



# **CS 412 Intro. to Data Mining**

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

**Jiawei Han, Computer Science, Univ. Illinois at Urbana-Champaign, 2017**





# **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

# **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

# 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 as similar as possible
- ❑ 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

# 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

# **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.

# 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 class)

## ❑ 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)

# Requirements and Challenges

---

## ❑ Quality

- ❑ Ability to deal with different types of attributes: Numerical, categorical, text, multimedia, networks, and mixture of multiple types
- ❑ Discovery of clusters with arbitrary shape
- ❑ Ability to deal with noisy data

## ❑ Scalability

- ❑ Clustering all the data instead of only on samples
- ❑ High dimensionality
- ❑ Incremental or stream clustering and insensitivity to input order

## ❑ Constraint-based clustering

- ❑ User-given preferences or constraints; domain knowledge; user queries

## ❑ Interpretability and usability

- ❑ The final generated clusters should be semantically meaningful and useful

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

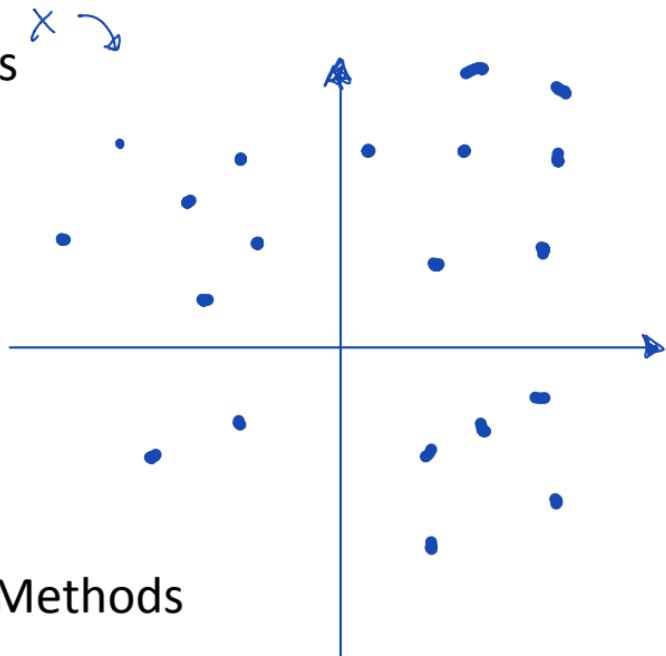
---

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



# 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



# 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, where  $c_k$  is the centroid or medoid of cluster  $C_k$ )
- ❑ A typical objective function: **Sum of Squared Errors (SSE)**

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

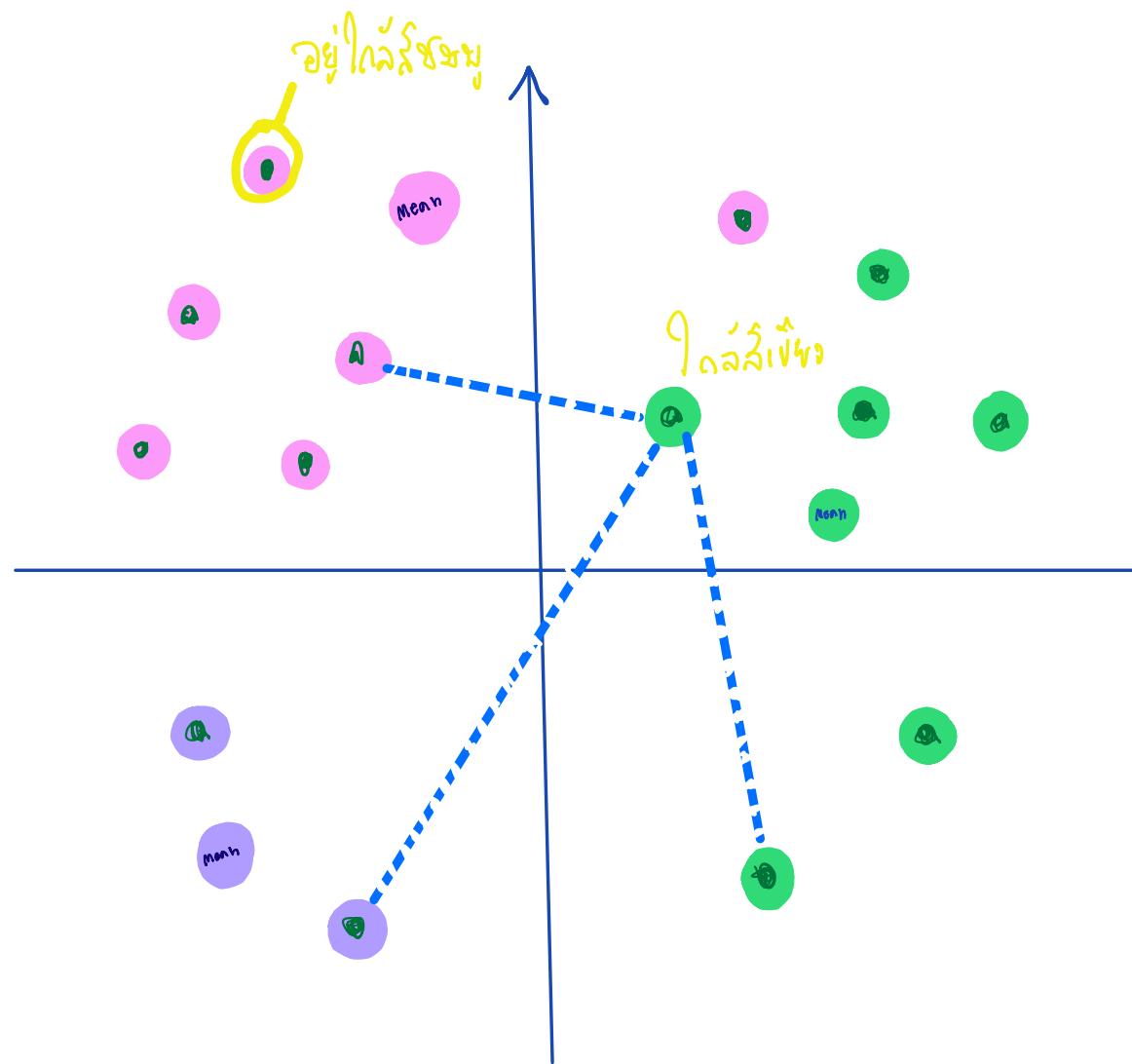
- ❑ Problem definition: Given  $K$ , find a partition of  $K$  clusters that optimizes the chosen partitioning criterion
  - ❑ Global optimal: Needs to exhaustively enumerate all partitions
  - ❑ Heuristic methods (i.e., greedy algorithms): *K-Means*, *K-Medians*, *K-Medoids*, etc.

# The *K-Means* Clustering Method

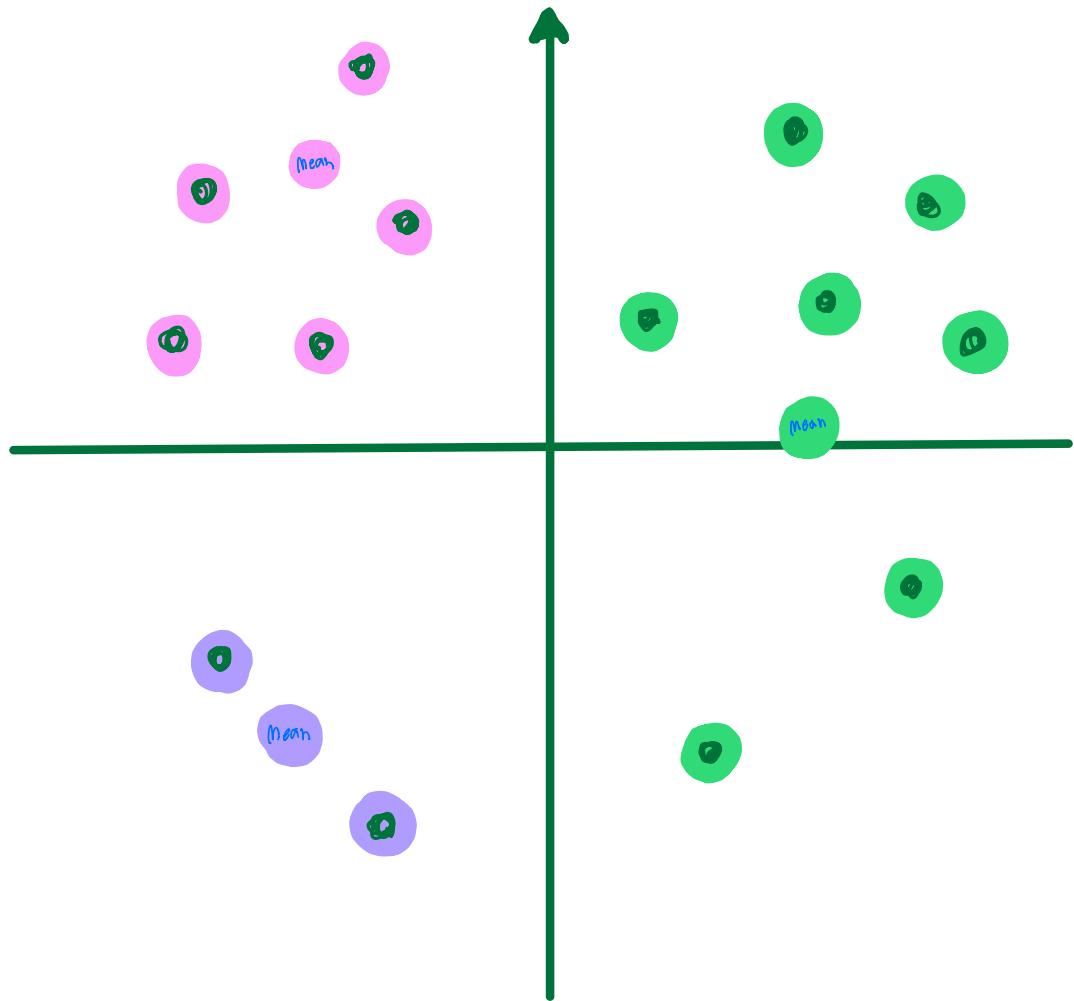
ກົດໍາ ຈົນແລະ ເຕັມກົດໍາ

- ❑ *K-Means* (MacQueen'67, Lloyd'57/'82)
    - ❑ Each cluster is represented by the center 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** → ກົດໍາໄປແຮງໝາຍ ດາວໂຫຼວດ ຂະໜາວອະນຸມາ
  - ❑ 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
- ❑ Different kinds of measures can be used
- ❑ Manhattan distance ( $L_1$  norm), Euclidean distance ( $L_2$  norm), Cosine similarity

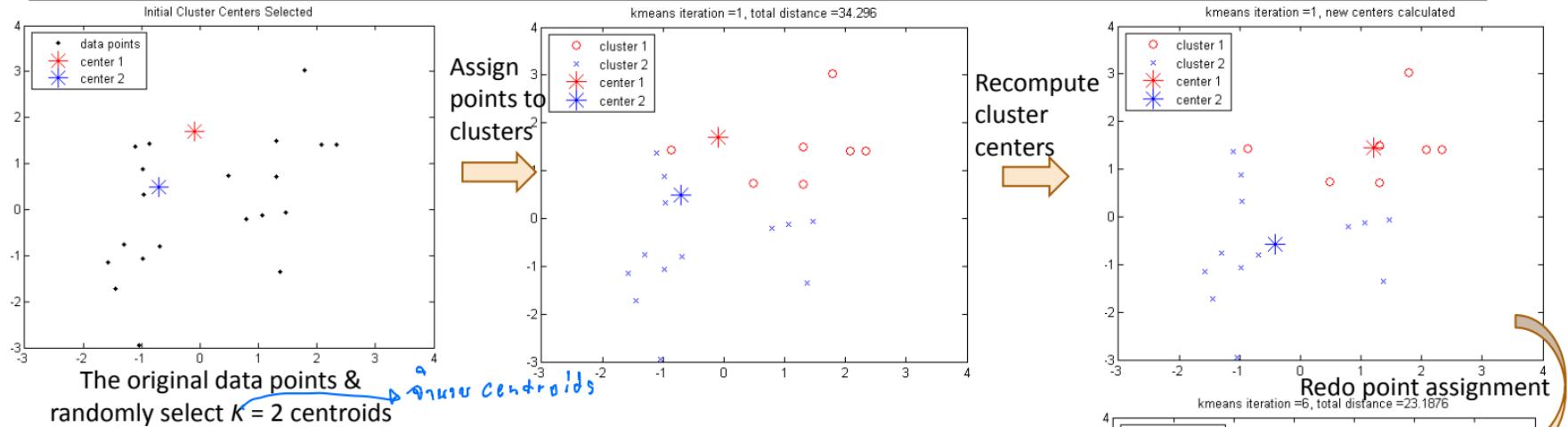
$K=3$



$K = 3$  ជាកំណត់លាយ  
ទម្រង់នាព័ត៌មាន Means ឬ មេដាច់



# Example: *K-Means* Clustering



## 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

# Discussion on the *K-Means* Method

---

- ❑ **Efficiency:**  $O(tKn)$  where  $n$ : # of objects,  $K$ : # of clusters, and  $t$ : # of iterations
  - ❑ Normally,  $K, t \ll n$ ; thus, an efficient method
- ❑ K-means clustering often ***terminates at a local optimal***
  - ❑ Initialization can be important to find high-quality clusters
- ❑ **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.

# 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*

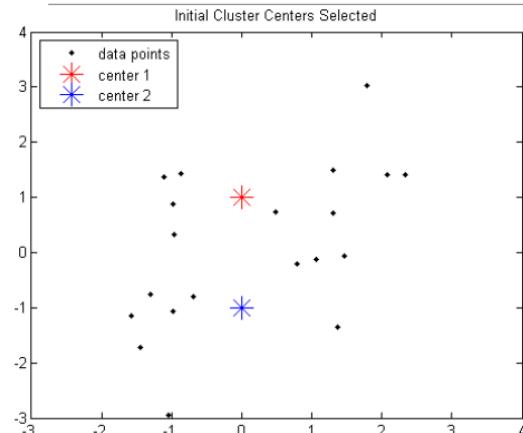
To be discussed in this lecture

- ❑ Applying feature transformation techniques

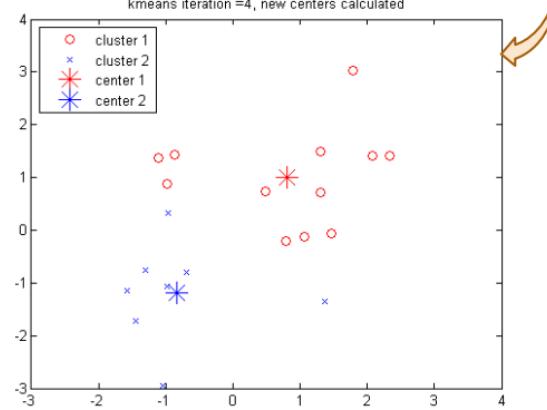
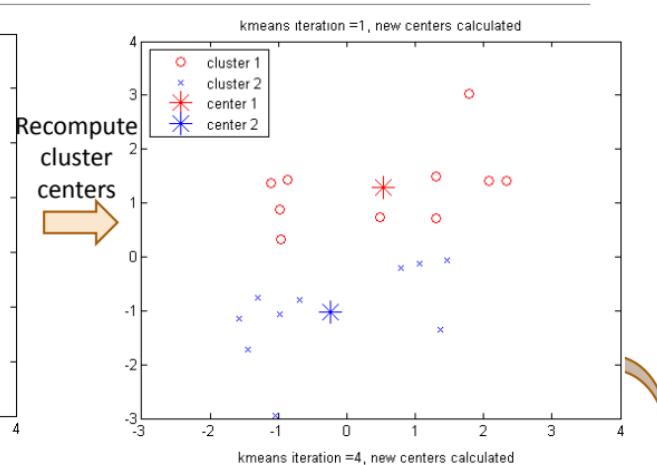
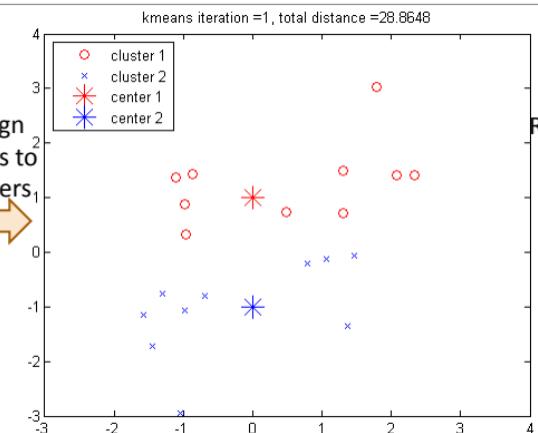
- ❑ *Weighted K-Means*, *Kernel K-Means*

To be discussed in this lecture

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



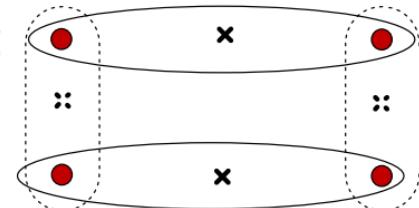
Another random selection of k centroids for the same data points



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

# 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 farthest from the currently selected (selection is based on a weighted probability score)
    - The selection continues until  $K$  centroids are obtained

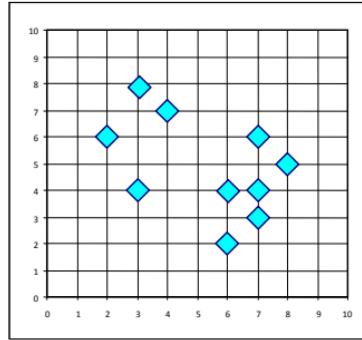


# Handling Outliers: From *K-Means* to *K-Medoids*

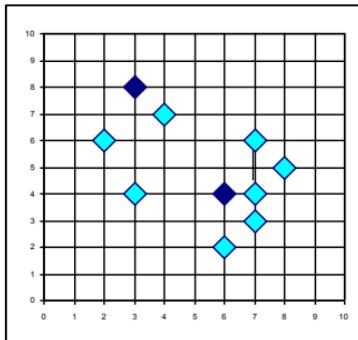
---

- 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 object 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  $o_i$
    - If  $S < 0$ , then swap  $m$  with  $o_i$  to form the new set of medoids
  - **Until** convergence criterion is satisfied

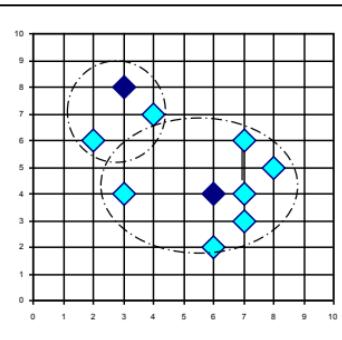
# PAM: A Typical $K$ -Medoids Algorithm



Arbitrary choose  $K$  object as initial medoids



Assign each remaining object to nearest medoids



Select initial  $K$  medoids randomly

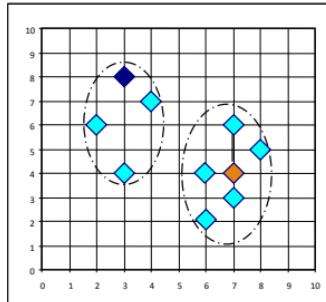
Repeat

Object re-assignment

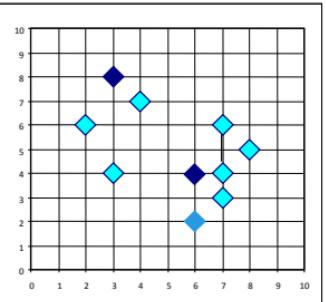
Swap medoid  $m$  with  $o_i$  if it improves the clustering quality

Until convergence criterion is satisfied

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



Compute total cost of swapping



# 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

# **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: 
$$S = \sum_{k=1}^K \sum_{x_{ij} \in C_k} |x_{ij} - med_{kj}|$$
- 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

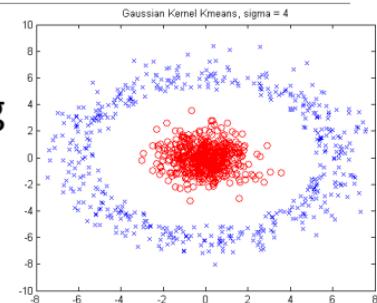
# K-Modes: Clustering Categorical Data

---

- ❑ *K-Means* cannot handle non-numerical (categorical) data
  - ❑ Mapping categorical value to 1/0 cannot generate quality clusters
- ❑ **K-Modes:** An extension to *K-Means* by replacing means of clusters with ***modes***
  - ❑ Mode: The value that appears most often in a **set** of data values
- ❑ Dissimilarity measure between object X and the center of a cluster Z
  - ❑  $\Phi(x_j, z_j) = 1 - n_j^r/n_j$ , when  $x_j = z_j$  ; 1 when  $x_j \neq z_j$ 
    - ❑ where  $z_j$  is the categorical value of attribute j in  $Z_i$ ,  $n_j$  is the number of objects in cluster  $i$ , 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 **fuzzy K-Modes** method is proposed to calculate a **fuzzy cluster membership value** for each object to each cluster
- ❑ A mixture of categorical and numerical data: Using a **K-Prototype** method

# ***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 is higher than K-Means
  - ❑ Need to compute and store  $n \times n$  kernel matrix generated from the kernel function on the original data, where  $n$  is the number of points
- ❑ *Spectral clustering* can be considered as a variant of Kernel K-Means clustering



# Kernel Functions and Kernel K-Means Clustering

---

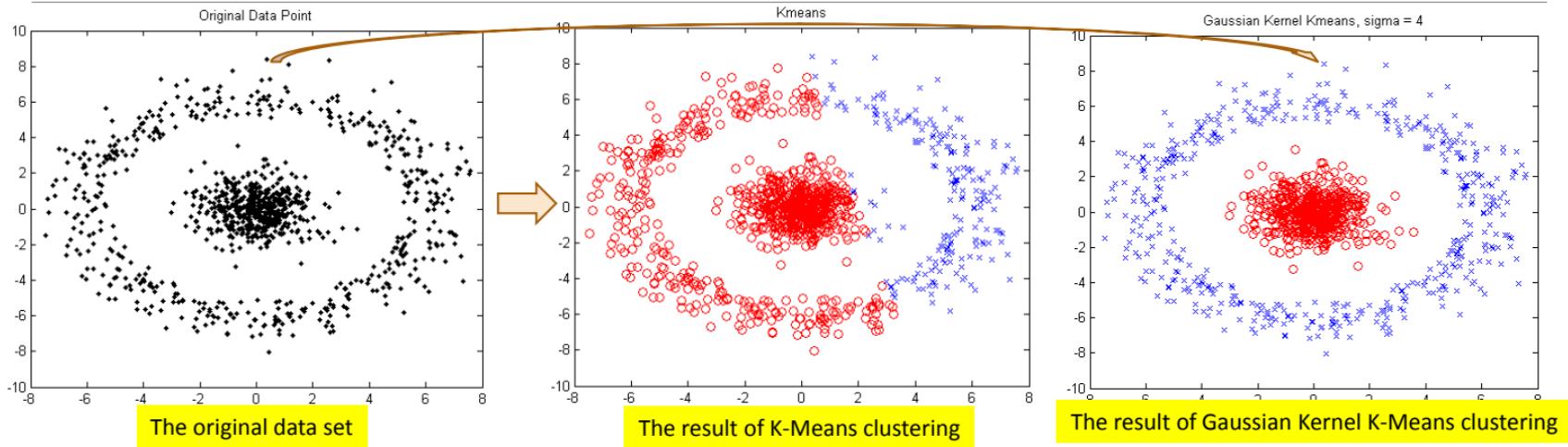
- Typical kernel functions:
  - Polynomial kernel of degree h:  $K(\mathbf{X}_i, \mathbf{X}_j) = (\mathbf{X}_i \cdot \mathbf{X}_j + 1)^h$
  - Gaussian radial basis function (RBF) kernel:  $K(\mathbf{X}_i, \mathbf{X}_j) = e^{-\|\mathbf{X}_i - \mathbf{X}_j\|^2 / 2\sigma^2}$
  - Sigmoid kernel:  $K(\mathbf{X}_i, \mathbf{X}_j) = \tanh(\kappa \mathbf{X}_i \cdot \mathbf{X}_j - \delta)$
- The formula for kernel matrix K for any two points  $\mathbf{x}_i, \mathbf{x}_j \in C_k$  is  $K_{x_i x_j} = \phi(\mathbf{x}_i) \bullet \phi(\mathbf{x}_j)$
- The SSE criterion of *kernel K-means*:  $SSE(C) = \sum_{k=1}^K \sum_{x_i \in C_k} \|\phi(\mathbf{x}_i) - c_k\|^2$ 
  - The formula for the cluster centroid:
$$c_k = \frac{\sum_{x_i \in C_k} \phi(\mathbf{x}_i)}{|C_k|}$$
- Clustering can be performed without the actual individual projections  $\phi(\mathbf{x}_i)$  and  $\phi(\mathbf{x}_j)$  for the data points  $\mathbf{x}_i, \mathbf{x}_j \in C_k$

# Example: Kernel Functions and Kernel K-Means Clustering

- Gaussian radial basis function (RBF) kernel:  $K(X_i, X_j) = e^{-\|X_i - X_j\|^2 / 2\sigma^2}$
- 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		RBF Kernel Space ( $\sigma = 4$ )				
	$x$	$K(x_i, x_1)$	$K(x_i, x_2)$	$K(x_i, x_3)$	$K(x_i, x_4)$	$K(x_i, x_5)$
$x_1$	0	0	$e^{-\frac{4^2+4^2}{2 \cdot 4^2}} = e^{-1}$	$e^{-1}$	$e^{-1}$	$e^{-1}$
$x_2$	4	0	$e^{-1}$	$e^{-2}$	$e^{-4}$	$e^{-2}$
$x_3$	-4	0	$e^{-1}$	0	$e^{-2}$	$e^{-4}$
$x_4$	-4	0	$e^{-1}$	$e^{-2}$	0	$e^{-2}$
$x_5$	4	0	$e^{-1}$	$e^{-2}$	$e^{-4}$	0

# Example: 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:  $K(X_i, X_j) = e^{-\|X_i - X_j\|^2 / 2\sigma^2}$
- K-Means clustering is conducted on the mapped data, generating quality clusters

# **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

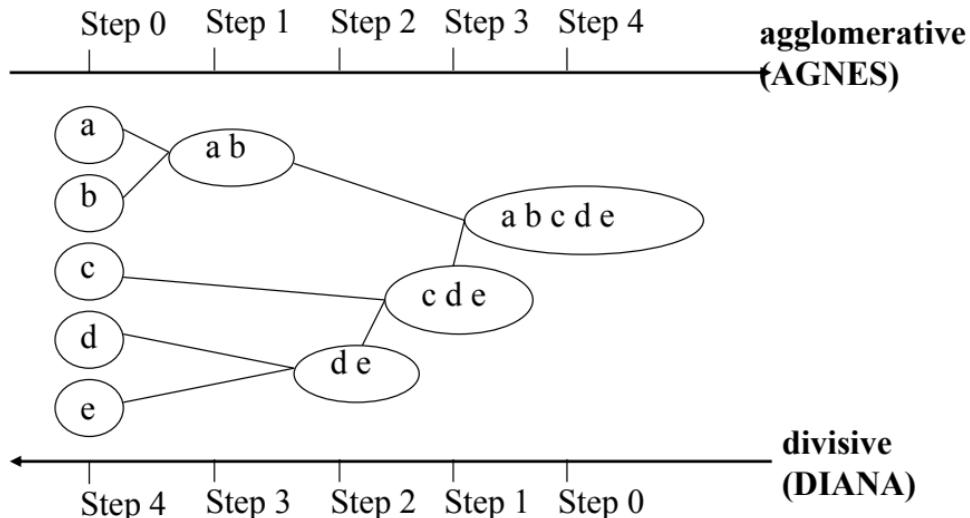
# Hierarchical Clustering Methods

---

- 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

# Hierarchical Clustering: Basic Concepts

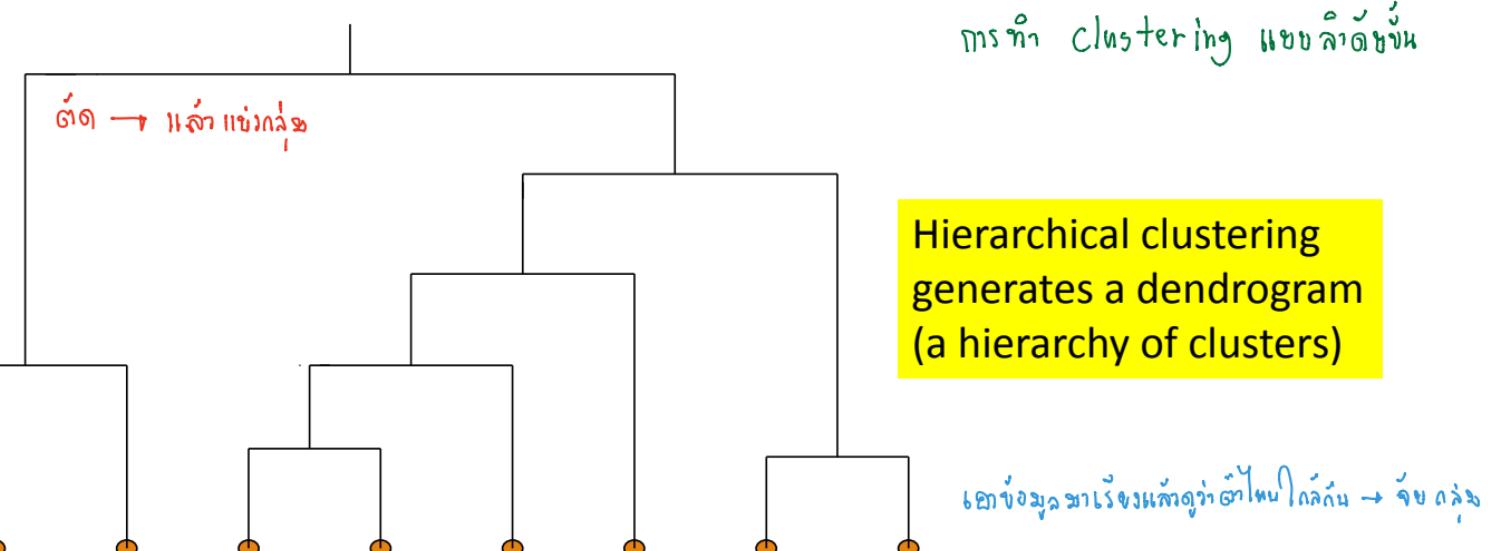
- ❑ Hierarchical clustering
  - ❑ Generate a clustering hierarchy (drawn as a **dendrogram**)
  - ❑ Not required to specify  $K$ , the number of clusters
  - ❑ More deterministic
  - ❑ No iterative refinement
- ❑ Two categories of algorithms:
  - ❑ **Agglomerative**: Start with singleton clusters, continuously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



# Dendrogram: Shows How Clusters are Merged

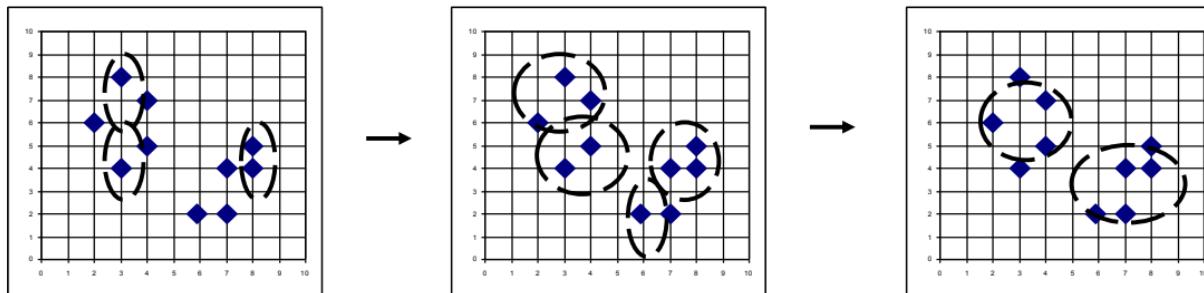
ລດໄຮ່ເກີດ ປຶ້ມາໃນການຈົດລົ່ງ ໂດຍແສ່ລະເພັນການລືບຕີ່ນ ຕົວການລົ່ບລ່າງ

- 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



# 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

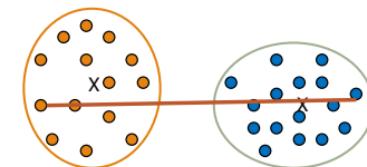
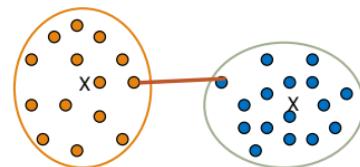


- ❑ Agglomerative clustering varies on different similarity measures among clusters
  - ❑ Single link (nearest neighbor)
  - ❑ Average link (group average)
  - ❑ Complete link (diameter)
  - ❑ Centroid link (centroid similarity)

# Single Link vs. Complete Link in Hierarchical Clustering

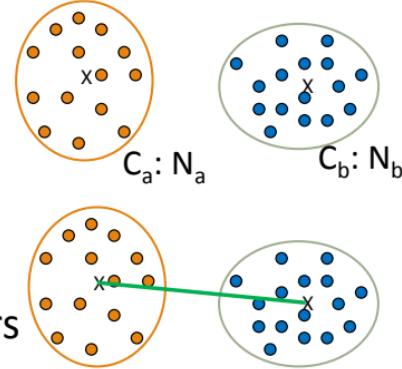
---

- 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 non-elliptical 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



# Agglomerative Clustering: Average vs. Centroid Links

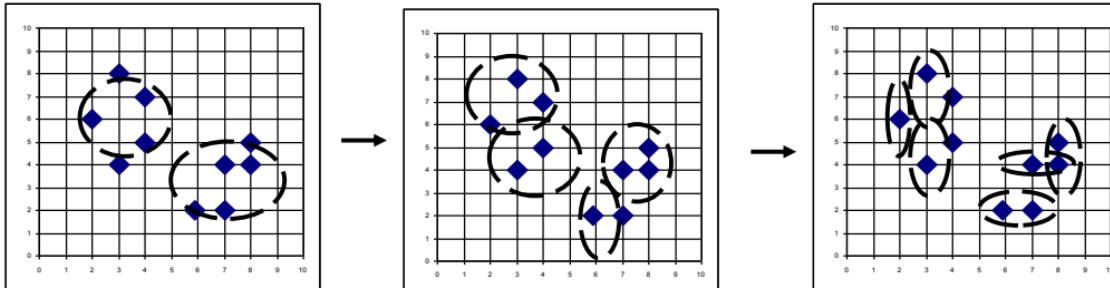
- ❑ Agglomerative clustering with **average link**
  - ❑ **Average link:** The average distance between an element in one cluster and an element in the other (i.e., all pairs in two clusters)
    - ❑ Expensive to compute
- ❑ Agglomerative clustering with **centroid link**
  - ❑ **Centroid link:** The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



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

# 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
  - ❑ More efficient when compared with agglomerative clustering

# 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
  - For categorical data, Gini-index can be used
- Handling the noise
  - Use a threshold to determine the termination criterion (do not generate clusters that are too small because they contain mainly noises)

# 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

# BIRCH (Balanced Iterative Reducing and Clustering Using Hierarchies)

---

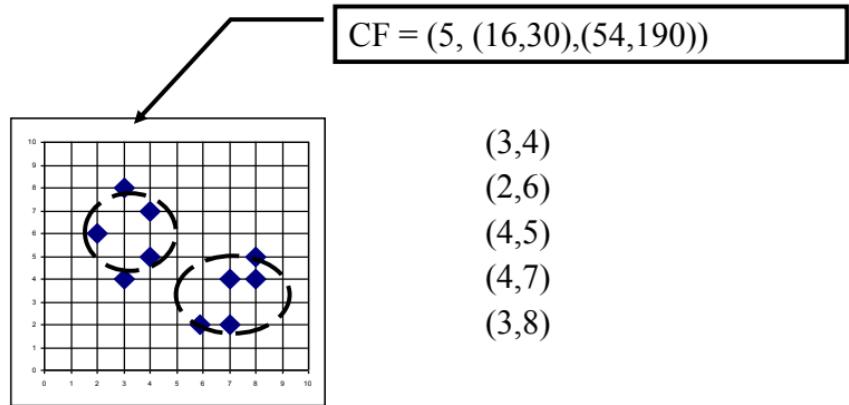
- ❑ A multiphase clustering algorithm (Zhang, Ramakrishnan & Livny, SIGMOD'96)
- ❑ Incrementally construct a CF (Clustering Feature) tree, a hierarchical data structure for multiphase clustering
- ❑ Phase 1: Scan DB to build an initial in-memory CF tree (a multi-level compression of the data that tries to preserve the inherent clustering structure of the data)
- ❑ Phase 2: Use an arbitrary clustering algorithm to cluster the leaf nodes of the CF-tree
- ❑ Key idea: Multi-level clustering
  - ❑ Low-level micro-clustering: Reduce complexity and increase scalability
  - ❑ High-level macro-clustering: Leave enough flexibility for high-level clustering
- ❑ *Scales linearly*: Find a good clustering with a single scan and improve the quality with a few additional scans

# Clustering Feature Vector in BIRCH

- Clustering Feature (CF):  $CF = (N, LS, SS)$

- $N$ : Number of data points

- $LS$ : linear sum of  $N$  points:  $\sum_{i=1}^N X_i$
  - $SS$ : square sum of  $N$  points:  $\sum_{i=1}^N X_i^2$



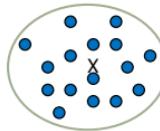
- Clustering feature:

- Summary of the statistics for a given sub-cluster: the 0-th, 1st, and 2nd moments of the sub-cluster from the statistical point of view
  - Registers crucial measurements for computing cluster and utilizes storage efficiently

# Measures of Cluster: Centroid, Radius and Diameter

---

- Centroid:  $\vec{x}_0$ 
  - the “middle” of a cluster
  - $n$ : number of points in a cluster
  - $\vec{x}_i$  is the  $i$ -th point in the cluster
- Radius: R
  - Average distance from member objects to the centroid
  - The square root of average distance from any point of the cluster to its centroid
- Diameter: D
  - Average pairwise distance within a cluster
  - The square root of average mean squared distance between all pairs of points in the cluster



$$\vec{x}_0 = \frac{\sum_i^n \vec{x}_i}{n}$$

$$R = \sqrt{\frac{\sum_i^n (\vec{x}_i - \vec{x}_0)^2}{n}}$$

$$D = \sqrt{\frac{\sum_i^n \sum_j^n (\vec{x}_i - \vec{x}_j)^2}{n(n-1)}}$$

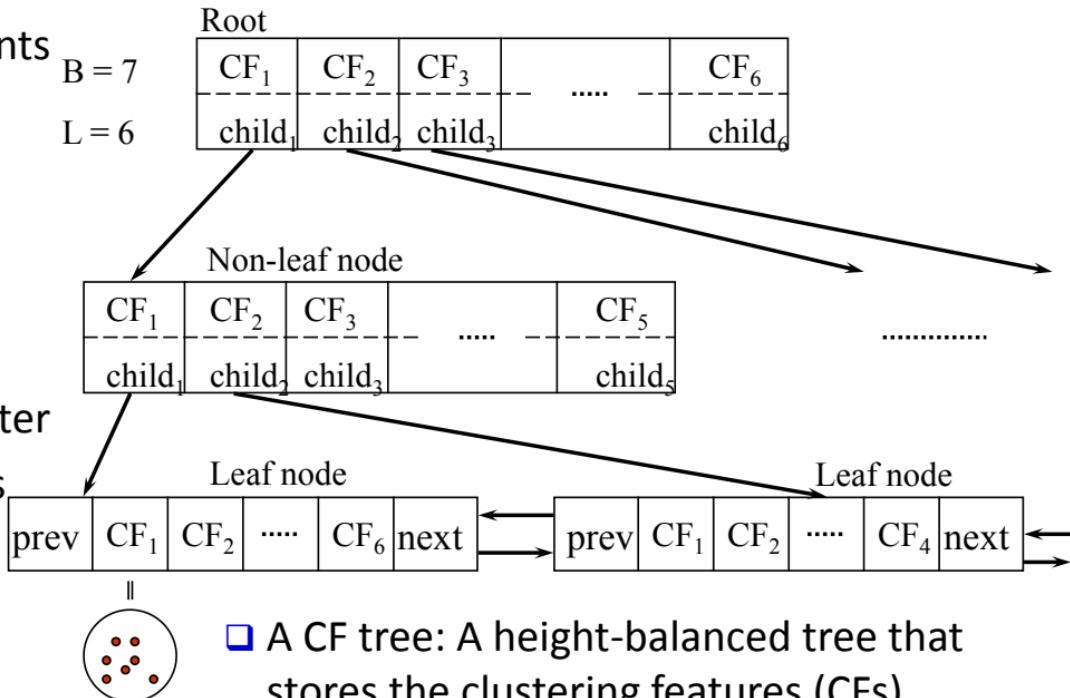
# The CF Tree Structure in BIRCH

- ❑ Incremental insertion of new points  
(similar to B+-tree)

- ❑ For each point in the input
  - ❑ Find closest leaf entry
  - ❑ Add point to leaf entry and update CF
  - ❑ If entry diameter > max\_diameter
    - ❑ split leaf, and possibly parents

- ❑ A CF tree has two parameters

- ❑ Branching factor: Maximum number of children
- ❑ Maximum diameter of sub-clusters stored at the leaf nodes



- ❑ A CF tree: A height-balanced tree that stores the clustering features (CFs)
- ❑ The non-leaf nodes store sums of the CFs of their children

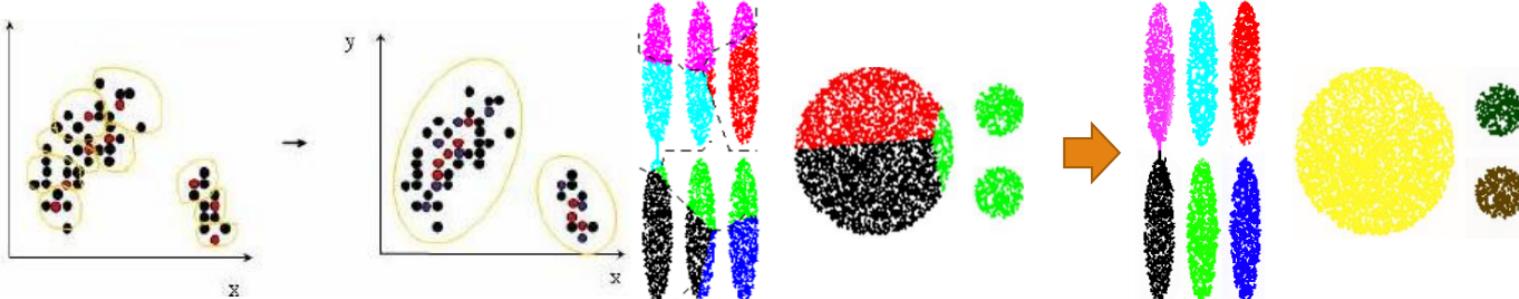
# BIRCH: A Scalable and Flexible Clustering Method

---

- ❑ An integration of agglomerative clustering with other (flexible) clustering methods
  - ❑ Low-level micro-clustering
  - ❑ Exploring CP-feature and BIRCH tree structure
  - ❑ Preserving the inherent clustering structure of the data
- ❑ Higher-level macro-clustering
  - ❑ Provide sufficient flexibility for integration with other clustering methods
- ❑ Impact to many other clustering methods and applications
- ❑ Concerns
  - ❑ Sensitive to insertion order of data points
  - ❑ Due to the fixed size of leaf nodes, clusters may not be so natural
  - ❑ Clusters tend to be spherical given the radius and diameter measures

# CURE: Clustering Using Representatives

- ❑ CURE (Clustering Using REpresentatives) (S. Guha, R. Rastogi, and K. Shim, 1998)
  - ❑ Represent a cluster using a set of well-scattered representative points
- ❑ Cluster distance: Minimum distance between the representative points chosen
  - ❑ This incorporates features of both single link and average link
- ❑ Shrinking factor  $\alpha$ : The points are shrunk towards the centroid by a factor  $\alpha$ 
  - ❑ Far away points are shrunk more towards the center: More robust to outliers
- ❑ Choosing scattered points helps CURE capture clusters of arbitrary shapes



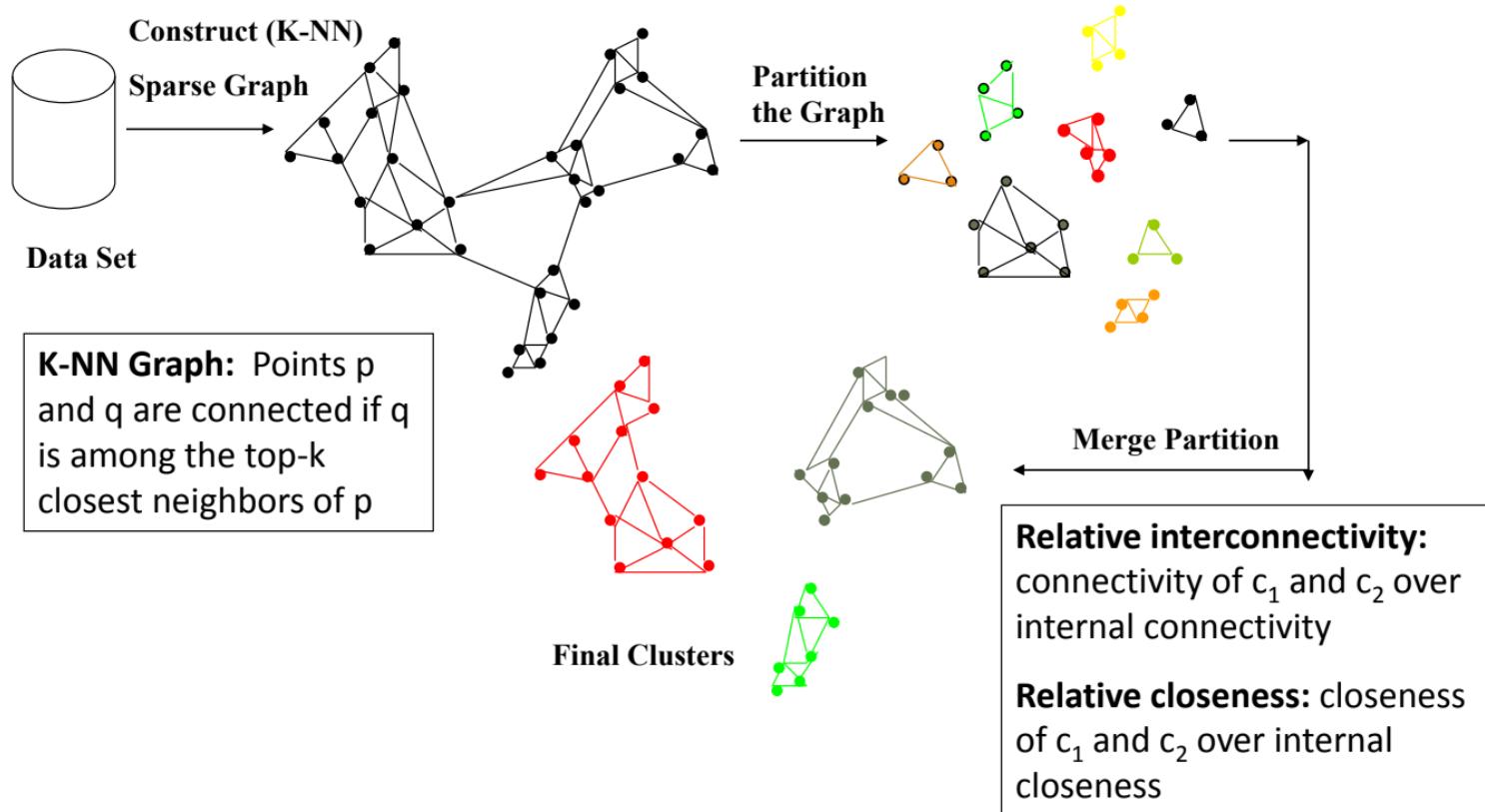
Courtesy: Kyuseok Shim@SNU.KR

# CHAMELEON: Hierarchical Clustering Using Dynamic Modeling

---

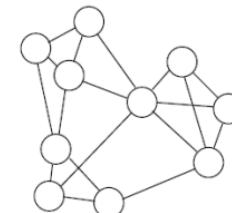
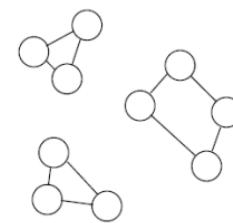
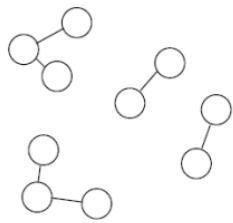
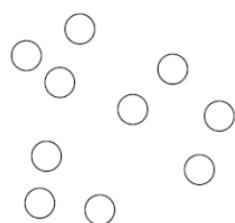
- ❑ CHAMELEON: A graph partitioning approach (G. Karypis, E. H. Han, and V. Kumar, 1999)
- ❑ Measures the similarity based on a dynamic model
  - ❑ Two clusters are merged only if the *interconnectivity* and *closeness (proximity)* between two clusters are high *relative to* the internal interconnectivity of the clusters and closeness of items within the clusters
- ❑ A graph-based, two-phase algorithm
  1. Use a graph-partitioning algorithm: Cluster objects into a large number of relatively small sub-clusters
  2. Use an agglomerative hierarchical clustering algorithm: Find the genuine clusters by repeatedly combining these sub-clusters

# Overall Framework of CHAMELEON



# 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

- ❑  $EC_{\{C_i, C_j\}}$ : 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  $EC_{C_i}$  (i.e., the weighted sum of edges that partition the graph into two roughly equal parts)
- ❑ Relative Interconnectivity (RI):
$$RI(C_i, C_j) = \frac{|EC_{\{C_i, C_j\}}|}{\frac{|EC_{C_i}| + |EC_{C_j}|}{2}}$$

# 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  $C_j$*

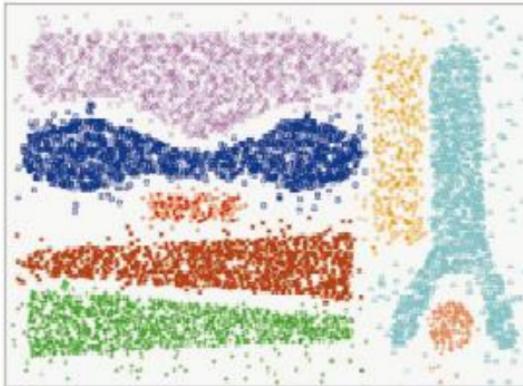
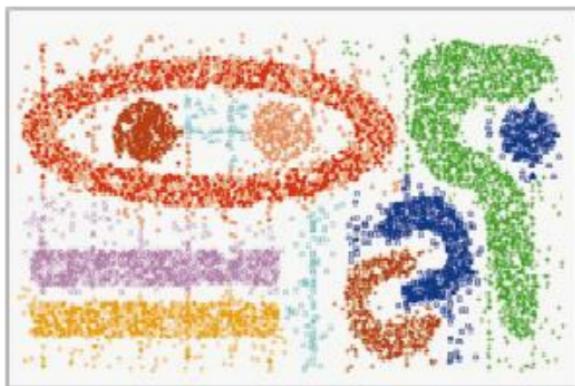
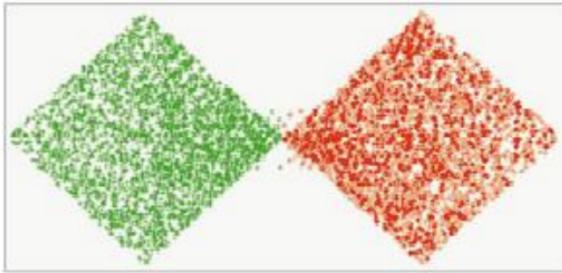
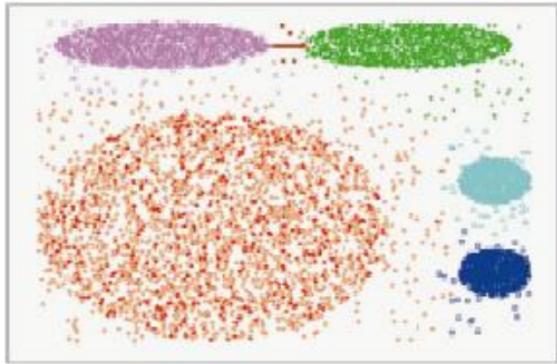
$$RC(C_i, C_j) = \frac{\overline{S}_{EC_{\{C_i, C_j\}}}}{\frac{|C_i|}{|C_i|+|C_j|}\overline{S}_{EC_{C_i}} + \frac{|C_j|}{|C_i|+|C_j|}\overline{S}_{EC_{C_j}}}$$

- 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

# CHAMELEON: Clustering Complex Objects



CHAMELEON is capable  
to generate quality  
clusters at clustering  
complex objects