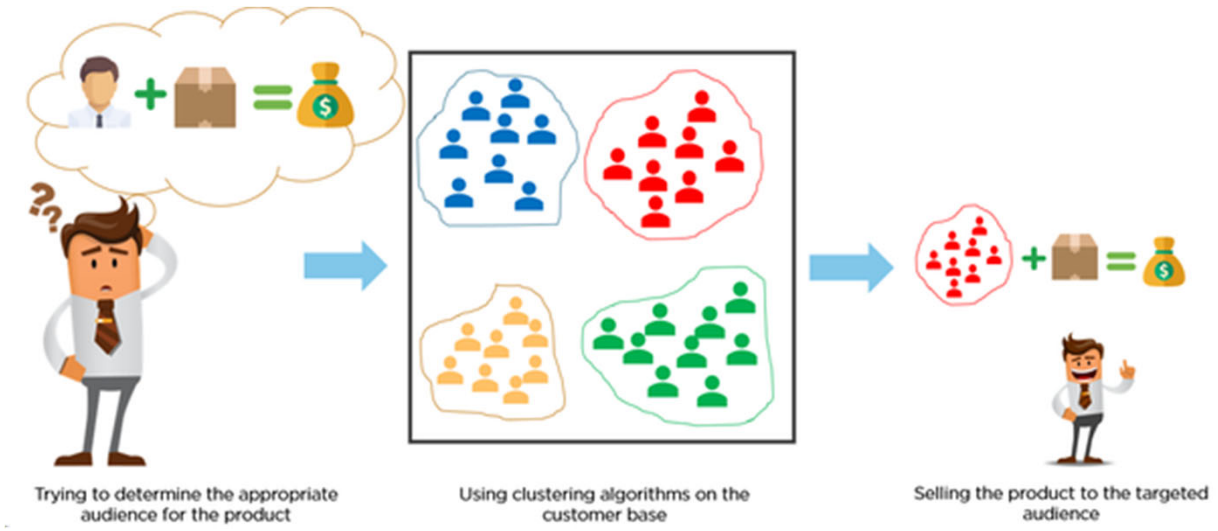



1

# Chapter 10. Cluster Analysis: Basic Concepts and Methods

- Cluster Analysis: An Introduction
- Partitioning Methods
- Hierarchical Methods
- Density- and Grid-Based Methods
- Evaluation of Clustering (Coverage will be based on the available time)
- Summary

2

2

# Cluster Analysis: An Introduction

- What Is Cluster Analysis?
- Applications of Cluster Analysis
- Cluster Analysis: Requirements and Challenges
- Cluster Analysis: A Multi-Dimensional Categorization
- An Overview of Typical Clustering Methodologies
- An Overview of Clustering Different Types of Data
- An Overview of User Insights and Clustering

3

3

# What Is Cluster Analysis?

- **What is a cluster?**
  - A cluster is a collection of data objects which are
    - Similar (or related) to one another within the same group (i.e., cluster)
    - Dissimilar (or unrelated) to the objects in other groups (i.e., clusters)
- **Cluster analysis** (or ***clustering***, ***data segmentation***, …)
  - Given a set of data points, partition them into a set of groups (i.e., clusters) which are <u>as similar as possible!</u>
- Cluster analysis is "**unsupervised" learning** (i.e., no predefined classes)
  - This contrasts with *classification* (i.e., *supervised learning*)
- Typical ways to use/apply cluster analysis
  - As a stand-alone tool to get insight into data distribution, or
  - As a preprocessing (or intermediate) step for other algorithms, e.g., classification algorithms.

4

4

# Cluster Analysis: Applications

- A key intermediate step for other data mining tasks
  - Generating a compact summary of data for classification, pattern discovery, hypothesis generation and testing, etc.
  - Outlier detection: Outliers—those "far away" from any cluster
- Data summarization, compression, and reduction
  - Ex. Image processing: Vector quantization
- Collaborative filtering, recommendation systems, or customer segmentation
  - Find like-minded users or similar products
- Dynamic trend detection
  - Clustering stream data and detecting trends and patterns
- Multimedia data analysis, biological data analysis and social network analysis
  - Ex. Clustering images or video/audio clips, gene/protein sequences, etc.

5

5

# What Is Good Clustering?

- A good clustering method will produce high quality clusters which should have
  - **High intra-class similarity:** Cohesive within clusters
  - **Low inter-class similarity:** Distinctive between clusters
- **Quality function**
  - There is usually a separate "quality" function that measures the "goodness" of a cluster
  - It is hard to define "similar enough" or "good enough"
    - The answer is typically highly subjective

- There exist many similarity measures and/or functions for different applications
- Similarity measure is critical for cluster analysis

6

6

# Considerations for Cluster Analysis

- **Partitioning criteria**
  - Single level vs. hierarchical partitioning (often, multi-level hierarchical partitioning is desirable, e.g., grouping topical terms)
- **Separation of clusters**
  - Exclusive (e.g., one customer belongs to only one region) vs. non-exclusive (e.g., one document may belong to more than one cluster)
- **Similarity measure**
  - Distance-based (e.g., Euclidean, road network, vector) vs. connectivity-based (e.g., density or contiguity)
- **Clustering space**
  - Full space (often when low dimensional) vs. subspaces (often in high-dimensional clustering)

7

7

# Chapter 10. Cluster Analysis: Basic Concepts and Methods

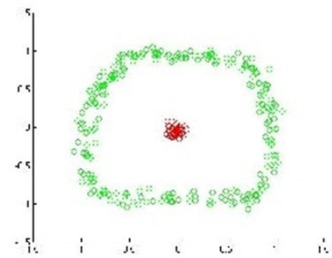
- Cluster Analysis: An Introduction
- Partitioning Methods
- Hierarchical Methods
- Density- and Grid-Based Methods
- Evaluation of Clustering
- Summary

10

10

# Partitioning-Based Clustering Methods

- Basic Concepts of Partitioning Algorithms
- The K-Means Clustering Method
- Initialization of K-Means Clustering
- The K-Medoids Clustering Method
- The K-Medians and K-Modes Clustering Methods
- The Kernel K-Means Clustering Method



11

11

# Partitioning Algorithms: Basic Concepts

- Partitioning method: Discovering the groupings in the data by optimizing a specific **objective function** and **iteratively** improving the quality of partitions
- ***K*-partitioning** method: Partitioning a dataset ***D*** of ***n*** objects into a set of ***K*** clusters so that an objective function is optimized (e.g., the sum of squared distances is minimized within each cluster, where $c_k$ is the centroid or medoid of cluster $C_k$)
  - A typical objective function: **Sum of Squared Errors** (**SSE**)

$$SSE(C) = \sum_{k=1}^{K} \sum_{x_{i \in C_k}} \| x_i - c_k \|^2$$

- Problem definition: Given *K*, find a partition of *K clusters* that optimizes the chosen partitioning criterion
  - Global optimal: Needs to exhaustively enumerate all partitions
  - Heuristic methods (i.e., greedy algorithms): *K-Means*, *K-Medians*, *K-Medoids*, etc.

12

12

# The *K-Means* Clustering Method

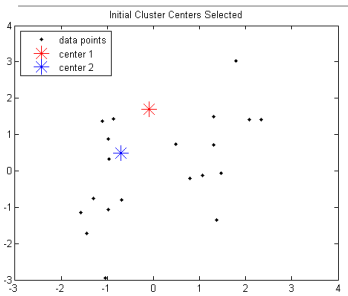
- *K-Means*
  - Each cluster is represented by the center/centroid of the cluster
- Given K, the number of clusters, the *K-Means* clustering algorithm is outlined as follows
  - Select ***K*** points as initial centroids
  - **Repeat**
    - Form *K* clusters by assigning each point to its closest centroid
    - Re-compute the centroid (i.e., *mean point*) of each cluster
  - **Until** convergence criterion is satisfied (e.g., no change of cluster membership, or a certain # of iterations have been reached, or, the SSE is < threshold)
- Different kinds of distance measures can be used
  - Manhattan distance ($L_1$ norm), Euclidean distance ($L_2$ norm), Cosine similarity

13

13

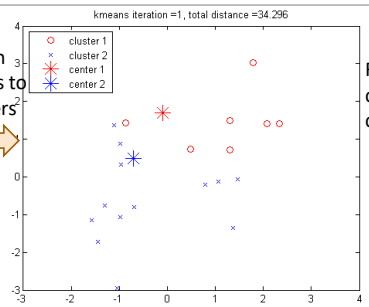
https://user.ceng.metu.edu.tr/~akifakkus/courses/ceng574/k-means/

# Example: *K-Means* Clustering

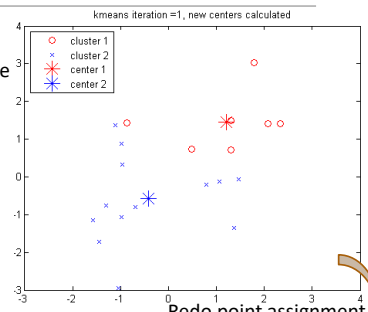



The original data points & randomly select ***K* = 2** centroids

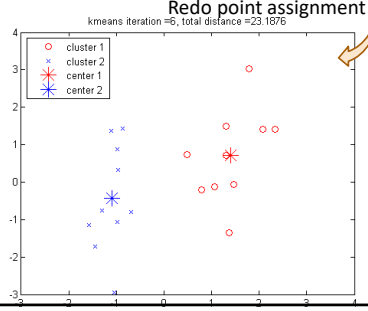
Assign points to clusters




Recompute cluster centers




Redo point assignment




*Execution of the K-Means* Clustering Algorithm

Select *K* points as initial centroids

**Repeat**

- Form *K* clusters by assigning each point to its closest centroid
- Re-compute the centroids (i.e., *mean point*) of each cluster

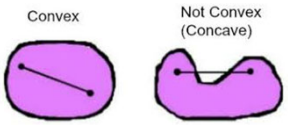
**Until** convergence criterion is satisfied

14

14

# Discussion on the *K-Means* Method

- **Efficiency**: $O(tKn)$ where $n$: # of objects, $K$: # of clusters, and $t$: # of iterations
  - Normally, $K$, $t << n$; thus, an efficient method
- K-means clustering often ***terminates at a local optimal***
  - Initialization can be important to find high-quality clusters
  - http://shabal.in/visuals/kmeans/1.html
- **Need to specify *K*,** the *number* of clusters, in advance
  - There are ways to automatically determine the "*best*" *K*
  - In practice, one often runs a range of values and selected the "*best*" *K* value
- **Sensitive to noisy data and *outliers***
  - Variations: Using K-medians, K-medoids, etc.
- K-means is applicable only to objects in a continuous n-dimensional space
  - Using the K-modes for ***categorical data***
- Not suitable to discover clusters with ***non-convex shapes***
  - Using density-based clustering, kernel *K*-means, etc.




15

15

# Variations of *K-Means*

- There are many variants of the *K-Means* method, varying in different aspects
  - Choosing better initial centroid estimates
    - *K-means++*, *Intelligent K-Means*, *Genetic K-Means*

To be discussed in this lecture

  - Choosing different representative prototypes for the clusters
    - *K-Medoids*, *K-Medians*, *K-Modes*

To be discussed in this lecture

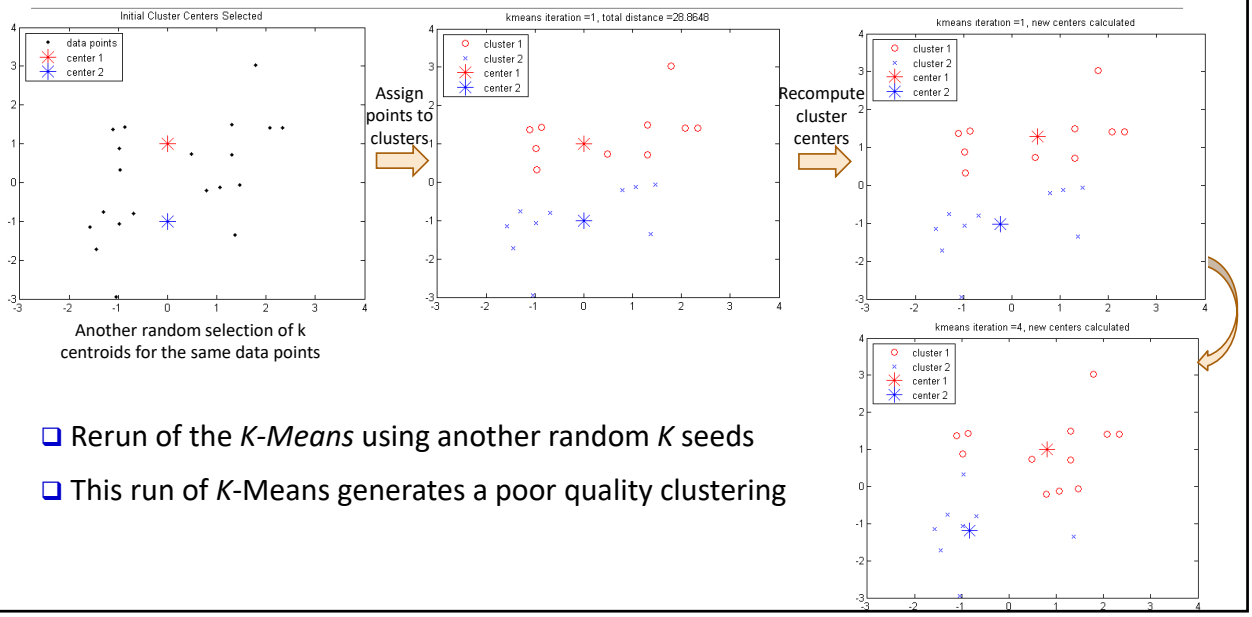
  - Applying feature transformation techniques
    - *Weighted K-Means*, *Kernel K-Means*

To be briefly discussed in this lecture

16

16

# Poor Initialization in K-Means May Lead to Poor Clustering

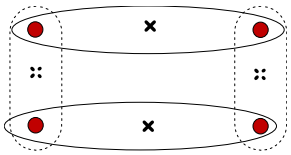



- Rerun of the *K-Means* using another random *K* seeds
- This run of *K*-Means generates a poor quality clustering

17

17

# Initialization of K-Means: Problem and Solution



- Different initializations may generate rather different clustering results (some could be far from optimal)
- Original proposal (MacQueen'67): Select *K* seeds randomly
  - Need to run the algorithm multiple times using different seeds
- There are many methods proposed for better initialization of *k* seeds
  - ***K-Means*++** (Arthur & Vassilvitskii'07):
    - The first centroid is selected at random
    - The next centroid selected is the one that is the farthest from the currently selected (selection is based on a weighted probability score)
    - The selection continues until *K* centroids are obtained

18

18

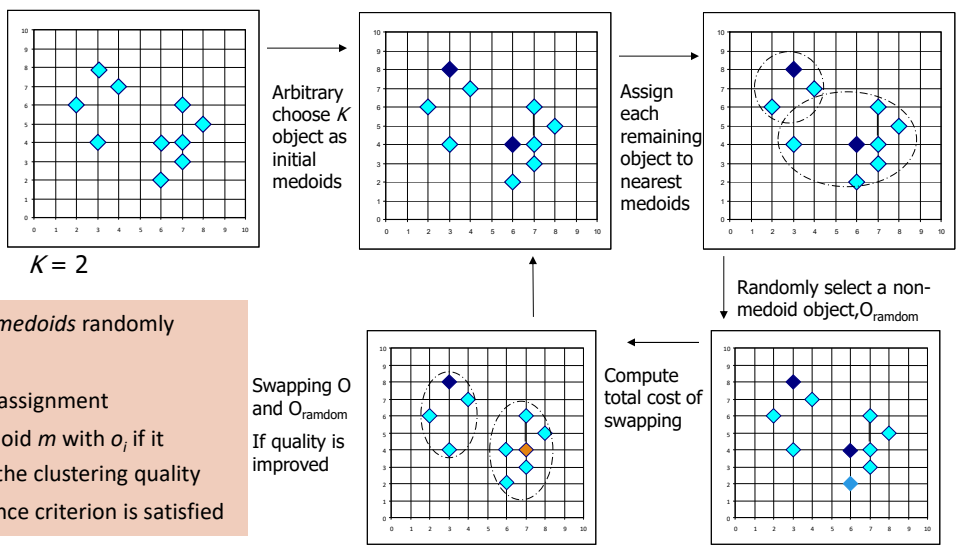
# Handling Outliers: From *K-Means* to *K-Medoids*

- 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 objects in a cluster as a reference point, **medoids** can be used, which is the **most centrally located** object in a cluster
- The *K-Medoids* clustering algorithm:
  - Select *K* points as the initial representative objects (i.e., as initial *K medoids*)
  - **Repeat**
    - Assigning each point to the cluster with the closest medoid
    - Randomly select a non-representative object $o_i$
    - Compute the total cost *S* of swapping the medoid *m* with $o_i$ *(e.g., use SSE to measure)*
    - If $S < 0$ (e.g., new SSE<previous SSE), then swap *m* with $o_i$ to form the new set of medoids
  - **Until** convergence criterion is satisfied

21

21

# PAM: A Typical *K-Medoids* Algorithm




Select initial *K medoids* randomly

**Repeat**

Object re-assignment

Swap medoid *m* with $o_i$ if it improves the clustering quality

**Until** convergence criterion is satisfied

22

22

# Discussion on *K-Medoids* Clustering

- *K-Medoids* Clustering: Find *representative* objects (medoids) in clusters
- *PAM* (Partitioning Around Medoids: Kaufmann & Rousseeuw 1987)
  - Starts from an initial set of medoids, and
  - Iteratively replaces one of the medoids by one of the non-medoids if it improves the total sum of the squared errors (SSE) of the resulting clustering
  - *PAM* works effectively for small data sets but does not scale well for large data sets (due to the computational complexity)
  - Computational complexity: PAM: $O(K(n-K)^2)$ (quite expensive!)
- Efficiency improvements on PAM
  - ***CLARA*** (Kaufmann & Rousseeuw, 1990):
    - PAM on samples; $O(Ks^2 + K(n-K))$, s is the sample size
  - ***CLARANS*** (Ng & Han, 1994): Randomized re-sampling, ensuring efficiency + quality

23

23

# *K-Medians*: Handling Outliers by Computing Medians

- Medians are less sensitive to outliers than means
  - Think of the median salary vs. mean salary of a large firm when adding a few top executives!
- ***K-Medians***: Instead of taking the **mean** value of the object in a cluster as a reference point, **medians** are used ($L_1$-norm as the distance measure)
- The criterion function for the *K-Medians* algorithm: $S = \sum_{k=1}^{K} \sum_{x_{i \in C_k}} | x_{ij} - med_{kj} |$
- The *K-Medians* clustering algorithm:
  - Select *K* points as the initial representative objects (i.e., as initial *K medians*)
  - **Repeat**
    - Assign every point to its nearest median
    - Re-compute the median using the median of each individual feature
  - **Until** convergence criterion is satisfied

24

24

# Last time

- K-means clustering
  - SSE
- K-median
- K-medoid
- K-means++

25

25

# K-Modes: Clustering Categorical Data

- *K-Means* cannot directly handle non-numerical (categorical) data – how to calculate the mean? What do they mean?
  - Mapping categorical value to 0/1 cannot generate quality clusters (in high-dimensional space)
- ***K-Modes***: An extension to *K-Means* by replacing means of clusters with ***modes***
  - Mode: The value that appears the most often in a **set** of data values
- Dissimilarity measure between object X and the center of a cluster $Z_l$
  - $\Phi(x_j, z_j) = 1 - n_j^r/n_l$ when $x_j = z_j = r$ ; 1 when $x_j \neq z_j$
    - where $z_j$ is the categorical value of attribute j in $Z_l$, $n_l$ is the number of objects in cluster *l*, and $n_j^r$ is the number of objects whose attribute value is r
- This dissimilarity measure (distance function) is **frequency-based**
- Algorithm is still based on iterative *object cluster assignment* and *centroid update*
- A mixture of categorical and numerical data: Using a ***K-Prototype*** method

26

26

# Update the mode

**Moving Frequency Based Method –**

Let $A = \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}$ $B = \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix}$ $C = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$ be three categorical objects having three attributes in binary format.

Then the new updated mode will be Mode $= \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix}$ as in the first row "0" has the highest frequency ,in the second row "1" has the highest frequency and in the third row also "1" holds the highest frequency. This is how the ***Moving Frequency Based Method*** works.

**Cost Function** –

Let there be "Ci" (i = 1 to K) number of clusters formed after using k-modes algorithm then the cost function is given by

$$J = \sum_{i=1}^{k} \sum_{X_j \in C_i} dissimilarity(X_j, Q_i)$$

Where, **Xj is categorical object of $i_{th}$ cluster**

**Qi is the Mode of $i_{th}$ cluster**

27

27

# K-Prototype

$$d(X_i, Q_l) = \sum_{j=1}^{m_r} (x_{ij}^r - q_{lj}^r)^2 + \gamma_l \sum_{j=1}^{m_c} \delta(x_{ij}^c, q_{lj}^c) \qquad (2.3)$$

where $\delta(p,q)=0$ for $p=q$ and $\delta(p,q)=1$ for $p \neq q$. $x_{ij}^r$ and $q_{lj}^r$ are values of numeric attributes, whereas $x_{ij}^c$ and $q_{lj}^c$ are values of categorical attributes for object $i$ and the prototype of cluster $l$. $m_r$ and $m_c$ are the numbers of numeric and categorical attributes. $\gamma_l$ is a weight for categorical attributes for cluster $l$.

28

28

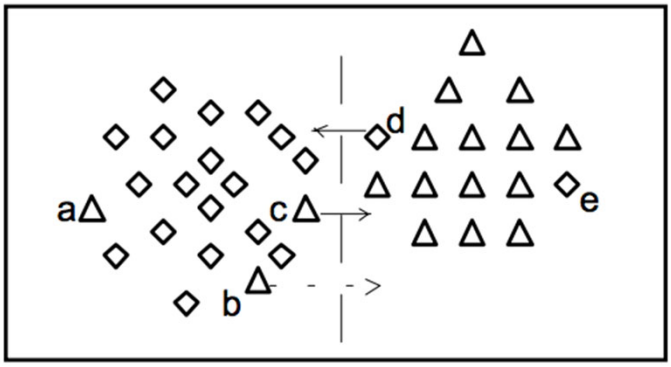



**Figure 1**. Influence of $\gamma_l$ in clustering.

29

29

30

30

# Variations of *K-Means*

- There are many variants of the *K-Means* method, varying in different aspects
  - Choosing better initial centroid estimates
    - *K-means++*, *Intelligent K-Means*, *Genetic K-Means*

To be discussed in this lecture

  - Choosing different representative prototypes for the clusters
    - *K-Medoids*, *K-Medians*, *K-Modes*

To be discussed in this lecture

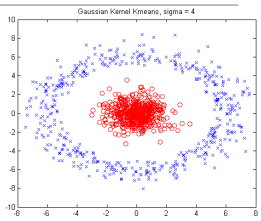
  - Applying feature transformation techniques
    - *Weighted K-Means*, *Kernel K-Means*

To be discussed in this lecture

31

31

# *Kernel K-Means* Clustering




- *Kernel K-Means* can be used to detect non-convex clusters
  - A region is **convex** if it contains all the line segments connecting any pair of its points. Otherwise, it is **concave**
  - *K-Means* can only detect clusters that are linearly separable
- Idea: Project data onto the high-dimensional kernel space, and then perform *K-Means* clustering
  - Map data points in the input space onto a high-dimensional feature space using the kernel function
  - Perform *K-Means* on the mapped feature space
- Computational complexity (time and space) is higher than K-Means
  - Need to compute and store *n* x *n* kernel matrix generated from the kernel function on the original data, where *n* is the number of points

32

32

# Kernel Functions and Kernel K-Means Clustering

- Typical kernel functions:
  - Polynomial kernel of degree h: $K(\boldsymbol{X}_i, \boldsymbol{X}_j) = (\boldsymbol{X}_i \cdot \boldsymbol{X}_j + 1)^h$
  - Gaussian radial basis function (RBF) kernel: $K(\boldsymbol{X}_i, \boldsymbol{X}_j) = e^{-\|X_i - X_j\|^2 / 2\sigma^2}$
  - Sigmoid kernel: $K(\boldsymbol{X}_i, \boldsymbol{X}_j) = \tanh(\kappa \boldsymbol{X}_i \cdot \boldsymbol{X}_j - \delta)$
- The formula for kernel matrix K for any two points $x_i, x_j \in C_k$ is $K_{x_i x_j} = \phi(x_i) \bullet \phi(x_j)$
- The SSE criterion of *kernel K-means*: $SSE(C) = \sum_{k=1}^{K} \sum_{x_{i \in C_k}} \|\phi(x_i) - c_k\|^2$
  - The formula for the cluster centroid:

$$c_k = \frac{\sum_{x_{i \in C_k}} \phi(x_i)}{|C_k|}$$

- Clustering can be performed without the actual individual projections $\phi(x_i)$ and $\phi(x_j)$ for the data points $x_i, x_j \in C_k$ (use K(Xi, Xj) instead)

33

33

# Example: Kernel Functions and Kernel K-Means Clustering

- Gaussian radial basis function (RBF) kernel: $K(\boldsymbol{X}_i, \boldsymbol{X}_j) = e^{-\|X_i - X_j\|^2 / 2\sigma^2}$
- Suppose there are 5 original 2-dimensional points:
  - $x_1(0, 0)$, $x_2(4, 4)$, $x_3(-4, 4)$, $x_4(-4, -4)$, $x_5(4, -4)$
- If we set $\sigma$ to 4, we will have the following points in the kernel space
  - E.g., $\|x_1 - x_2\|^2 = (0-4)^2 + (0-4)^2 = 32$, thus, $K(x_1, x_2) = e^{-\frac{32}{2 \cdot 4^2}} = e^{-1}$

Original Space

| | $x$ | $y$ |
|---|---|---|
| $x_1$ | 0 | 0 |
| $x_2$ | 4 | 4 |
| $x_3$ | $-4$ | 4 |
| $x_4$ | $-4$ | $-4$ |
| $x_5$ | 4 | $-4$ |

RBF Kernel Space ($\sigma = 4$)

| $K(x_i, x_1)$ | $K(x_i, x_2)$ | $K(x_i, x_3)$ | $K(x_i, x_4)$ | $K(x_i, x_5)$ |
|---|---|---|---|---|
| 1 | $e^{-\frac{4^2+4^2}{2 \cdot 4^2}} = e^{-1}$ | $e^{-1}$ | $e^{-1}$ | $e^{-1}$ |
| $e^{-1}$ | 1 | $e^{-2}$ | $e^{-4}$ | $e^{-2}$ |
| $e^{-1}$ | $e^{-2}$ | 1 | $e^{-2}$ | $e^{-4}$ |
| $e^{-1}$ | $e^{-4}$ | $e^{-2}$ | 1 | $e^{-2}$ |
| $e^{-1}$ | $e^{-2}$ | $e^{-4}$ | $e^{-2}$ | 1 |

34

34

$$\mathcal{D}(\{\pi_c\}_{c=1}^k) = \sum_{c=1}^{k} \sum_{\mathbf{a}_i \in \pi_c} \|\phi(\mathbf{a}_i) - \mathbf{m}_c\|^2, \text{ where } \mathbf{m}_c = \frac{\sum_{\mathbf{a}_i \in \pi_c} \phi(\mathbf{a}_i)}{|\pi_c|}.$$

$$\phi(\mathbf{a}_i) \cdot \phi(\mathbf{a}_i) - \frac{2\sum_{\mathbf{a}_j \in \pi_c} \phi(\mathbf{a}_i) \cdot \phi(\mathbf{a}_j)}{|\pi_c|} + \frac{\sum_{\mathbf{a}_j, \mathbf{a}_l \in \pi_c} \phi(\mathbf{a}_j) \cdot \phi(\mathbf{a}_l)}{|\pi_c|^2}.$$

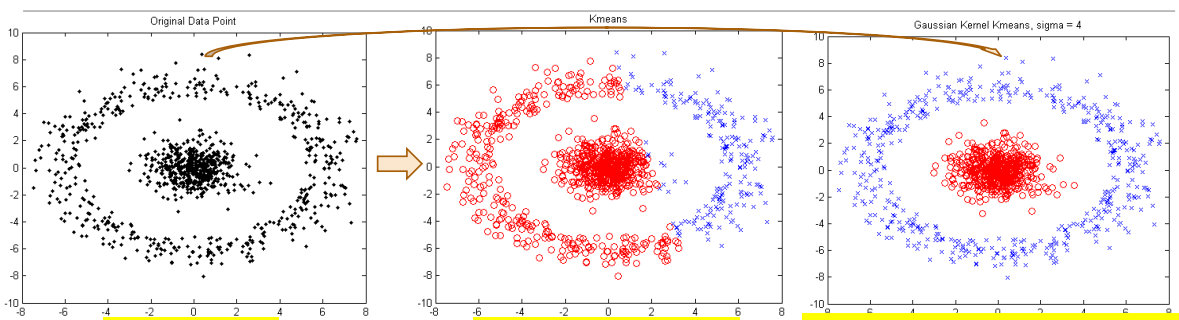
where,

$c^{th}$ cluster is denoted by $\pi_c$.
'$m_c$' denotes the mean of the cluster $\pi_c$.
'$\Phi(a_i)$' denotes the data point $a_i$ in transformed space.
$\Phi(a_i)$. $\Phi(aj) = K(ai, aj)$ for gaussian kernel.

35

35

# Example: Kernel K-Means Clustering




The original data set | The result of K-Means clustering | The result of Gaussian Kernel K-Means clustering

- The above data set cannot generate quality clusters by K-Means since it contains non-convex clusters
- Gaussian RBF Kernel transformation maps data to a kernel matrix K for any two points $x_i$, $x_j$: $K_{x_i x_j} = \phi(x_i) \bullet \phi(x_j)$ and Gaussian kernel: $K(\boldsymbol{X_i}, \boldsymbol{X_j}) = e^{-\|X_i - X_j\|^2 / 2\sigma^2}$
- K-Means clustering is conducted on the mapped data, generating quality clusters

36

36

## Chapter 10. Cluster Analysis: Basic Concepts and Methods

- Cluster Analysis: An Introduction
- Partitioning Methods
- Hierarchical Methods
- Density- and Grid-Based Methods
- Evaluation of Clustering
- Summary

37

37

## Hierarchical Clustering Methods

- Basic Concepts of Hierarchical Algorithms
- Agglomerative Clustering Algorithms
- Divisive Clustering Algorithms
- Extensions to Hierarchical Clustering
- BIRCH: A Micro-Clustering-Based Approach
- CURE: Exploring Well-Scattered Representative Points
- CHAMELEON: Graph Partitioning on the KNN Graph of the Data
- Probabilistic Hierarchical Clustering

38

38

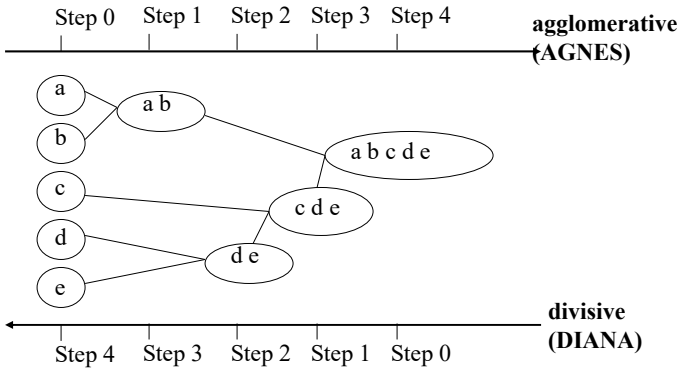

"How much for NO stems ?"

39

39

# Hierarchical Clustering: Basic Concepts

- Hierarchical clustering
  - Generate a clustering hierarchy (drawn as a **dendrogram**)
  - Not required to specify ***K,*** the number of clusters
  - More deterministic
  - No iterative refinement
- Two categories of algorithms:



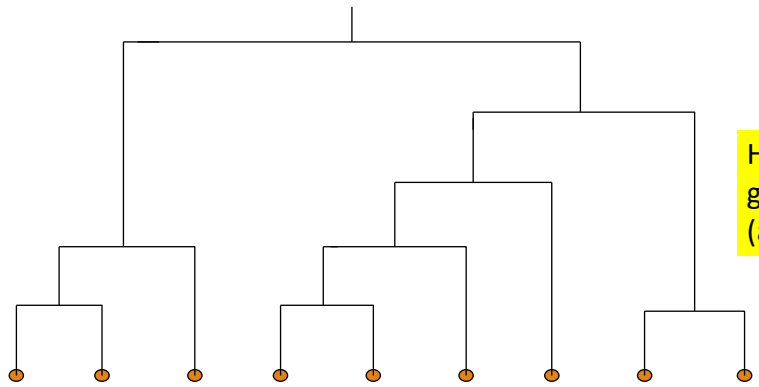

  - **Agglomerative**: Start with singleton clusters, continuously merge two clusters at a time to build a **bottom-up** hierarchy of clusters
  - **Divisive:** Start with a huge macro-cluster, split it continuously into two groups, generating a **top-down** hierarchy of clusters

40

40

# Dendrogram: Shows How Clusters are Merged

- Dendrogram: Decompose a set of data objects into a tree of clusters by multi-level nested partitioning
- A clustering of the data objects is obtained by cutting the dendrogram at the desired level, then each connected component forms a cluster



Hierarchical clustering generates a dendrogram (a hierarchy of clusters)

41

41

# Strengths of Hierarchical Clustering

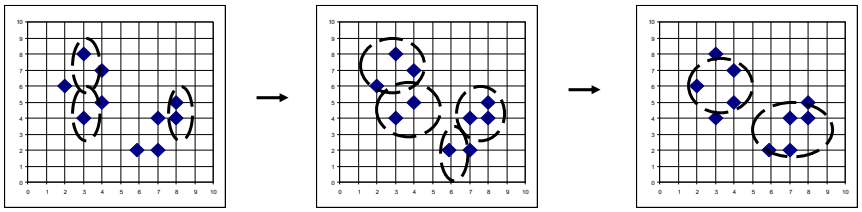
- Do not have to assume any particular number of clusters
  - Any desired number of clusters can be obtained by 'cutting' the dendogram at the proper level
- They may correspond to meaningful taxonomies
  - Example in biological sciences (e.g., animal kingdom, phylogeny reconstruction, …)

42

42

# Agglomerative Clustering Algorithm

- AGNES (AGglomerative NESting) (Kaufmann and Rousseeuw, 1990)
  - Use the **single-link** method and the dissimilarity matrix
  - Continuously merge nodes that have the least dissimilarity
  - Eventually all nodes belong to the same cluster



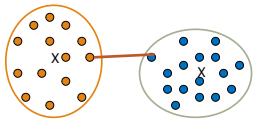
- Agglomerative clustering varies on different similarity measures among clusters
  - Single link (nearest neighbor)
  - Complete link (diameter)
  - Average link (group average)
  - Centroid link (centroid similarity)

43

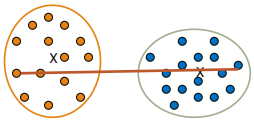
43

# Single Link vs. Complete Link in Hierarchical Clustering

- Single link (nearest neighbor)
  - The similarity between two clusters is the similarity between their most similar (nearest neighbor) members
  - Local similarity-based: Emphasizing more on close regions, ignoring the overall structure of the cluster
  - Capable of clustering <u>non-elliptical</u> shaped group of objects
  - Sensitive to noise and outliers



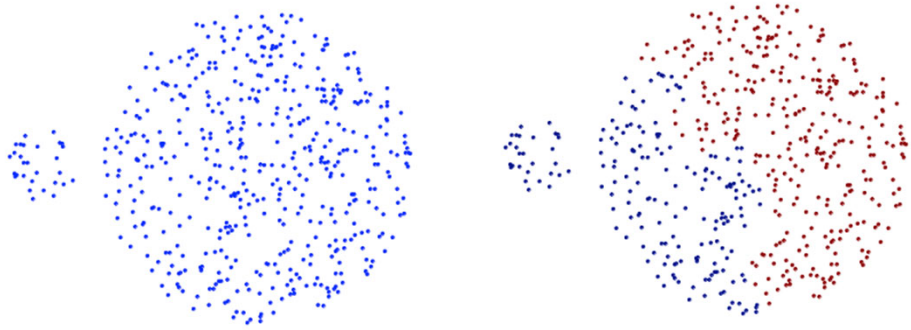
- Complete link (diameter)
  - The similarity between two clusters is the similarity between their most dissimilar members
  - Merge two clusters to form one with the smallest diameter
  - Nonlocal in behavior, obtaining **compact** shaped clusters
  - Sensitive to outliers



44

44

# Problems with Complete Link



Original Points  Two Clusters

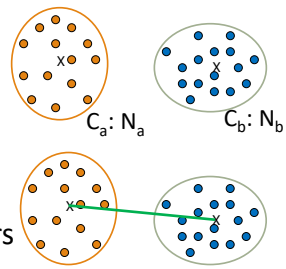
- Tends to break large clusters
- Biased towards globular clusters

45

45

# Agglomerative Clustering: Average vs. Centroid Links

- Agglomerative clustering with **average link**
  - **Average link**: The average distance between an element in one cluster and an element in the other (i.e., all pairs in two clusters)
    - Expensive to compute
- Agglomerative clustering with **centroid link**
  - **Centroid link**: The distance between the centroids of two clusters
- **Group Averaged Agglomerative Clustering (GAAC)**
  - Let two clusters $C_a$ and $C_b$ be merged into $C_{aUb}$. The new centroid is:
    - $N_a$ is the cardinality of cluster $C_a$, and $c_a$ is the centroid of $C_a$
  - The similarity measure for GAAC is the average of their distances
- Agglomerative clustering with **Ward's criterion**
  - **Ward's criterion:** minimize the increase in the value of the SSE criterion for the clustering obtained by merging them into $C_a$ U $C_b$: $W(C_{a\cup b}, c_{a\cup b}) - W(C, c) = \frac{N_a N_b}{N_a + N_b} d(c_a, c_b)$



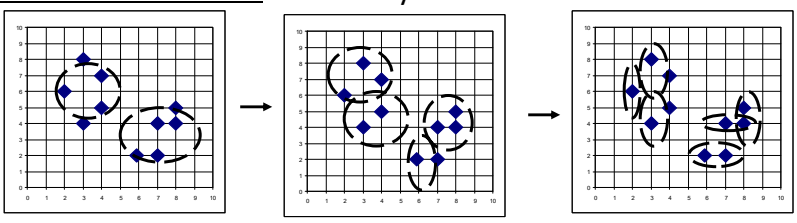

$$c_{a\cup b} = \frac{N_a c_a + N_b c_b}{N_a + N_b}$$

46

46

# Divisive Clustering

- DIANA (Divisive Analysis) (Kaufmann and Rousseeuw,1990)
  - Implemented in some statistical analysis packages, e.g., Splus
- Inverse order of AGNES: Eventually each node forms a cluster on its own



- Divisive clustering is a top-down approach
  - The process starts at the root with all the points as one cluster
  - It recursively splits the higher level clusters to build the dendrogram
  - Can be considered as a global approach

47

47

# More on Algorithm Design for Divisive Clustering

- Choosing which cluster to split
  - Check the sums of squared errors of the clusters and choose the one with the largest value
- Splitting criterion: Determining how to split
  - One may use Ward's criterion to chase for greater reduction in the difference in the SSE criterion as a result of a split
- Handling the noise
  - Use a threshold to determine the termination criterion (do not generate clusters that are too small because they contain mainly noises, especially on the upper level splits)

48

48

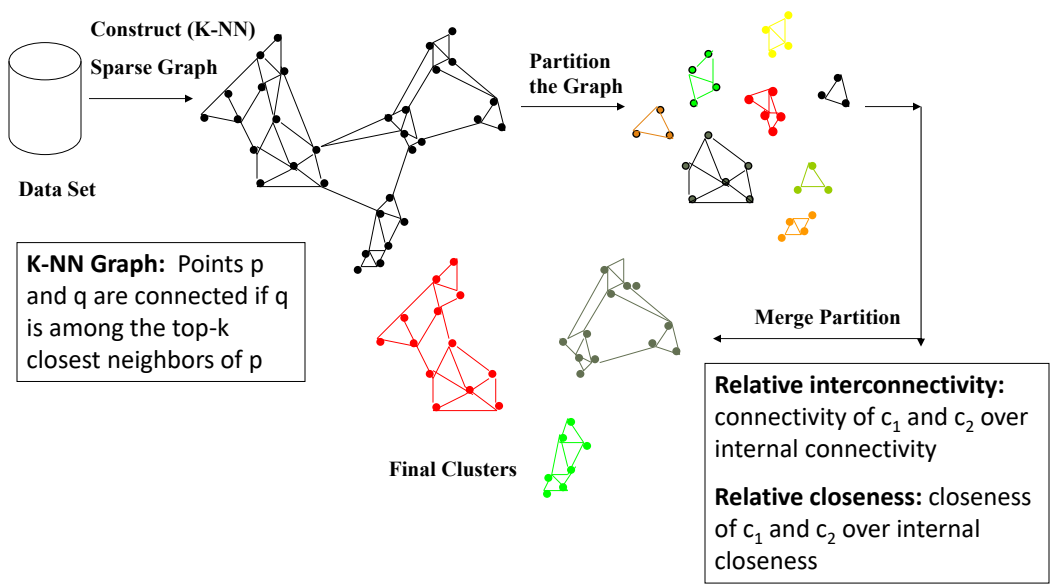
# Extensions to Hierarchical Clustering

- Major weaknesses of hierarchical clustering methods
  - Can never undo what was done previously
  - Do not scale well
    - Time complexity of at least $O(n^2)$, where $n$ is the number of total objects
- Other hierarchical clustering algorithms
  - BIRCH (1996): Use CF-tree and incrementally adjust the quality of sub-clusters
  - CURE (1998): Represent a cluster using a set of well-scattered representative points
  - CHAMELEON (1999): Use graph partitioning methods on the K-nearest neighbor graph of the data

49

49

# Overall Framework of CHAMELEON

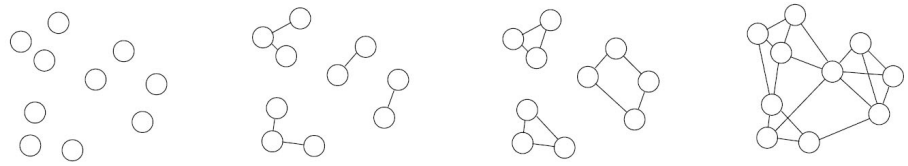



57

57

# KNN Graphs and Interconnectivity

- K-nearest neighbor (KNN) graphs from an original data in 2D:



(a) Original Data in 2D (b) 1-nearest neighbor graph (c) 2-nearest neighbor graph (d) 3-nearest neighbor graph

- $EC_{\{Ci,Cj\}}$: The absolute interconnectivity between $C_i$ *and* $C_j$:
  - *The sum* of the weight of the edges that connect vertices in $C_i$ *to vertices in* $C_j$
- Internal interconnectivity of a cluster $C_i$ : *The size of its min-cut bisector* $EC_{Ci}$ *(i.e., the weighted sum* of edges that partition the graph into two roughly equal parts)
- Relative Interconnectivity (RI): $$RI(C_i, C_j) = \frac{|EC_{\{C_i,C_j\}}|}{\frac{|EC_{C_i}|+|EC_{C_j}|}{2}}$$

58

58

# Relative Closeness & Merge of Sub-Clusters

- **Relative closeness** between a pair of clusters $C_i$ *and* $C_j$ *: The absolute closeness* between $C_i$ *and* $C_j$ *normalized w.r.t. the internal closeness of the two clusters* $C_i$ *and* $C_j$

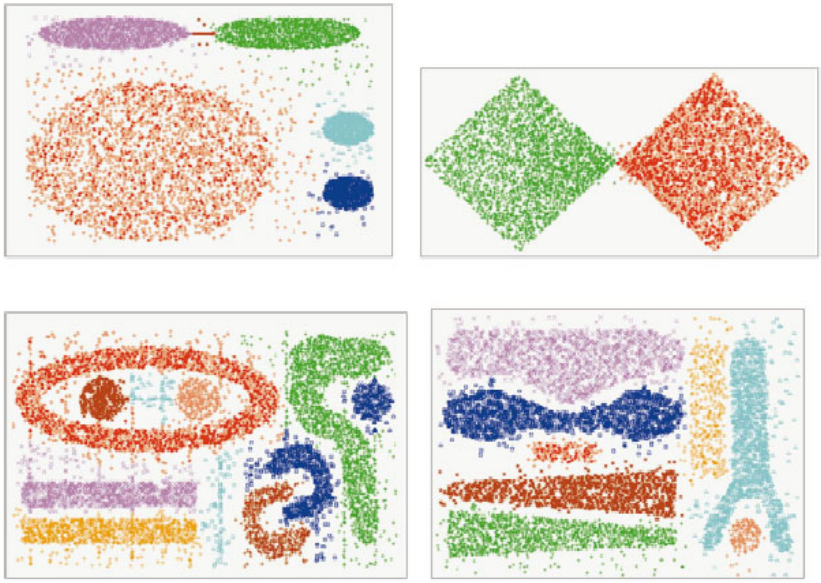
$$RC(C_i, C_j) = \frac{\overline{S}_{EC_{\{C_i,C_j\}}}}{\frac{|C_i|}{|C_i|+|C_j|}\overline{S}_{EC_{C_i}} + \frac{|C_j|}{|C_i|+|C_j|}\overline{S}_{EC_{C_j}}}$$

  - where $\overline{S}_{EC_{C_i}}$ and $\overline{S}_{EC_{C_j}}$ are the average weights of the edges that belong to the min-cut bisector of clusters $C_i$ *and* $C_j$ , respectively, and $\overline{S}_{EC_{\{C_i,C_j\}}}$ is the average weight of the edges that connect vertices in $C_i$ *to vertices in* $C_j$
- **Merge Sub-Clusters:**
  - Merges only those pairs of clusters whose RI and RC are both above some user-specified thresholds
  - Merge those maximizing the function that combines RI and RC

59

59

# CHAMELEON: Clustering Complex Objects



CHAMELEON is capable to generate quality clusters at clustering complex objects

60

60

# Chapter 10. Cluster Analysis: Basic Concepts and Methods

- Cluster Analysis: An Introduction
- Partitioning Methods
- Hierarchical Methods
- Density- and Grid-Based Methods
- Evaluation of Clustering
- Summary

65

65

# Density-Based Clustering Methods

- Clustering based on density (a local cluster criterion), such as density-connected points
- Major features:
  - Discover clusters of **arbitrary** shape
  - Handle noise
  - One scan (only examine the local region to justify density)
  - Need density parameters as termination condition
- Several interesting studies:
  - DBSCAN: Ester, et al. (KDD'96) To be covered in this lecture
  - OPTICS: Ankerst, et al (SIGMOD'99) To be covered in this lecture
  - DENCLUE: Hinneburg & D. Keim (KDD'98)
  - CLIQUE: Agrawal, et al. (SIGMOD'98) (also, grid-based)

67

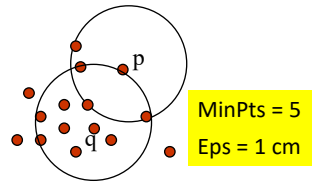
67

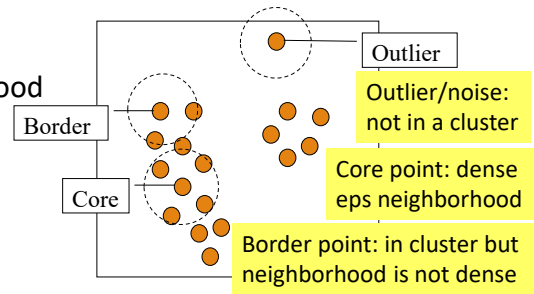
68

68

# DBSCAN: A Density-Based Spatial Clustering Algorithm

- DBSCAN (M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, KDD'96)
  - Discovers clusters of arbitrary shape: Density-Based Spatial Clustering of Applications with Noise
- A *density-based* notion of cluster
  - A *cluster* is defined as a maximal set of density-connected points
- Two parameters:
  - *Eps* ($\varepsilon$): Maximum radius of the neighborhood
  - *MinPts*: Minimum number of points in the Eps-neighborhood of a point
- The Eps($\varepsilon$)-neighborhood of a point $q$:
  - $N_{Eps}(q)$: {p belongs to D | dist(p, q) ≤ Eps}



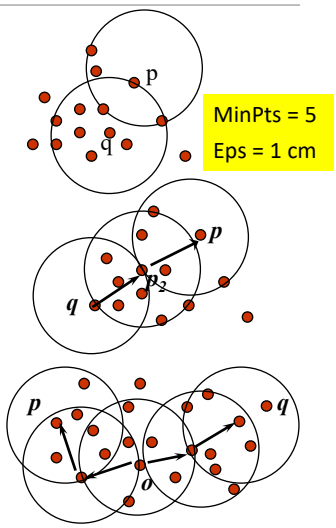




69

69

# DBSCAN: Density-Reachable and Density-Connected

- **Directly density-reachable**:
  - A point $p$ is directly density-reachable from a point $q$ w.r.t. *Eps* ($\varepsilon$), *MinPts* if
    - $p$ belongs to $N_{Eps}(q)$
    - **core point** condition: $|N_{Eps}(q)| \geq MinPts$
- **Density-reachable**:
  - A point $p$ is density-reachable from a point $q$ w.r.t. *Eps*, *MinPts* if there is a chain of points $p_1, \ldots, p_n$, $p_1 = q$, $p_n = p$ such that $p_{i+1}$ is directly density-reachable from $p_i$
- **Density-connected:**
  - A point $p$ is density-connected to a point $q$ w.r.t. *Eps*, *MinPts* if there is a point $o$ such that both $p$ and $q$ are density-reachable from $o$ w.r.t. *Eps* and *MinPts*

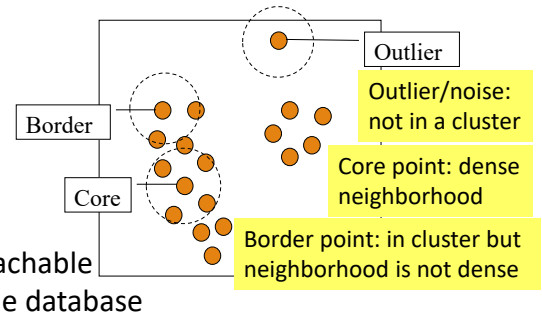



70

70

# DBSCAN: The Algorithm

- **Algorithm**
  - Arbitrarily select a point *p*
  - Retrieve all points density-reachable from *p* w.r.t. *Eps* and *MinPts*
    - If *p* is a core point, a cluster is formed
    - If *p* is a border point, no points are density-reachable from *p,* and DBSCAN visits the next point of the database
  - Continue the process until all of the points have been processed (pp. 474)
- **Computational complexity**
  - If a spatial index is used, the computational complexity of DBSCAN is O(nlogn), where n is the number of database objects
  - Otherwise, the worst case complexity is $O(n^2)$




71

71

# A demo:

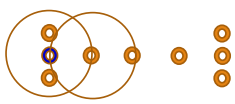
https://www.youtube.com/watch?v=h53WMIImUuc

https://www.naftaliharris.com/blog/visualizing-dbscan-clustering/

72

72

Pseudo code from the textbook pp. 474: MinPts=4, $\varepsilon = 1$



1 7
2 4 5 6 8
3 9

73

73

# DBSCAN Is Sensitive to the Setting of Parameters

*Figure 8. DBScan results for DS1 with MinPts at 4 and Eps at (a) 0.5 and (b) 0.4.*

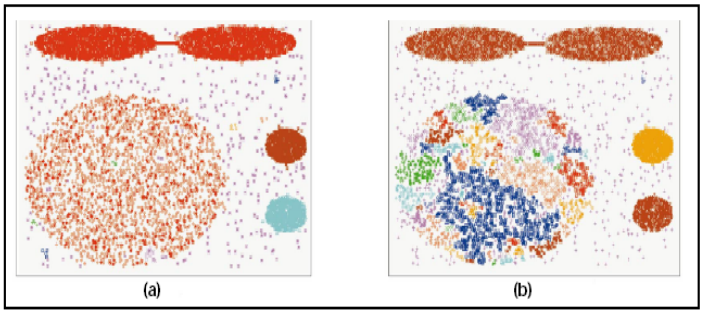
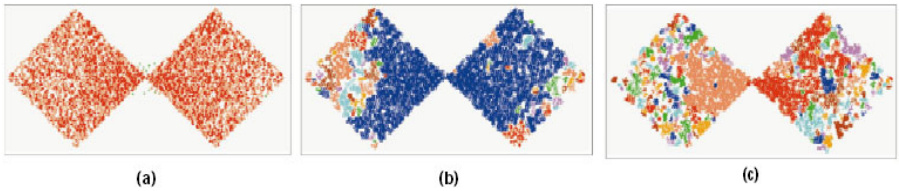



*Figure 9. DBScan results for DS2 with MinPts at 4 and Eps at (a) 5.0, (b) 3.5, and (c) 3.0.*




Ack. Figures from G. Karypis, E.-H. Han, and V. Kumar, *COMPUTER*, 32(8), 1999

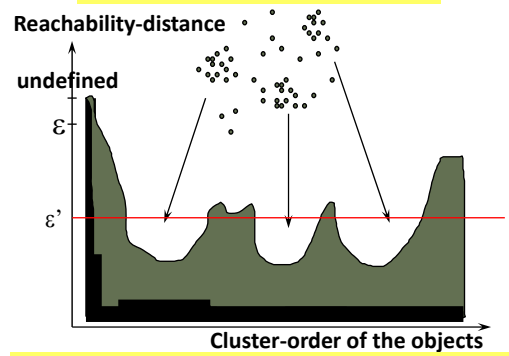
74

74

# OPTICS: Ordering Points To Identify Clustering Structure

- OPTICS (Ankerst, Breunig, Kriegel, and Sander, SIGMOD'99)
  - DBSCAN is sensitive to parameter setting
  - An extension: finding clustering structure
- Observation: Given a *MinPts*, density-based clusters w.r.t. a higher density are completely contained in clusters w.r.t. to a lower density
- Idea: Higher density points should be processed first—find high-density clusters first
- OPTICS stores such a clustering order using two pieces of information:
  - *Core distance* and *reachability distance*

Reachability plot for a dataset




- Since points belonging to a cluster have a low reachability distance to their nearest neighbor, valleys correspond to clusters
- The deeper the valley, the denser the cluster

75

75

# OPTICS: An Extension from DBSCAN

- **Core distance** of an object *p*: The smallest value *ε* such that the *ε*-neighborhood of *p* has at least *MinPts* objects

  Let $N_\varepsilon(p)$: *ε*-neighborhood of *p*

  *ε* is a distance value

  Core-distance$_{\varepsilon, MinPts}(p)$ = Undefined if card($N_\varepsilon(p)$) < *MinPts*

  *MinPts*-distance(p), otherwise

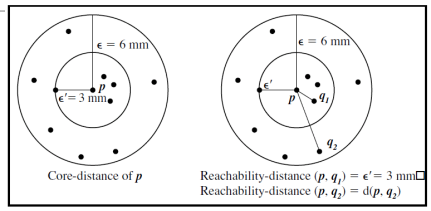



Figure 10.16: OPTICS terminology. Based on [ABKS99].

- **Reachability distance** of object *q* from core object *p* is the min. radius value that makes *q* density-reachable from *p*

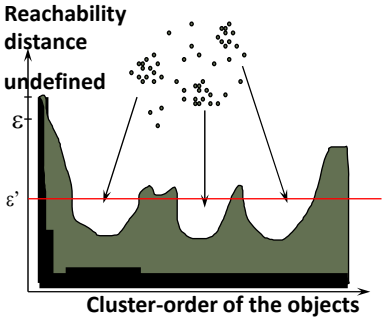
  Reachability-distance$_{\varepsilon, MinPts}(p, q)$ =

  Undefined, *if p is not a core object*

  max(core-distance(*p*), distance (*q*, *p*)), *otherwise*

- Complexity: O(*N* log*N*) (if index-based)

  where *N*: # of points




76

76

# OPTICS (cont.)

- OPTICS does not explicitly produce a data set clustering.
- It outputs a cluster ordering.
  - It is a linear list of all objects under analysis and
  - Represents the density-based clustering structure of the data.
- Objects in a denser cluster are listed closer to each other in the cluster ordering
- Ordering is equivalent to density-based clustering obtained from a wide range of parameter settings.
- Thus OPTICS does not require the user to provide a specific density threshold.
- The cluster ordering can be used to extract basic clustering information (e.g., cluster centers, or arbitrary-shaped clusters), derive the intrinsic clustering structure, as well as provide a visualization of the clustering.

77

77

# OPTICS (Cont.)

- It computes an ordering of all objects in a given database. And
- It stores the core-distance and a suitable reachability-distance for each object in the database.
- OPTICS maintains a list called **OrderSeeds** to help generate the output ordering.
- Objects in **OrderSeeds**
  - Are sorted by the reachability-distance from their respective closest core objects,
  - That is, by the smallest reachability-distance of each object.

78

78

## Main Algorithm

- OPTICS (SetOfObjects, e, MinPts, OrderedFile)
  - OrderedFile.open();
  - FOR i FROM 1 TO SetOfObjects.size DO
    - Object := SetOfObjects.get(i);
    - IF NOT Object.Processed THEN
      - ExpandClusterOrder(SetOfObjects, Object, e, MinPts, OrderedFile)
  - OrderedFile.close();
- END; // OPTICS

79

79

- ExpandClusterOrder(SetOfObjects, Object, $\varepsilon$, MinPts, OrderedFile);
  - neighbors := SetOfObjects.neighbors(Object, $\varepsilon$);
  - Object.Processed := TRUE;
  - Object.reachability_distance := UNDEFINED;
  - Object.setCoreDistance(neighbors, $\varepsilon$, MinPts);
  - OrderedFile.write(Object);
  - IF Object.core_distance <> UNDEFINED THEN
    - OrderSeeds.update(neighbors, Object);
    - WHILE NOT OrderSeeds.empty() DO
      - currentObject := OrderSeeds.next();
      - neighbors:=SetOfObjects.neighbors(currentObject, $\varepsilon$);
      - currentObject.Processed := TRUE;
      - currentObject.setCoreDistance(neighbors, $\varepsilon$, MinPts);
      - OrderedFile.write(currentObject);
      - IF currentObject.core_distance<>UNDEFINED THEN
        - OrderSeeds.update(neighbors, currentObject);
- END; // ExpandClusterOrder

80

80

# OrderSeeds::update()

```
OrderSeeds::update(neighbors, CenterObject);
    c_dist := CenterObject.core_distance;
    FORALL Object FROM neighbors DO
        IF NOT Object.Processed THEN
            new_r_dist:=max(c_dist,CenterObject.dist(Object));
            IF Object.reachability_distance=UNDEFINED THEN
                Object.reachability_distance := new_r_dist;
                insert(Object, new_r_dist);
            ELSE // Object already in OrderSeeds
                IF new_r_dist<Object.reachability_distance THEN
                    Object.reachability_distance := new_r_dist;
                    decrease(Object, new_r_dist);
END; // OrderSeeds::update
```

81

81

# Extract Clusters

- Having generated the cluster-ordering of a database with respect to e and MinPts,
- Extract any density-based clustering from this order w.r.t. MinPts and a clustering distance
  - By simply "scanning" the cluster-ordering
  - And assigning cluster-memberships depending on the reachability-distance and the core-distance of the objects.

82

82

# ExtractDBSCAN-Clustering (ClusterOrderedObjs, ε', MinPts)

- ExtractDBSCAN-Clustering (ClusterOrderedObjs, ε', MinPts)
- // Precondition: ε' ≤ generating dist ε for ClusterOrderedObjs
  - ClusterId := NOISE;
  - FOR i FROM 1 TO ClusterOrderedObjs.size DO
    - Object := ClusterOrderedObjs.get(i);
    - IF Object.reachability_distance > ε' THEN
    - // UNDEFINED > ε
      - IF Object.core_distance ≤ ε' THEN
        - ClusterId := nextId(ClusterId);
        - Object.clusterId := ClusterId;
      - ELSE
        - Object.clusterId := NOISE;
    - ELSE // Object.reachability_distance ≤ ε'
      - Object.clusterId := ClusterId;
- END; // ExtractDBSCAN-Clustering

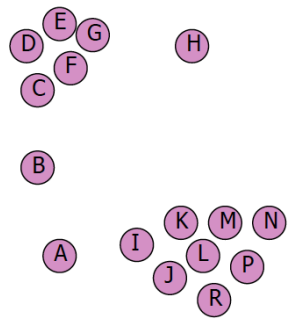
83

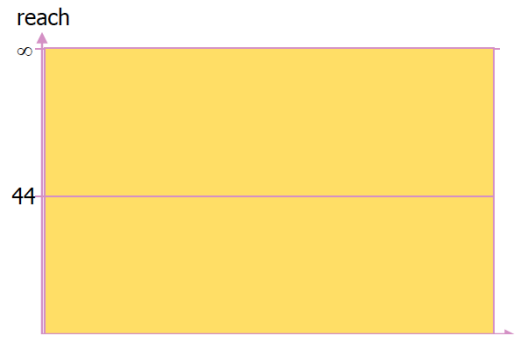
83

84

84

- Example Database (2-dimensional, 16 points)
- $\varepsilon = 44$, $MinPts = 3$



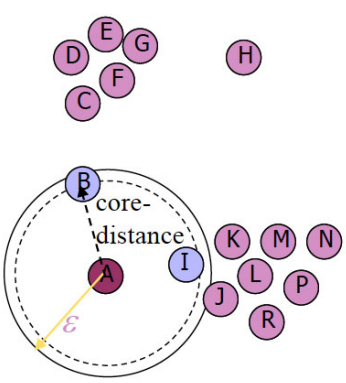



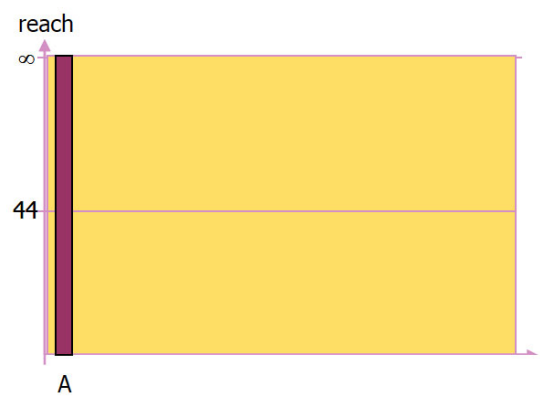

seedlist:

85

85

- $\varepsilon = 44$, $MinPts = 3$



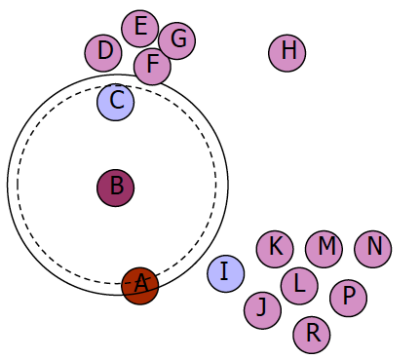

seedlist: (B,40) (I, 40)

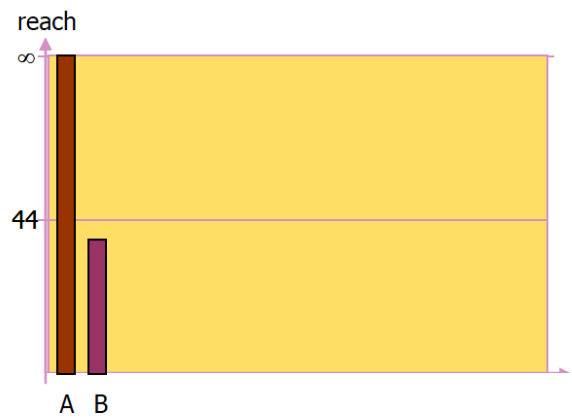
- Write A(reachability: undefined, core dist: 40)

86

86

- $\varepsilon = 44$, $MinPts = 3$



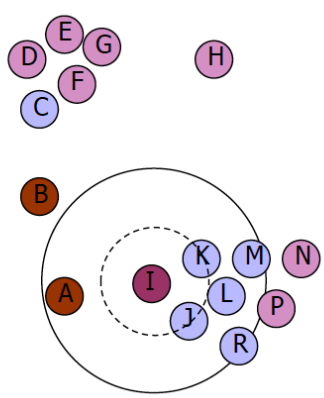

seedlist: (I, 40) (C, 40)

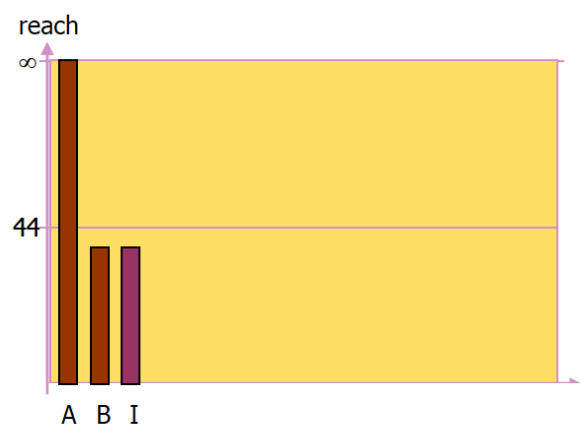
- Write B(reachability: 40, core dist: 40)

87

87

- $\varepsilon = 44$, $MinPts = 3$



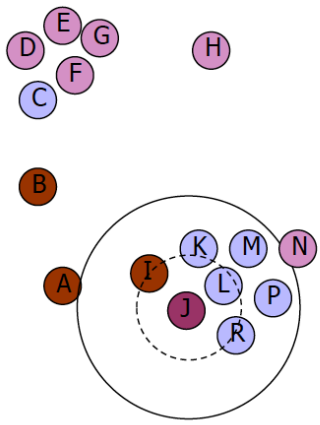

seedlist: (J, 20) (K, 20) (L, 31) (C, 40) (M, 40) (R, 43)

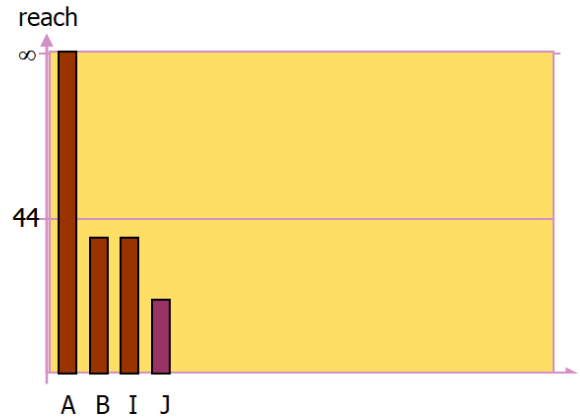
- Write I (reachability: 40, core dist: 20)

88

88

- $\varepsilon = 44$, $MinPts = 3$



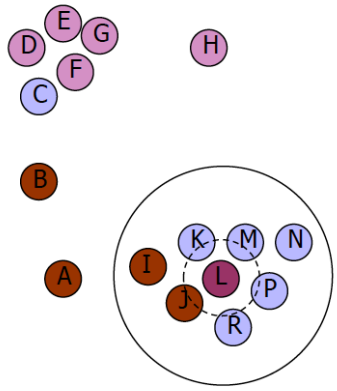

seedlist: (L, 19) (K, 20) (R, 21) (M, 30) (P, 31) (C, 40)

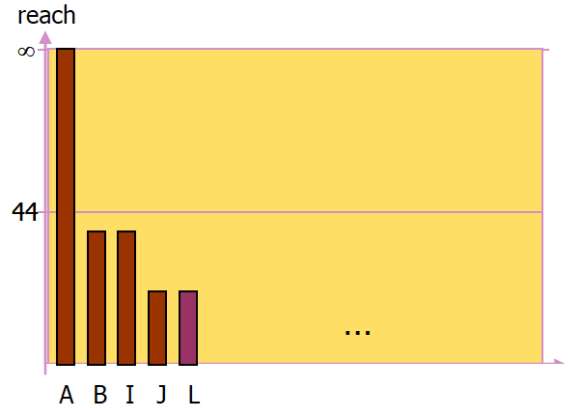
- Write J (reachability: 20, core dist: 20)

89

89

- $\varepsilon = 44$, $MinPts = 3$



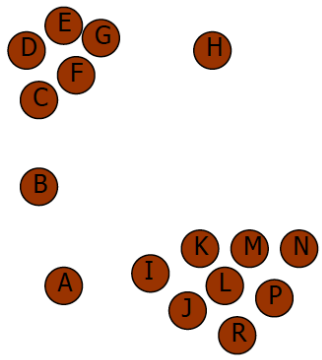



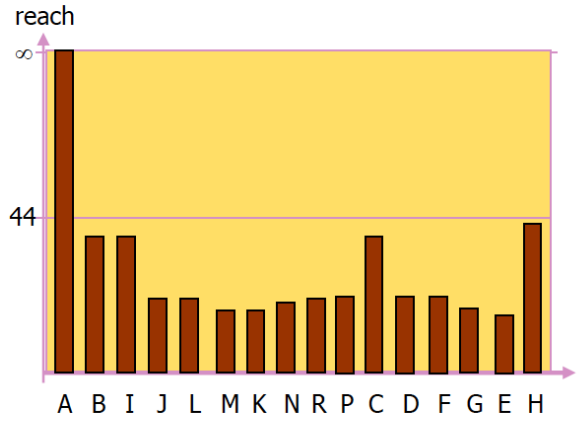

seedlist: (M, 18) (K, 18) (R, 20) (P, 21) (N, 35) (C, 40)

90

90

- $\varepsilon = 44$, $MinPts = 3$



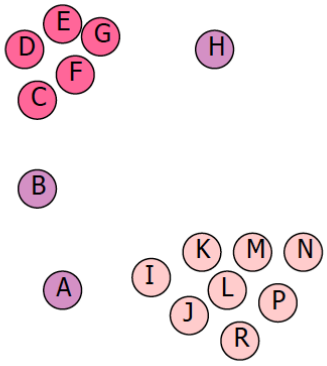



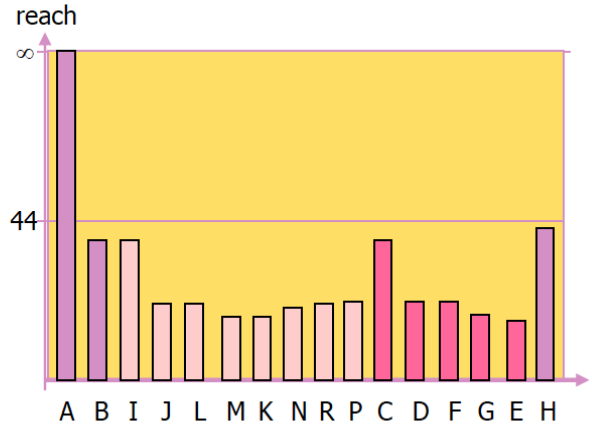

seedlist: -

91

91

- $\varepsilon = 44$, $MinPts = 3$ $\varepsilon' = 39$




seedlist: -

92

92

# OPTICS

- To produce a consistent result
  - Obey a specific order in which objects are processed when expanding a cluster.
- Select an object which is density-reachable with respect to the lowest **ε value**
  - To guarantee that clusters w.r.t higher density(i.e. smaller **ε** values) are finished first.
- OPTICS works in principle like such an extended DBSCAN algorithm for an infinite number of distance parameters **εi which are smaller than a "generating distance" ε(i.e.0≤εi≤ε).**
- The only difference is that we do not assign cluster memberships.
- Instead, we store the order in which the objects are processed and the information which would be used by an extended DBSCAN algorithm to assign cluster memberships if this were at all possible for an infinite number of parameters.

93

93

# Algorithm Performance

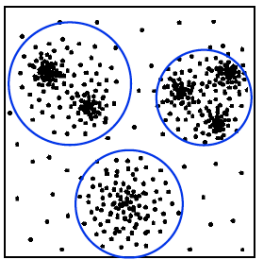
- To retrieve the e-neighborhood of an object, a region query with the center **o** and the radius **e** are used.
- Without any index support, to answer such a region query, a scan through the whole database has to be performed.
- In this case the run-time of optics would be $O(N^2)$
- If a tree-based spatial index can be used, the run-time is reduced to O(nlogn)
- Typically about 1.6 times of DBSCAN

94

94

**Flat Clustering**

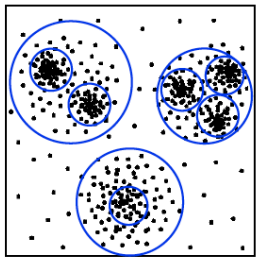
**one level of clusters**



e.g. density-based clustering algorithm
DBSCAN [KDD 96]

**Hierarchical Clustering**

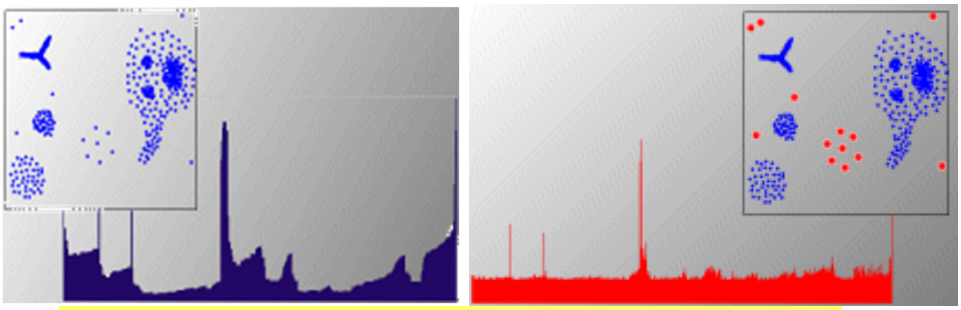
**nested clusters**



e.g. density-based clustering algorithm
OPTICS [SIGMOD 99]

95

95



Finding nested clustering structures with different parameter settings

96

96

- Scikitlearn: https://scikit-learn.org/stable/modules/generated/sklearn.cluster.OPTICS.html
- A matlab implementation: https://github.com/alexgkendall/OPTICS_Clustering
- Java ELKI data mining framework
- Optics Demo: https://www.youtube.com/watch?v=8kJjgowewOs

- More questions:
  - Do we need to specify K?
  - How does density-based methods find outliers?
  - Do we need a distance function?
  - Scalability compared to K-means?

97

97

# Chapter 10. Cluster Analysis: Basic Concepts and Methods

- Cluster Analysis: An Introduction
- Partitioning Methods
- Hierarchical Methods
- Density- and Grid-Based Methods
- Evaluation of Clustering
- Summary

105

105

# Clustering Validation

- Clustering Validation: Basic Concepts
- Clustering Evaluation: Measuring Clustering Quality
- External Measures for Clustering Validation
  - I: Matching-Based Measures
  - II: Entropy-Based Measures
  - III: Pairwise Measures
- Internal Measures for Clustering Validation
- Relative Measures
- Cluster Stability
- Clustering Tendency

106

106

# Clustering Validation and Assessment

- Major issues on clustering validation and assessment
  - **Clustering evaluation**
    - Evaluating the goodness of the clustering
  - **Clustering stability**
    - To understand the sensitivity of the clustering result to various algorithm parameters, e.g., # of clusters
  - **Clustering tendency**
    - Assess the suitability of clustering, i.e., whether the data has any inherent grouping structure

107

107

# Measuring Clustering Quality

- **Clustering Evaluation**: Evaluating the goodness of clustering results
  - **No** commonly recognized best suitable measure in practice
- **Three categorization of measures**: External, internal, and relative
  - **External**: Supervised, employ criteria not inherent to the dataset
    - Compare a clustering against prior or expert-specified knowledge (i.e., the ground truth) using certain clustering quality measure
  - **Internal**: Unsupervised, criteria derived from data and the grouping itself
    - Evaluate the goodness of a clustering by considering how well the clusters are separated and how compact the clusters are, e.g., silhouette coefficient
  - **Relative**: Directly compare different clusterings, usually those obtained via different parameter settings for the same algorithm

108

108

# Measuring Clustering Quality: External Methods

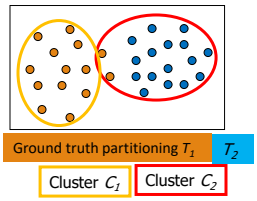
- Given the **ground truth** *T*, *Q*(*C, T*) is the **quality measure** for a clustering *C*
- *Q*(*C, T*) is good if it satisfies the following **four** essential criteria
  - **Cluster homogeneity**
    - The purer, the better
  - **Cluster completeness**
    - Assign objects belonging to the same category in the ground truth to the same cluster
  - **Rag bag better than alien**
    - 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

109

109

# Commonly Used External Measures

- **Matching-based measures** (To be covered)
  - Purity, maximum matching, F-measure
- **Entropy-Based Measures** [cover if time allows]
  - Conditional entropy
  - Normalized mutual information (NMI)
  - Variation of information
- **Pairwise measures** (To be covered)
  - Four possibilities: True positive (TP), FN, FP, TN
  - Jaccard coefficient, Rand statistic, Fowlkes-Mallow measure
- **Correlation measures**
  - Discretized Huber static, normalized discretized Huber static

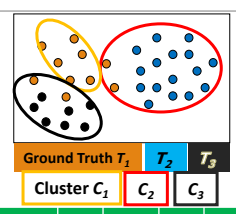



110

110

# Matching-Based Measures (I): Purity vs. Maximum Matching

- **Purity**: Quantifies the extent that cluster $C_i$ contains points only from one (ground truth) partition: $purity_i = \frac{1}{n_i}\max_{j=1}^{k}\{n_{ij}\}$
  - Total purity of clustering $C$: $purity = \sum_{i=1}^{r}\frac{n_i}{n}purity_i = \frac{1}{n}\sum_{i=1}^{r}\max_{j=1}^{k}\{n_{ij}\}$
  - Perfect clustering if purity = 1 and $r = k$ (the number of clusters obtained is the same as that in the ground truth)
  - Ex. 1 (green or orange): $purity_1$ = 30/50; $purity_2$ = 20/25; $purity_3$ = 25/25; $purity$ = (30 + 20 + 25)/100 = 0.75
  - Two clusters may share the same majority partition
- **Maximum matching**: Only one cluster can match one partition
  - **Match**: Pairwise matching, weight $w(e_{ij}) = n_{ij}$ $\quad w(M) = \sum_{e \in M} w(e)$
  - Maximum weight matching: $match = \arg\max_{M}\{\frac{w(M)}{n}\}$
  - Ex2. (green) *match* = *purity* = 0.75; (orange) *match* = 0.65 > 0.6



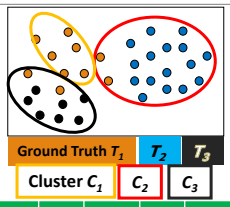

| C\T | $T_1$ | $T_2$ | $T_3$ | Sum |
|---|---|---|---|---|
| $C_1$ | 0 | 20 | 30 | 50 |
| $C_2$ | 0 | 20 | 5 | 25 |
| $C_3$ | 25 | 0 | 0 | 25 |
| $m_j$ | 25 | 40 | 35 | 100 |

| C\T | $T_1$ | $T_2$ | $T_3$ | Sum |
|---|---|---|---|---|
| $C_1$ | 0 | 30 | 20 | 50 |
| $C_2$ | 0 | 20 | 5 | 25 |
| $C_3$ | 25 | 0 | 0 | 25 |
| $m_j$ | 25 | 50 | 25 | 100 |

111

111

# Matching-Based Measures (II): F-Measure




| C\T | $T_1$ | $T_2$ | $T_3$ | Sum |
|---|---|---|---|---|
| $C_1$ | 0 | 20 | 30 | 50 |
| $C_2$ | 0 | 20 | 5 | 25 |
| $C_3$ | 25 | 0 | 0 | 25 |
| $m_j$ | 25 | 40 | 35 | 100 |

- **Precision**: The fraction of points in $C_i$ from the majority partition $T_{j_i}$ (i.e., the same as purity), where $j_i$ is the partition that contains the maximum # of points from $C_i$

$$prec_i = \frac{1}{n_i}\max_{j=1}^{k}\{n_{ij}\} = \frac{n_{ij_i}}{n_i}$$

  - Ex. For the green table
    - $prec_1$ = 30/50; $prec_2$ = 20/25; $prec_3$ = 25/25
- **Recall**: The fraction of point in partition $T_{j_i}$ shared in common with cluster $C_i$, where $m_{j_i} = |T_{j_i}|$

$$recall_i = \frac{n_{ij_i}}{|T_{j_i}|} = \frac{n_{ij_i}}{m_{j_i}}$$

  - Ex. For the green table
    - $recall_1$ = 30/35; $recall_2$ = 20/40; $recall_3$ = 25/25
- **F-measure** for $C_i$: The harmonic means of $prec_i$ and $recall_i$: $F_i = \frac{2n_{ij_i}}{n_i + m_{j_i}}$
- F-measure for clustering $C$: average of all clusters: $F = \frac{1}{r}\sum_{i=1}^{r} F_i$
  - Ex. For the green table
    - $F_1$ = 60/85; $F_2$ = 40/65; $F_3$ = 1; $F$ = 0.774

112

112

# General formula for F measures

- Recall is $\beta$ times as important as precision (putting more emphasis on false negatives!)

$$F_\beta = (1+\beta^2) * \frac{Precision * recall}{(\beta^2 * precision) + recall}$$

113

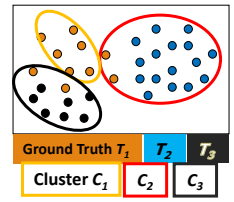
113

# Pairwise Measures: Four Possibilities for Truth Assignment

- **Four possibilities** based on the agreement between cluster label and partition label
  - *TP*: true positive—Two points $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ belong to the same partition *T*, and they also in the same cluster *C*

$$TP = |\{(\mathbf{x}_i, \mathbf{x}_j) : y_i = y_j \text{ and } \hat{y}_i = \hat{y}_j\}|$$

where $y_i$: the true partition label , and $\hat{y}_i$ : the cluster label for point $\boldsymbol{x}_i$




  - *FN*: false negative: $FN = |\{(\mathbf{x}_i, \mathbf{x}_j) : y_i = y_j \text{ and } \hat{y}_i \neq \hat{y}_j\}|$
  - *FP: false positive* $FP = |\{(\mathbf{x}_i, \mathbf{x}_j) : y_i \neq y_j \text{ and } \hat{y}_i = \hat{y}_j\}|$
  - *TN*: true negative $TN = |\{(\mathbf{x}_i, \mathbf{x}_j) : y_i \neq y_j \text{ and } \hat{y}_i \neq \hat{y}_j\}|$
- Calculate the four measures:

$$N = \binom{n}{2}$$ Total # of pairs of points

$$TP = \sum_{i=1}^{r}\sum_{j=1}^{k}\binom{n_{ij}}{2} = \frac{1}{2}\left(\left(\sum_{i=1}^{r}\sum_{j=1}^{k} n_{ij}^2\right) - n\right) \quad FN = \sum_{j=1}^{k}\binom{m_j}{2} - TP$$

$$FP = \sum_{i=1}^{r}\binom{n_i}{2} - TP \quad TN = N - (TP + FN + FP) = \frac{1}{2}\left(n^2 - \sum_{i=1}^{r} n_i^2 - \sum_{j=1}^{k} m_j^2 + \sum_{i=1}^{r}\sum_{j=1}^{k} n_{ij}^2\right)$$

114

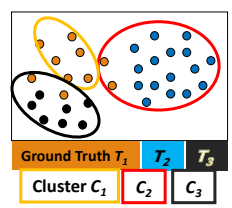
114

# Pairwise Measures: Jaccard Coefficient and Rand Statistic

- **Jaccard coefficient:** Fraction of true positive point pairs, but after ignoring the true negatives (thus asymmetric)
  - *Jaccard* = *TP*/(*TP* + *FN* + *FP*) [i.e., denominator ignores *TN*]
  - Perfect clustering: *Jaccard* = 1
- **Rand Statistic**:
  - *Rand* = (*TP* + *TN*)/*N*
  - Symmetric; perfect clustering: *Rand* = 1
- **Fowlkes-Mallow Measure**:
  - Geometric mean of precision and recall

$$FM = \sqrt{prec \times recall} = \frac{TP}{\sqrt{(TP+FN)(TP+FP)}}$$

- Using the above formulas, one can calculate all the measures for the green table (leave as an exercise)



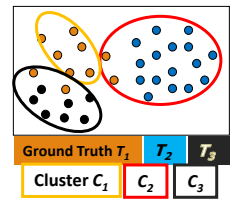

| C\T | $T_1$ | $T_2$ | $T_3$ | Sum |
|---|---|---|---|---|
| $C_1$ | 0 | 20 | 30 | 50 |
| $C_2$ | 0 | 20 | 5 | 25 |
| $C_3$ | 25 | 0 | 0 | 25 |
| $m_j$ | 25 | 40 | 35 | 100 |

115

115

# Entropy-Based Measures (I): Conditional Entropy*

- **Entropy of clustering** *C*: $H(\mathcal{C}) = -\sum_{i=1}^{r} p_{C_i} \log p_{C_i}$ $\quad p_{C_i} = \frac{n_i}{n}$ (i.e., the probability of cluster $C_i$)
- **Entropy of partitioning** *T*: $H(\mathcal{T}) = -\sum_{j=1}^{k} p_{T_i} \log p_{T_j}$
- **Entropy of *T* with respect to cluster $C_i$**: $H(\mathcal{T}|C_i) = -\sum_{j=1}^{k} (\frac{n_{ij}}{n_i}) \log(\frac{n_{ij}}{n_i})$
- **Conditional entropy of *T* with respect to clustering** *C*: $H(\mathcal{T}|\mathcal{C}) = -\sum_{i=1}^{r} (\frac{n_i}{n}) H(\mathcal{T}|C_i) = -\sum_{i=1}^{r}\sum_{j=1}^{k} p_{ij} \log(\frac{p_{ij}}{p_{C_i}})$




  - The more a cluster's members are split into different partitions, the higher the conditional entropy
  - For a perfect clustering, the conditional entropy value is 0, where the worst possible conditional entropy value is *log k*

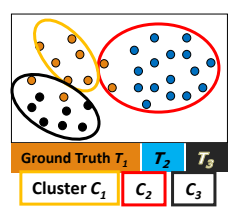
$$H(\mathcal{T}|\mathcal{C}) = -\sum_{i=1}^{r}\sum_{j=1}^{k} p_{ij}(\log p_{ij} - \log p_{C_i}) = -\sum_{i=1}^{r}\sum_{j=1}^{k} p_{ij} \log p_{ij} + \sum_{i=1}^{r}(\log p_{C_i} \sum_{j=1}^{k} p_{ij})$$
$$= -\sum_{i=1}^{r}\sum_{j=1}^{k} p_{ij} \log p_{ij} + \sum_{i=1}^{r}(p_{C_i} \log p_{C_i}) = H(\mathcal{C},\mathcal{T}) - H(\mathcal{C})$$

116

116

# Entropy-Based Measures (II): Normalized Mutual Information (NMI)*

- **Mutual information**:
  - Quantifies the amount of shared info between the clustering *C* and partitioning *T* $\quad I(C,T) = \sum_{i=1}^{r}\sum_{j=1}^{k} p_{ij} \log(\frac{p_{ij}}{p_{C_i} \cdot p_{T_j}})$
  - Measures the dependency between the observed joint probability $p_{ij}$ of *C* and *T*, and the expected joint probability $p_{Ci} \cdot p_{Tj}$ under the independence assumption
  - When *C* and *T* are independent, $p_{ij} = p_{Ci} \cdot p_{Tj}$, $I(C, T) = 0$. However, there is no upper bound on the mutual information




- **Normalized mutual information** (NMI)

$$NMI(\mathcal{C},\mathcal{T}) = \sqrt{\frac{I(\mathcal{C},\mathcal{T})}{H(\mathcal{C})} \cdot \frac{I(\mathcal{C},\mathcal{T})}{H(\mathcal{T})}} = \frac{I(\mathcal{C},\mathcal{T})}{\sqrt{H(\mathcal{C}) \cdot H(\mathcal{T})}}$$

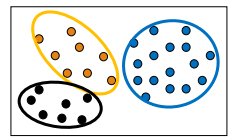
  - Value range of NMI: [0,1]. Value close to 1 indicates a good clustering

117

117

# Internal Measures (I): BetaCV Measure

- A trade-off in maximizing intra-cluster compactness and inter-cluster separation
- Given a clustering $C = \{C_1, \ldots, C_k\}$ with $k$ clusters, cluster $C_i$ contains $n_i = |C_i|$ points
  - Let $W(S, R)$ be sum of weights on all edges with one vertex in $S$ and the other in $R$
  - The sum of all the intra-cluster weights over all clusters: $W_{in} = \frac{1}{2}\sum_{i=1}^{k} W(C_i, C_i)$
  - The sum of all the inter-cluster weights: $W_{out} = \frac{1}{2}\sum_{i=1}^{k} W(C_i, \overline{C_i}) = \sum_{i=1}^{k-1}\sum_{j>i} W(C_i, C_j)$
  - The number of distinct intra-cluster edges: $N_{in} = \sum_{i=1}^{k}\binom{n_i}{2}$
  - The number of distinct inter-cluster edges: $N_{out} = \sum_{i=1}^{k-1}\sum_{j=i+1}^{k} n_i n_j$



- **Beta-CV measure**: $BetaCV = \frac{W_{in} / N_{in}}{W_{out} / N_{out}}$
  - The ratio of the mean intra-cluster distance to the mean inter-cluster distance
  - The smaller, the better the clustering

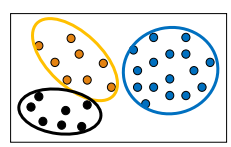
118

118

# Internal Measures (II): Normalized Cut and Modularity

- **Normalized cut**: $NC = \sum_{i=1}^{k}\frac{W(C_i, \overline{C_i})}{vol(C_i)} = \sum_{i=1}^{k}\frac{W(C_i, \overline{C_i})}{W(C_i, V)} = \sum_{i=1}^{k}\frac{W(C_i, \overline{C_i})}{W(C_i, C_i) + W(C_i, \overline{C_i})} = \sum_{i=1}^{k}\frac{1}{\frac{W(C_i, C_i)}{W(C_i, \overline{C_i})} + 1}$

  where $vol(C_i) = W(C_i, V)$ is the volume of cluster $C_i$
  - The higher normalized cut value, the better the clustering



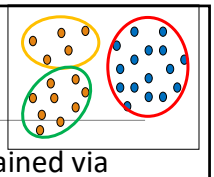
- **Modularity** (for graph clustering) $Q = \sum_{i=1}^{k}\left(\frac{W(C_i, C_i)}{W(V, V)} - \left(\frac{W(C_i, V)}{W(V, V)}\right)^2\right)$
  - Modularity $Q$ is defined as

    where $W(V, V) = \sum_{i=1}^{k} W(C_i, V) = \sum_{i=1}^{k} W(C_i, C_i) + \sum_{i=1}^{k} W(C_i, \overline{C_i}) = 2(W_{in} + W_{out})$
  - Modularity measures the difference between the observed and expected fraction of weights on edges within the clusters.
  - The smaller the value, the better the clustering—the intra-cluster distances are lower than expected (i.e., a smaller ratio of intra-cluster distance to inter-cluster distance)

119

119

# Relative Measure



- Relative measure: Directly compare different clusterings, usually those obtained via different parameter settings for the same algorithm
- **Silhouette coefficient** as an **internal measure**: Check cluster cohesion and separation
  - For each point $\boldsymbol{x}_i$, its silhouette coefficient $s_i$ is: $s_i = \frac{\mu_{out}^{\min}(\mathbf{x}_i) - \mu_{in}(\mathbf{x}_i)}{\max\{\mu_{out}^{\min}(\mathbf{x}_i), \mu_{in}(\mathbf{x}_i)\}}$

    where $\mu_{in}(\mathbf{x}_i)$ is the mean distance from $\boldsymbol{x}_i$ to points in its own cluster

    $\mu_{out}^{\min}(\mathbf{x}_i)$ is the mean distance from $\boldsymbol{x}_i$ to points in its closest cluster
  - Silhouette coefficient (*SC*) is the mean values of $s_i$ across all the points: $SC = \frac{1}{n}\sum_{i=1}^{n} s_i$
  - *SC* close to +1 implies good clustering
    - Points are close to their own clusters but far from the other clusters
- **Silhouette coefficient** as a **relative measure**: Estimate the # of clusters in the data

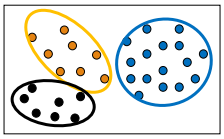
$$SC_i = \frac{1}{n_i}\sum_{x_j \in C_i} s_j$$

Pick the *k* value that yields the best clustering, i.e., yielding high values for *SC* and $SC_i$ ($1 \le i \le k$)

120

120

# Cluster Stability



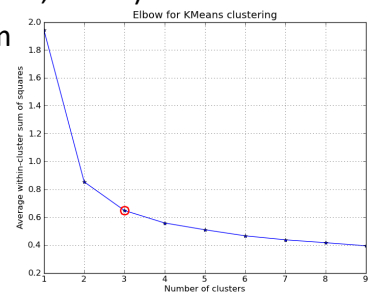
- Clusterings obtained from several datasets sampled from the same underlying distribution as ***D*** should be similar or "stable"
- Typical approach:
  - Find good parameter values for a given clustering algorithm
- Example: Find a good value of *k*, the correct number of clusters
- A **bootstrapping approach** to find the best value of *k* (judged on stability)
  - Generate *t* sets of samples of size *n* by sampling from ***D***
  - For each sample $\boldsymbol{D}_i$, run the same clustering algorithm with *k* values from 2 to $k_{max}$
  - Compare the distance between all pairs of clusterings $C_k(\boldsymbol{D}_i)$ and $C_k(\boldsymbol{D}_j)$ via some distance function
    - Compute the expected pairwise distance for each value of *k*
  - The value *k** that exhibits the least deviation between the clusterings obtained from the resampled datasets is the best choice for *k* since it exhibits the highest stability

121

121

# Other Methods for Finding K, the Number of Clusters

- **Empirical method**
  - # of clusters: $k \approx \sqrt{n/2}$ for a dataset of n points (e.g., $n$ = 200, $k$ = 10)
- **Elbow method**: Use the turning point in the curve of the sum of within cluster variance with respect to 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
    - For example, 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

122

122

# Chapter 10. Cluster Analysis: Basic Concepts and Methods

- Cluster Analysis: An Introduction
- Partitioning Methods
- Hierarchical Methods
- Density- and Grid-Based Methods
- Evaluation of Clustering
- Summary

125

125

# Summary

- Cluster Analysis: An Introduction
- Partitioning Methods
- Hierarchical Methods
- Density- and Grid-Based Methods
- Evaluation of Clustering

126

126

# References: (I) Cluster Analysis: An Introduction


- Jiawei Han, Micheline Kamber, and Jian Pei. Data Mining: Concepts and Techniques. Morgan Kaufmann, 3rd ed. , 2011 (Chapters 10 & 11)
- Charu Aggarwal and Chandran K. Reddy (eds.). Data Clustering: Algorithms and Applications. CRC Press, 2014
- Mohammed J. Zaki and Wagner Meira, Jr.. Data Mining and Analysis: Fundamental Concepts and Algorithms. Cambridge University Press, 2014
- L. Kaufman and P. J. Rousseeuw, Finding Groups in Data: An Introduction to Cluster Analysis, John Wiley & Sons, 1990
- Charu Aggarwal. An Introduction to Clustering Analysis. *in* Aggarwal and Reddy (eds.). Data Clustering: Algorithms and Applications (Chapter 1). CRC Press, 2014


127

127

# References: (II) Partitioning Methods


- J. MacQueen. Some Methods for Classification and Analysis of Multivariate Observations. In *Proc. of the 5th Berkeley Symp. on Mathematical Statistics and Probability*, 1967
- S. Lloyd. Least Squares Quantization in PCM. *IEEE Trans. on Information Theory*, 28(2), 1982
- A. K. Jain and R. C. Dubes. Algorithms for Clustering Data. Prentice Hall, 1988
- R. Ng and J. Han. Efficient and Effective Clustering Method for Spatial Data Mining. VLDB'94
- B. Schölkopf, A. Smola, and K. R. Müller. Nonlinear Component Analysis as a Kernel Eigenvalue Problem. *Neural computation*, 10(5):1299–1319, 1998
- I. S. Dhillon, Y. Guan, and B. Kulis. Kernel K-Means: Spectral Clustering and Normalized Cuts. *KDD*'04
- D. Arthur and S. Vassilvitskii. K-means++: The Advantages of Careful Seeding. SODA'07
- C. K. Reddy and B. Vinzamuri. A Survey of Partitional and Hierarchical Clustering Algorithms, in (Chap. 4) Aggarwal and Reddy (eds.), Data Clustering: Algorithms and Applications. CRC Press, 2014


128

128

# References: (III) Hierarchical Methods


- A. K. Jain and R. C. Dubes. Algorithms for Clustering Data. Prentice Hall, 1988
- L. Kaufman and P. J. Rousseeuw. Finding Groups in Data: An Introduction to Cluster Analysis. John Wiley & Sons, 1990
- T. Zhang, R. Ramakrishnan, and M. Livny. BIRCH: An Efficient Data Clustering Method for Very Large Databases. SIGMOD'96
- S. Guha, R. Rastogi, and K. Shim. Cure: An Efficient Clustering Algorithm for Large Databases. SIGMOD'98
- G. Karypis, E.-H. Han, and V. Kumar. CHAMELEON: A Hierarchical Clustering Algorithm Using Dynamic Modeling. *COMPUTER*, 32(8): 68-75, 1999.
- C. K. Reddy and B. Vinzamuri. A Survey of Partitional and Hierarchical Clustering Algorithms, in (Chap. 4) Aggarwal and Reddy (eds.), Data Clustering: Algorithms and Applications. CRC Press, 2014


129

129

# References: (IV) Density- and Grid-Based Methods


- M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases. KDD'96
- W. Wang, J. Yang, R. Muntz, STING: A Statistical Information Grid Approach to Spatial Data Mining, VLDB'97
- R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan. Automatic Subspace Clustering of High Dimensional Data for Data Mining Applications. SIGMOD'98
- A. Hinneburg and D. A. Keim. An Efficient Approach to Clustering in Large Multimedia Databases with Noise. KDD'98
- M. Ankerst, M. M. Breunig, H.-P. Kriegel, and J. Sander. Optics: Ordering Points to Identify the Clustering Structure. SIGMOD'99
- M. Ester. Density-Based Clustering. In (Chapter 5) Aggarwal and Reddy (eds.), Data Clustering: Algorithms and Applications . CRC Press. 2014
- W. Cheng, W. Wang, and S. Batista. Grid-based Clustering. In (Chapter 6) Aggarwal and Reddy (eds.), Data Clustering: Algorithms and Applications. CRC Press. 2014


130

130

# References: (IV) Evaluation of Clustering


- M. J. Zaki and W. Meira, Jr.. Data Mining and Analysis: Fundamental Concepts and Algorithms. Cambridge University Press, 2014
- L. Hubert and P. Arabie. Comparing Partitions. *Journal of Classification*, 2:193–218, 1985
- A. K. Jain and R. C. Dubes. Algorithms for Clustering Data. Printice Hall, 1988
- M. Halkidi, Y. Batistakis, and M. Vazirgiannis. On Clustering Validation Techniques. *Journal of Intelligent Info. Systems*, 17(2-3):107–145, 2001
- J. Han, M. Kamber, and J. Pei. Data Mining: Concepts and Techniques. Morgan Kaufmann, 3rd ed. , 2011
- H. Xiong and Z. Li. Clustering Validation Measures. in (Chapter 23) C. Aggarwal and C. K. Reddy (eds.), Data Clustering: Algorithms and Applications. CRC Press, 2014


131

131