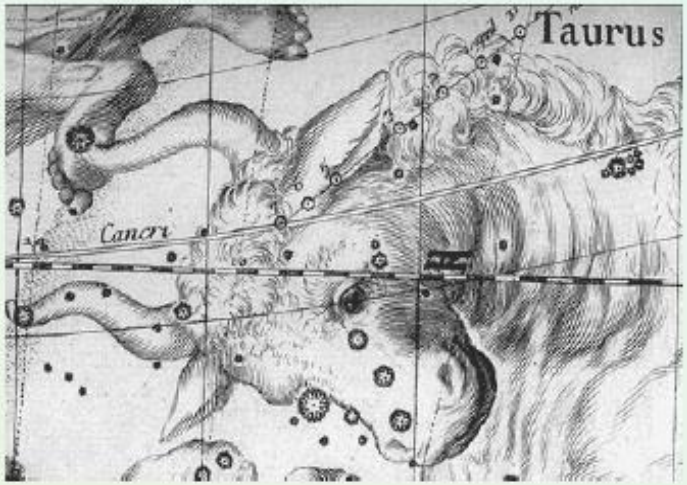
# Cluster Analysis

## UNIT – 4

# Cluster Analysis

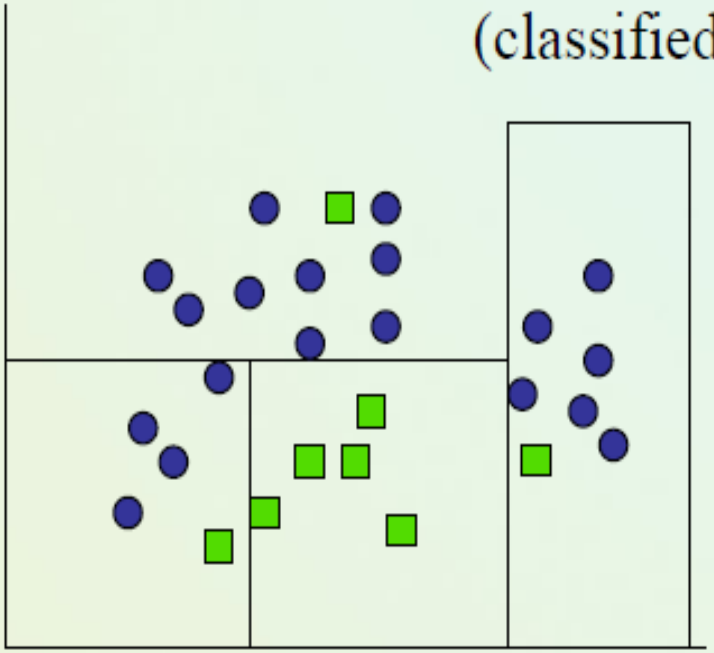
Astronomy - aggregation of stars, galaxies, or super galaxies, …



# Classification vs. Clustering

Classification: Supervised learning:

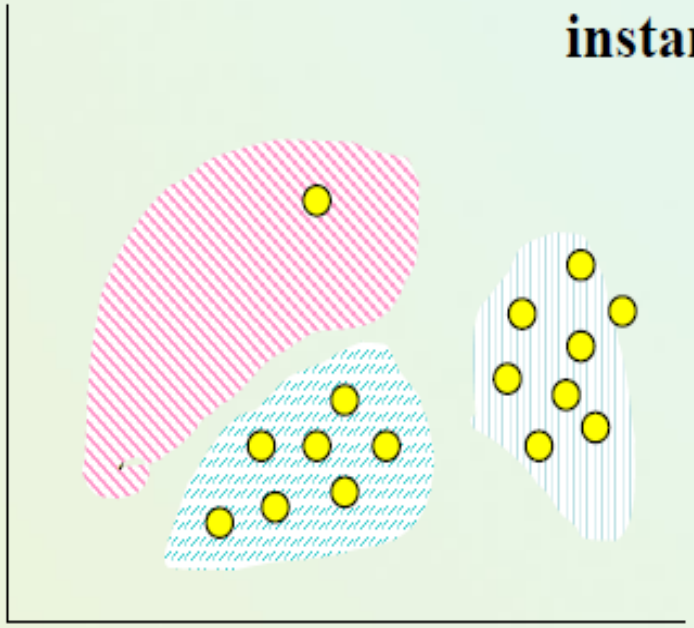
Learns a method for predicting the instance class from pre-labeled (classified) instances



# Clustering

**Unsupervised learning:**

**Finds "natural" grouping of instances given un-labeled data**



# What is a cluster?

1. A cluster is a subset of objects which are “similar”
2. A subset of objects such that the distance between any two objects in the cluster is less than the distance between any object in the cluster and any object not located inside it.
3. A connected region of a multidimensional space containing a relatively high density of objects.

# What Is Clustering ?

- Clustering is a process of partitioning a set of data (or objects) into a set of meaningful sub-classes, called clusters.
  - Help users understand the natural grouping or structure in a data set.
- Clustering: unsupervised classification: no predefined classes.
- Used either as a stand-alone tool to get insight into data distribution or as a preprocessing step for other algorithms.
  - Moreover, data compression, outliers detection, understand human concept formation.

# What Is Good Clustering?

- A good clustering method will produce high quality clusters in which:
  - the intra-class (that is, intra-cluster) similarity is high.
  - the inter-class similarity is low.
- The quality of a clustering result also depends on both the similarity measure used by the method and its implementation.
- The quality of a clustering method is also measured by its ability to discover some or all of the hidden patterns.
- However, objective evaluation is problematic: usually done by human / expert inspection.

# Clustering: Rich Applications and Multidisciplinary Efforts

- Pattern Recognition
- Spatial Data Analysis
  - Create thematic maps in GIS by clustering feature spaces
  - Detect spatial clusters or for other spatial mining tasks
- Image Processing
- Economic Science (especially market research)
- WWW
  - Document classification
  - Cluster Weblog data to discover groups of similar access patterns

# Examples of Clustering Applications

- Marketing: Help marketers discover distinct groups in their customer bases, and then use this knowledge to develop targeted marketing programs
- Land use: Identification of areas of similar land use in an earth observation database
- Insurance: Identifying groups of motor insurance policy holders with a high average claim cost
- City-planning: Identifying groups of houses according to their house type, value, and geographical location
- Earth-quake studies: Observed earth quake epicenters should be clustered along continent faults

# Requirements of Clustering in Data Mining

- Scalability – small datasets, large datasets
- Ability to deal with different types of attributes – nominal, numeric, ordinal, binary or mixture of these data types.
- Ability to handle dynamic data
- Discovery of clusters with arbitrary shape - develop alg that can detect clusters of arbitrary shape.
- Minimal requirements for domain knowledge to determine input parameters
- Able to deal with noise and outliers – avoid clusters of poor quality.
- Insensitive to order of input records – some clustering alg cannot incorporate newly inserted data.
- High dimensionality – have good alg to handle high dimensional data.
- Incorporation of user(Constrain based clustering)-specified constraints
- Interpretability and usability

# Types of Data in Cluster Analysis

**Data matrix (or *object-by-variable structure*):**

This represents *n* objects, such as persons, with *p* variables (also called *measurements* or *attributes*), such as age, height, weight, gender, and so on.

- The structure is in the form of a **relational table, or *n*-by-*p* matrix (*n* objects *p* variables)**

$$\begin{bmatrix} x_{11} & \cdots & x_{1f} & \cdots & x_{1p} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ x_{i1} & \cdots & x_{if} & \cdots & x_{ip} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ x_{n1} & \cdots & x_{nf} & \cdots & x_{np} \end{bmatrix}$$

**Data matrix**

$$\begin{bmatrix} 0 & & & & \\ d(2,1) & 0 & & & \\ d(3,1) & d(3,2) & 0 & & \\ \vdots & \vdots & \vdots & & \\ d(n,1) & d(n,2) & \cdots & \cdots & 0 \end{bmatrix}$$

**Dissimilarity matrix**

**Dissimilarity matrix (or *object-by-object structure*):**

This stores a collection of proximities that are available for all pairs of *n* objects. It is often represented by an *n*-by-*n* table.

- d(i, j) - difference or dissimilarity between objects i and j.
- d(i, j) - **nonnegative number** that is **close to 0** when objects i and j are **highly similar or "near"** each other, and becomes larger the more they differ.
- Since d(i, j)=d( j, i), and d(i, i)=0.
- Data matrix - two-mode matrix.(Deals with different entities)
- Dissimilarity matrix - one-mode matrix.(Deals with same entity)

# Interval-Scaled Variables

- Interval-scaled variables are continuous measurements of a roughly linear scale. Eg: weight and height

- The measurement unit used can affect the clustering analysis.

- For example, changing measurement units from meters to inches for height, or from kilograms to pounds for weight, may lead to a very different clustering structure.

- To avoid dependence on the choice of measurement units, the data should be standardized.

- Standardizing measurements attempts to give all variables an equal weight.

***How can the data for a variable be standardized?***

- To standardize measurements, one choice is to convert the original measurements to unitless variables.

1. Calculate the mean absolute deviation, $s_f$:

$$s_f = \frac{1}{n}(|x_{1f} - m_f| + |x_{2f} - m_f| + \cdots + |x_{nf} - m_f|), \tag{7.3}$$

where $x_{1f}, \ldots, x_{nf}$ are $n$ measurements of $f$, and $m_f$ is the *mean* value of $f$, that is, $m_f = \frac{1}{n}(x_{1f} + x_{2f} + \cdots + x_{nf})$.

2. Calculate the standardized measurement, or z-score:

$$z_{if} = \frac{x_{if} - m_f}{s_f}. \tag{7.4}$$

After standardization, or without standardization in certain applications, the dissimilarity (or similarity) between the objects described by interval-scaled variables is typically computed based on the distance between each pair of objects. The most popular distance measure is **Euclidean distance**

$$d(i, j) = \sqrt{(x_{i1} - x_{j1})^2 + (x_{i2} - x_{j2})^2 + \cdots + (x_{in} - x_{jn})^2},$$

Another well-known metric is **Manhattan (or city block) distance**, defined as

$$d(i, j) = |x_{i1} - x_{j1}| + |x_{i2} - x_{j2}| + \cdots + |x_{in} - x_{jn}|.$$

- Both the Euclidean distance and Manhattan distance satisfy the following mathematic requirements of a distance function

1. $d(i, j) \geq 0$: Distance is a nonnegative number.
2. $d(i, i) = 0$: The distance of an object to itself is 0.
3. $d(i, j) = d(j, i)$: Distance is a symmetric function.
4. $d(i, j) \leq d(i, h) + d(h, j)$: Going directly from object $i$ to object $j$ in space is no more than making a detour over any other object $h$ (*triangular inequality*).

- **Minkowski distance** is a generalization of both Euclidean distance and Manhattan distance. It is defined as

$$d(i, j) = (|x_{i1} - x_{j1}|^p + |x_{i2} - x_{j2}|^p + \cdots + |x_{in} - x_{jn}|^p)^{1/p},$$

- $p$ - positive integer. Such a distance is also called $Lp$ norm.
- It represents the Manhattan distance when $p$ = 1 (i.e., $L1$ norm)
- Euclidean distance when p = 2 (i.e., L2 norm).

- If each variable is assigned a weight according to its perceived importance, the weighted Euclidean distance can be computed as

$$d(i, j) = \sqrt{w_1|x_{i1} - x_{j1}|^2 + w_2|x_{i2} - x_{j2}|^2 + \cdots + w_m|x_{in} - x_{jn}|^2}.$$

Euclidean distance and Manhattan distance. Let $x_1 = (1, 2)$ and $x_2 = (3, 5)$ represent two objects as in Figure 7.1. The Euclidean distance between the two is $\sqrt{(2^2 + 3^2)} = 3.61$. The Manhattan distance between the two is $2 + 3 = 5$. ■

## Example: Data Matrix and Dissimilarity Matrix

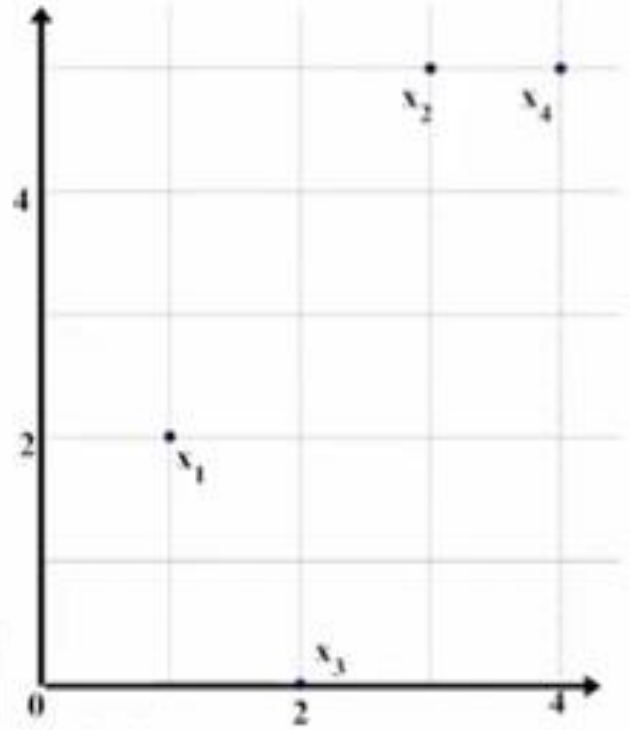

Data Matrix

| point | attribute1 | attribute2 |
|---|---|---|
| *x1* | 1 | 2 |
| *x2* | 3 | 5 |
| *x3* | 2 | 0 |
| *x4* | 4 | 5 |

Dissimilarity Matrix (by Euclidean Distance)

| | *x1* | *x2* | *x3* | *x4* |
|---|---|---|---|---|
| *x1* | 0 | | | |
| *x2* | 3.61 | 0 | | |
| *x3* | 2.24 | 5.1 | 0 | |
| *x4* | 4.24 | 1 | 5.39 | 0 |

| point | attribute 1 | attribute 2 |
|---|---|---|
| x1 | 1 | 2 |
| x2 | 3 | 5 |
| x3 | 2 | 0 |
| x4 | 4 | 5 |




## Manhattan ($L_1$)

| L | x1 | x2 | x3 | x4 |
|---|---|---|---|---|
| x1 | 0 | | | |
| x2 | 5 | 0 | | |
| x3 | 3 | 6 | 0 | |
| x4 | 6 | 1 | 7 | 0 |

## Euclidean ($L_2$)

| L2 | x1 | x2 | x3 | x4 |
|---|---|---|---|---|
| x1 | 0 | | | |
| x2 | 3.61 | 0 | | |
| x3 | 2.24 | 5.1 | 0 | |
| x4 | 4.24 | 1 | 5.39 | 0 |

# Binary Variables

- How to compute the dissimilarity between objects described by either *symmetric* or *asymmetric binary variables*.
- A binary variable has only **two states: 0 or 1**
- **0 – absent & 1- present**
- Treating binary variables as if they are interval-scaled can lead to misleading clustering results.
- If all binary variables are thought of as having the same weight, we have the 2-by-2 contingency table as

A contingency table for binary variables.

| | | object $j$ | | |
|---|---|---|---|---|
| | | 1 | 0 | sum |
| object $i$ | 1 | $q$ | $r$ | $q+r$ |
| | 0 | $s$ | $t$ | $s+t$ |
| | sum | $q+s$ | $r+t$ | $p$ |

- The total number of variables is $p$, where $p = q+r+s+t$.

- A binary variable is symmetric if both of its states are equally valuable and carry the same weight.

- There is no preference on which outcome should be coded as 0 or 1.

$$d(i, j) = \frac{r+s}{q+r+s+t}.$$

- Dissimilarity that is based on symmetric binary variables is called symmetric binary dissimilarity.

- A binary variable is asymmetric if the outcomes of the states are not equally important, such as the positive and negative outcomes of a disease test.

- The number of negative matches, *t*, is considered unimportant and thus is ignored in the computation, as

$$d(i, j) = \frac{r + s}{q + r + s}.$$

- we can measure the distance between two binary variables based on the notion of similarity instead of dissimilarity.

$$sim(i, j) = \frac{q}{q + r + s} = 1 - d(i, j).$$

- The coefficient sim(i, j) is called the **Jaccard coefficient**.

Dissimilarity between binary variables. Suppose that a patient record table (Table 7.2) contains the attributes *name*, *gender*, *fever*, *cough*, *test-1*, *test-2*, *test-3*, and *test-4*, where *name* is an object identifier, *gender* is a symmetric attribute, and the remaining attributes are asymmetric binary.

For asymmetric attribute values, let the values *Y* (*yes*) and *P* (*positive*) be set to 1, and the value *N* (*no* or *negative*) be set to 0. Suppose that the distance between objects (patients) is computed based only on the asymmetric variables. According to Equation (7.10), the distance between each pair of the three patients, Jack, Mary, and Jim, is

$$d(Jack, Mary) = \frac{0+1}{2+0+1} = 0.33$$

$$d(Jack, Jim) = \frac{1+1}{1+1+1} = 0.67$$

$$d(Mary, Jim) = \frac{1+2}{1+1+2} = 0.75$$

# Example: Dissimilarity between Asymmetric Binary Variables

| Name | Gender | Fever | Cough | Test-1 | Test-2 | Test-3 | Test-4 |
|---|---|---|---|---|---|---|---|
| Jack | M | Y | N | P | N | N | N |
| Mary | F | Y | N | P | N | P | N |
| Jim | M | Y | P | N | N | N | N |

- Gender is a symmetric attribute (not counted in)
- The remaining attributes are asymmetric binary
- Let the values Y and P be 1, and the value N be 0
- Distance: $d(i, j) = \frac{r + s}{q + r + s}$

Jack vs Mary

| Jack \ Mary | 1 | 0 | $\sum_{row}$ |
|---|---|---|---|
| 1 | 2 | 0 | 2 |
| 0 | 1 | 3 | 4 |
| $\sum_{col}$ | 3 | 3 | 6 |

Jack vs Jim

| Jack \ Jim | 1 | 0 | $\sum_{row}$ |
|---|---|---|---|
| 1 | 1 | 1 | 2 |
| 0 | 1 | 3 | 4 |
| $\sum_{col}$ | 2 | 4 | 6 |

Jim vs Mary

| Jim \ Mary | 1 | 0 | $\sum_{row}$ |
|---|---|---|---|
| 1 | 1 | 1 | 2 |
| 0 | 2 | 2 | 4 |
| $\sum_{col}$ | 3 | 3 | 6 |

- These measurements suggest that **Mary and Jim** are **unlikely to have a similar disease** because they have the **highest dissimilarity** value among the three pairs.

- Of the three patients, **Jack and Mary** are the **most likely to have a similar disease.**

# Categorical, Ordinal, and Ratio-Scaled Variables

- A **categorical variable** is a generalization of the binary variable in that it can take on more than two states.
- For example, *map color* is a categorical variable that may have, say, five states: *red, yellow, green, pink*, and *blue*.
- Let the number of **states** of a categorical variable be **M**. The states can be denoted by letters, symbols, or a set of integers, such as 1, 2, : : : , M.
- The dissimilarity between two objects i and j can be computed based on the ratio of mismatches

$$d(i, j) = \frac{p - m}{p},$$

- m - number of matches
- P - total number of variables.

# Proximity Measure for Categorical Attributes

- Categorical data, also called nominal attributes
  - Example: Color (red, yellow, blue, green), profession, etc.
- Method 1: Simple matching
  - *m*: # of matches, *p*: total # of variables

$$d(i, j) = \frac{p - m}{p}$$

- Method 2: Use a large number of binary attributes
  - Creating a new binary attribute for each of the *M* nominal states

A sample data table containing variables of mixed type.

| object identifier | test-1 (categorical) | test-2 (ordinal) | test-3 (ratio-scaled) |
|---|---|---|---|
| 1 | code-A | excellent | 445 |
| 2 | code-B | fair | 22 |
| 3 | code-C | good | 164 |
| 4 | code-A | excellent | 1,210 |

*test-1*, we set $p = 1$

$d(i, j)$ evaluates to 0 if objects $i$ and $j$ match, and 1 if the objects differ.

$$\begin{bmatrix} 0 & & & \\ d(2,1) & 0 & & \\ d(3,1) & d(3,2) & 0 & \\ d(4,1) & d(4,2) & d(4,3) & 0 \end{bmatrix}$$

$$\begin{bmatrix} 0 & & & \\ 1 & 0 & & \\ 1 & 1 & 0 & \\ 0 & 1 & 1 & 0 \end{bmatrix}$$

# Ordinal Variables

- An ordinal variable can be discrete or continuous
- Order is important, e.g., rank (e.g., freshman, sophomore, junior, senior)
- Can be treated like interval-scaled
  - Replace *an ordinal variable value* by its rank: $r_{if} \in \{1, \ldots, M_f\}$
  - Map the range of each variable onto [0, 1] by replacing *i*-th object in the *f*-th variable by

$$z_{if} = \frac{r_{if} - 1}{M_f - 1}$$

    - Example: freshman: 0; sophomore: 1/3; junior: 2/3; senior 1
      - Then distance: d(freshman, senior) = 1, d(junior, senior) = 1/3
  - Compute the dissimilarity using methods for interval-scaled variables

**Dissimilarity between ordinal variables:**

- There are three states for *test*-2, namely *fair*, *good*, and *excellent*, that is *Mf* =3.
- step 1, if we replace each value for *test*-2 by its rank, the four objects are assigned the ranks 3, 1, 2, and 3, respectively.
- Step 2 normalizes the ranking by mapping rank 1 to 0.0, rank 2 to 0.5, and rank 3 to 1.0.
- For step 3, we can use, say, the Euclidean distance, which results in the following dissimilarity matrix:

$$\begin{bmatrix} 0 & & & \\ 1 & 0 & & \\ 0.5 & 0.5 & 0 & \\ 0 & 1.0 & 0.5 & 0 \end{bmatrix}$$

## Ratio-Scaled Variables

- A ratio-scaled variable makes a positive measurement on a nonlinear scale, such as an exponential scale.

$$Ae^{Bt} \quad \text{or} \quad Ae^{-Bt}$$

- where $A$ and $B$ are positive constants, and $t$ typically represents time. Eg: growth of a bacteria population or the decay of a radioactive element.

- There are three methods to handle ratio-scaled variables for computing the dissimilarity between objects.

- Treat ratio-scaled variables like interval-scaled variables.
- Apply logarithmic transformation to a ratio-scaled variable $f$ having value $x_{if}$ for object $i$ by using the formula $y_{if}$ = log($x_{if}$ ). The $y_{if}$ values can be treated as interval-valued.
- Treat $x_{if}$ as continuous ordinal data and treat their ranks as interval-valued.
- The latter two methods are the most effective, although the choice of method used may depend on the given application.

## Dissimilarity between ratio-scaled variables

- The ratio-scaled variable, *test-3*, are available. Let's try a logarithmic transformation.

- Taking the *log* of *test-3* results in the values 2.65, 1.34, 2.21, and 3.08 for the objects 1 to 4, respectively.

$$\begin{bmatrix} 0 & & & \\ 1.31 & 0 & & \\ 0.44 & 0.87 & 0 & \\ 0.43 & 1.74 & 0.87 & 0 \end{bmatrix}$$

## Variables of Mixed Types

- *how can we compute the dissimilarity between objects of mixed variable types?"*

- One approach is to group each kind of variable together, performing a separate cluster analysis for each variable type.

- A more preferable approach is to process all variable types together, performing a single cluster analysis.

- Suppose that the data set contains p variables of mixed type. The dissimilarity d(i, j) between objects i and j is defined as

$$d(i, j) = \frac{\sum_{f=1}^{p} \delta_{ij}^{(f)} d_{ij}^{(f)}}{\sum_{f=1}^{p} \delta_{ij}^{(f)}},$$

where the indicator $\delta_{ij}^{(f)} = 0$ if either (1) $x_{if}$ or $x_{jf}$ is missing (i.e., there is no measurement of variable $f$ for object $i$ or object $j$), or (2) $x_{if} = x_{jf} = 0$ and variable $f$ is asymmetric binary; otherwise, $\delta_{ij}^{(f)} = 1$. The contribution of variable $f$ to the dissimilarity between $i$ and $j$, that is, $d_{ij}^{(f)}$, is computed dependent on its type:

- If $f$ is interval-based: $d_{ij}^{(f)} = \frac{|x_{if} - x_{jf}|}{max_h x_{hf} - min_h x_{hf}}$, where $h$ runs over all nonmissing objects for variable $f$.
- If $f$ is binary or categorical: $d_{ij}^{(f)} = 0$ if $x_{if} = x_{jf}$; otherwise $d_{ij}^{(f)} = 1$.
- If $f$ is ordinal: compute the ranks $r_{if}$ and $z_{if} = \frac{r_{if} - 1}{M_f - 1}$, and treat $z_{if}$ as interval-scaled.
- If $f$ is ratio-scaled: either perform logarithmic transformation and treat the transformed data as interval-scaled; or treat $f$ as continuous ordinal data, compute $r_{if}$ and $z_{if}$, and then treat $z_{if}$ as interval-scaled.

Apply logarithmic transformation to its values. Based on the transformed values of 2.65, 1.34, 2.21, and 3.08 obtained for the objects 1 to 4.
$max_h x_h$ =3.08 and $min_h\ x_h$=1.34.
Then normalize the values in the dissimilarity matrix obtained in Example 7.5 by dividing each one by (3.08 – 1.34) = 1.74.

$$\begin{bmatrix} 0 & & & \\ 0.75 & 0 & & \\ 0.25 & 0.50 & 0 & \\ 0.25 & 1.00 & 0.50 & 0 \end{bmatrix}$$

We can now use the dissimilarity matrices for the three variables in our computation.

$$d(2, 1) = \frac{1(1)+1(1)+1(0.75)}{3} = 0.92.$$

$$\begin{bmatrix} 0 & & & \\ 0.92 & 0 & & \\ 0.58 & 0.67 & 0 & \\ 0.08 & 1.00 & 0.67 & 0 \end{bmatrix}$$

## Vector Objects

- There are several ways to define such a similarity function, $s(\boldsymbol{x}, \boldsymbol{y})$, to compare two vectors $\boldsymbol{x}$ and $\boldsymbol{y}$.
- One popular way is to define the similarity function as a cosine measure

$$s(\boldsymbol{x}, \boldsymbol{y}) = \frac{\boldsymbol{x}^t \cdot \boldsymbol{y}}{||\boldsymbol{x}||||\boldsymbol{y}||},$$

- where $x'$ is a transposition of vector $\boldsymbol{x}$, ||x|| is the Euclidean norm of vector $x'$ ,||y|| is the Euclidean norm of vector $\boldsymbol{y}$, and $s$ is essentially the cosine of the angle between vectors $\boldsymbol{x}$ and $\boldsymbol{y}$.

Nonmetric similarity between two objects using cosine. Suppose we are given two vectors, $\boldsymbol{x} = (1, 1, 0, 0)$ and $\boldsymbol{y} = (0, 1, 1, 0)$. By Equation (7.16), the similarity between $\boldsymbol{x}$ and $\boldsymbol{y}$ is $s(\boldsymbol{x}, \boldsymbol{y}) = \frac{(0+1+0+0)}{\sqrt{2}\sqrt{2}} = 0.5$. ■

A simple variation of the above measure is

$$s(\boldsymbol{x}, \boldsymbol{y}) = \frac{\boldsymbol{x}^t \cdot \boldsymbol{y}}{\boldsymbol{x}^t \cdot \boldsymbol{x} + \boldsymbol{y}^t \cdot \boldsymbol{y} - \boldsymbol{x}^t \cdot \boldsymbol{y}} \tag{7.17}$$

which is the ratio of the number of attributes shared by $\boldsymbol{x}$ and $\boldsymbol{y}$ to the number of attributes possessed by $\boldsymbol{x}$ or $\boldsymbol{y}$. This function, known as the **Tanimoto coefficient or Tanimoto distance**.

# A Categorization of Major Clustering Methods

**Partitioning methods**

- In a database of *n* objects or data tuples, a partitioning method constructs *k* partitions of the data, where each partition represents a cluster and ***k <= n***.

It classifies the data into *k* groups, which together satisfy the following requirements:

- (1) each group must contain at least one object
- (2) each object must belong to exactly one group.
- Given k, the number of partitions to construct, a partitioning method creates an **initial partitioning**.
- It then uses an **iterative relocation technique** that attempts to improve the partitioning by moving objects from one group to another.

**Popular heuristic methods**

- The ***k-means* algorithm**, where each cluster is represented by the mean value of the objects in the cluster.

- The ***k-medoids* algorithm**, where each cluster is represented by one of the objects located near the center of the cluster.

- Heuristic clustering methods work well for finding **spherical-shaped clusters** in **small to medium-sized databases.**

## Hierarchical methods

- A hierarchical method creates a hierarchical **decomposition** of the given set of data objects.

- It is classified into ***agglomerative*** **or** ***divisive****.*

- ***Agglomerative*** *approach*, also called the ***bottom-up*** approach, starts with each object forming a separate group.

- It successively merges the objects or groups that are close to one another, until all of the groups are merged into one (the topmost level of the hierarchy), or until a termination condition holds.

- The ***divisive approach***, also called the ***top-down* approach**, starts with all of the objects in the same cluster.

- In each successive iteration, a cluster is split up into smaller clusters, until eventually each object is in one cluster, or until a termination condition holds.

- Hierarchical methods suffer from the fact that once a step (merge or split) is done, it can never be undone.

- There are **two** approaches to **improving the quality** of hierarchical clustering:

  (1) perform careful analysis of **object "linkages"** at each hierarchical partitioning

  (2) **Integrate** hierarchical agglomeration and other approaches by first using a hierarchical agglomerative algorithm to group object into ***microclusters***, and then performing ***macroclustering*** on the microclusters using another clustering method such as iterative relocation, as in BIRCH.

## Density-based methods

- General idea is to continue growing the given cluster as long as the density (number of objects or data points) in the "neighborhood" exceeds some threshold.

- That is, for each data point within a given cluster, the neighborhood of a given radius has to contain at least a minimum number of points.

- Such a method can be used to filter out noise (outliers) and discover clusters of arbitrary shape.

- Eg: DBSCAN, OPTICS, DENCLUE.

## Grid-based methods

- Grid-based methods **quantize the object space** into a finite number of cells that form a grid structure.

- All of the clustering operations are performed on the grid structure.

- It's processing time is fast, which is typically independent of the number of data objects and dependent only on the number of cells in each dimension in the quantized space.

- Eg: STING

## Model-based methods

- Model-based methods hypothesize a model for each of the clusters and find the **best fit** of the data to the given model.

- A model-based algorithm may locate clusters by constructing a density function that reflects the spatial distribution of the data points.

- Eg: **EM** is an algorithm that performs **expectation-maximization analysis** based on **statistical modeling**.

- **COBWEB** is a conceptual learning algorithm that performs **probability analysis** and takes concepts as a model for clusters.

There are **two classes** of clustering

- Tasks that require special attention. One is *clustering high-dimensional data*, and the other is *constraint-based clustering*.

**Clustering high-dimensional data**

- Particularly important task in cluster analysis because many applications require the analysis of objects containing a large number of features or dimensions.
- As the number of dimensions increases, the data become increasingly sparse so that the distance measurement between pairs of points become meaningless and the average density of points anywhere in the data is likely to be low

- **CLIQUE** and **PROCLUS** are two influential ***subspace clustering methods***, which search for clusters in subspaces (or subsets of dimensions) of the data, rather than over the entire data space.

- **Frequent pattern–based clustering**, another clustering methodology, extracts ***distinct frequent patterns*** among subsets of dimensions that occur frequently.

- Eg: Pcluster is the frequent pattern based clustering.

**Constraint-based clustering**

- It performs clustering by incorporation of user-specified or application-oriented constraints.

- ***spatial clustering*** with the existence of obstacles and clustering under user-specified constraints.

- ***semi-supervised clustering*** - which employs, for example, pairwise constraints (such as pairs of instances labelled as belonging to the same or different clusters) in order to improve the quality of the resulting clustering.

# Partitioning Methods

**Classical Partitioning Methods: *k*-Means and *k*-Medoids**

**Centroid-Based Technique: The *k*-Means Method**

- The *k*-means algorithm takes the input parameter, *k*, and partitions a set of *n* objects into *k* clusters so that the resulting **intracluster similarity is high** but the **intercluster similarity is low**.

- Cluster similarity is measured in regard to the ***mean* value** of the objects in a cluster, which can be viewed as the **cluster's *centroid* or *center of gravity***.

- Randomly selects *k* of the objects, each of which initially represents a cluster mean or center.

- For each of the remaining objects, an object is assigned to the cluster to which it is the most similar, based on the distance between the object and the cluster mean.

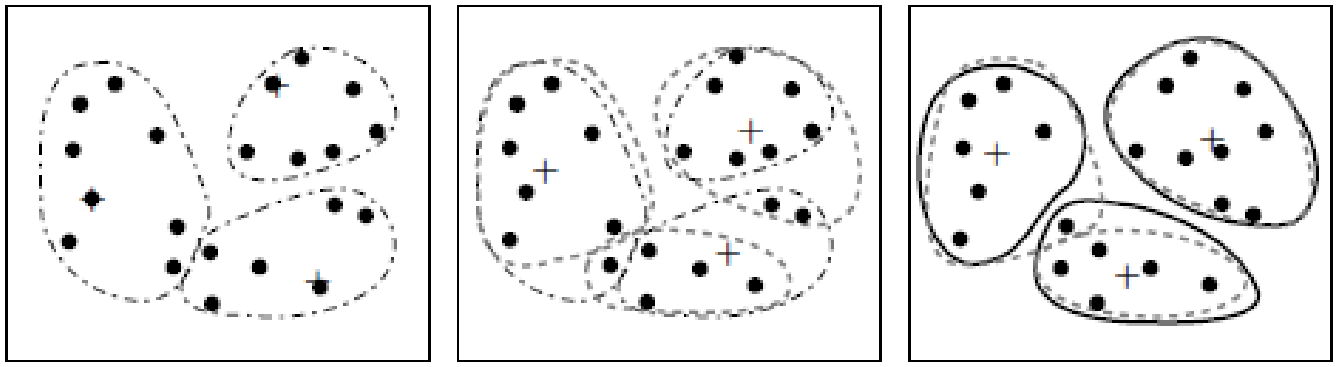
- It then computes the new mean for each cluster.
- This process iterates until the criterion function converges.

- The square-error criterion is used

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

- ***E*** - sum of the square error for all objects in the data set.
- ***p*** **-** point in space representing a given object.
- ***mi*** **-** mean of cluster *Ci.*

- Suppose that there is a set of objects located in space as depicted in the rectangle
- Let k = 3; that is, the user would like the objects to be partitioned into three clusters.




Clustering of a set of objects based on the $k$-means method. (The mean of each cluster is marked by a "+".)

- we arbitrarily choose three objects as the three initial cluster centers, where cluster centers are marked by a "+".
- Each object is distributed to a cluster based on the cluster center to which it is the nearest.

- The cluster centers are updated. That is, the mean value of each cluster is recalculated based on the current objects in the cluster.

- Using the new cluster centers, the objects are redistributed to the clusters based on which cluster center is the nearest.

- The process of **iteratively reassigning objects** to clusters to improve the partitioning is referred to as iterative relocation.
- Eventually, no redistribution of the objects in any cluster occurs, and so the process terminates.
- The resulting clusters are returned by the clustering process.

**Algorithm: *k*-means.** The *k*-means algorithm for partitioning, where each cluster's center is represented by the mean value of the objects in the cluster.

**Input:**

- *k*: the number of clusters,
- *D*: a data set containing *n* objects.

**Output:** A set of *k* clusters.

**Method:**

(1) arbitrarily choose *k* objects from *D* as the initial cluster centers;
(2) **repeat**
(3) (re)assign each object to the cluster to which the object is the most similar, based on the mean value of the objects in the cluster;
(4) update the cluster means, i.e., calculate the mean value of the objects for each cluster;
(5) **until** no change;

---

The *k*-means partitioning algorithm.

- It works well when the clusters are **compact clouds** that are rather **well separated** from one another.

- The method is **relatively scalable** and **efficient** in processing large data sets because the **computational complexity** of the algorithm is $O(nkt)$.

- Where $n$ is the total number of objects, $k$ is the number of clusters, and $t$ is the number of iterations.

- Normally, $k<<n$ and $t<<n$. The method often terminates at a local optimum.

## Limitations of K-means

- The *k*-means method, however, can be applied only when the mean of a cluster is defined.

- The necessity for users to **specify *k***, the number of clusters, in advance can be seen as a disadvantage.

- The *k*-means method is **not suitable** for discovering clusters with **non convex shapes** or **clusters of very different size.**

- It is **sensitive to noise** and **outlier data** points because a small number of such data can substantially influence the mean value.

There are quite a **few variants** of the ***k*-means method**

- These can differ in the **selection** of the **initial *k* means**, the calculation of dissimilarity, and the **strategies** for calculating cluster means.

- The ***k*-modes method**, which extends the *k*-means paradigm to cluster categorical data by replacing the means of clusters with modes.

- *How can we make the k-means algorithm more scalable?*

By identifying **three kinds of regions** in data

1. Regions that are compressible.
2. Regions that must be maintained in main memory.
3. Regions that are discardable.

- An object is **discardable** if its membership in a cluster is ascertained.
- An object is **compressible** if it is not discardable but belongs to a tight subcluster.
- If an object is **neither discardable nor compressible**, then it should be retained in main memory.

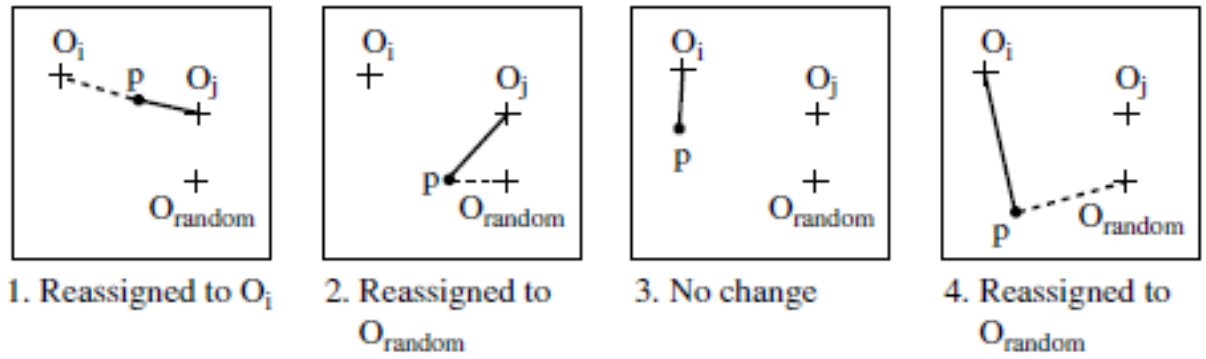
# Representative Object-Based Technique: The *k*-Medoids Method

- The ***k*-means** algorithm is **sensitive to outliers** because an object with an extremely large value may substantially distort the distribution of data.
- Instead of taking the **mean value** of the objects in a cluster as a reference point, we take **actual objects** to represent the clusters, using **one representative object** per cluster.
- Each remaining object is clustered with the representative object to which it is the most similar.
- Absolute-error criterion is used

$$E = \sum_{j=1}^{k} \sum_{p \in C_j} |p - o_j|,$$

- $E$ - sum of the absolute error for all objects in the data set
- $\boldsymbol{p}$ - point in space representing a given object in cluster$Cj$
- $\boldsymbol{oj}$ - representative object of$Cj$.

- Case 1: $\boldsymbol{p}$ currently belongs to representative object, $\boldsymbol{o_j}$. If $\boldsymbol{o_j}$ is replaced by $\boldsymbol{o}_{random}$ as a representative object and $\boldsymbol{p}$ is closest to one of the other representative objects, $\boldsymbol{o_i}$, $i \neq j$, then $\boldsymbol{p}$ is reassigned to $\boldsymbol{o_i}$.
- Case 2: $\boldsymbol{p}$ currently belongs to representative object, $\boldsymbol{o_j}$. If $\boldsymbol{o_j}$ is replaced by $\boldsymbol{o}_{random}$ as a representative object and $\boldsymbol{p}$ is closest to $\boldsymbol{o}_{random}$, then $\boldsymbol{p}$ is reassigned to $\boldsymbol{o}_{random}$.
- Case 3: $\boldsymbol{p}$ currently belongs to representative object, $\boldsymbol{o_i}$, $i \neq j$. If $\boldsymbol{o_j}$ is replaced by $\boldsymbol{o}_{random}$ as a representative object and $\boldsymbol{p}$ is still closest to $o_i$, then the assignment does not change.
- Case 4: $\boldsymbol{p}$ currently belongs to representative object, $\boldsymbol{o_i}$, $i \neq j$. If $\boldsymbol{o_j}$ is replaced by $\boldsymbol{o}_{random}$ as a representative object and $\boldsymbol{p}$ is closest to $\boldsymbol{o}_{random}$, then $\boldsymbol{p}$ is reassigned to $\boldsymbol{o}_{random}$.




Four cases of the cost function for $k$-medoids clustering.

- The cost function calculates the *difference* in absolute-error value if a current representative object is replaced by a nonrepresentative object.
- Total cost - negative, then ***oj*** is replaced or swapped with ***o****random.*
- Total cost - positive, the current representative object, **oj**, is considered acceptable, and nothing is changed in the iteration.

- **PAM(Partitioning Around Medoids)** was one of the first *k*-medoids algorithms introduced.
- The complexity of each iteration is $O(k(n-k)^2)$

**Algorithm:** ***k*-medoids.** PAM, a *k*-medoids algorithm for partitioning based on medoid or central objects.

**Input:**

- *k*: the number of clusters,
- *D*: a data set containing *n* objects.

**Output:** A set of *k* clusters.

**Method:**

(1) arbitrarily choose *k* objects in *D* as the initial representative objects or seeds;
(2) **repeat**
(3) assign each remaining object to the cluster with the nearest representative object;
(4) randomly select a nonrepresentative object, $o_{random}$;
(5) compute the total cost, *S*, of swapping representative object, $o_j$, with $o_{random}$;
(6) **if** $S < 0$ **then** swap $o_j$ with $o_{random}$ to form the new set of *k* representative objects;
(7) **until** no change;

PAM, a *k*-medoids partitioning algorithm.

## Advantage & Disadvantage of K-Medoid

- The *k*-medoids method is more robust than *k*-means in the presence of noise and outliers, because a medoid is less influenced by outliers or other extreme values than a mean.

- K-Medoid processing is more costly than the *k*-means method.

- Both methods require the user to specify *k*, the number of clusters.

- The *median* can be used, resulting in the ***k-median*** method, where the median or "**middle value**" is taken for each ordered attribute.

- ***k-modes*** method, the **most frequent value** for each attribute is used.

# Hierarchical Methods

- A hierarchical clustering method works by grouping data objects into a **tree of clusters**.

- Hierarchical clustering methods can be further classified as either ***agglomerative*** **or** ***divisive***, depending on whether the hierarchical decomposition is formed in a **bottom-up (merging) or top-down (splitting)** fashion**.**

## Agglomerative and Divisive Hierarchical Clustering

- **Agglomerative hierarchical clustering:** This bottom-up strategy starts by placing each object in its own cluster and then merges these atomic clusters into larger and larger clusters, until all of the objects are in a single cluster or until certain termination conditions are satisfied.

- **Divisive hierarchical clustering:** This top-down strategy does the reverse of agglomerative hierarchical clustering by starting with all objects in one cluster.
- It subdivides the cluster into smaller and smaller pieces, until each object forms a cluster on its own or until it satisfies certain termination conditions

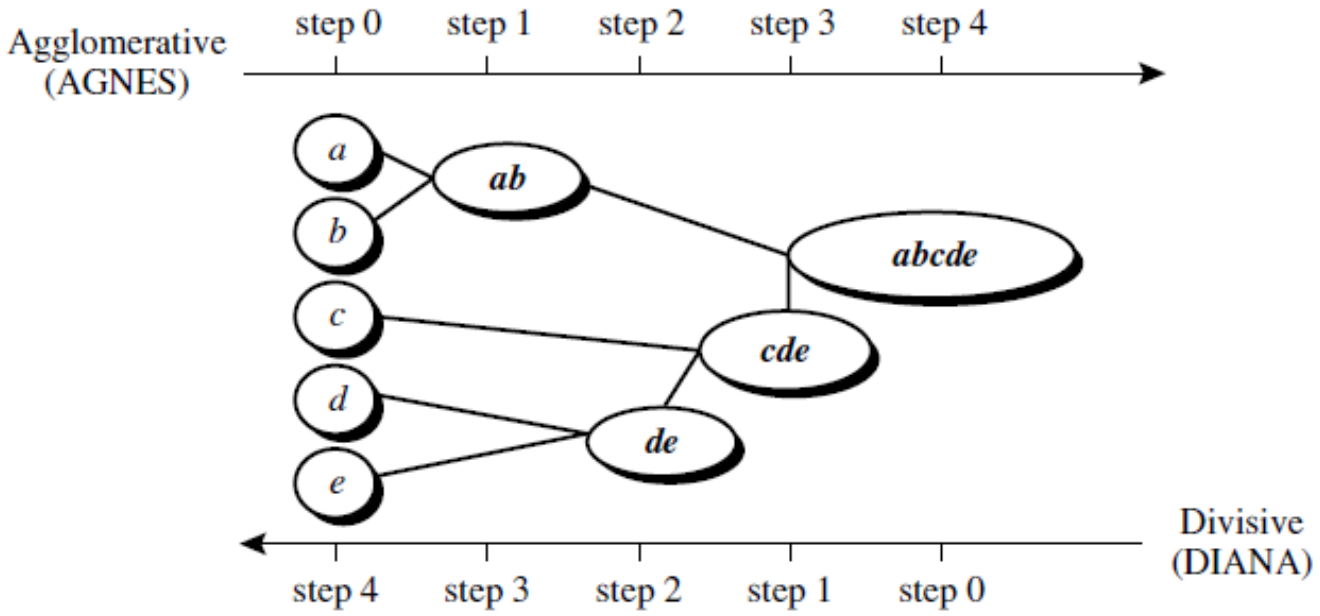
**Agglomerative versus divisive hierarchical cluster**ing

- The application of **AGNES (AGglomerative NESting),** an agglomerative hierarchical clustering method, and **DIANA (DIvisive ANAlysis),** a divisive hierarchical clustering method, to a data set of five objects, {***a***, ***b***, ***c***, ***d***, ***e***}.

- **AGNES** - The **minimum Euclidean distance** between any two objects from different clusters. This is a **single-linkage** approach.

- **DIANA –** The **maximum Euclidean distance** between the closest neighboring objects in the cluster.

- In agglomerative or divisive hierarchical clustering, the user can **specify** the desired **number of clusters** as a termination condition.




Agglomerative and divisive hierarchical clustering on data objects $\{a, b, c, d, e\}$.

- A **tree structure** called a **dendrogram** is commonly used to represent the process of hierarchical clustering.
- It shows how objects are grouped together step by step.

**Four** widely used measures for distance between clusters

- where $|p - p'|$ $-$ distance between two objects or points, $p$ and $p'$
- $m_i$ - mean for cluster $C_i$.
- $n_i$ - number of objects in $C_i$.

- When an algorithm uses the ***minimum distance*, *dmin*(*Ci*, *Cj*),** to measure the distance between clusters, it is sometimes called a **nearest-neighbor clustering** algorithm.

- If the clustering process is terminated when the distance between nearest clusters exceeds an arbitrary threshold, it is called a **single-linkage algorithm**.

- An agglomerative hierarchical clustering algorithm that uses the **minimum distance** measure is also called a **minimal spanning tree algorithm.**

- When an algorithm uses the ***maximum distance*, *dmax*(*Ci*, *Cj*)**, to measure the distance between clusters, it is sometimes called a **farthest-neighbor clustering** algorithm.

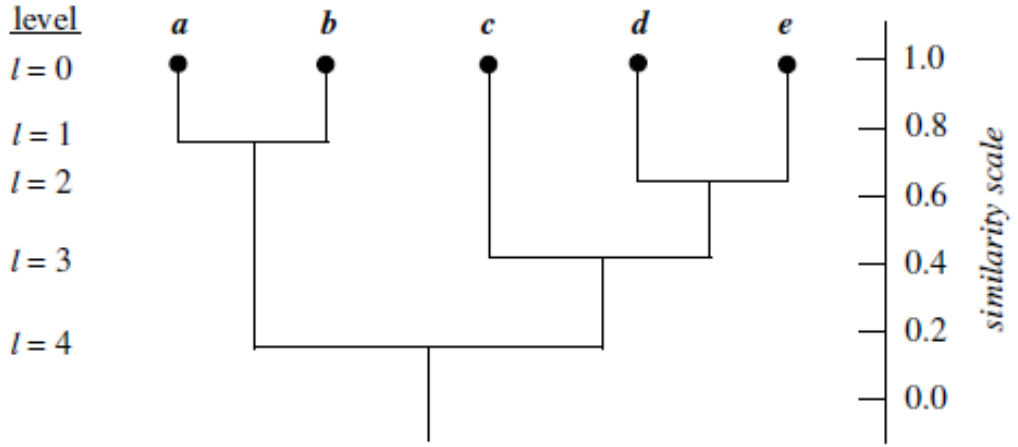
- If the clustering process is terminated when the **maximum distance** between nearest clusters exceeds an arbitrary threshold, it is called a **complete-linkage** algorithm.

$$\text{Minimum distance}: \quad d_{min}(C_i, C_j) = \quad min_{p \in C_i,\, p' \in C_j} |p - p'|$$

$$\text{Maximum distance}: \quad d_{max}(C_i, C_j) = \quad max_{p \in C_i,\, p' \in C_j} |p - p'|$$

$$\text{Mean distance}: \quad d_{mean}(C_i, C_j) = \quad |m_i - m_j|$$

$$\text{Average distance}: \quad d_{avg}(C_i, C_j) = \quad \frac{1}{n_i n_j} \sum_{p \in C_i} \sum_{p' \in C_j} |p - p'|$$




Dendrogram representation for hierarchical clustering of data objects $\{a, b, c, d, e\}$.

***Difficulties with hierarchical clustering***

- Difficulties regarding the selection of merge or split points.

- One promising direction for **improving** the **clustering quality** of hierarchical methods is to **integrate** hierarchical clustering with other clustering techniques, resulting in **multiple-phase clustering.**

**Three methods** are introduced

- **BIRCH** - partitioning objects hierarchically using tree structures.
- **ROCK –** merges clusters based on their interconnectivity.
- **Chameleon –** explores dynamic modeling in hierarchical clustering.

# Density Based Methods

**DBSCAN(Density Based Spatial Clustering of Applications with Noise)**

- To discover clusters with arbitrary shape, density-based clustering methods have been developed.
- DBSCAN grows clusters according to a density-based connectivity analysis.
- The algorithm grows regions with sufficiently high density into clusters and discovers clusters of arbitrary shape in spatial databases with noise.
- It defines a cluster as a maximal set of density-connected points.

- The neighborhood within a radius $\varepsilon$ of a given object is called the neighborhood of the object.

- If the $\varepsilon$ -neighborhood of an object contains at least a **minimum number**, *MinPts*, of objects, then the object is called a **core object**.

- Given a set of objects, *D*, we say that an object ***p*** is **directly density-reachable** from object ***q*** if ***p*** is within the $\varepsilon$ -neighborhood of ***q***, and ***q*** is a core object.

- An object ***p*** is **density-reachable** from object ***q*** with respect to $\varepsilon$ and *MinPts* in a set of objects, *D*, if there is a chain of objects ***p*1**,…, ***pn***, where ***p*1** = ***q*** and ***pn*** = ***p*** such that ***pi*+1** is directly density-reachable from ***pi*** with respect to $\varepsilon$ and *MinPts*, for 1<= *i* <= *n*, ***pi*** ∈ *D*.

- An object ***p*** is **density-connected** to object ***q*** with respect to $\varepsilon$ and *MinPts* in a set of objects, *D*, if there is an object **o**∈ *D* such that both ***p*** and ***q*** are density-reachable from ***o*** with respect to $\varepsilon$ and *MinPts*.

- Density reachability is the **transitive closure** of **direct density reachability**, and this relationship is **asymmetric**.
- Only **core objects** are mutually **density reachable**. Density connectivity, however, is a **symmetric** relation

**Density-reachability and density connectivity**

- For a given $\varepsilon$ represented by the radius of the circles, and, say, let *MinPts* = 3.
- Of the labeled points,***m***, ***p***, ***o***, and ***r*** are core objects because each is in an $\varepsilon$ -neighborhood containing at least three points.

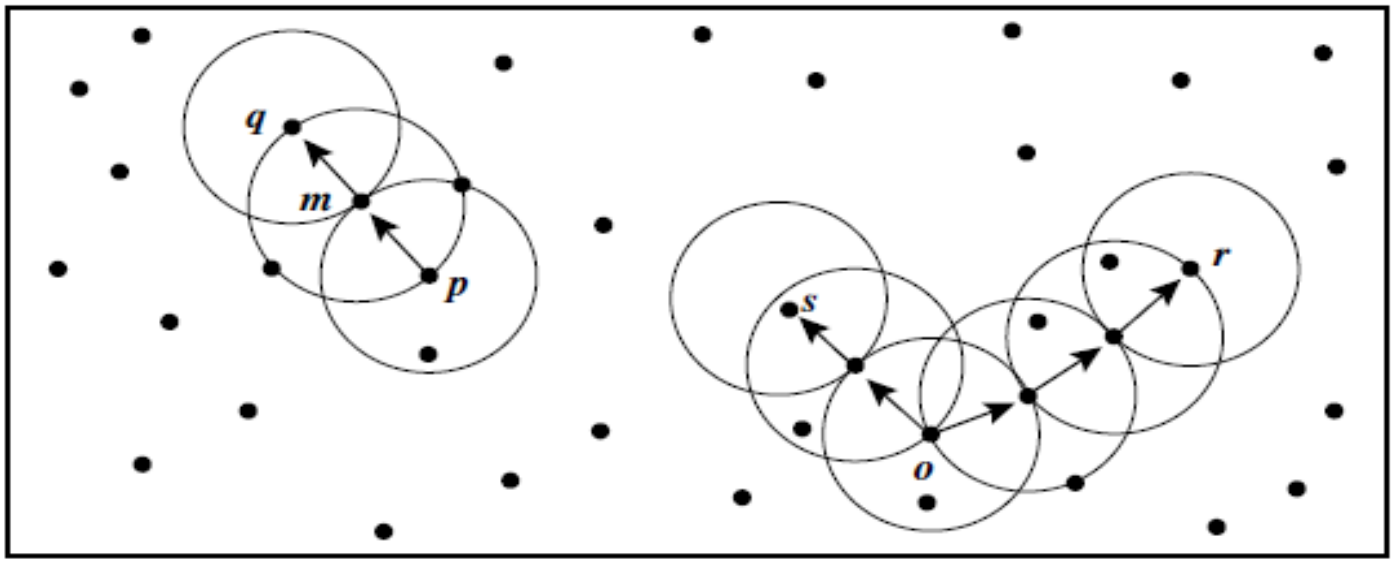
- ***q*** is **directly density-reachable** from ***m***. ***m*** is directly density-reachable from ***p*** and vice versa.

- ***q*** is **(indirectly) density-reachable** from ***p*** because ***q*** is directly density-reachable from ***m*** and ***m*** is directly density-reachable from ***p***. However, ***p*** is not density-reachable from ***q*** because ***q*** is not a core object.

- Similarly, ***r*** and ***s*** are density-reachable from ***o***, and ***o*** is density-reachable from ***r***.

- ***o***, ***r***, and ***s*** are all density-connected.

- A density-based cluster is a set of density-connected objects that is maximal with respect to density-reachability.
- Every object not contained in any cluster is considered to be *noise*.




Density reachability and density connectivity in density-based clustering.

***How does DBSCAN find clusters?***

- DBSCAN searches for clusters by checking the $\varepsilon$-neighborhood of each point in the database.

- If the $\varepsilon$-neighborhood of a point ***p*** contains more than *MinPts*, a new cluster with ***p*** as a core object is created.

- DBSCAN then iteratively collects directly density-reachable objects from these core objects, which may involve the merge of a few density-reachable clusters.

- The process terminates when no new point can be added to any cluster.

- If a **spatial index** is used, the computational complexity of DBSCAN is $O(\mathrm{n}log_n)$, where $n$ is the number of database objects.

- Otherwise, it is $O(n^2)$.