

# Cluster Analysis

## Unit - 3


Reference:

Jiawei Han

Department of Computer Science

University of Illinois at Urbana-Champaign

# Cluster Analysis

1. What is Cluster Analysis? 
2. Types of Data in Cluster Analysis
3. A Categorization of Major Clustering Methods
4. Partitioning Methods
5. Hierarchical Methods
6. Density-Based Methods
7. Grid-Based Methods
8. Model-Based Methods
9. Clustering High-Dimensional Data
10. Constraint-Based Clustering
11. Outlier Analysis
12. Data Mining Applications

# What is Cluster Analysis?

- Cluster: a collection of data objects
  - Similar to one another within the same cluster
  - Dissimilar to the objects in other clusters
- Cluster analysis
  - Finding similarities between data according to the characteristics found in the data and grouping similar data objects into clusters
- **Unsupervised learning**: no predefined classes
- Typical applications
  - As a **stand-alone tool** to get insight into data distribution
  - As a **preprocessing step** for other algorithms

# 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

# Quality: What Is Good Clustering?

- A good clustering method will produce high quality clusters with
  - high intra-class similarity
  - low inter-class similarity
- The quality of a clustering result 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

# Measure the Quality of Clustering

- **Dissimilarity/Similarity metric**: Similarity is expressed in terms of a distance function, typically metric:  $d(i, j)$
- There is a separate “quality” function that measures the “goodness” of a cluster.
- The definitions of **distance functions** are usually very different for interval-scaled, boolean, categorical, ordinal ratio, and vector variables.
- Weights should be associated with different variables based on applications and data semantics.
- It is hard to define “similar enough” or “good enough”
  - the answer is typically highly subjective.

# Requirements of Clustering in Data Mining

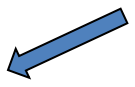
- Scalability
- Ability to deal with different types of attributes
- Ability to handle dynamic data
- Discovery of clusters with arbitrary shape
- Minimal requirements for domain knowledge to determine input parameters
- Able to deal with noise and outliers
- Insensitive to order of input records
- High dimensionality
- Incorporation of user-specified constraints
- Interpretability and usability

# Orthogonal aspects for comparing clustering methods

- **The partitioning criteria** – with hierarchy or without hierarchy
- **Separation of clusters** – mutually exclusive or overlapping clusters.
- **Similarity measure** - distance-based methods or density- and continuity-based methods
- **Clustering space** – entire dataspace or subspace



# Cluster Analysis

1. What is Cluster Analysis?
2. Types of Data in Cluster Analysis 
3. A Categorization of Major Clustering Methods
4. Partitioning Methods
5. Hierarchical Methods
6. Density-Based Methods
7. Grid-Based Methods
8. Model-Based Methods
9. Clustering High-Dimensional Data
10. Constraint-Based Clustering
11. Outlier Analysis
12. Data Mining Applications

# Types of Data in Cluster Analysis

- Data Matrix versus Dissimilarity Matrix
- Proximity Measures for Nominal Attributes
- Proximity Measures for Binary Attributes
- Dissimilarity of Numeric Data: Minkowski Distance
- Proximity Measures for Ordinal Attributes
- Dissimilarity for Attributes of Mixed Types
- Cosine Similarity

# Data Matrix versus Dissimilarity Matrix (1)

- Suppose that we have  $n$  objects (e.g., persons, items, or courses) described by  $p$  attributes (also called measurements or features, such as age, height, weight, or gender).
- The objects are  $x_1 = (x_{11}, x_{12}, \dots, x_{1p})$ ,  $x_2 = (x_{21}, x_{22}, \dots, x_{2p})$ , and so on, where  $x_{ij}$  is the value for object  $x_i$  of the  $j$ th attribute.
- The objects may be tuples in a relational database, and are also referred to as data samples or feature vectors.
- Main memory-based clustering and nearest-neighbor algorithms typically operate on either of the following two data structures: Data matrix and Dissimilarity matrix

# Data Matrix versus Dissimilarity Matrix (2)

## Data matrix (or *object-by-attribute structure*):

- This structure stores the  $n$  data objects in the form of a relational table, or  $n$ -by- $p$  matrix ( $n$  objects  $\times$   $p$  attributes)
- Each row corresponds to an object.
- A data matrix is made up of two entities or “things,” namely rows (for objects) and columns (for attributes).
- Therefore, the data matrix is often called a **two-mode** matrix.
- As part of our notation, we may use  $f$  to index through the  $p$  attributes.

## Data Matrix versus Dissimilarity Matrix (3)

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

Data Matrix

# Data Matrix versus Dissimilarity Matrix (4)

- **Dissimilarity matrix** (or *object-by-object structure*):
  - This structure stores a collection of proximities that are available for all pairs of  $n$  objects.
  - It is often represented by an  $n$ -by- $n$  table: where  $d(i,j)$  is the measured **dissimilarity** or “difference” between objects  $i$  and  $j$ .
  - In general,  $d(i,j)$  is a non-negative 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.
  - Note that  $d(i,i) = 0$ ; that is, the difference between an object and itself is 0.

# Data Matrix versus Dissimilarity Matrix (5)

- Furthermore,  $d(i,j) = d(j,i)$ . (For readability,  $d(i,j)$  entries are not shown ; the matrix is Symmetric)
- The dissimilarity matrix contains one kind of entity (dissimilarities) and so is called a **one-mode** matrix.
- Many clustering and nearest-neighbor algorithms operate on a dissimilarity matrix.
- Data in the form of a data matrix can be transformed into a dissimilarity matrix before applying such algorithms

# Data Matrix versus Dissimilarity Matrix (6)

- Measures of similarity can often be expressed as a function of measures of dissimilarity.
- For example, for nominal data,

$$\text{sim}(i, j) = 1 - d(i, j),$$

- where  $\text{sim}(i, j)$  is the similarity between objects  $i$  and  $j$ .

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

Dissimilarity Matrix



# Proximity Measures for Nominal Attributes (1)

- A nominal attribute can take on two or more states.
- For example, *map color* is a nominal attribute that may have, say, five states: *red*, *yellow*, *green*, *pink*, and *blue*.
- Let the number of states of a nominal attribute be  $M$ .
- The states can be denoted by letters, symbols, or a set of integers, such as  $1, 2, \dots, M$ .
- “How is dissimilarity computed between objects described by nominal attributes?”

# Proximity Measures for Nominal Attributes (2)

- The dissimilarity between two objects  $i$  and  $j$  can be computed based on the ratio of mismatches:

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

- where  $m$  is the number of *matches* (i.e., the number of attributes for which  $i$  and  $j$  are in the same state), and  $p$  is the total number of attributes describing the objects.
- Weights can be assigned to increase the effect of  $m$  or to assign greater weight to the matches in attributes having a larger number of states.

# Proximity Measures for Nominal Attributes (3)

- **Example 2.17** Dissimilarity between nominal attributes.
- Suppose that we have the sample data of Table 2.2 a where *test-1* is nominal.

<b>Object Identifier</b>	<b>test-1 (nominal)</b>
1	code A
2	code B
3	code C
4	code A

# Proximity Measures for Nominal Attributes (4)

- Since here we have one nominal attribute, *test-1*, we set  $p = 1$  in Eq. (2.11) so that  $d(i, j)$  evaluates to 0 if objects  $i$  and  $j$  match, and 1 if the objects differ.
- Thus, we get the following dissimilarity matrix

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

- From this, we see that all objects are dissimilar except objects 1 and 4 (i.e.,  $d(4,1)=0$ )

# Proximity Measures for Nominal Attributes (5)

- Alternatively, similarity can be computed as

$$\text{sim}(i, j) = 1 - d(i, j) = \frac{m}{p}.$$

- Proximity between objects described by nominal attributes can be computed using an alternative encoding scheme.
- Nominal attributes can be encoded using asymmetric binary attributes by creating a new binary attribute for each of the  $M$  states.
- For an object with a given state value, the binary attribute representing that state is set to 1, while the remaining binary attributes are set to 0.

# Proximity Measures for Binary Attributes (1)

- A binary attribute has only one of two states: 0 and 1, where 0 means that the attribute is absent, and 1 means that it is present.
- Given the attribute *smoker* describing a patient, for instance, 1 indicates that the patient smokes, while 0 indicates that the patient does not.
- *“So, how can we compute the dissimilarity between two binary attributes?”*
- One approach involves computing a dissimilarity matrix from the given binary data.

# Proximity Measures for Binary Attributes (2)

- If all binary attributes are thought of as having the same weight, we have the 2 x 2 contingency table of Table 2.3, where  $q$  is the number of attributes that equal 1 for both objects  $i$  and  $j$ ,  $r$  is the number of attributes that equal 1 for object  $i$  but equal 0 for object  $j$ ,  $s$  is the number of attributes that equal 0 for object  $i$  but equal 1 for object  $j$ , and  $t$  is the number of attributes that equal 0 for both objects  $i$  and  $j$ .

Contingency Table for Binary Attributes

		Object $j$		
		1	0	sum
Object $i$	1	$q$	$r$	$q + r$
	0	$s$	$t$	$s + t$
sum		$q + s$	$r + t$	$p$

# Proximity Measures for Binary Attributes (3)

- The total number of attributes is  $p$ , where  $p = q + r + s + t$
- For symmetric binary attributes, each state is equally valuable.
- Dissimilarity that is based on symmetric binary attributes is called **symmetric binary dissimilarity**.
- If objects  $i$  and  $j$  are described by symmetric binary attributes, then the dissimilarity between  $i$  and  $j$  is

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

- For asymmetric binary attributes, the two states are not equally important, such as the *positive* (1) and *negative* (0) outcomes of a disease test.



# Proximity Measures for Binary Attributes (4)

- For asymmetric binary attributes, the two states are not equally important, such as the *positive* (1) and *negative* (0) outcomes of a disease test.
- Given two asymmetric binary attributes, the agreement of two 1s (a positive match) is then considered more significant than that of two 0s (a negative match).
- Therefore, such binary attributes are often considered “monary” (having one state).
- The dissimilarity based on these attributes is called **asymmetric binary dissimilarity**, where the number of negative matches,  $t$ , is considered unimportant and is thus ignored in the following computation:

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

# Proximity Measures for Binary Attributes (5)

- Complementarily, we can measure the difference between two binary attributes based on the notion of similarity instead of dissimilarity.
- For example, the **asymmetric binary similarity** between the objects  $i$  and  $j$  can be computed as

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

- The coefficient  $\text{sim}(i, j)$  is called the **Jaccard coefficient**.
- When both symmetric and asymmetric binary attributes occur in the same data set, the mixed attributes approach can be applied.

# Proximity Measures for Binary Attributes (6)

- **Example 2.18 Dissimilarity between binary attributes.** Suppose that a patient record table (Table 2.4) 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.

**Table 2.4** Relational Table Where Patients Are Described by Binary Attributes

<i>name</i>	<i>gender</i>	<i>fever</i>	<i>cough</i>	<i>test-1</i>	<i>test-2</i>	<i>test-3</i>	<i>test-4</i>
Jack	M	Y	N	P	N	N	N
Jim	M	Y	Y	N	N	N	N
Mary	F	Y	N	P	N	P	N
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

# Proximity Measures for Binary Attributes (7)

- 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 attributes.
- The distance between each pair of the three patients—Jack, Mary, and Jim—is

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

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

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

# Proximity Measures for Binary Attributes (8)

- These measurements suggest that Jim and Mary 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

# Dissimilarity of Numeric Data (1)

- Euclidean, Manhattan, and Minkowski distances are commonly used for computing the dissimilarity of objects described by numeric attributes.
- In some cases, the data are normalized before applying distance calculations.
- This involves transforming the data to fall within a smaller or common range, such as  $[-1, 1]$  or  $[0.0, 1.0]$ .
- Consider a *height* attribute, for example, which could be measured in either meters or inches.
- In general, expressing an attribute in smaller units will lead to a larger range for that attribute, and thus tend to give such attributes greater effect or “weight.”

## Dissimilarity of Numeric Data (2)

- Normalizing the data attempts to give all attributes an equal weight.
- It may or may not be useful in a particular application.
- The most popular distance measure is **Euclidean distance** (i.e., straight line or “as the crow flies”).
- Let  $i = (x_{i1}, x_{i2}, \dots, x_{ip})$ ,  $x_2 = (x_{j1}, x_{j2}, \dots, x_{jp})$  be two objects described by  $p$  numeric attributes.
- The Euclidean distance between objects  $i$  and  $j$  is defined as

$$d(i, j) = \sqrt{(x_{i1} - x_{j1})^2 + (x_{i2} - x_{j2})^2 + \dots + (x_{ip} - x_{jp})^2}.$$

## Dissimilarity of Numeric Data (3)

- Another well-known measure is the **Manhattan (or city block) distance**, named so because it is the distance in blocks between any two points in a

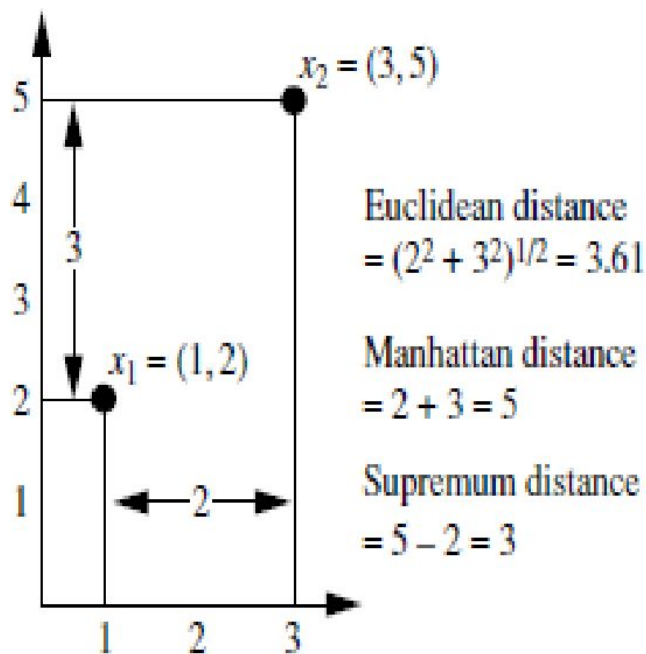
$$d(i, j) = |x_{i1} - x_{j1}| + |x_{i2} - x_{j2}| + \cdots + |x_{ip} - x_{jp}|.$$

- It is defined as Both the Euclidean and the Manhattan distance satisfy the following mathematical properties:
  - **Non-negativity:**  $d(i, j) > 0$  i.e Distance is a non-negative number.
  - **Identity of indiscernibles:**  $d(i, i) = 0$  i.e The distance of an object to itself is 0.
  - **Symmetry:**  $d(i, j) = d(j, i)$  Distance is a symmetric function.
  - **Triangle inequality:**  $d(i, j) \leq d(i, k) + d(k, j)$  Going directly from object  $i$  to object  $j$  in space is no more than making a detour over any other object  $k$ .
- A measure that satisfies these conditions is known as **metric**.



## Dissimilarity of Numeric Data (4)

- **Example 2.19 Euclidean distance and Manhattan distance.** Let  $x_1 = (1, 2)$  and  $x_2 = (3, 5)$  represent two objects as shown in Figure p 2.23.



**Figure 2.23** Euclidean, Manhattan, and supremum distances between two objects.

## Dissimilarity of Numeric Data (5)

- The Euclidean distance between the two is  $\sqrt{2^2 + 3^2} = 3.61$ .
- The Manhattan distance between the two is  $2+3 = 5$ .
- **Minkowski distance** is a generalization of the Euclidean and Manhattan distances.
- It is defined as

$$d(i, j) = \sqrt[h]{|x_{i1} - x_{j1}|^h + |x_{i2} - x_{j2}|^h + \dots + |x_{ip} - x_{jp}|^h},$$

- here  $h$  is a real number such that  $h \geq 1$ .
- It represents the Manhattan distance when  $h = 1$  (i.e.,  $L1$  norm) and Euclidean distance when  $h = 2$  (i.e.,  $L2$  norm).

## Dissimilarity of Numeric Data (6)

- The **supremum distance** (also referred to as ***Lmax***, ***L1 norm*** and as the **Chebyshev distance**) is a generalization of the Minkowski distance for  $h \rightarrow \infty$ .
- To compute it, we find the attribute  $f$  that gives the maximum difference in values between the two objects.
- This difference is the supremum distance, defined more formally as:

$$d(i, j) = \lim_{h \rightarrow \infty} \left( \sum_{f=1}^p |x_{if} - x_{jf}|^h \right)^{\frac{1}{h}} = \max_f |x_{if} - x_{jf}|.$$

- The  $L1$  norm is also known as the *uniform norm*.

## Dissimilarity of Numeric Data (7)

- **Example 2.20 Supremum distance.** Let's use the same two objects,  $\mathbf{x1} = (1, 2)$  and  $\mathbf{x2} = (3, 5)$ , as in Figure 2.23.
- The second attribute gives the greatest difference between values for the objects, which is  $5 - 2 = 3$ .
- This is the supremum distance between both objects.
- If each attribute is assigned a weight according to its perceived importance, the **weighted Euclidean distance** can be computed as

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

- Weighting can also be applied to other distance measures as well.

# Exercise

Given two objects represented by the tuples (22, 1, 42, 10) and (20, 0, 36, 8)

- (a) Compute the Euclidean distance between the two objects.
- (b) Compute the Manhattan distance between the two objects.
- (c) Compute the Minkowski distance between the two objects, using  $q = 3$ .

# Proximity Measures for Ordinal Attributes (1)

- The values of an ordinal attribute have a meaningful order or ranking about them, yet the magnitude between successive values is unknown.
- An example includes the sequence *small, medium, large* for a *size* attribute.
- Ordinal attributes may also be obtained from the discretization of numeric attributes by splitting the value range into a finite number of categories.
- These categories are organized into ranks.
- That is, the range of a numeric attribute can be mapped to an ordinal attribute  $f$  having  $M_f$  states.

# Proximity Measures for Ordinal Attributes (2)

- For example, the range of the interval-scaled attribute *temperature* (in Celsius) can be organized into the following states: -30 to -10, -10 to 10, 10 to 30, representing the categories *cold temperature*, *moderate temperature*, and *warm temperature*, respectively.
- Let  $M$  represent the number of possible states that an ordinal attribute can have.
- These ordered states define the ranking  $1, \dots, M_f$ .
- “How are ordinal attributes handled?”
- The treatment of ordinal attributes is quite similar to that of numeric attributes when computing dissimilarity between objects.

# Proximity Measures for Ordinal Attributes (3)

- Suppose that  $f$  is an attribute from a set of ordinal attributes describing  $n$  objects.
- The dissimilarity computation with respect to  $f$  involves the following steps:
  1. The value of  $f$  for the  $i$ th object is  $x_{if}$ , and  $f$  has  $M_f$  ordered states, representing the ranking  $1, \dots, M_f$ . Replace each  $x_{if}$  by its corresponding rank,  $r_{if} \in \{1, \dots, M_f\}$ .
  2. Since each ordinal attribute can have a different number of states, it is often necessary to map the range of each attribute onto  $[0.0, 1.0]$  so that each attribute has equal weight. We perform such data normalization by replacing the rank  $r_{if}$  of the  $i$ th object in the  $f_{th}$  attribute by

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



# Proximity Measures for Ordinal Attributes (4)

- Dissimilarity can then be computed using any of the distance measures for numeric attributes, using  $z_{if}$  to represent the  $f$  value for the  $i$ th object.
- **Example 2.21 Dissimilarity between ordinal attributes.**
- Suppose that we have the sample data shown below

<b>Object Identifier</b>	<b>test-2 (ordinal)</b>
1	excellent
2	fair
3	good
4	excellent

# Proximity Measures for Ordinal Attributes (5)

- There are three states for *test-2*: *fair*, *good*, and *excellent*, that is,  $M_f = 3$ .
- For 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:

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

# Proximity Measures for Ordinal Attributes (6)

- Therefore, objects 1 and 2 are the most dissimilar, as are objects 2 and 4 (i.e.,  $d(2,1) = 1.0$  and  $d(4,2) = 1.0$ ).
- Similarity values for ordinal attributes can be interpreted from dissimilarity as  $\text{Sim}(i, j) = 1 - d(i, j)$ .

# Dissimilarity for Attributes of Mixed Types (1)

- In many real databases, objects are described by a *mixture* of attribute types.
- In general, a database can contain all of these attribute types.
- *“So, how can we compute the dissimilarity between objects of mixed attribute types?”*
- One approach is to group each type of attribute together, performing separate data mining (e.g., clustering) analysis for each type.
- This is feasible if these analyses derive compatible results.
- However, in real applications, it is unlikely that a separate analysis per attribute type will generate compatible results.

# Dissimilarity for Attributes of Mixed Types (2)

- A more preferable approach is to process all attribute types together, performing a single analysis.
- One such technique combines the different attributes into a single dissimilarity matrix, bringing all of the meaningful attributes onto a common scale of the interval [0.0, 1.0].
- Suppose that the data set contains  $p$  attributes 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 attribute  $f$  for object  $i$  or object  $j$ ), or (2)  $x_{if} = x_{jf} = 0$  and attribute  $f$  is asymmetric binary; otherwise,  $\delta_{ij}^{(f)} = 1$ .

# Dissimilarity for Attributes of Mixed Types (3)

- The contribution of attribute  $f$  to the dissimilarity between  $i$  and  $j$  is computed dependent on its type:
  - If  $f$  is numeric:  $d_{ij}^{(f)} = \frac{|x_{if} - x_{jf}|}{\max_h x_{hf} - \min_h x_{hf}}$ , where  $h$  runs over all nonmissing objects for attribute  $f$ .
  - If  $f$  is nominal or binary:  $d_{ij}^{(f)} = 0$  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 numeric.
- These steps are identical to what we have already seen for each of the individual attribute types.
- The only difference is for numeric attributes, where we normalize so that the values map to the interval  $[0.0, 1.0]$ .
- Thus, the dissimilarity between objects can be computed even when the attributes describing the objects are of different types.

# Dissimilarity for Attributes of Mixed Types (4)

- **Example 2.22 Dissimilarity between attributes of mixed type.**
- Let's compute a dissimilarity matrix for the objects shown below. Now we will consider *all* of the attributes, which are of different types.

<b>Object Identifier</b>	<b>test-1 (nominal)</b>	<b>test-2 (ordinal)</b>	<b>test-3 (numeric)</b>
1	code A	excellent	45
2	code B	fair	22
3	code C	good	64
4	code A	excellent	28

- In Examples 2.17 and 2.21, we worked out the dissimilarity matrices for each of the individual attributes.
- The procedures we followed for *test-1* (which is nominal) and *test-2* (which is ordinal) are the same as outlined earlier for processing attributes of mixed types.

# Dissimilarity for Attributes of Mixed Types (5)

- First, we need to compute the dissimilarity matrix for the third attribute, *test-3* (which is numeric). That is, we must compute  $d_{ij}$ .
- Following the case for numeric attributes, we let  $\max_h x_h = 64$  and  $\min_h x_h = 22$ .
- The difference between the two is used to normalize the values of the dissimilarity matrix.
- The resulting dissimilarity matrix for *test-3* is

$$\begin{bmatrix} 0 & & & \\ 0.55 & 0 & & \\ 0.45 & 1.00 & 0 & \\ 0.40 & 0.14 & 0.86 & 0 \end{bmatrix}.$$

- We can now use the dissimilarity matrices for the three attributes in our computation
- We get, for example,  $d(3, 1) = \frac{1(1)+1(0.50)+1(0.45)}{3} = 0.65.$



# Dissimilarity for Attributes of Mixed Types (6)

- The resulting dissimilarity matrix obtained for the data described by the three attributes of mixed types is:

$$\begin{bmatrix} 0 & & & \\ 0.85 & 0 & & \\ 0.65 & 0.83 & 0 & \\ 0.13 & 0.71 & 0.79 & 0 \end{bmatrix}.$$

- From Table 2.2, we can intuitively guess that objects 1 and 4 are the most similar, based on their values for *test-1* and *test-2*.
- This is confirmed by the dissimilarity matrix, where  $d(4, 1)$  is the lowest value for any pair of different objects.
- Similarly, the matrix indicates that objects 1 and 2 are the least similar.

# Cosine Similarity (1)

- A document can be represented by thousands of attributes, each recording the frequency of a particular word (such as a keyword) or phrase in the document.
- Thus, each document is an object represented by what is called a *term-frequency vector*.
- For example, in Table below, we see that *Document1* contains five instances of the word *team*, while *hockey* occurs three times.
- The word *coach* is absent from the entire document, as indicated by a count value of 0.
- Such data can be highly asymmetric.

# Cosine Similarity (2)

Document Vector or Term-Frequency Vector

<i>Document</i>	<i>team</i>	<i>coach</i>	<i>hockey</i>	<i>baseball</i>	<i>soccer</i>	<i>penalty</i>	<i>score</i>	<i>win</i>	<i>loss</i>	<i>season</i>
<i>Document1</i>	5	0	3	0	2	0	0	2	0	0
<i>Document2</i>	3	0	2	0	1	1	0	1	0	1
<i>Document3</i>	0	7	0	2	1	0	0	3	0	0
<i>Document4</i>	0	1	0	0	1	2	2	0	3	0

- Term-frequency vectors are typically very long and **sparse** (i.e., they have many 0 values).
- Applications using such structures include information retrieval, text document clustering, biological taxonomy, and gene feature mapping.
- The traditional distance measures that we have studied in this chapter do not work well for such sparse numeric data.

# Cosine Similarity (3)

- For example, two term-frequency vectors may have many 0 values in common, meaning that the corresponding documents do not share many words, but this does not make them similar
- We need a measure that will focus on the words that the two documents *do* have in common, and the occurrence frequency of such words.
- In other words, we need a measure for numeric data that ignores zero-matches.
- **Cosine similarity** is a measure of similarity that can be used to compare documents or, say, give a ranking of documents with respect to a given vector of query words.

# Cosine Similarity (4)

- Let  $\mathbf{x}$  and  $\mathbf{y}$  be two vectors for comparison. Using the cosine measure as a similarity function, we have

$$\text{sim}(\mathbf{x}, \mathbf{y}) = \frac{\mathbf{x} \cdot \mathbf{y}}{||\mathbf{x}|| ||\mathbf{y}||},$$

- where  $||\mathbf{x}||$  is the Euclidean norm of vector  $\mathbf{x} = (x_1, x_2, \dots, x_p)$ , defined as

$$\sqrt{x_1^2 + x_2^2 + \dots + x_p^2}.$$

- Conceptually, it is the length of the vector.
- Similarly,  $||\mathbf{y}||$  is the Euclidean norm of vector  $\mathbf{y}$ .
- The measure computes the cosine of the angle between vectors  $\mathbf{x}$  and  $\mathbf{y}$ .

# Cosine Similarity (5)

- A cosine value of 0 means that the two vectors are at 90 degrees to each other (orthogonal) and have no match.
- The closer the cosine value to 1, the smaller the angle and the greater the match between vectors.
- Note that because the cosine similarity measure does not obey all of the properties defining metric measures, it is referred to as a *nonmetric measure*.
- **Example 2.23 Cosine similarity between two term-frequency vectors.** Suppose that  $\mathbf{x}$  and  $\mathbf{y}$  are the first two term-frequency vectors in Table below. That is,  $\mathbf{x} = (5, 0, 3, 0, 2, 0, 0, 2, 0, 0)$  and  $\mathbf{y} = (3, 0, 2, 0, 1, 1, 0, 1, 0, 1)$ . How similar are  $\mathbf{x}$  and  $\mathbf{y}$ ?

# Cosine Similarity (6)

Document Vector or Term-Frequency Vector

<i>Document</i>	<i>team</i>	<i>coach</i>	<i>hockey</i>	<i>baseball</i>	<i>soccer</i>	<i>penalty</i>	<i>score</i>	<i>win</i>	<i>loss</i>	<i>season</i>
<i>Document1</i>	5	0	3	0	2	0	0	2	0	0
<i>Document2</i>	3	0	2	0	1	1	0	1	0	1
<i>Document3</i>	0	7	0	2	1	0	0	3	0	0
<i>Document4</i>	0	1	0	0	1	2	2	0	3	0

- cosine similarity between the two vectors, is:

$$x^t \cdot y = 5 \times 3 + 0 \times 0 + 3 \times 2 + 0 \times 0 + 2 \times 1 + 0 \times 1 + 0 \times 0 + 2 \times 1 + 0 \times 0 + 0 \times 1 = 25$$

$$||x|| = \sqrt{5^2 + 0^2 + 3^2 + 0^2 + 2^2 + 0^2 + 0^2 + 2^2 + 0^2 + 0^2} = 6.48$$

$$||y|| = \sqrt{3^2 + 0^2 + 2^2 + 0^2 + 1^2 + 1^2 + 0^2 + 1^2 + 0^2 + 1^2} = 4.12$$

$$sim(x, y) = 0.94$$

# Cosine Similarity (7)

- Therefore, if we were using the cosine similarity measure to compare these documents, they would be considered quite similar.
- When attributes are binary-valued, the cosine similarity function can be interpreted in terms of shared features or attributes.
- Suppose an object  $\mathbf{x}$  possesses the  $i$ th attribute if  $x_i = 1$ . Then  $\mathbf{x}^t \mathbf{y}$  is the number of attributes possessed (i.e., shared) by both  $\mathbf{x}$  and  $\mathbf{y}$ , and  $|\mathbf{x}| |\mathbf{y}|$  is the geometric mean of the number of attributes possessed by  $\mathbf{x}$  and the number possessed by  $\mathbf{y}$ .
- Thus,  $\text{sim}(\mathbf{x}, \mathbf{y})$  is a measure of relative possession of common attributes.



## Cosine Similarity (8)

- A simple variation of cosine similarity for the preceding scenario is

$$\text{sim}(x, y) = \frac{x \cdot y}{x \cdot x + y \cdot y - x \cdot y},$$

- which is the ratio of the number of attributes shared by  $x$  and  $y$  to the number of attributes possessed by  $x$  or  $y$ .
- This function, known as the **Tanimoto coefficient** or **Tanimoto distance**, is frequently used in information retrieval and biology taxonomy.

# Cluster Analysis

1. What is Cluster Analysis?
2. Types of Data in Cluster Analysis
3. A Categorization of Major Clustering Methods
4. Partitioning Methods
5. Hierarchical Methods
6. Density-Based Methods
7. Grid-Based Methods
8. Model-Based Methods
9. Clustering High-Dimensional Data
10. Constraint-Based Clustering
11. Outlier Analysis
12. Data Mining Applications



# Categorization of Major Clustering Methods (1)

- **Partitioning methods:**

- Given a set of  $n$  objects, a partitioning method constructs  $k$  partitions of the data, where each partition represents a cluster and  $k \leq n$ .
- That is, it divides the data into  $k$  groups such that each group must contain at least one object.
- The basic partitioning methods typically adopt *exclusive cluster separation*. i.e., each object must belong to exactly one group.
- This requirement may be relaxed, for example, in fuzzy partitioning techniques.
- Most partitioning methods are distance-based

# Categorization of Major Clustering Methods (2)

- 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.
- Traditional partitioning methods can be extended for subspace clustering, rather than searching the full data space.
- This is useful when there are many attributes and the data are sparse.

# Categorization of Major Clustering Methods (3)

- Achieving global optimality in partitioning-based clustering is often computationally prohibitive, potentially requiring an exhaustive enumeration of all the possible partitions.
- Instead, most applications adopt popular heuristic methods, such as greedy approaches like the *k-means* and the *k-medoids* algorithms, which progressively improve the clustering quality and approach a local optimum.
- These heuristic clustering methods work well for finding spherical-shaped clusters in small- to medium-size databases.
- To find clusters with complex shapes and for very large data sets, partitioning-based methods need to be extended.

# Categorization of Major Clustering Methods (4)

- **Hierarchical methods:**

- A hierarchical method creates a hierarchical decomposition of the given set of data objects.
- A hierarchical method can be classified as being either *agglomerative* or *divisive*, based on how the hierarchical decomposition is formed.
- The *agglomerative approach*, also called the *bottom-up* approach, starts with each object forming a separate group.
- It successively merges the objects or groups close to one another, until all the groups are merged into one (the topmost level of the hierarchy), or a termination condition holds.
- The divisive approach, also called the top-down approach, starts with all the objects in the same cluster.

# Categorization of Major Clustering Methods (5)

- In each successive iteration, a cluster is split into smaller clusters, until eventually each object is in one cluster, or a termination condition holds.
- Hierarchical clustering methods can be distance-based or density- and continuity based.
- Various extensions of hierarchical methods consider clustering in subspaces as well.
- Hierarchical methods suffer from the fact that once a step (merge or split) is done, it can never be undone.
- Such techniques cannot correct erroneous decisions

# Categorization of Major Clustering Methods (6)

- **Density-based methods:**

- Most partitioning methods cluster objects based on the distance between objects.
- Such methods can find only spherical-shaped clusters and encounter difficulty in discovering clusters of arbitrary shapes.
- Other clustering methods have been developed based on the notion of *density*.
- Their general idea is to continue growing a given cluster as long as the density (number of objects or data points) in the “neighborhood” exceeds some threshold.



# Categorization of Major Clustering Methods (7)

- For example, 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 or outliers and discover clusters of arbitrary shape.
- Density-based methods can divide a set of objects into multiple exclusive clusters, or a hierarchy of clusters.
- Typically, density-based methods consider exclusive clusters only, and do not consider fuzzy clusters.
- Density-based methods can be extended from full space to subspace clustering.

# Categorization of Major Clustering Methods (8)

- **Grid-based methods:**

- Grid-based methods quantize the object space into a finite number of cells that form a grid structure.
- All the clustering operations are performed on the grid structure (i.e., on the quantized space).
- The main advantage of this approach is its fast processing time, 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.
- Using grids is often an efficient approach to many spatial data mining problems, including clustering.

# Categorization of Major Clustering Methods (9)

- Therefore, grid-based methods can be integrated with other clustering methods such as density-based methods and hierarchical methods.
- These methods are briefly summarized in Figure 10.1.
- Some clustering algorithms integrate the ideas of several clustering methods, so that it is sometimes difficult to classify a given algorithm as uniquely belonging to only one clustering method category.
- Furthermore, some applications may have clustering criteria that require the integration of several clustering techniques.

# Categorization of Major Clustering Methods (10)

Method	General Characteristics
Partitioning methods	<ul style="list-style-type: none"><li>– Find mutually exclusive clusters of spherical shape</li><li>– Distance-based</li><li>– May use mean or medoid (etc.) to represent cluster center</li><li>– Effective for small- to medium-size data sets</li></ul>
Hierarchical methods	<ul style="list-style-type: none"><li>– Clustering is a hierarchical decomposition (i.e., multiple levels)</li><li>– Cannot correct erroneous merges or splits</li><li>– May incorporate other techniques like microclustering or consider object “linkages”</li></ul>
Density-based methods	<ul style="list-style-type: none"><li>– Can find arbitrarily shaped clusters</li><li>– Clusters are dense regions of objects in space that are separated by low-density regions</li><li>– Cluster density: Each point must have a minimum number of points within its “neighborhood”</li><li>– May filter out outliers</li></ul>
Grid-based methods	<ul style="list-style-type: none"><li>– Use a multiresolution grid data structure</li><li>– Fast processing time (typically independent of the number of data objects, yet dependent on grid size)</li></ul>

# Cluster Analysis

1. What is Cluster Analysis?
2. Types of Data in Cluster Analysis
3. A Categorization of Major Clustering Methods
4. Partitioning Methods
5. Hierarchical Methods
6. Density-Based Methods
7. Grid-Based Methods
8. Model-Based Methods
9. Clustering High-Dimensional Data
10. Constraint-Based Clustering
11. Outlier Analysis
12. Data Mining Applications



# Partitioning Methods

- Partitioning organizes the objects of a set into several exclusive groups or clusters.
- Given a data set,  $D$ , of  $n$  objects, and  $k$ , the number of clusters to form, a **partitioning algorithm** organizes the objects into  $k$  partitions ( $k \leq n$ ), where each partition represents a cluster.

# k-means: A centroid Based Technique (1)

- Suppose a data set,  $D$ , contains  $n$  objects in Euclidean space. Partitioning methods distribute the objects in  $D$  into  $k$  clusters,  $C_1, \dots, C_k$ , that is,  $C_i$  is subset of  $D$  and  $C_i \cap C_j = \text{null}$ ; for  $(1 \leq i, j \leq k)$ .
- An objective function is used to assess the partitioning quality so that objects within a cluster are similar to one another but dissimilar to objects in other clusters. i.e it aims for high intracluster similarity and low intercluster similarity.
- A centroid-based partitioning technique uses the *centroid* of a cluster,  $C_i$ , to represent that cluster.
- Conceptually, the centroid of a cluster is its center point.

# k-means: A centroid Based Technique (2)

- The centroid can be defined in various ways such as by the mean or medoid of the objects assigned to the cluster.
- The difference between an object  $\mathbf{p} \in C_i$  and  $\mathbf{c}_i$ , the representative of the cluster, is measured by  $dist(\mathbf{p}, \mathbf{c}_i)$ , where  $dist(\mathbf{x}, \mathbf{y})$  is the Euclidean distance between two points  $\mathbf{x}$  and  $\mathbf{y}$ .
- The quality of cluster  $C_i$  can be measured by the **within cluster variation**, which is the sum of *squared error* between all objects in  $C_i$  and the centroid  $\mathbf{c}_i$ , defined as

$$E = \sum_{i=1}^k \sum_{\mathbf{p} \in C_i} dist(\mathbf{p}, \mathbf{c}_i)^2,$$

- where  $E$  is the sum of the squared error for all objects in the data set;  $\mathbf{p}$  is the point in space representing a given object; and  $\mathbf{c}_i$  is the centroid of cluster  $C_i$ .



## k-means: A centroid Based Technique (3)

- In other words, for each object in each cluster, the distance from the object to its cluster center is squared, and the distances are summed.
- This objective function tries to make the resulting  $k$  clusters as compact and as separate as possible.

# k-means Algorithm – working (1)

- The  $k$ -means algorithm defines the centroid of a cluster as the mean value of the points within the cluster.
- First, it randomly selects  $k$  of the objects in  $D$ , 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 Euclidean distance between the object and the cluster mean.
- The  $k$ -means algorithm then iteratively improves the within-cluster variation.
- For each cluster, it computes the new mean using the objects assigned to the cluster in the previous iteration.

## k-means Algorithm – working (2)

- All the objects are then reassigned using the updated means as the new cluster centers.
- The iterations continue until the assignment is stable, that is, the clusters formed in the current round are the same as those formed in the previous round.

# K-means Algorithm (3)

**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:**

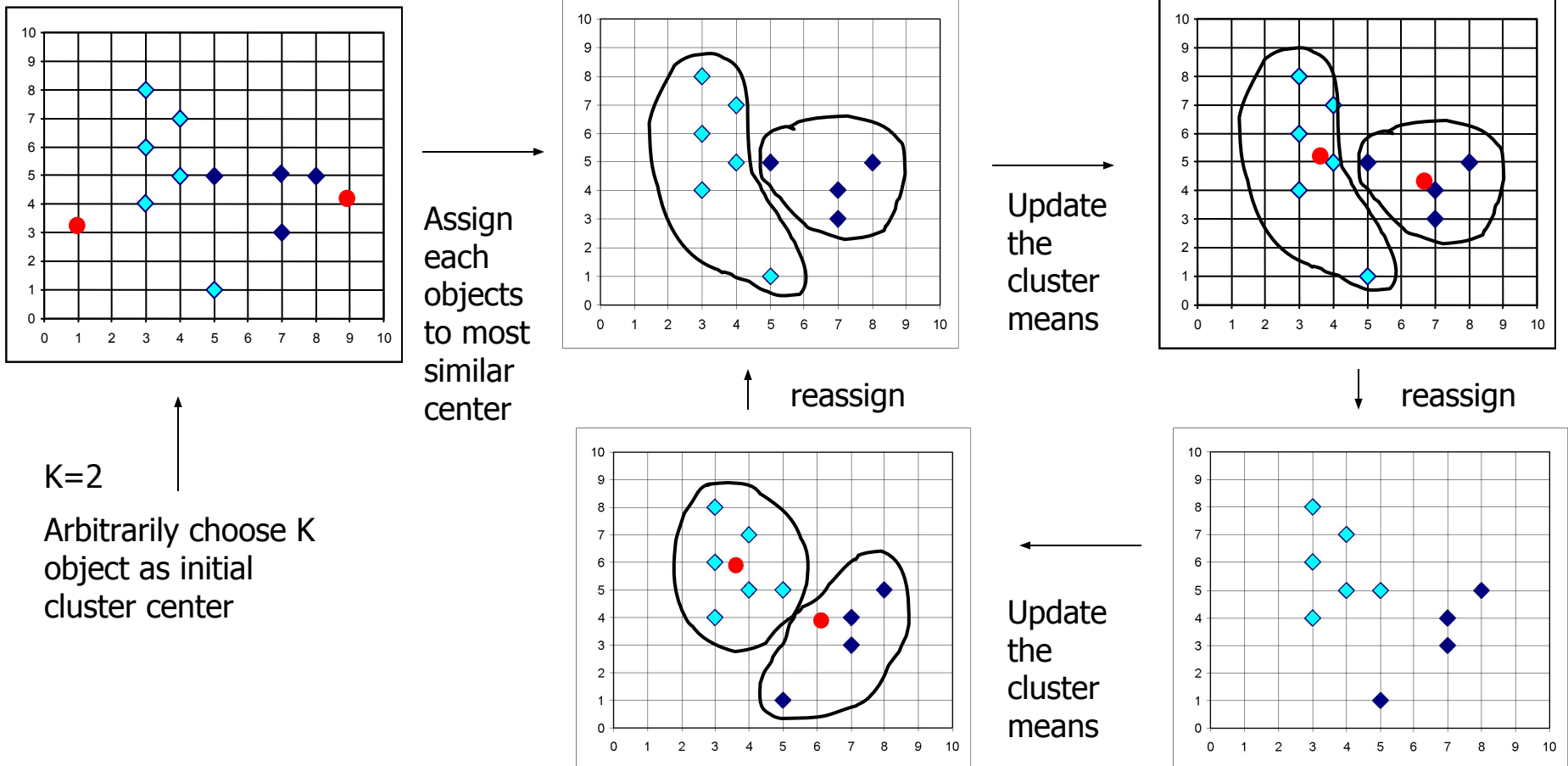
- (1) arbitrarily choose *k* objects from *D* as the initial cluster centers;
- (2) **repeat**
- (3)     (re)assign each object to the cluster to which the object is the most similar,  
          based on the mean value of the objects in the cluster;
- (4)     update the cluster means, that is, calculate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

---

**Figure 10.2** The *k*-means partitioning algorithm.

# The *K-Means* Clustering Method (4)

- Example



# Comments on the *K-Means* Method

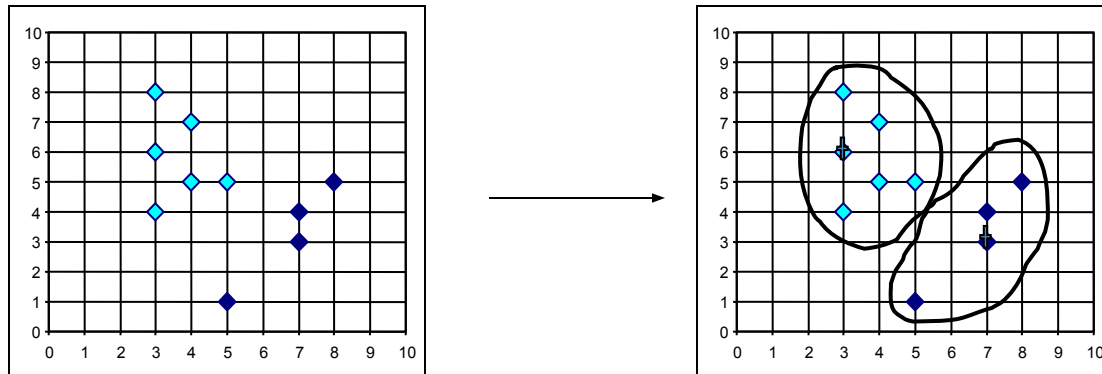
- Strength: *Relatively efficient*:  $O(tkn)$ , where  $n$  is # objects,  $k$  is # clusters, and  $t$  is # iterations. Normally,  $k, t \ll n$ .
  - Comparing: PAM:  $O(k(n-k)^2)$ , CLARA:  $O(ks^2 + k(n-k))$
- Weakness
  - Applicable only when *mean* is defined, then what about categorical data?
  - 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*

# Variations of the *K-Means* Method

- A few variants of the *k-means* which differ in
  - Selection of the initial *k* means
  - Dissimilarity calculations
  - Strategies to calculate cluster means
- Handling categorical data: *k-modes* (Huang'98)
  - Replacing means of clusters with modes
  - Using new dissimilarity measures to deal with categorical objects
  - Using a frequency-based method to update modes of clusters

# What Is the Problem of the K-Means Method?

- The k-means algorithm is sensitive to outliers !
  - Since an object with an extremely large value may substantially distort the distribution of the data.
- K-Medoids: Instead of taking the **mean** value of the object in a cluster as a reference point, **medoids** can be used, which is the **most centrally located** object in a cluster.





# *“How can we make the $k$ -means algorithm more scalable?”*

- One approach to making the  $k$ -means method more efficient on large data sets is to use a good-sized set of samples in clustering.
- Another is to employ a filtering approach that uses a spatial hierarchical data index to save costs when computing means.
- A third approach explores the microclustering idea, which first groups nearby objects into “microclusters” and then performs  $k$ -means clustering on the microclusters.

# ***k*-Medoids: A Representative Object-Based Technique (1)**

- **A drawback of *k-means* : Sensitive to outliers**
- Consider six points in 1-D space having the values 1, 2, 3, 8, 9, 10, and 25, respectively.
- Intuitively, by visual inspection we may imagine the points partitioned into the clusters {1, 2,3} and {8, 9,10}, where point 25 is excluded because it appears to be an outlier. How would *k-means partition the values?*
- *If we apply k-means using  $k = 2$  and*
$$E = \sum_{i=1}^k \sum_{p \in C_i} \text{dist}(p, c_i)^2,$$
- *the partitioning {1, 2,3}, {8, 9, 10,25} has*
- *the within-cluster variation*

# ***k*-Medoids: A Representative Object-Based Technique (2)**

- If we apply k-means using  $k = 2$  and  $E = \sum_{i=1}^k \sum_{p \in C_i} \text{dist}(p, c_i)^2$ ,
- the partitioning  $\{1, 2, 3\}, \{8, 9, 10, 25\}$  has the within-cluster variation

$$(1-2)^2 + (2-2)^2 + (3-2)^2 + (8-13)^2 + (9-13)^2 + (10-13)^2 + (25-13)^2 = 196,$$

- given that the mean of cluster  $\{1, 2, 3\}$  is 2 and the mean of  $\{8, 9, 10, 25\}$  is 13

# ***k*-Medoids: A Representative Object-Based Technique (3)**

- Compare this to the partitioning  $\{1, 2, 3, 8\}$ ,  $\{9, 10, 25\}$ , for which *k*-means computes the within cluster variation as

$$(1 - 3.5)^2 + (2 - 3.5)^2 + (3 - 3.5)^2 + (8 - 3.5)^2 + (9 - 14.67)^2 \\ + (10 - 14.67)^2 + (25 - 14.67)^2 = 189.67,$$

- given that 3.5 is the mean of cluster  $\{1, 2, 3, 8\}$  and 14.67 is the mean of cluster  $\{9, 10, 25\}$ .
- The *k*-means method assigns the value 8 to a cluster different from that containing 9 and 10 due to the outlier point 25.
- Moreover, the center of the second cluster, 14.67, is substantially far from all the members in the cluster.

# The *K-Medoids* Clustering Method (4)

- Instead of taking the mean value of the objects in a cluster as a reference point, we can pick 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.
- The partitioning method is then performed based on the principle of minimizing the sum of the dissimilarities between each object and its corresponding reference point.
- That is, an absolute-error criterion is used, defined as

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

- where  $E$  is the sum of the absolute error for all objects in the data set;  $p$  is the point in space representing a given object in cluster  $C_j$ ; and  $o_j$  is the representative object of  $C_j$ .

# Partitioning Around Medoids (PAM) (1)

- PAM is a popular  $k$ -medoids clustering method.
- Like the  $k$ -means algorithm, the initial representative objects (called seeds) are chosen arbitrarily.
- We consider whether replacing a representative object by a non representative object would improve the clustering quality.
- All the possible replacements are tried out.

## PAM (2)

- The iterative process of replacing representative objects by other objects continues until the quality of the resulting clustering cannot be improved by any replacement.
- This quality is measured by a cost function of the average dissimilarity between an object and the representative object of its cluster.

# PAM (3)

**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.



## PAM (4)

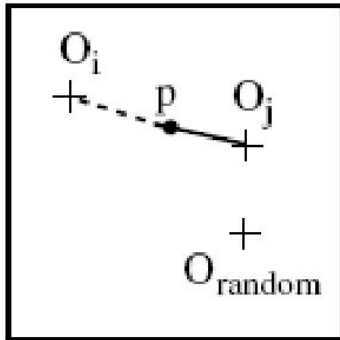
Case 1:  $p$  currently belongs to representative object,  $o_j$ . If  $o_j$  is replaced by  $o_{random}$  as a representative object and  $p$  is closest to one of the other representative objects,  $o_i$ ,  $i \neq j$ , then  $p$  is reassigned to  $o_i$ .

Case 2:  $p$  currently belongs to representative object,  $o_j$ . If  $o_j$  is replaced by  $o_{random}$  as a representative object and  $p$  is closest to  $o_{random}$ , then  $p$  is reassigned to  $o_{random}$ .

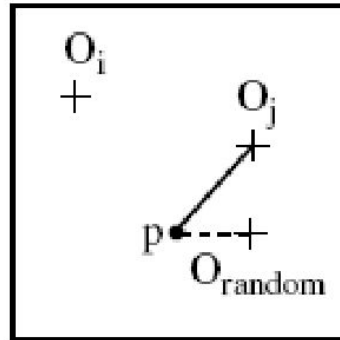
Case 3:  $p$  currently belongs to representative object,  $o_i$ ,  $i \neq j$ . If  $o_j$  is replaced by  $o_{random}$  as a representative object and  $p$  is still closest to  $o_i$ , then the assignment does not change.

Case 4:  $p$  currently belongs to representative object,  $o_i$ ,  $i \neq j$ . If  $o_j$  is replaced by  $o_{random}$  as a representative object and  $p$  is closest to  $o_{random}$ , then  $p$  is reassigned to  $o_{random}$ .

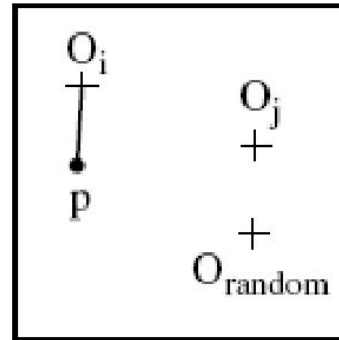
# PAM (5)



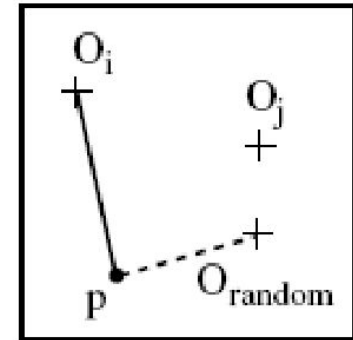
1. Reassigned to  $O_i$



2. Reassigned to  $O_{\text{random}}$



3. No change

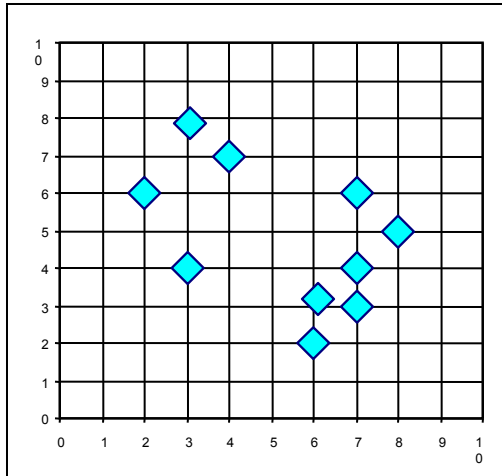


4. Reassigned to  $O_{\text{random}}$

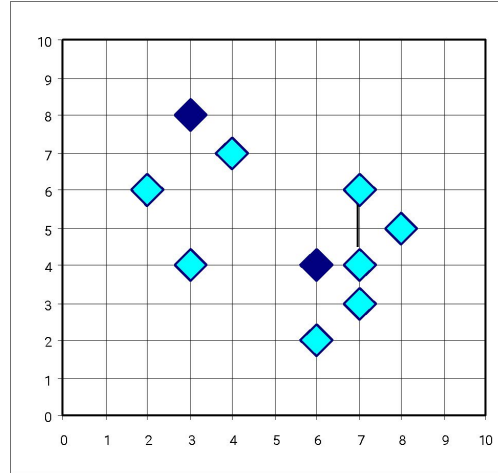
- data object
- + cluster center
- before swapping
- after swapping

# PAM (6)

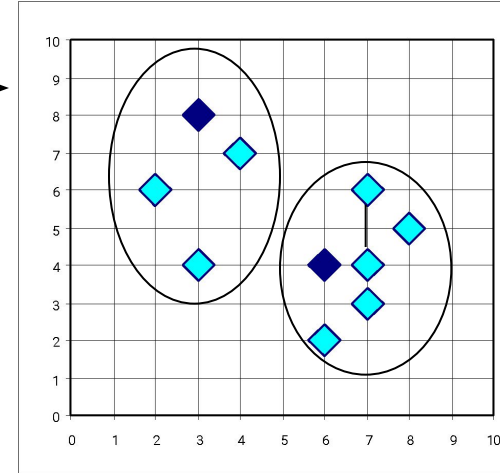
Total Cost = 20



Arbitrary  
choose  $k$   
object as  
initial  
medoids



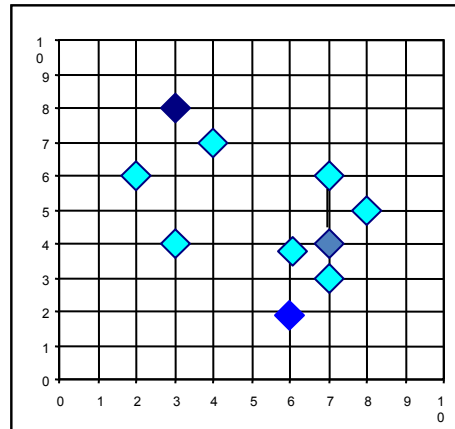
Assign  
each  
remainin  
g object  
to  
nearest  
medoids



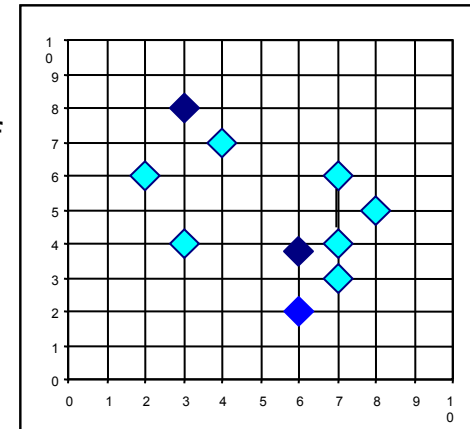
$K=2$

Randomly select a  
nonmedoid object,  $O_{\text{random}}$

Total Cost = 26



Compute  
total cost of  
swapping



Swapping  $O$   
and  $O_{\text{random}}$   
If quality is  
improved.

**Do loop  
Until no  
change**

# “Which method is more robust— $k$ -means or $k$ -medoids?”

- 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.
- However, the complexity of each iteration in the  $k$ -medoids algorithm is  $O(k(n-k)^2)$ . For large values of  $n$  and  $k$ , such computation becomes very costly, and much more costly than the  $k$ -means method.
- Both methods require the user to specify  $k$ , the number of clusters.

# CLARA (Clustering LARge Applications) (1)

- “How can we scale up the k-medoids method?”
- A typical *k*-medoids partitioning algorithm like PAM works effectively for small data sets, but does not scale well for large data sets.
- To deal with larger data sets, a *sampling*-based method called **CLARA (Clustering LARge Applications)** can be used.
- Instead of taking the whole data set into consideration, CLARA uses a random sample of the data set.
- The PAM algorithm is then applied to compute the best medoids from the sample.
- Ideally, the sample should closely represent the original data set.

## CLARA (Clustering LARge Applications) (2)

- In many cases, a large sample works well if it is created so that each object has equal probability of being selected into the sample.
- The representative objects (medoids) chosen will likely be similar to those that would have been chosen from the whole data set.
- CLARA builds clusterings from multiple random samples and returns the best clustering as the output.
- CLARA can deal with larger data sets than PAM.

## CLARA (Clustering LARge Applications) (3)

- The effectiveness of CLARA depends on the sample size.
- Notice that PAM searches for the best  $k$ -medoids among a given data set, whereas CLARA searches for the best  $k$ -medoids among the selected sample of the data set.
- CLARA cannot find a good clustering if any of the best sampled medoids is far from the best  $k$ -medoids.
- If an object is one of the best  $k$ -medoids but is not selected during sampling, CLARA will never find the best clustering.

# CLARANS (Clustering Large Applications based upon RANdomized Search) (1)


- *“How might we improve the quality and scalability of CLARA?”*
- **CLARANS** presents a trade-off between the cost and the effectiveness of using samples to obtain clustering.
- First, it randomly selects  $k$  objects in the data set as the current medoids.
- It then randomly selects a current medoid  $\mathbf{x}$  and an object  $\mathbf{y}$  that is not one of the current medoids.
- Can replacing  $\mathbf{x}$  by  $\mathbf{y}$  improve the absolute-error criterion?
- If yes, the replacement is made.



## CLARANS (2)

- CLARANS conducts such a randomized search  $l$  times.
- The set of the current medoids after the  $l$  steps is considered a local optimum.
- CLARANS repeats this randomized process  $m$  times and returns the best local optimal as the final result.

# Cluster Analysis

1. What is Cluster Analysis?
2. Types of Data in Cluster Analysis
3. A Categorization of Major Clustering Methods
4. Partitioning Methods
5. Hierarchical Methods 
6. Density-Based Methods
7. Grid-Based Methods
8. Model-Based Methods
9. Clustering High-Dimensional Data
10. Constraint-Based Clustering
11. Outlier Analysis
12. Data Mining Applications

# Hierarchical methods (1)

- A **hierarchical clustering method** works by grouping data objects into a hierarchy or “tree” of clusters.
- Representing data objects in the form of a hierarchy is useful for data summarization and visualization.
- For example, as the manager of human resources at *AllElectronics*, you may organize your employees into major groups such as executives, managers, and staff.
- You can further partition these groups into smaller subgroups.

# Hierarchical methods (2)

- For instance, the general group of staff can be further divided into subgroups of senior officers, officers, and trainees.
- All these groups form a hierarchy.
- We can easily summarize or characterize the data that are organized into a hierarchy, which can be used to find, say, the average salary of managers and of officers.

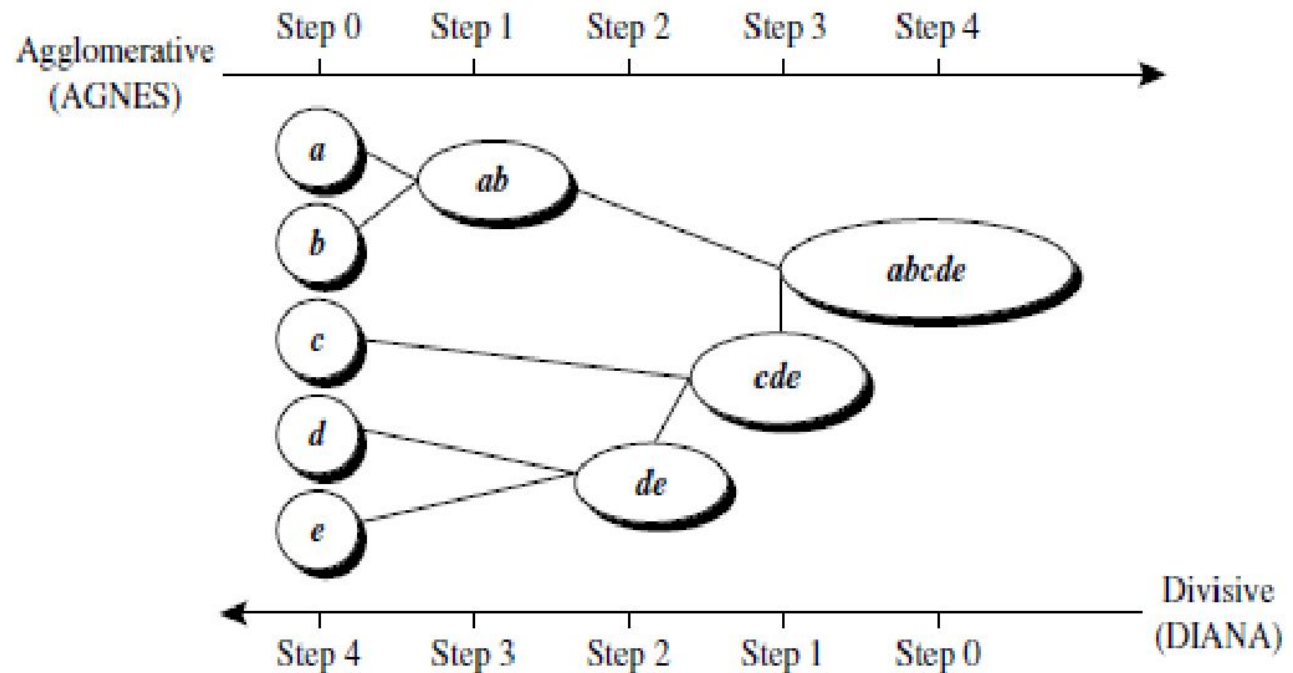
# Agglomerative hierarchical clustering method

- An **agglomerative hierarchical clustering method** uses a bottom-up strategy.
- It typically starts by letting each object form its own cluster and iteratively merges clusters into larger and larger clusters, until all the objects are in a single cluster or certain termination conditions are satisfied.
- The single cluster becomes the hierarchy's root.
- For the merging step, it finds the two clusters that are closest to each other (according to some similarity measure), and combines the two to form one cluster.
- Because two clusters are merged per iteration, where each cluster contains at least one object, an agglomerative method requires at most  $n$  iterations.

# Divisive hierarchical clustering method

- A **divisive hierarchical clustering method** employs a top-down strategy.
- It starts by placing all objects in one cluster, which is the hierarchy's root.
- It then divides the root cluster into several smaller subclusters, and recursively partitions those clusters into smaller ones.
- The partitioning process continues until each cluster at the lowest level is coherent enough—either containing only one object, or the objects within a cluster are sufficiently similar to each other.
- In either agglomerative or divisive hierarchical clustering, a user can specify the desired number of clusters as a termination condition.

# Agglomerative versus divisive hierarchical clustering (1)



**Figure 10.6** Agglomerative and divisive hierarchical clustering on data objects  $\{a, b, c, d, e\}$ .

# Agglomerative versus divisive hierarchical clustering (2)

- Figure 10.6 shows the application of **AGNES** (AGglomerative NESTing), an agglomerative hierarchical clustering method, and **DIANA** (Dlvisive ANALysis), a divisive hierarchical clustering method, on a data set of five objects, { **a**, **b**, **c**, **d**, **e** }.
- Initially, AGNES, the agglomerative method, places each object into a cluster of its own.
- The clusters are then merged step-by-step according to some criterion.
- For example, clusters  $C_1$  and  $C_2$  may be merged if an object in  $C_1$  and an object in  $C_2$  form the minimum Euclidean distance between any two objects from different clusters.



# Agglomerative versus divisive hierarchical clustering (3)

- This is a **single-linkage** approach in that each cluster is represented by all the objects in the cluster, and the similarity between two clusters is measured by the similarity of the *closest* pair of data points belonging to different clusters.
- The cluster-merging process repeats until all the objects are eventually merged to form one Cluster.

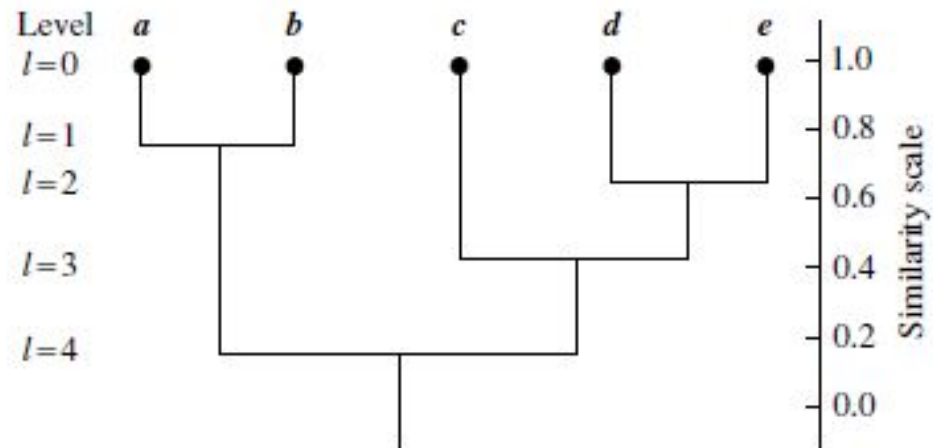
# Agglomerative versus divisive hierarchical clustering (4)

- DIANA, the divisive method, proceeds in the contrasting way.
- All the objects are used to form one initial cluster.
- The cluster is split according to some principle such as the maximum Euclidean distance between the closest neighboring objects in the cluster.
- The cluster-splitting process repeats until, eventually, each new cluster contains only a single object.

# Dendrogram (2)

- A tree structure called a **dendrogram** is commonly used to represent the process of hierarchical clustering.
- It shows how objects are grouped together (in an agglomerative method) or partitioned (in a divisive method) step-by-step.
- At  $l = 1$ , objects ***a*** and ***b*** are grouped together to form the first cluster, and they stay together at all subsequent levels.
- We can also use a vertical axis to show the similarity scale between clusters.
- For example, when the similarity of two groups of objects,  **$\{a, b\}$**  and  **$\{c, d, e\}$** , is roughly 0.16, they are merged together to form a single cluster

# Dendrogram (1)



**Figure 10.7** Dendrogram representation for hierarchical clustering of data objects  $\{a, b, c, d, e\}$ .

# Assessing Clustering Tendency

- Assess if non-random structure exists in the data by measuring the probability that the data is generated by a uniform data distribution
- Test spatial randomness by statistic test: Hopkins Static
  - Given a dataset D regarded as a sample of a random variable o, determine how far away o is from being uniformly distributed in the data space
  - Sample  $n$  points,  $p_1, \dots, p_n$ , uniformly from D. For each  $p_i$ , find its nearest neighbor in D:  $x_i = \min\{\text{dist}(p_i, v)\}$  where  $v$  in D
  - Sample  $n$  points,  $q_1, \dots, q_n$ , uniformly from D. For each  $q_i$ , find its nearest neighbor in D –  $\{q_i\}$ :  $y_i = \min\{\text{dist}(q_i, v)\}$  where  $v$  in D and  $v \neq q_i$
  - Calculate the Hopkins Statistic:
 
$$H = \frac{\sum_{i=1}^n y_i}{\sum_{i=1}^n x_i + \sum_{i=1}^n y_i}$$
  - If D is uniformly distributed,  $\sum x_i$  and  $\sum y_i$  will be close to each other and H is close to 0.5. If D is highly skewed, H is close to 0

# Determine the Number of Clusters

- Empirical method
  - # of clusters  $\approx \sqrt{n}/2$  for a dataset of  $n$  points
- Elbow method
  - Use the turning point in the curve of sum of within cluster variance w.r.t the # of clusters
- Cross validation method
  - Divide a given data set into  $m$  parts
  - Use  $m - 1$  parts to obtain a clustering model
  - Use the remaining part to test the quality of the clustering
    - E.g., For each point in the test set, find the closest centroid, and use the sum of squared distance between all points in the test set and the closest centroids to measure how well the model fits the test set
  - For any  $k > 0$ , repeat it  $m$  times, compare the overall quality measure w.r.t. different  $k$ 's, and find # of clusters that fits the data the best

# Measuring Clustering Quality

- Two methods: extrinsic vs. intrinsic
- Extrinsic: supervised, i.e., the ground truth is available
  - Compare a clustering against the ground truth using certain clustering quality measure
  - Ex. BCubed precision and recall metrics
- Intrinsic: unsupervised, i.e., the ground truth is unavailable
  - Evaluate the goodness of a clustering by considering how well the clusters are separated, and how compact the clusters are
  - Ex. Silhouette coefficient

# Measuring Clustering Quality: Extrinsic Methods

- Clustering quality measure:  $Q(C, C_g)$ , for a clustering  $C$  given the ground truth  $C_g$ .
- $Q$  is good if it satisfies the following **4** essential criteria
  - Cluster homogeneity: the purer, the better
  - Cluster completeness: should assign objects belong to the same category in the ground truth to the same cluster
  - Rag bag: putting a heterogeneous object into a pure cluster should be penalized more than putting it into a *rag bag* (i.e., “miscellaneous” or “other” category)
  - Small cluster preservation: splitting a small category into pieces is more harmful than splitting a large category into pieces