqrt{\sum_{r=1}^{p}(x_{ir})^2}\sqrt{\sum_{r=1}^{p}(x_{jr})^2}}}$$

where $\bar{x}_r = \frac{1}{n}\sum_{i=1}^{n} x_{ir}$ is ***mean value*** of ***n*** objects.

Clustering: Data Typing[17]

- Data are a collection of **objects** and their **attributes**.
- An **attribute** is a property or characteristic of an object.
  - Examples: eye color of a person, temperature, etc.
  - Attribute is also known as variable, field, characteristic, or feature.
- A collection of attributes describe an **object**
  - Object is also known as record, point, case, sample, entity, or instance.

Attributes

Objects

Tid	**Refund**	**Marital Status**	**Taxable Income**	**Cheat**
1	Yes	Single	125K	**No**
2	No	Married	100K	**No**
3	No	Single	70K	**No**
4	Yes	Married	120K	**No**
5	No	Divorced	95K	**Yes**
6	No	Married	60K	**No**
7	Yes	Divorced	220K	**No**
8	No	Single	85K	**Yes**
9	No	Married	75K	**No**
10	No	Single	90K	**Yes**

Clustering: Data Typing[17]

- There are different types of **attributes**

  - **Binary:** only two states: variable is absent or present.
    - Examples: *male and female, on and off.*
  - **Nominal:** generalization of the binary variable in that it can take more than 2 states
    - Examples: *ID numbers, eye color, zip codes.*
  - **Ordinal:** An ordinal $\boldsymbol{q}$ variable can be discrete or continuous and his $M_{\boldsymbol{q}}$ values can be mapped to ranking: 1,…, $M_{\boldsymbol{q}}$.
    - Examples: *rankings (e.g., taste of potato chips on a scale from 1-10), grades, height in {tall, medium, short}.*
  - **Ratio-scaled:** makes a positive measurement on a scale, such as at exponential scale: Ae^{Bt} or Ae^{-Bt}.
    - Examples: *temperature in Kelvin, length, time, counts.*
  - **Variables of Mixed Types:** a collection of all other previous variables.
    - Examples: *temperature in Kelvin, grades, ID numbers, counts.*
  - **Categorical:** when there is no inherent distance measure between data values.
    - Examples: *We consider a relation that stores information about movies. A movie is a object or tuple characterized by the values or attributes: 'director', 'actor/actress', and 'genre'.*

Clustering: Data Typing

- **Dissimilarities and similarities of clustering (quality of clustering)**
  - Suppose that ***D*** is data set which contains ***n*** objects, where each has ***p*** variables or attributes. Mathematically data structure is in the form of relational table and there are two types of rappresentation:

Data matrix (<u>two mode</u>: *n* objects x *p* variables)

$$(x_{iq})_{i=1,q=1}^{n,p} = \begin{bmatrix} x_{11} & \cdots & x_{1p} \\ \vdots & \ddots & \vdots \\ x_{n1} & \cdots & x_{np} \end{bmatrix}$$

Dissimilarity matrix (<u>one-mode</u>: *n* objects x *n* objects)

$$[\delta(i,j)]_{i=1,j=1}^{n,n} = \begin{bmatrix} 0 & \cdots & \delta(2,n) & \delta(1,n) \\ \delta(2,1) & & \ddots & \\ \vdots & & & \vdots \\ \delta(n,1) & \delta(n,2) & \cdots & 0 \end{bmatrix}$$

  - <u>Clusters of objects</u> are computed based on their **Similarities** or **Dissimilarities**

Similarities

$$[s(i,j)]_{i=1,j=1}^{n,n} = \begin{cases} [s(\vec{x}_i,\vec{x}_j)_{\text{Pearson}}]_{i=1,j=1}^{n,n} \\ or \\ [s(\vec{x}_i,\vec{x}_j)_{\text{Jaccard}}]_{i=1,j=1}^{n,n} \end{cases}$$

Dissimilarities

$$[d(i,j)]_{i=1,j=1}^{n,n} = \begin{cases} [1-s(i,j)]_{i=1,j=1}^{n,n} \\ or \\ [d(\vec{x}_i,\vec{x}_j)_{\text{Minkowski}}]_{i=1,j=1}^{n,n} \end{cases}$$

Clustering: Introduction

- **Partitioning methods** constructs a partition of a database ***D*** of ***n*** objects into a set ***K*** $=\{C_1, C_2, \ldots, C_k\}$ of ***k*** clusters C_r, with $r = 1, 2, \ldots$***k*** and $1 \leq k \leq n$, to minimize a *similarity function,* such as *distance*, so that the objects in a cluster are 'similar', but the objects of different clusters are 'dissimilar'.
  The general method works as follow:
  given ***k*** clusters, an initial partition is made; ***n*** objects are than moved between partitions in an attempt to improve a *similarity function.*
  To find a global optimum solution we need to consider all possible partitions, whose numbers coincide with Stirling's number of the second kind *S(n,k)* [16].
  With increasing ***n*** this soon becomes impossible, so partitioning algorithms don't consider all partitions and may only find a local optima: heuristic methods.
  The most important heuristic methods are:

  - **k-means** [2],
  - **k-medoids or PAM** [3],
  - **CLARA** [4],
  - **CLARANS** [5].

Partitioning methods: K-means

- **Algorithm (K-means)**

Input: ***k*** clusters, ***n*** objects of database ***D***.
Output: A set of ***k*** clusrers which minimizes the squared-error function *E*.

Algorithm:

1) Choose ***k*** objects as the initial **cluster centers**.
2) Assign each object to the cluster which has the closest *mean point* (*centroid*) under squared Euclidean distance metric.
3) When all objects have been assigned, recalculate the positions of *k mean point* (*centroid*).
4) Repeat Steps 2) and 3) until the *centroids* do not change any more.

Pseudo Code

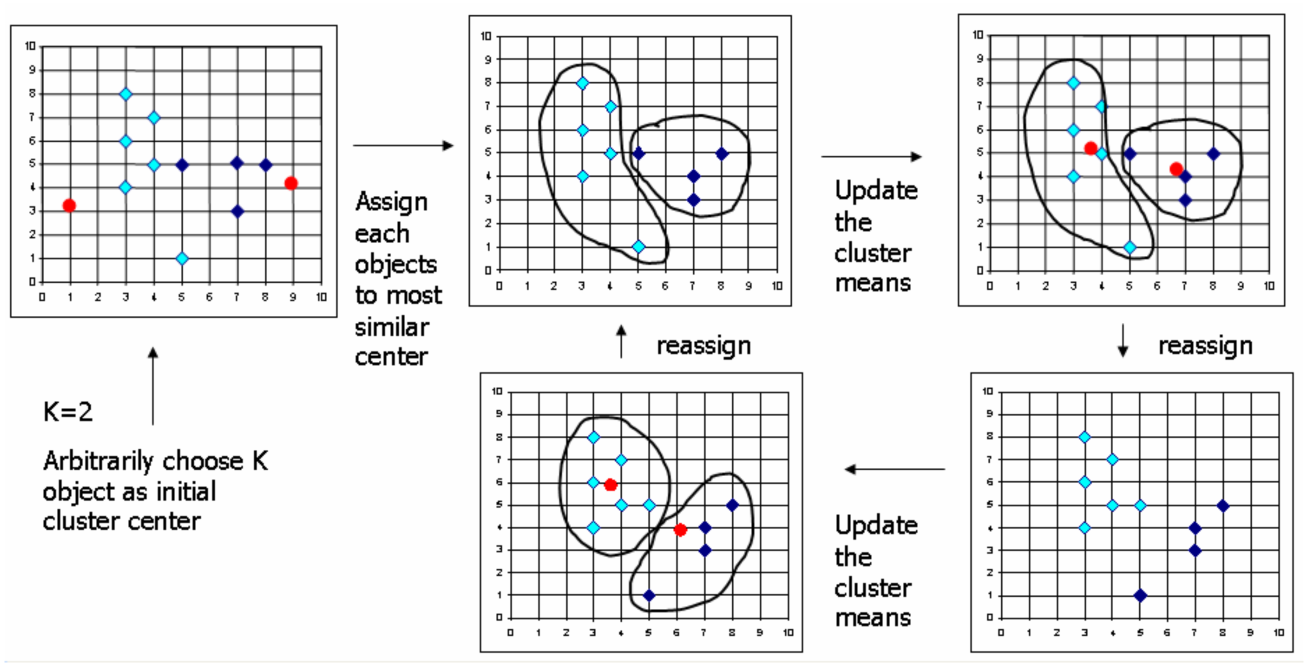
```
Input:   k                          // Desired number of clusters
         D={$\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n$}     // Set of elements
Output:  K={$C_1, C_2, \ldots, C_k$} // Set of k clusters which
         minimizes the squared-error function E (*)
K-means algorithm
    1) assign initial values for
       means point $\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \ldots \boldsymbol{\mu}_k$     // k seeds (**)
    repeat
        2.1) assign each item $\mathbf{x}_i$ to the cluster which
             has closest mean;
        2.2) calculate new mean for each cluster;
    until convergence criteria is meat;
```

(*) $E = \Sigma_{r=1}^{k} \Sigma_{i=1, \vec{x}_i \in C_r}^{n} (\vec{x}_i - \vec{\mu}_r)^2$

(**) $\vec{\mu}_r = \dfrac{1}{size(C_r)} \sum_{i=1, \vec{x}_i \in C_r}^{n} \vec{x}_i$

Partitioning methods: K-means

- **Example (K-means)**



Assign each objects to most similar center
Update the cluster means
reassign
reassign
Update the cluster means
K=2
Arbitrarily choose K object as initial cluster center

Partitioning methods: K-means

- **Comment (K-means)**

• The ***K-means*** method is is *relatively efficient*: *O*(***tkn***), where ***n*** is objects number, ***k*** is clusters number, and ***t*** is iterations number. Normally, ***k***, ***t*** << ***n***.

• Often terminates at a *local optimum*. The *global optimum* may be found using techniques such as: *deterministic annealing* [6] and *genetic algorithms* [7] *.*

• **Weakness** :

- Not applicable in categorical data .

- Need to specify ***k**,* the *number* of clusters, in advance.

- Unable to handle noisy data and *outliers .*

- Not suitable to discover clusters with *non-convex shapes .*

- To overcome some of these problems is introduced the k-medoids or PAM method.

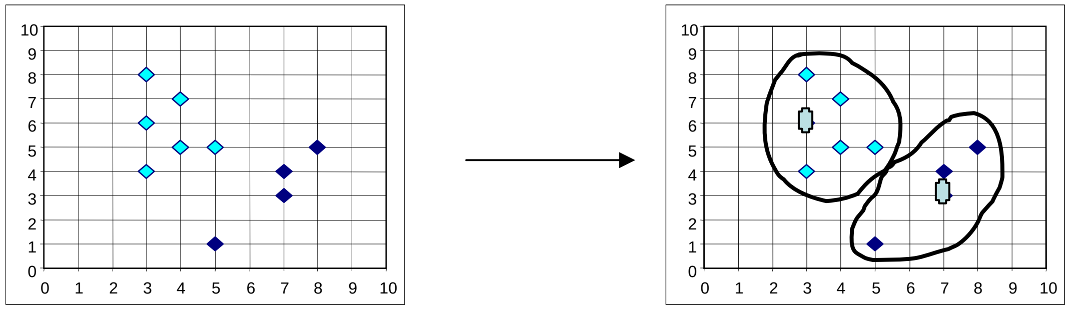
Partitioning methods: K-medoid or PAM

- **Definition of medoid**

  - The method **K-medoid** or **PAM** (**P**artitioning **A**round **M**edoids) uses **medoid** ***m***$_q$ (*q = 1,2,…,**k***) as object more representative of cluster.

    medoid is the **most centrally located** object in a cluster.

    Infact instead of taking the **mean** value of the object in a cluster as a reference point (see K-means), can be used **medoid** .



Partitioning methods: K-medoid or PAM

- **Algorithm (K-medoid or PAM)**

Input: ***k*** clusters, ***n*** objects of database ***D***.

Output: A set of ***k*** clusrers which minimizes the sum of the dissimilarities $\delta(\vec{x}_j, \vec{m}_q)$ of all ***n*** objects to their nearest *q*-th medoid (*q = 1,2,…,**k***).

Algorithm:

```
1)  Randomly choose k objects from the data set to be the cluster
    medoids at the initial state.
2)  For each pair of non-selected object h and selected object i, calculate the total
    swapping cost $T_{ih}$.
3)  For each pair of i and h,
    - If $T_{ih} < 0$, i is replaced by h
    - Then assign each non-selected object to the most simila representative object.
4)  Repeat steps 2 and 3 until no change happens.
```

Calculate T_{ih} , the ‘ total swap contribution’ for the pair of objects (***i,h***), as

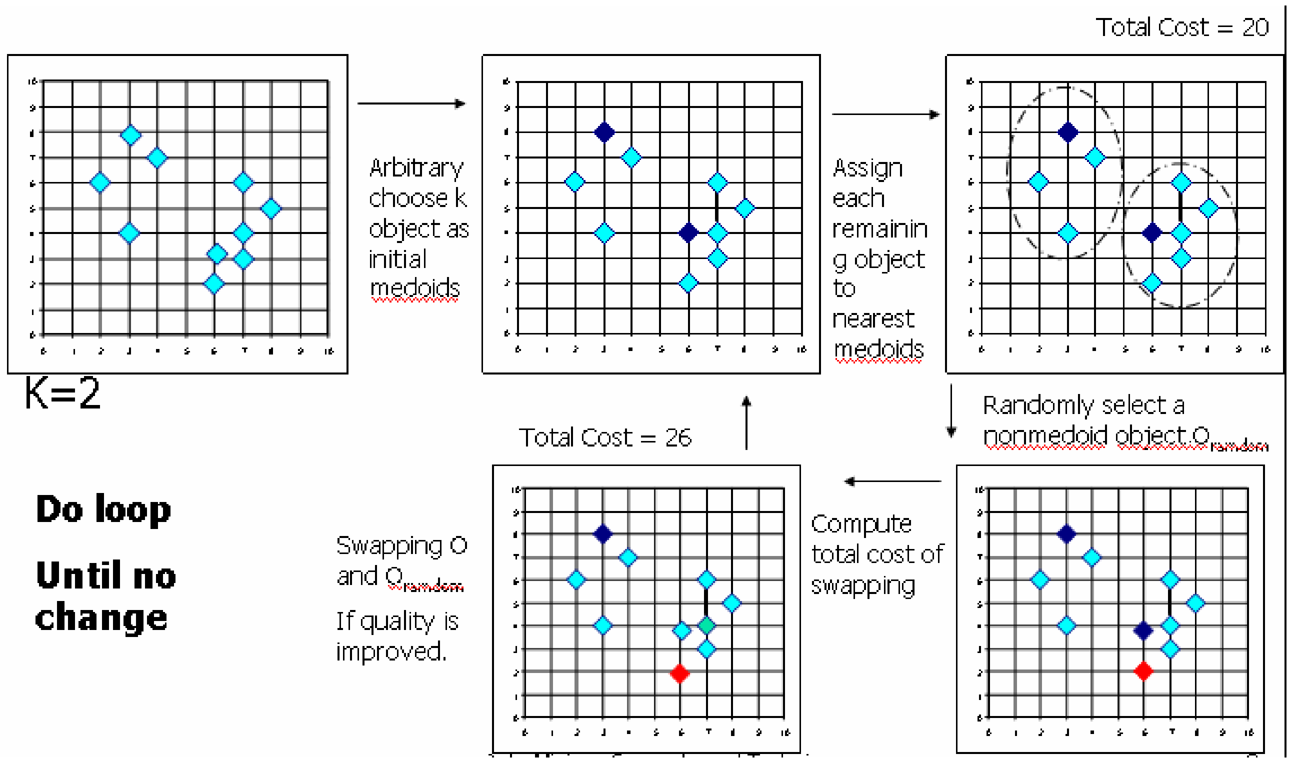
$$\boxed{T_{ih} = \sum_{j=1} C_{jih}}$$

where C_{jih} is the contribution to swapping the pair of objects (***i,h***) (***i*** <-> ***h***) from object *j*, defined below.

There are *four* possibilities to consider when calculating C_{jih}, see Tab.1 in Appendix.

Partitioning methods: K-medoid or PAM

- **Example (K-medoid or PAM)**


Total Cost = 20
Arbitrary choose k object as initial medoids
Assign each remaining object to nearest medoids
K=2
Randomly select a nonmedoid object, O_{random}
Total Cost = 26
Do loop
Until no change
Swapping O and O_{random}
If quality is improved.
Compute total cost of swapping

Partitioning methods: K-medoid or PAM

- **Comment (K-medoid or PAM)**

• **PAM** is more robust than **K-means** in the presence of <u>noise</u> and <u>outliers</u> because a medoid is less influenced by outliers or other extreme values than a mean.

• <u>**Weakness**</u>:

PAM works efficiently for small data sets but does not **scale well** for large data sets. Infact: O($\boldsymbol{k}(\boldsymbol{n}-\boldsymbol{k})^2$) for each iteration where ***n*** is data numbers, ***k*** is clusters numbers.

To overcome these problems is introduced :

• <u>**CLARA**</u> (**C**lustering **LAR**ge **A**pplications) **- >** <u>Sampling based method</u>

• <u>***CLARANS***</u> **- >** <u>A Clustering Algorithm based on Randomized Search</u>.

Partitioning methods: CLARA

- **CLARA** (**C**lustering **LAR**ge **A**pplications) is a method that instead of taking the whole set of data into consideration, only a small portion of the real data is chosen (in random manner) as a representative of the data, and **medoids** are chosen from this sample using **PAM**.
- Deals with larger data sets than **PAM**.
- **Weakness**:
  - Efficiency depends on the sample size.
  - A good clustering based on samples will not necessarily represent a good clustering of the whole data set if the sample is biased.

Partitioning methods: CLARANS

- **CLARANS** (**“Randomized” CLARA**) is a method that draws sample of neighbors dynamically.
- The clustering process can be presented as searching a graph where every node is a potential solution, that is, a set of ***k* medoids***.*
- If the local optimum is found, **CLARANS** starts with new randomly selected node in search for a new local optimum.
- It is more efficient and scalable than both **PAM** and **CLARA**

Hierarchical methods: Introduction

- **Hierarchical clustering methods** works by grouping data objects into e tree of clusters and uses distance matrix as clustering criteria.This method does not require the number of clusters ***k*** as an input, but needs only number of objects ***n*** and a termination condition.
- There are two principal types of hierarchical methods:
  Agglomerative (bottom-up): merge clusters iteratively.
  - Start by placing each object in its own cluster;
  - merge these atomic clusters into larger and larger clusters;
  - until all objects are in a single cluster.

  Most hierarchical methods belong to this category.

  They differ only in their definition of ***between-cluster similarity***.

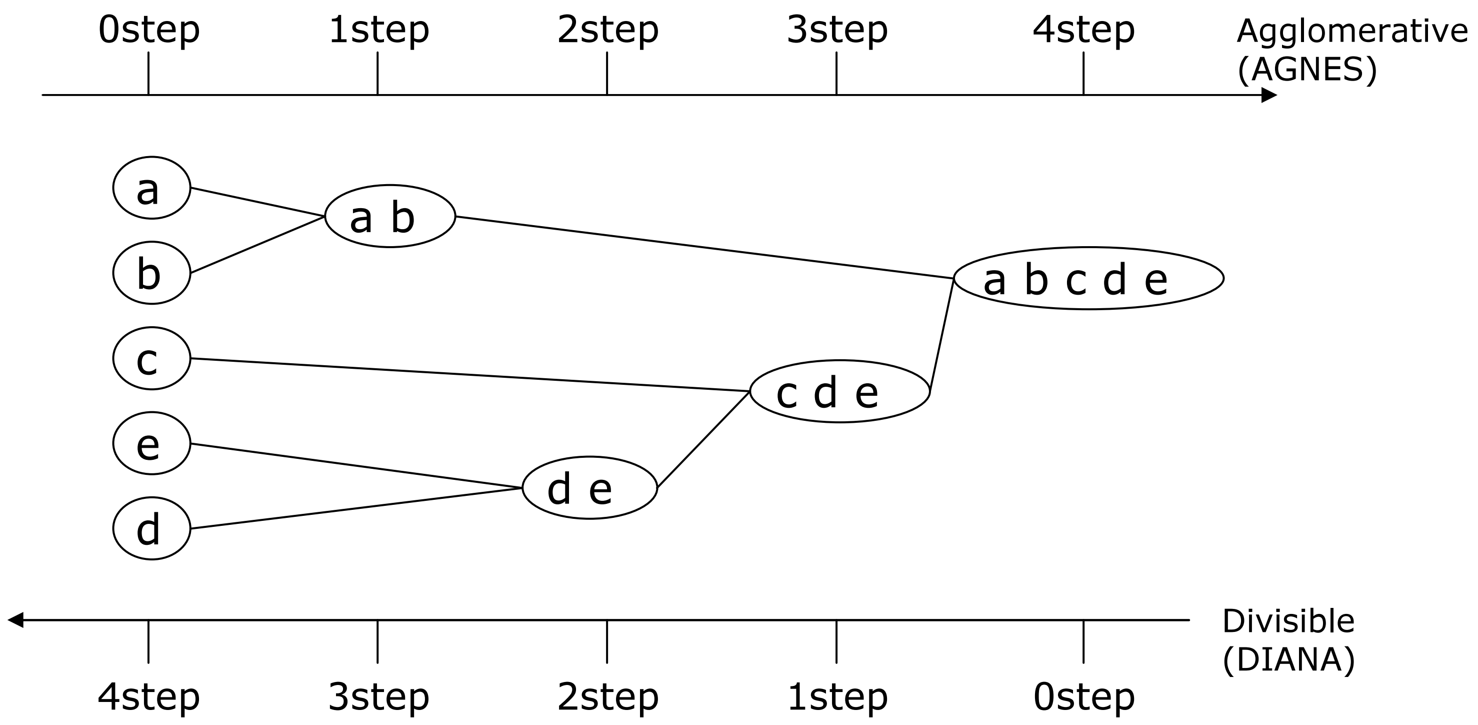
  An example is **AGNES** (**AG**glomerative **NES**ting), [8].

  Divisive (top-down):split a cluster iteratively.
  - It does the reverse by starting with all objects in one cluster and subdividing them into small pieces. Divisive methods are not generally available, and rarely have been applied.

    An example is **DIANA** (**DI**visive **ANA**lysis), [9].

Hierarchical methods: Introduction

- Application of **AGNES** and **DIANA** to a data set of five objects, {a, b, c, d, e}.



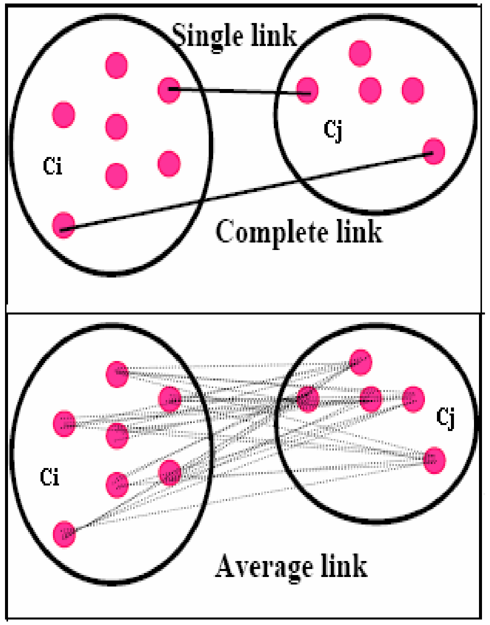
Hierarchical methods: Distance between clusters

- *Merging of clusters* is based on the **distance between clusters**:
- **Single-Linkage**: it is the shortest distance from any member P of one cluster C_i to any member P' of the other cluster C_j.

$$\boxed{d_{\min}(C_i, C_j) = \min_{p \in C_i, p' \in C_j} |p - p'|}$$

- **Complete-Linkage**: it is the the greatest distance from any member P of one cluster C_i to any member P' of the other cluster C_j.

$$\boxed{d_{\max}(C_i, C_j) = \max_{p \in C_i, p' \in C_j} |p - p'|}$$



- **Average-Linkage**: it is the the average distance between each element in one cluster C_i and each element in the other C_j.

$$\boxed{d_{avg}(C_i, C_j) = \frac{1}{n_i n_j} \sum_{p \in C_i} \sum_{p' \in C_j} | p - p'|}$$

Hierarchical methods: Agglomerative Algorithm

- **Algorithm with Single-Linkage (Pseudo Code)**[2]

```
Input: D={x_1, x_2,…, x_n}   // Set of elements;
    A  //  n*n proximity  or adyacency matrix  A = [d(i,j)] that showing distance between x_i , x_j ;
    C_r  //  r-th cluster, with  1 ≤ r ≤. n ;  d [ C_r ,C_s]   // Proximity between clusters C_r and C_s;
    k  // Sequence number, with k=0,1,…n -1;  L(k)  // Distance-level of the k-th clustering;
Output:    //  Dendrogram;
Algorithm:
1. Begin with n clusters, each containing one object and having level L(0) = 0 and sequence number k = 0.
2. Find the least dissimilar pair (C_r ,C_s) in the current clustering, according to
    $d[C_r ,C_s ] = \mathbf{min}(d[C_i ,C_j ])$
   where the minimum is over all pairs of clusters (C_i ,C_j) in the current clustering.
3. Increment the sequence number : k = k +1. Merge clusters C_r and C_s into a single cluster to form the
   next clustering k. Set the level of this clustering to L(k) = d[C_r ,C_s].
4. Update the proximity matrix, D, by deleting the rows and columns corresponding to clusters C_r and C_s
   and adding a row and column corresponding to the newly formed cluster. The proximity between the new
   cluster, denoted C_{r+s} and old cluster C_a is defined in this way:
    $d[C_a ,C_{r+s} ] = \mathbf{min}(d[C_a ,C_r ] , d[C_a ,C_s ])$.
5. If all objects are in one cluster, stop. Else, go to step 2.
```

(2) *Ref. S. C. Johnson (1967): "Hierarchical Clustering Schemes" Psychometrika, 2:241-254*

Hierarchical methods: Dendrogram

- Agglomerative Algorithm decompose data objects into a several levels of nested partitioning (tree of clusters), called a dendrogram see **Fig.2**.

  A clustering of the data objects is obtained by cutting the dendrogram at the desired level, then each connected component forms a cluster.

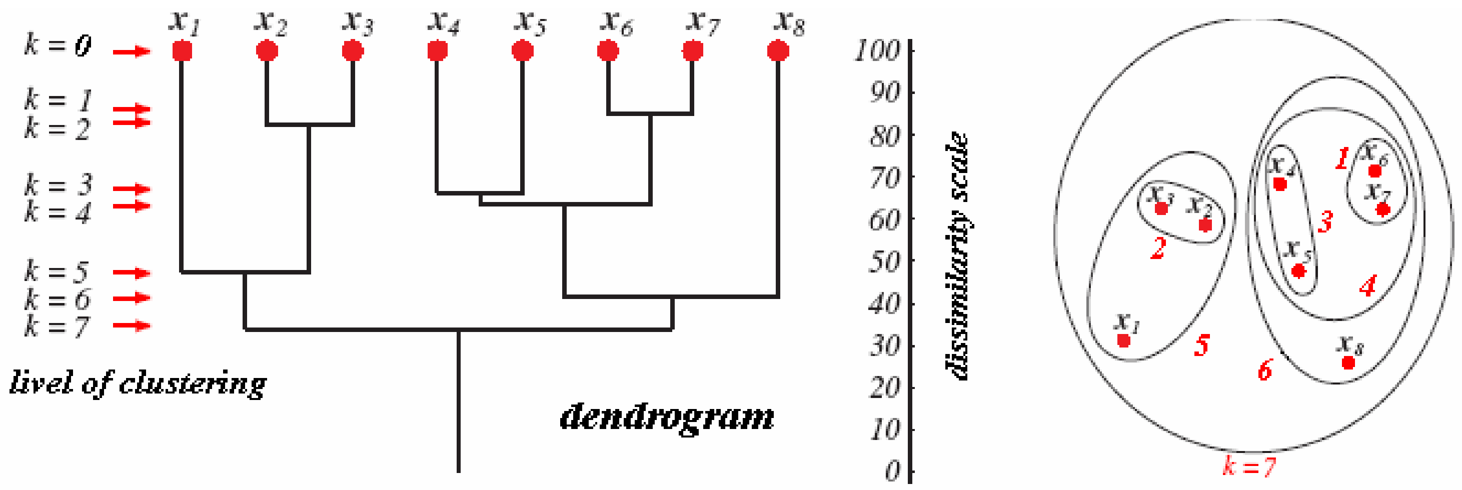
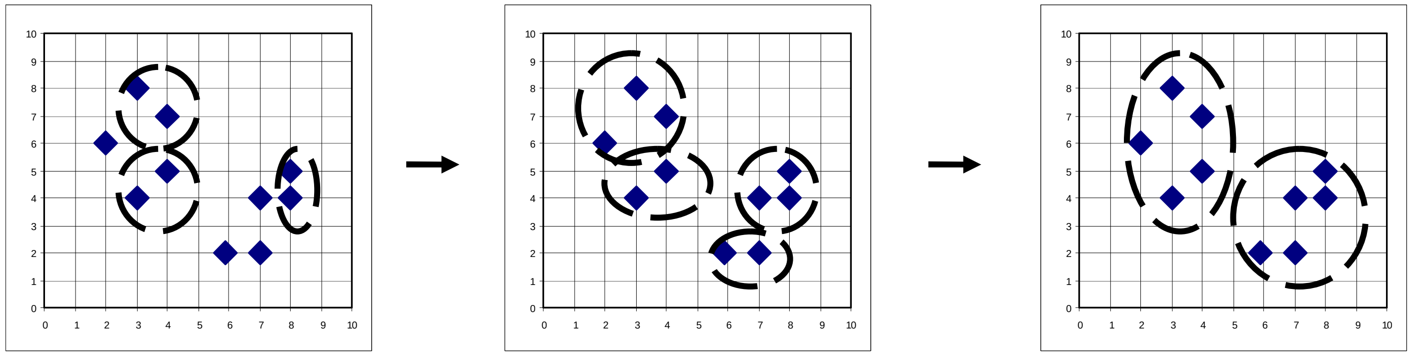


Fig. 2 The vertical axis shows a generalized measure of similarity among clusters. Here, at level **0** all eight points lie in singleton clusters; each point in a cluster is highly similar to itself, of course. Points x_6 and x_7 happen to be the most similar, and are merged at level **1**, and so forth. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

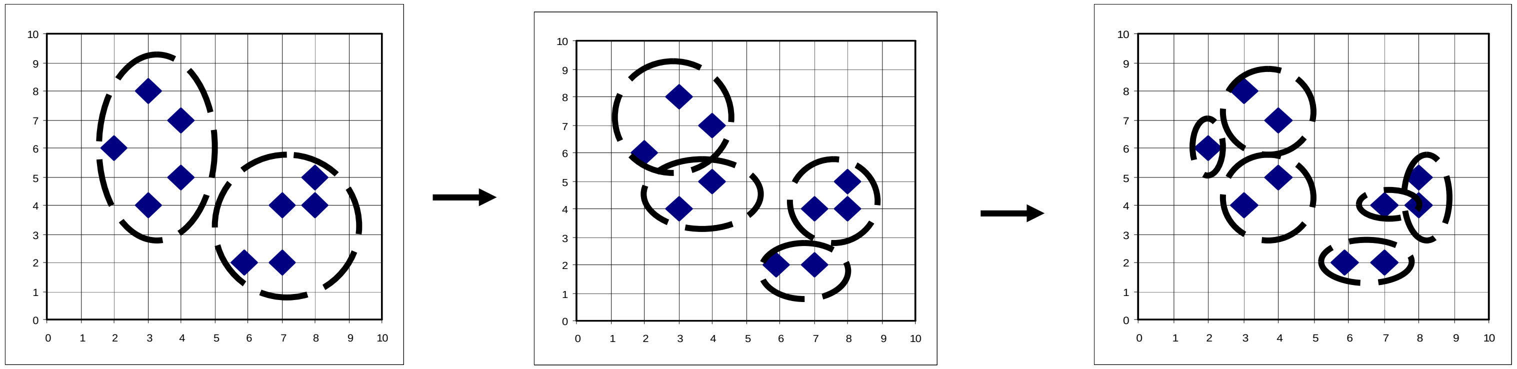
Hierarchical methods: AGNES

- Use the Single-Link method and the dissimilarity matrix.
- Merge nodes that have the least dissimilarity.
- Go on in a non-descending fashion
- Eventually all nodes belong to the same cluster.



Hierarchical methods: DIANA

- Inverse order of AGNES.
- Eventually each node forms a cluster on its own.



Hierarchical methods: more on AGNES and DIANA

- **Weakness** of agglomerative (AGNES) and divisive (DIANA) clustering methods.
  - Do not scale well: time complexity of at least $O(n^2)$; n is the number of objects.
  - It encounters difficulties regarding the selection of *merge* (agglomerative) and *split* (divisive) points. Such a decision is critical because once a group of objects is merged or split, the process at the next step will operate on the newly generated clusters. It will not undo what was done previously. Thus split or merge decisions may lead to low-quality clusters.

- **Integration** of *hierarchical* with *distance-based* clustering
  - **BIRCH** (1996), [10]: uses **CF-tree** and incrementally adjusts the quality of sub-clusters.
  - **CURE** (1998), [11]: selects well-scattered points from the cluster and then shrinks them towards the center of the cluster by a specified fraction.
  - **CHAMELEON** (1999), [12]: hierarchical clustering using dynamic modeling.

Hierarchical methods: BIRCH

- **BIRCH** (**B**alanced **I**terative **R**educing and **C**lustering using **H**ierarchies) is a method that indroduces two concepts: **clustering feature** and **CF tree** (**Clustering Feature tree**);

  incrementally uses CF tree as summarize cluster representation to achieve good speed and clustering scalability in large DB. This goal is spitted into phases.

  - Phase A: scan DB to build an initial in-memory CF tree (a multi-level compression of the data that tries to preserve the structure of the data).
  - Phase B: use an arbitrary clustering, such as partitioning algorithm k-means, to cluster the leaf nodes of the CF-tree.

- A **clustering feature CF** is a triplet of the points {*X*$_i$}, where

$$\boxed{CF = (N, \quad L\vec{S}, \quad SS) = (N, \quad \sum_{i=1}^{N} \vec{X}_i \quad \sum_{i=1}^{N} \vec{X}^2{}_i)}$$

N is the number of points (0-th statistical moment), ***LS*** is the linear sum on **N** points (1-st statistical moment) , and ***SS*** is the square sum of data points (2-nd statistical moment) . These registers measurements for computing cluster and utilizes storage efficiently.

Hierarchical methods: BIRCH

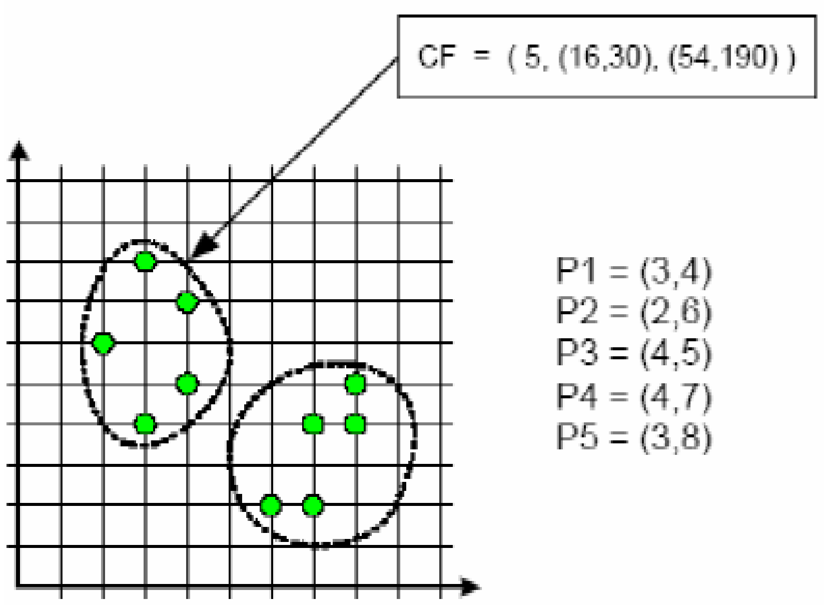
- Example of CF

Clustering feature = CF=(N, LS, SS)

N = 5

LS = (16, 30)

SS = (54, 190)



Hierarchical methods: BIRCH

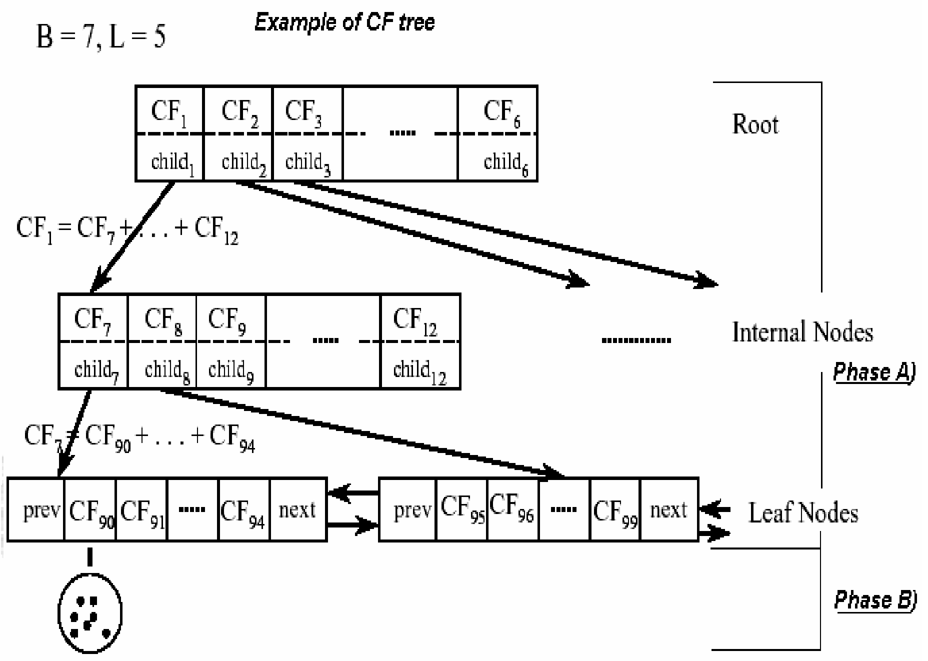
- **Algorithm (a draft)**

❑ **Phase A**

- A **CF tree** is a height-balanced tree that stores the clustering features for a hierarchical clustering. It has two parameters:
  - Branching factor ***B***: specify the maximum number of children.
  - Threshold ***L***: is the max diameter of sub-clusters stored at the leaf nod.
- The entry in each non leaf node has the form [**CF$_i$, child$_i$**].
- The entry in each leaf node is a **CF**; each leaf node has two pointers: `*prev*' and`*next*'.
- The **CF tree** is basically a tree used to store all the clustering features.

❑ **Phase B**

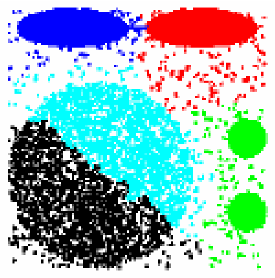
Partitioning algorithm, such as **k-means**, is used to cluster the leaf nodes of the **CF tree**.



Hierarchical methods: BIRCH

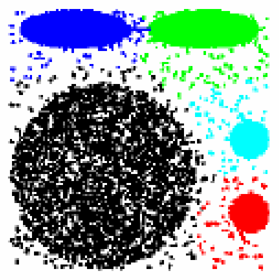
- **Advantages**:
  - *Scales linearly*: finds a good clustering with a single scan and improves the quality with a few additional scans.
  - Computation complexity of the algorithm is *O(N),* where N is number-objects.
- **Weakness**:
  - Handles only numeric data, and sensitive to the order of the data record.
  - Favors only clusters with spherical shape and similar sizes, becouse it uses the notion of diameter to control the boundary of a cluster, see **FIG.3**.[17]

FIG. 3



BIRCH
(centroid)

FIG. 4



CURE

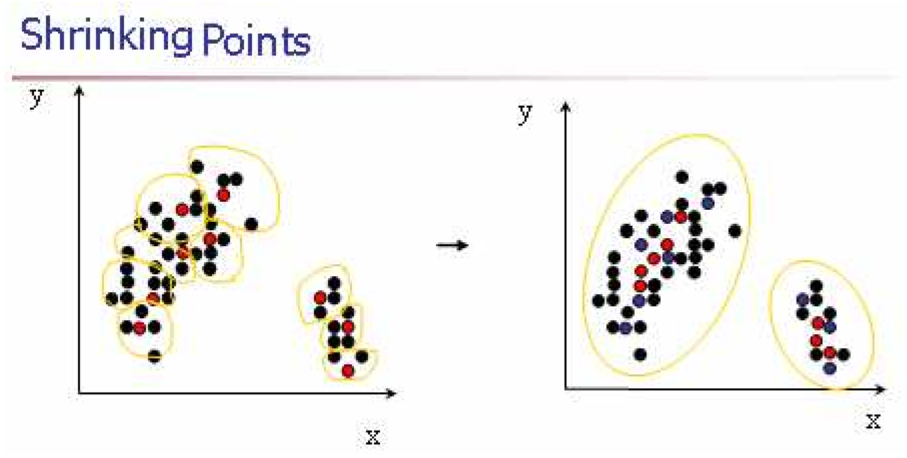
Hierarchical methods: CURE

- The method **CURE** (**C**lustering **U**sing **RE**presentatives) integrates hierarchical and partitioning algorithms to favor clusters with arbtrary shape (see **FIG. 4**[17]).

  It employs a novel hierarchical clustering algorithm:

  instead of using a single centroid to represent a cluster, a fixed number of points are chosen to represent a cluster.

  These points are generate by first selecting points from the cluster and then shrinking them toward the center of the cluster by a specified fraction ***p*** (shrinking factor) by a user.



Hierarchical methods: CURE

- **Algorithm**

Input: ***p*** clusters, ***s*** objects of database ***D***.

Output: A set of ***p*** clusters

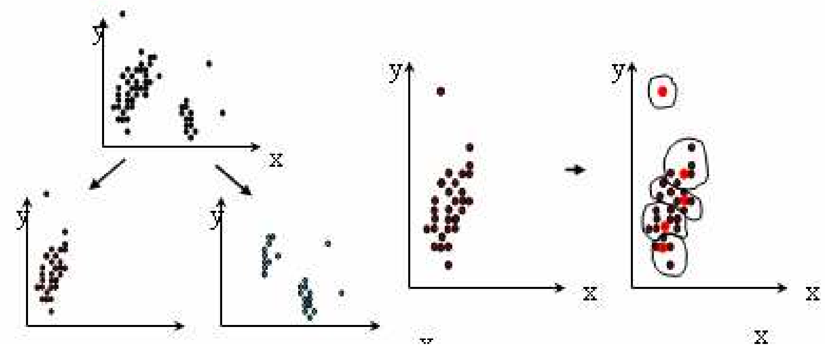
Algorithm:

1) Draw random sample ***s***.
2) Partition sample to p partitions with size ***s*** */****p****.*
3) Partially cluster the points in each cluster using hierarchical clustering algorithm to obtain partitions into ***s /pq*** clusters in each partition and a total of ***s/q*** clusters.
4) Eliminate outliers.
5) Cluster, using hierarchical clustering, partial clusters.
6) Label data in disk.

Example

- s = 50
- p = 2
- s/p = 25

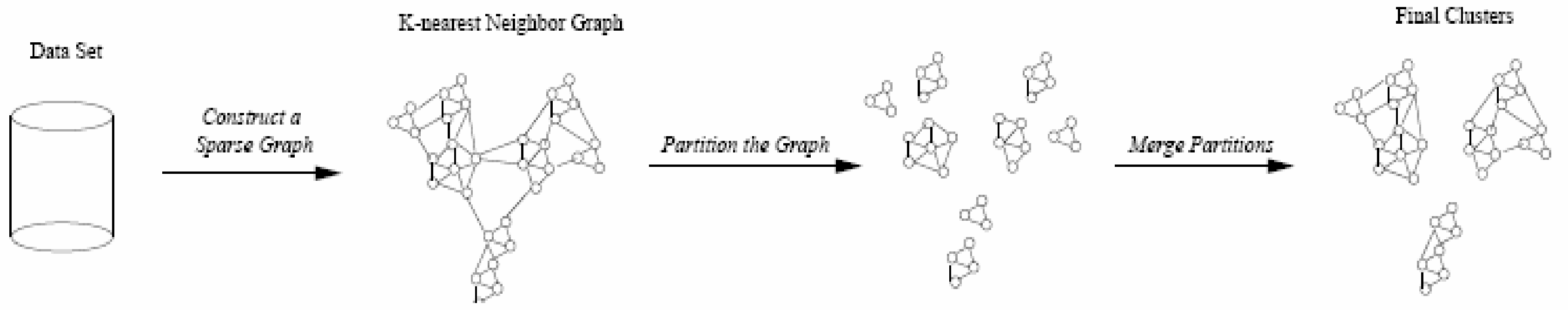
- s/pq = 5



Weakness: Cure ignore the information about the aggregate *inter-connectivity* of objects in two clusters. So it is introduced Chameleon algorithm.

Hierarchical methods: CHAMELEON

- **CHAMELEON** (**Hierarchical clustering using dynamic modeling**) algorithm explores dynamic modeling in hierarchical clustering.
  It solves two great weakness of hierarchical clustering:
  inter-connectivity of two clusters, which is in Cure algorithm; *closeness* of two clusters, which is in Rock algorithm [9].
  The algorithm works in two-phase:
  - In first phase use a graph partitioning algorithm to cluster objects into a large number of relatively small sub-clusters.
  - In second phase use an agglomerative hierarchical clustering algorithm to find the genuine clusters by repeatedly combining these sub-clusters.



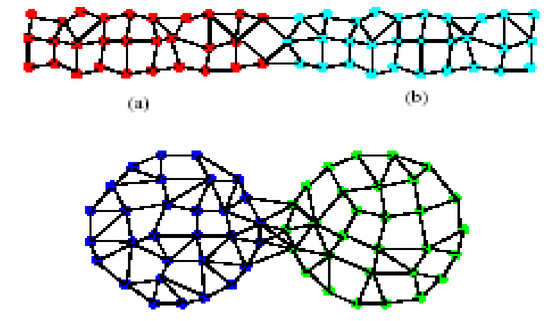
Hierarchical methods: CHAMELEON

- **Algorithm**

1) Preprocessin step.
   Represent the Data by a Graph:
   - Given a set of points, construct the k-nearest-neighbor (k-NN) graph to capture the relationship between a point and its k nearest neighbors.
   - Concept of neighborhood is captured dynamically (even if region is sparse).
2) Phase 1
   Use a multilevel graph partitioning algorithm on the graph to find a large number of clusters of well-connected vertices.
   - Each cluster should contain mostly points from one “true” cluster, i.e. is a sub cluster of a “real” cluster.
3) Phase 2
   Use Hierarchical Agglomerative Clustering to merge sub-clusters
   - Two clusters are combined if the resulting cluster shares certain properties with the constituent clusters.
   - Two key properties used to model cluster similarity:
     Relative Inter-connectivity: Absolute interconnectivity of two clusters normalized by the internal connectivity of the clusters.
     Relative Closeness: Absolute closeness of two clusters normalized by the internal closeness of the clusters.

Hierarchical methods: CHAMELEON

- **Relative Inter-Connectivity**:



$$RI(C_i, C_j) = \frac{|EC_{\{C_i,C_j\}}|}{\frac{|EC_{C_i}|+|EC_{C_j}|}{2}},$$

- **Relative Closeness**:

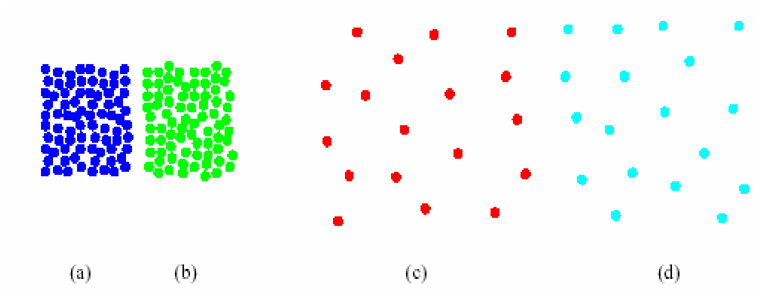


Figure 2: Example of clusters for merging choices.

$$RC(C_i, C_j) = \frac{\bar{S}_{EC_{\{C_i,C_j\}}}}{\frac{|C_i|}{|C_i|+|C_j|}\bar{S}_{EC_{C_i}} + \frac{|C_j|}{|C_i|+|C_j|}\bar{S}_{EC_{C_j}}},$$

Weakness: In Chameleon the processing cost for high dimensional data is $O(n^2)$ time for n objects.

Density-Based methods: Background[17]

- ***Density*** = number of points of data base ***D*** within a specified radius (***Eps***).
- ***Eps-neighbourhood*** of point *p* of data base ***D*** is

$$\boxed{N_{Eps}(p) = \{q \in D \mid dist(p,q) \le Eps\}}$$

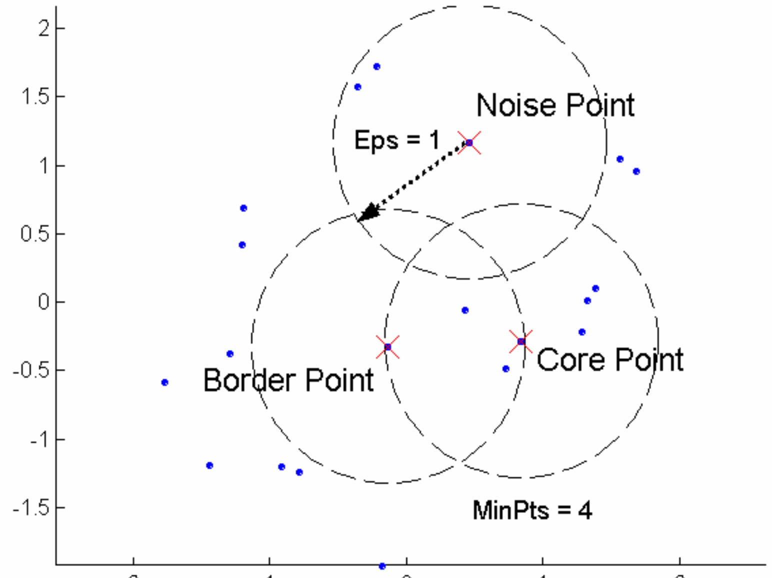
where *dist(p,q)* is the matric between points *p* and *q* of ***D***.

- A point is a ***core point*** if it has more than a specified number of points (***MinPts***) in ***Eps***.

$$\boxed{|N_{Eps}(p)| \ge MinPts}$$

where $|N_{Eps}(p)|$ is cardinality of set $N_{Eps}(p)$.

- A ***border point*** has fewer than ***MinPts*** within ***Eps***, but is in the neighborhood of a core point.
- A ***noise point*** is any point that is not a core point or a border point.

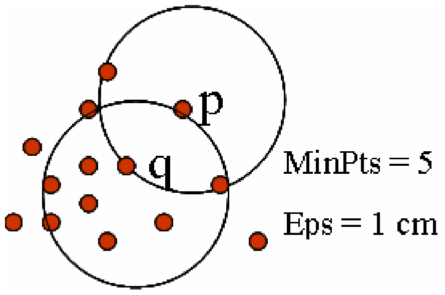


Density-Based methods: Background[17]

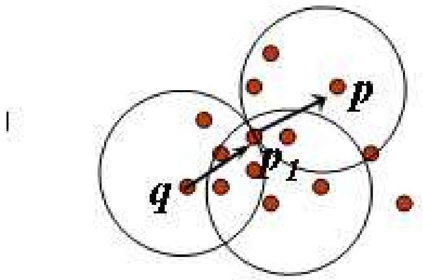
- A point p is ***directly density-reachable*** from a point q wrt. ***Eps***, ***MinPts*** if

    1) p belongs to $N_{Eps}(q)$

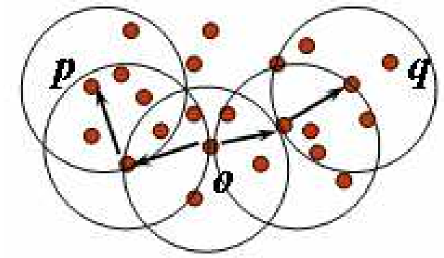
    2) p is a ***core point***.



- A point p is ***density-reachable*** from a point q wrt. *Eps*, *MinPts* if there is a chain of points $p_1, \ldots, p_n$, $p_1 = q$, $p_n = p$ such that p_{i+1} is directly density-reachable from p_i .



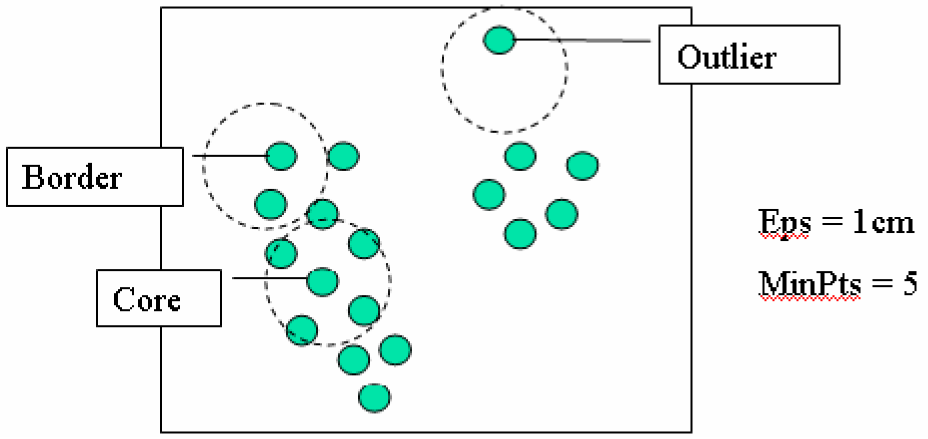
- A point p is ***density-connected*** to a point q wrt. ***Eps***, ***MinPts*** if there is a point o such that both, p and q are density-reachable from o wrt. ***Eps*** and ***MinPts***.



Density-Based methods: Background

- A ***density-based cluster*** (***cluster***) is a sub-set *C* of ***D*** so that:
  1) consists of the *density-connected* points.
  2) is maximal respect to *density-reachability,*
     i.e. for every pair of points (*p,q),* if *p* belongs to *C* and *q*
     is ***density-reachable*** from *p,* wrt. *Eps*, *MinPts,* than *q* belongs to *C.*

- Avery point that does not belong to the cluster *C* is ***noise***.
  DBSCAN (**D**ensity **B**ased **S**patial **C**lustering of **A**pplication with **N**ois)
  is the first density-based method that uses these concepts.
  Discovers clusters of arbitrary shape in spatial databases with noise.



Density-Based methods: DBSCAN

- **Algorithm**

Input: N objects to be clustered and global parameters *Eps*, *MinPts.*

Output: Clusters of objects.

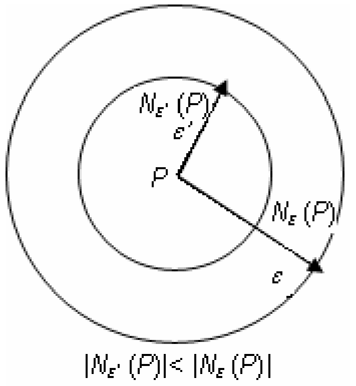
Algorithm:

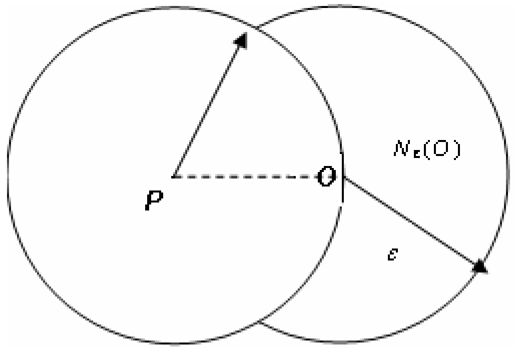
1) Arbitrary select a point *P*.
2) Retrieve all points density-reachable from *P* wrt ***Eps*** and ***MinPts***.
3) If *P* is a core point, a cluster is formed.
4) If *P* is a border point, no points are density-reachable from *P* and **DBSCAN** visits the next point of the database.
5) Continue the process until all of the points have been processed.

Weakness: Need to specify global parameters ***Eps***, ***MinPts*** in advance from user.

Density-Based methods: OPTICS

- **OPTICS** (**O**rdering **P**oints **T**o **I**dentify the **C**lustering **S**tructure) is introduced to overcome **DBSCAN**'s difficulty.
  It computes an augmented <u>clustering-ordering</u> for automatic cluster analysis.
  Based on this idea, two values need to be introduced:
  - The ***core distance*** of an point *P* is the smallest distance ***ε'*** between *P* and a point in its $N_\varepsilon(P)$ neighbourhood such that *P* is a *core point* with respect to ***ε'*** ($|N_{\varepsilon'}(P)|>=MinPits$) if $|N_{\varepsilon'}(P)|<|N_\varepsilon(P)|$.
    Otherwise the core distance is UNDEFINED.
  - The ***reachability–distance*** of an point *P* with respect to anather point *O* is the smallest distance ***ε*** such that *P* is ***directly density-reachable*** from *O* if *O* is a *core point.* If *O* isn't a core point ***reachability–distance*** is UNDEFINED.

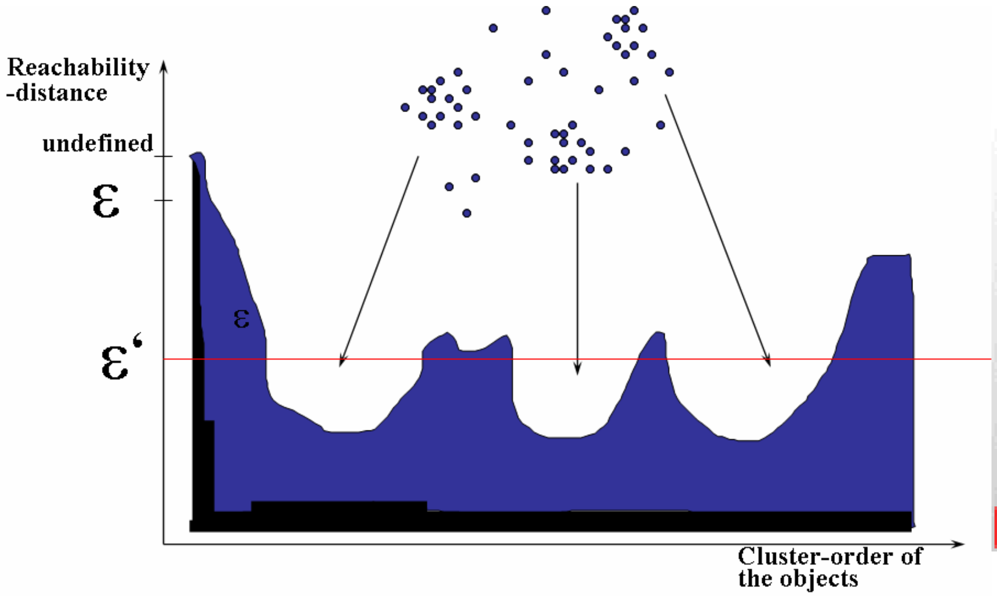


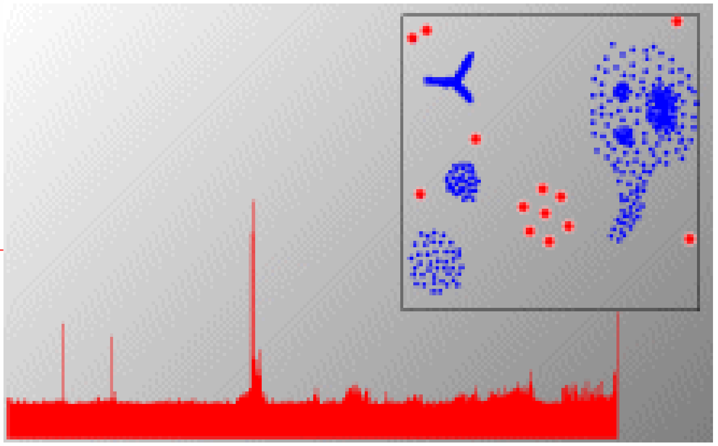


Density-Based methods: OPTICS

- **Algorithm**

  - The **OPTICS** algorithm creates an ordering of a database, additionally storing the ***core-distance*** and a suitable ***reachability-distance*** for each points.
  - Then such information is used to extract all density-based clusterings with respect to any distance ***ε'*** smaller than the generating distance ***ε*** from this order.





Density-Based methods: DENCLUE

- **DENCLUE** (**DEN**sity based **CLU**st**E**ring) is clustering method based on a set of ***density functions***.
  First introduce the ***influence functions*** and then the ***density functions***.

- The ***influence functions*** of a data point *y* belongs to F^{d}, where F^{d} is a **d**-dimensional feature space, is a basic influence function f_B so that

$$f_B^{y} : F^{d} \rightarrow f_B^{y} = f_B(x,y) > 0$$

- The ***density functions*** f_B^{D} is defined as the sum of ***influence functions*** of all data points.
  Given ***N*** data points described by a set of feature-vectors $D=\{x_1,\dots, x_N\}$ sub-set of F^{d}

$$\boxed{f_B^{D} = \sum_{i=1}^{N} f^{x_i}{}_B(x)}$$

Density-Based methods: DENCLUE

- **Algorithm**

1) **DENCLUE** uses grid cells but only keeps information about grid cells that do actually contain data points and manages these cells in a tree-based access structure.
2) ***Influence function***: describes the impact of a data point within its neighborhood.
3) The overall density of the data space can be modeled by ***density function*** , that is the sum of the ***influence functions*** of all data points.
4) Clusters can be determined mathematically by identifying ***density attractors***.
5) ***Density attractors*** are local *maxima* of the overall ***density function***.
6) The local *maxima* is calculated with *hill climbing algorithm* that uses the *gradient* of ***density function***.

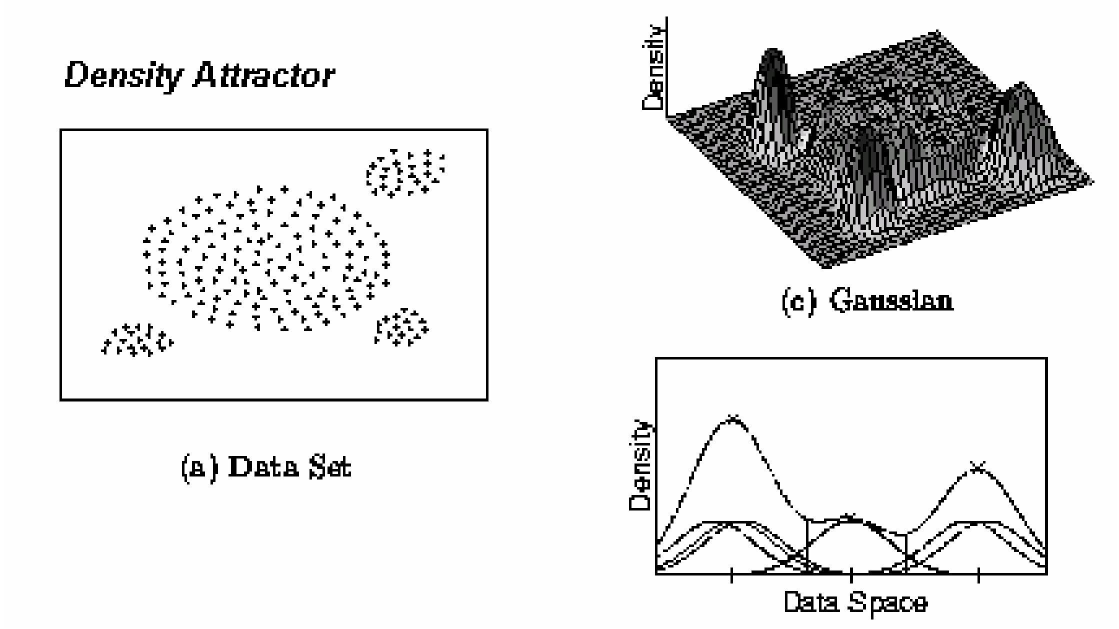
- For exemple if density function is gaussian we have ***density function*** and ***gradient*** :

$$f^D_{Gaussian}(x) = \sum_{i=1}^{N} e^{-\frac{d(x,x_i)^2}{2\sigma^2}}$$

$$\nabla f^D_{Gaussian}(x, x_i) = \sum_{i=1}^{N} (x_i - x) \cdot e^{-\frac{d(x,x_i)^2}{2\sigma^2}}$$

Density-Based methods: DENCLUE

- Example: *Density Attractor*



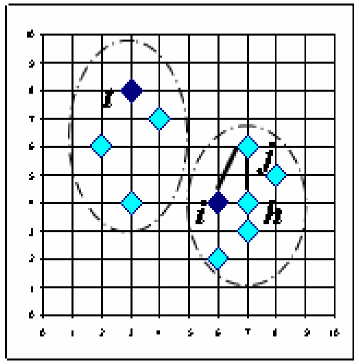
Clustering: Tools open source

- The flexclust package for R
- COMPACT - Comparative Package for Clustering Assessment (in Matlab)
- YALE (Yet Another Learning Environment): freely available open-source software for data pre-processing, knowledge discovery, data mining, machine learning, visualization, etc. also including a plugin for clustering, fully integrating Weka, easily extendible, and featuring a graphical user interface as well as a XML-based scripting language for data mining;
- Mixmod : Model Based Cluster And Discriminant Analysis. Code in C++, interface with Matlab and Scilab
- LingPipe Clustering Tutorial Tutorial for doing complete- and single-link clustering using LingPipe, a Java text data mining package distributed with source.
- Weka : Weka contains tools for data pre-processing, classification, regression, clustering, association rules, and visualization. It is also well-suited for developing new machine learning schemes.
- Tanagra : a free data mining software including several clustering algorithms such as K-MEANS, SOM, Clustering Tree, HAC and more.
- Cluster : Open source clustering software. The routines are available in the form of a C clustering library, an extension module to Python, a module to Perl.
- python-cluster Pure python implementation

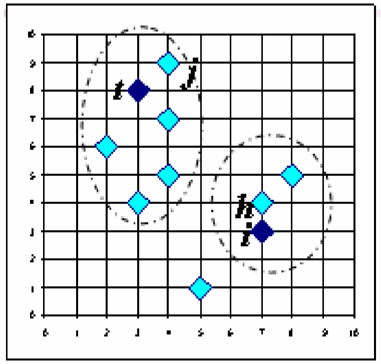
Also useful graph manipulation software:

- JUNG: Java Universal Network/Graph Framework
- Webgraph: WebGraph is a framework to study the web graph

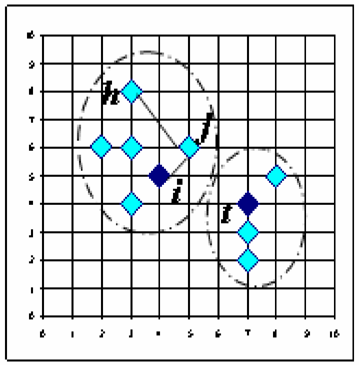
Appendix: Calculus of *Cjih*



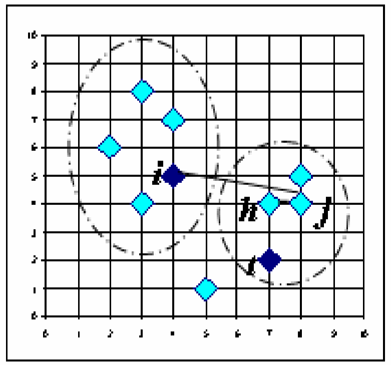
$C_{jih} = d(j, h) - d(j, i)$



$C_{jih} = 0$



$C_{jih} = d(j, t) - d(j, i)$



$C_{jih} = d(j, h) - d(j, t)$

Tab.1

A) If ***j*** belongs to the cluster defined by medoid ***i,*** consider the distance ***δ(x$_J$, x$_h$)*** between object ***j*** and object ***h***.

- If ***h*** is further from ***j*** than the second best medoid ***i '*** is from ***j***, then the contribution from object ***j*** to the swap is:

$$C_{Jih} = \delta(x_J , x_{i'}) - \delta(x_J , x_i)$$

The result of ***i*** <-> ***h*** would be that object ***j*** now belongs to cluster ***i '*** .

- Else, if ***h*** is closer to ***j*** than ***i '*** is to ***j***, the contribution from ***j*** to the swap is:

$$C_{Jih} = \delta(x_J , x_h) - \delta(x_J , x_i)$$

The result of ***i*** <-> ***h*** would be that object ***j*** now belongs to cluster ***h***.

B) If ***j*** belongs to cluster ***r***, where ***r*** =/= ***i***, check the distance between object ***j*** and object ***h***.

- If ***h*** is further from ***j*** than the medoid ***r*** is from ***j***, then the contribution from ***j*** to the swap is:

$$C_{Jih} = 0$$

The result of ***i*** <-> ***h*** would be that object ***j*** still belongs to cluster ***r***.

- Else, if ***h*** is closer to ***j*** than ***r*** is to ***j***, the contribution from ***j*** to the swap is:

$$C_{Jih} = \delta(x_J , x_h) - \delta(x_J , x_r)$$

The result of ***i*** <-> ***h*** would be that object ***j*** now belongs to cluster ***h***.

References

- [1] J. Han and M. Kamber. Data Mining: Concepts and Techniques. Morgan Kaufmann Publishers, August 2000.
- [2], [3] J. Han and M. Kamber. Data Mining: Concepts and Techniques. Morgan Kaufmann Publishers, August 2000 (*k-means, k-medoids or PAM*).
- [4], [8], [9] L. Kaufman and P. J. Rousseeuw. Finding Groups in Data: an Introduction to Cluster Analysis. John Wiley & Sons, 1990 (*CLARA, AGNES, DIANA*).
- [5] R. Ng and J. Han. Efficient and effective clustering method for spatial data mining. VLDB'94 (*CLARANS*).
- [6], [7] J. Han and M. Kamber. Data Mining: Concepts andTechniques. Morgan Kaufmann Publishers, August 2000 (*deterministic annealing, genetic algorithms*).
- [10] T. Zhang, R. Ramakrishnan, and M. Livny. BIRCH : an efficient data clustering method for very large databases. SIGMOD'96 (*BIRCH*).
- [11] S. Guha, R. Rastogi, and K. Shim. Cure: An efficient clustering algorithm for large databases. SIGMOD'98 (*CURE*).

References

- [12] Karypis G., Eui-Hong Han, Kumar V. Chameleon: hierarchical clustering using dynamic modeling (*CHAMELEON*).
- [13] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases. KDD'96 (*DBSCAN*).
- [14] M. Ankerst, M. Breunig, H.-P. Kriegel, and J. Sander. Optics: Ordering points to identify the clustering structure, SIGMOD'99 (*OPTICS*).
- [15] A. Hinneburg D., A. Keim: An Efficient Approach to Clustering in Large Multimedia Database with Noise. Proceedings of the 4-th ICKDDM, New York '98 (*DENCLUE*).
- [16] Abramowitz, M. and Stegun, I. A. (Eds.). "*Stirling Numbers of the Second Kind*." §24.1.4 in Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables, 9th printing. New York: Dover, pp. 824-825, 1972.
- [17] Introduction to Data Mining Pang-Ning Tan, Michigan State University Michael Steinbach,Vipin Kumar, University of Minnesota Publisher: Addison-Wesley Copyright: 2006.