

Data Mining: Concepts and Techniques

(3rd ed.)

— Chapter 10 —

Jianjun Cheng

School of Information Science & Engineering

Lanzhou University

©2011 Han, Kamber & Pei. All rights reserved.

Chapter 10. Cluster Analysis: Basic Concepts and Methods



- Cluster Analysis: Basic Concepts
- Partitioning Methods
- Hierarchical Methods
- Density-Based Methods
- Grid-Based Methods
- Evaluation of Clustering
- Summary

What is Cluster Analysis?

- Cluster: A collection of data objects
 - similar (or related) to one another within the same group
 - dissimilar (or unrelated) to the objects in other groups
- Cluster analysis (or clustering, data segmentation, ...)
 - Finding similarities between data according to the characteristics found in the data and grouping similar data objects into clusters
- **Unsupervised learning**: no predefined classes (i.e., learning by observations vs. learning by examples: supervised)
- Typical applications
 - As a **stand-alone tool** to get insight into data distribution
 - As a **preprocessing step** for other algorithms

Clustering for Data Understanding and Applications

- Biology: taxonomy of living things: kingdom, phylum, class, order, family, genus and species
- Information retrieval: document clustering
- Land use: Identification of areas of similar land use in an earth observation database
- Marketing: Help marketers discover distinct groups in their customer bases, and then use this knowledge to develop targeted marketing programs
- City-planning: Identifying groups of houses according to their house type, value, and geographical location
- Earth-quake studies: Observed earth quake epicenters should be clustered along continent faults
- Climate: understanding earth climate, find patterns of atmospheric and ocean
- Economic Science: market research

Clustering as a Preprocessing Tool (Utility)

- Summarization:
 - Preprocessing for regression, PCA, classification, and association analysis
- Compression:
 - Image processing: vector quantization
- Finding K-nearest Neighbors
 - Localizing search to one or a small number of clusters
- Outlier detection
 - Outliers are often viewed as those "far away" from any cluster

Quality: What Is Good Clustering?

- A good clustering method will produce high quality clusters
 - high intra-class similarity: **cohesive** within clusters
 - low inter-class similarity: **distinctive** between clusters
- The quality of a clustering method depends on
 - the similarity measure used by the method
 - its implementation, and
 - its ability to discover some or all of the hidden patterns

Measure the Quality of Clustering

- **Dissimilarity/Similarity metric**
 - Similarity is expressed in terms of a distance function, typically metric: $d(i, j)$
 - The definitions of **distance functions** are usually rather different for interval-scaled, boolean, categorical, ordinal ratio, and vector variables
 - Weights should be associated with different variables based on applications and data semantics
- **Quality of clustering:**
 - 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

Requirements and Challenges

- Scalability
 - Clustering all the data instead of only on samples
- Ability to deal with different types of attributes
 - Numerical, binary, categorical, ordinal, linked, and mixture of these
- Constraint-based clustering
 - User may give inputs on constraints
 - Use domain knowledge to determine input parameters
- Interpretability and usability
- Others
 - Discovery of clusters with arbitrary shape
 - Ability to deal with noisy data
 - Incremental clustering and insensitivity to input order
 - High dimensionality

Considerations for Cluster Analysis

- Partitioning criteria
 - Single level vs. hierarchical partitioning (often, multi-level hierarchical partitioning is desirable)
- 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., Euclidian, 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)

Major Clustering Approaches (I)

- Partitioning approach:
 - Construct various partitions and then evaluate them by some criterion, e.g., minimizing the sum of square errors
 - Typical methods: k-means, k-medoids, CLARANS
- Hierarchical approach:
 - Create a hierarchical decomposition of the set of data (or objects) using some criterion
 - Typical methods: Diana, Agnes, BIRCH, CAMELEON
- Density-based approach:
 - Based on connectivity and density functions
 - Typical methods: DBSACN, OPTICS, DenClue
- Grid-based approach:
 - based on a multiple-level granularity structure
 - Typical methods: STING, WaveCluster, CLIQUE

Major Clustering Approaches (II)

- Model-based:
 - A model is hypothesized for each of the clusters and tries to find the best fit of that model to each other
 - Typical methods: EM, SOM, COBWEB
- Frequent pattern-based:
 - Based on the analysis of frequent patterns
 - Typical methods: p-Cluster
- User-guided or constraint-based:
 - Clustering by considering user-specified or application-specific constraints
 - Typical methods: COD (obstacles), constrained clustering
- Link-based clustering:
 - Objects are often linked together in various ways
 - Massive links can be used to cluster objects: SimRank, LinkClus

Chapter 10. Cluster Analysis: Basic Concepts and Methods

- Cluster Analysis: Basic Concepts
- Partitioning Methods
- Hierarchical Methods
- Density-Based Methods
- Grid-Based Methods
- Evaluation of Clustering
- Summary



Partitioning Algorithms: Basic Concept

- Partitioning method: Partitioning a database **D** of **n** objects into a set of **k** clusters, such that the sum of squared distances is minimized (where c_i is the centroid or medoid of cluster C_i)

$$E = \sum_{i=1}^k \sum_{p \in C_i} (p - c_i)^2$$

- Given k , find a partition of k *clusters* that optimizes the chosen partitioning criterion
 - Global optimal: exhaustively enumerate all partitions
 - Heuristic methods: **k-means** and **k-medoids** algorithms
 - **k-means** (MacQueen'67, Lloyd'57/'82): Each cluster is represented by the center of the cluster
 - **k-medoids** or PAM (Partition around medoids) (Kaufman & Rousseeuw'87): Each cluster is represented by one of the objects in the cluster

The *K-Means* Clustering Method

- Given k , the **k-means** algorithm is implemented in four steps:
 1. Partition objects into k nonempty subsets
 2. Compute seed points as the centroids of the clusters of the current partitioning (the centroid is the center, i.e., **mean point**, of the cluster)
 3. Assign each object to the cluster with the nearest seed point
 4. Go back to step 2, stop when the assignment does not change

The *K-Means* Clustering Method

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

Input:

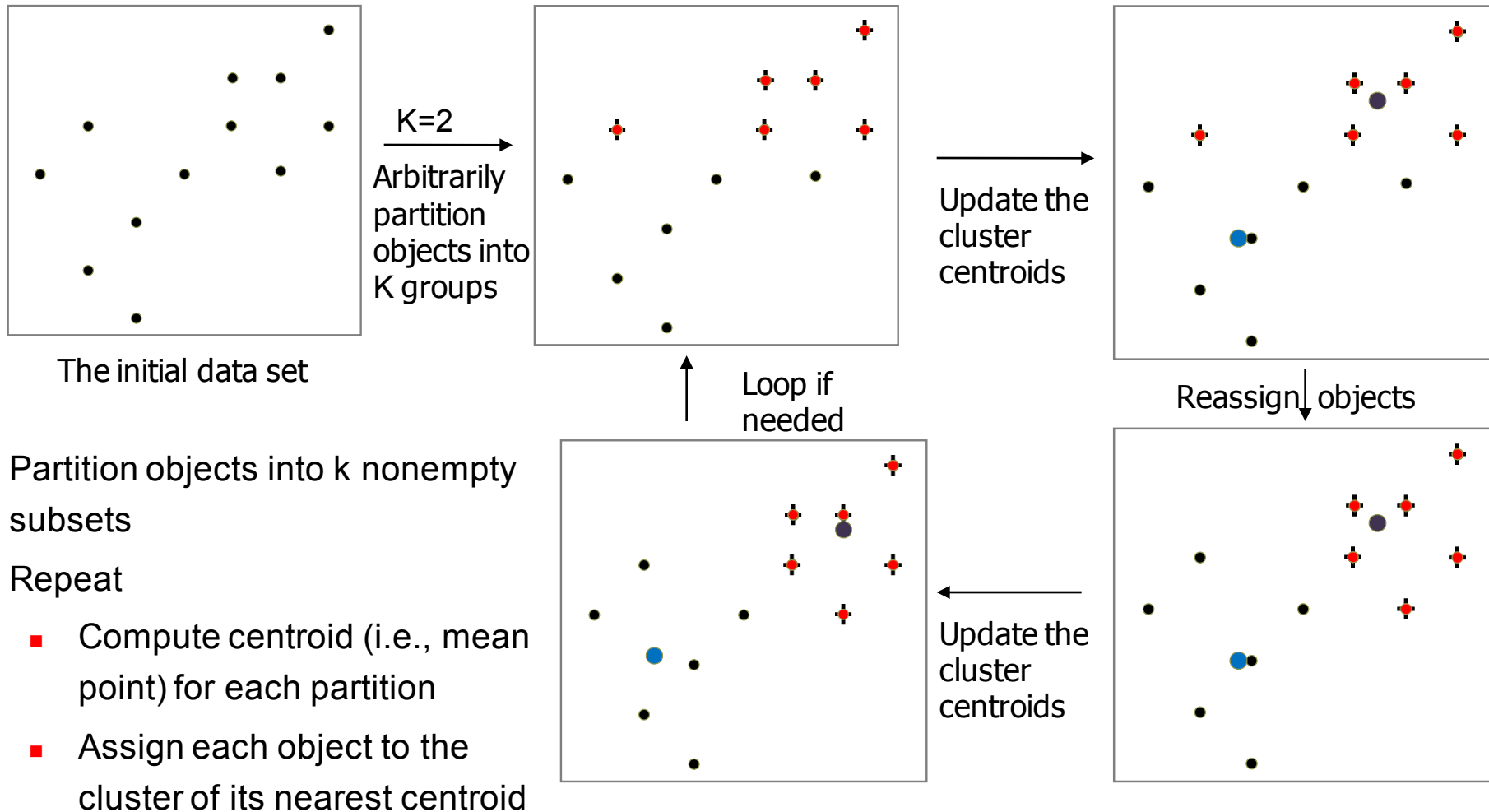
- k : the number of clusters,
- D : a data set containing n objects.

Output: A set of k clusters.

Method:

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

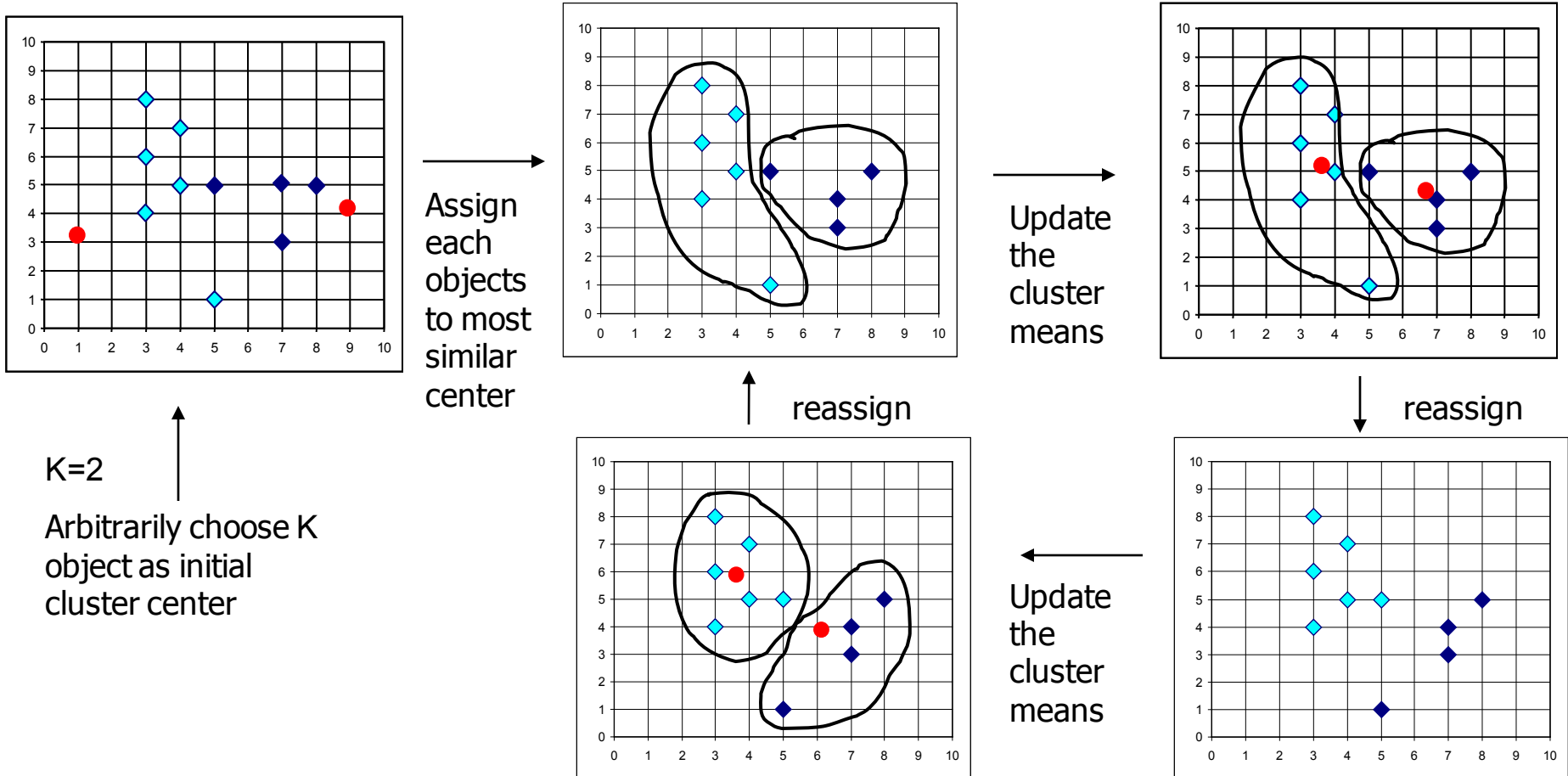
An Example of K-*Means* Clustering

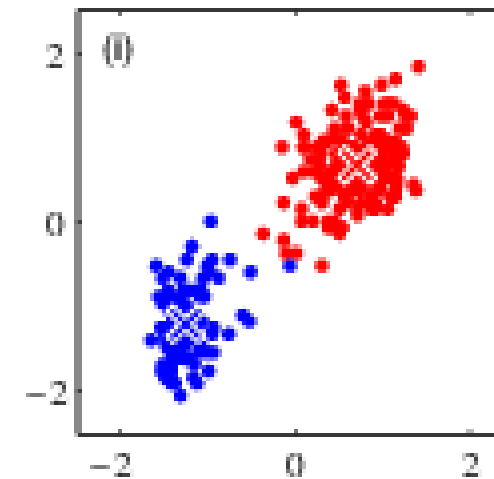
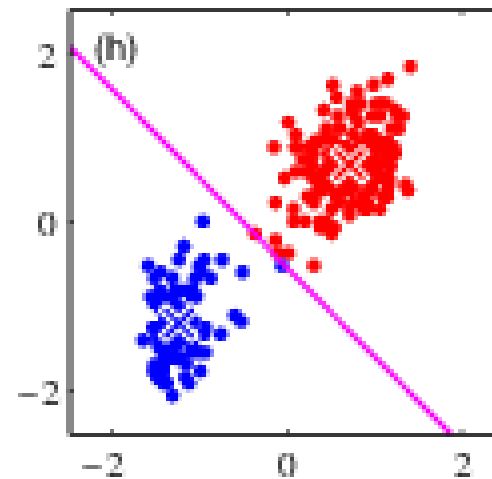
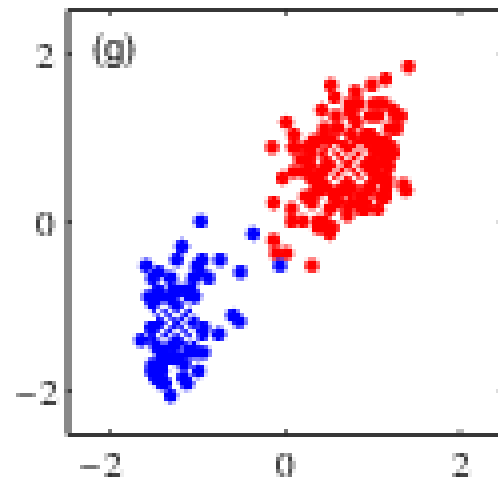
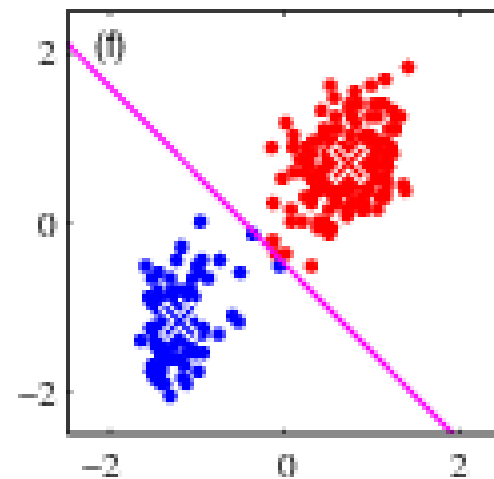
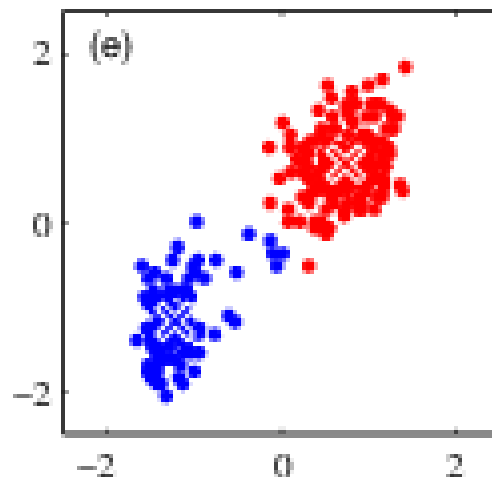
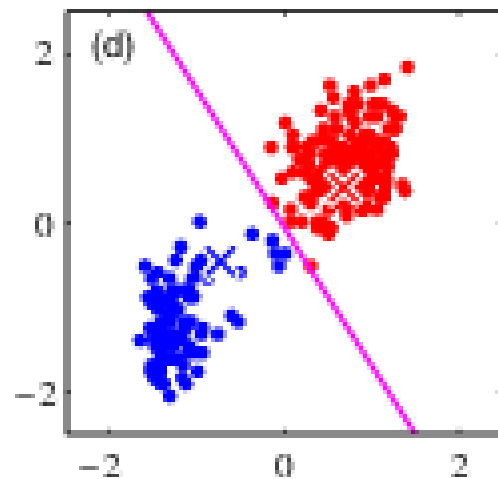
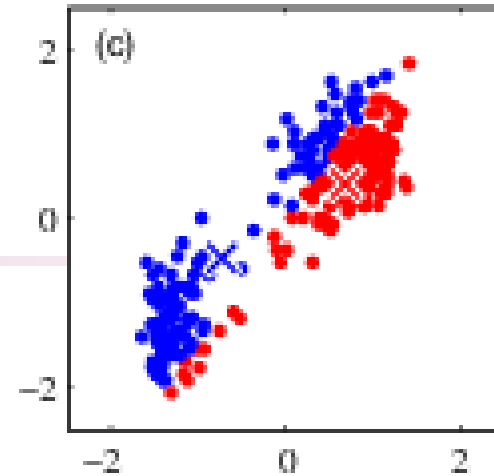
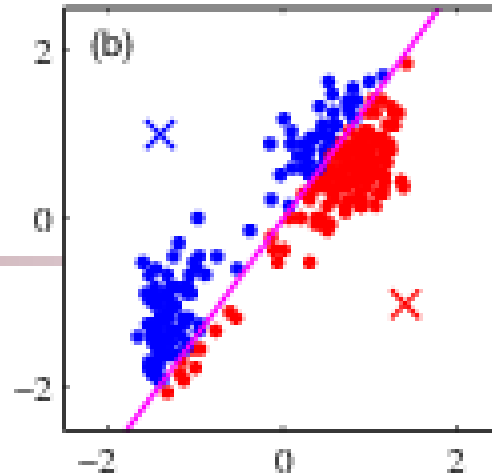
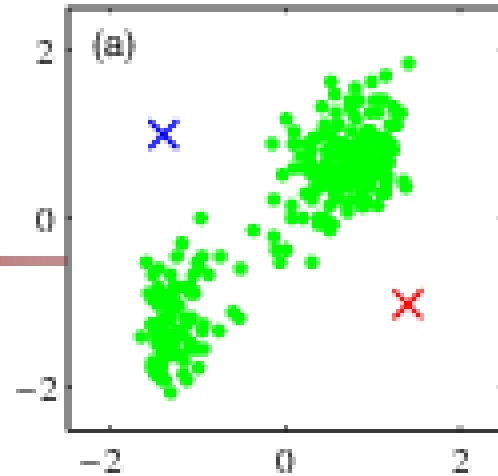


- Partition objects into k nonempty subsets
- Repeat
 - Compute centroid (i.e., mean point) for each partition
 - Assign each object to the cluster of its nearest centroid
- Until no change

The K-Means Clustering Method

■ Example



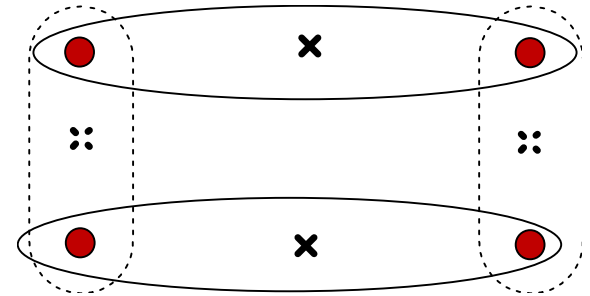


Comments on the K-*Means* Method

- Strength: **Efficient**: $O(tkn)$, where n is # objects, k is # clusters, and t is # iterations. Normally, $k, t \ll n$.
 - Comparing: PAM: $O(k(n-k)^2)$, CLARA: $O(ks^2 + k(n-k))$
- Comment: Often terminates at a *local optimal*.
- Weakness
 - Applicable only to objects in a continuous n -dimensional space
 - Using the k-modes method for categorical data
 - In comparison, k-medoids can be applied to a wide range of data
 - Need to specify k , the **number** of clusters, in advance (there are ways to automatically determine the best k (see Hastie et al., 2009))
 - Sensitive to noisy data and **outliers**
 - Not suitable to discover clusters with **non-convex** shapes

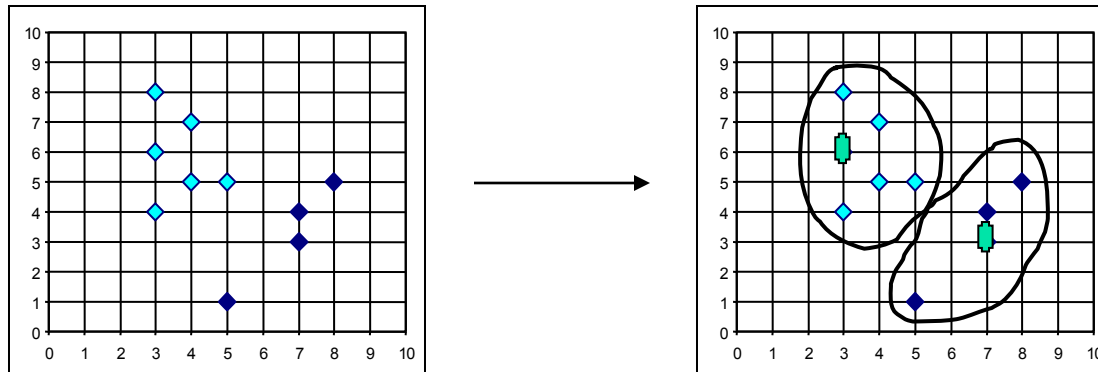
Variations of the *K-Means* Method

- Most of the variants of the *k-means* which differ in
 - Selection of the initial *k* means
 - Dissimilarity calculations
 - Strategies to calculate cluster means
- Handling categorical data: *k-modes*
 - Replacing means of clusters with modes
 - Using new dissimilarity measures to deal with categorical objects
 - Using a frequency-based method to update modes of clusters
 - A mixture of categorical and numerical data: *k-prototype* method



What Is the Problem of the K-Means Method?

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



The *K-Medoids* Clustering Method

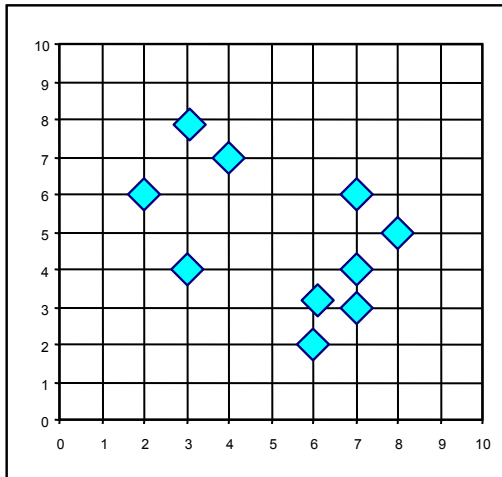
- Find **representative** objects, called medoids, in clusters
- **PAM** (Partitioning Around Medoids, 1987)
 - starts from an initial set of medoids and iteratively replaces one of the medoids by one of the nonmedoids if it improves the total distance of the resulting clustering

PAM (Partitioning Around Medoids) (1987)

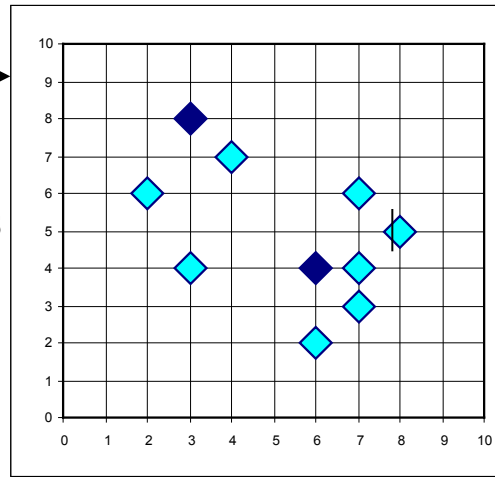
- PAM (Kaufman and Rousseeuw, 1987), built in Splus
- Uses real object to represent the cluster
 1. Select k representative objects arbitrarily
 2. For each pair of non-selected object h and selected object i , calculate the total swapping cost TC_{ih}
 3. For each pair of i and h ,
 - If $TC_{ih} = \sum_j C_{jih} < 0$, i is replaced by h
 - Assign each non-selected object to the most similar representative object
 4. Repeat steps 2-3 until there is no change

PAM: A Typical K-Medoids Algorithm

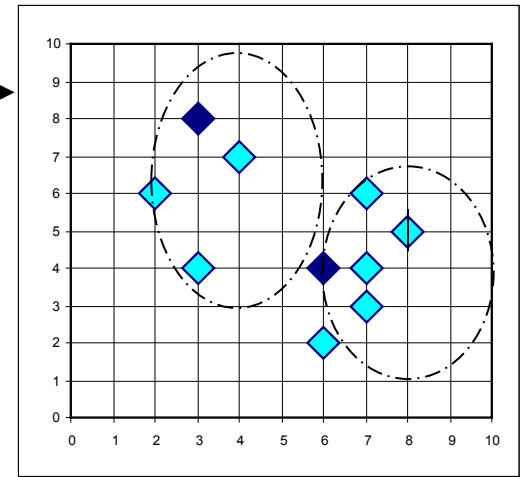
Total Cost = 20



Arbitrary
choose k
object as
initial
medoids



Assign
each
remainin
g object
to
nearest
medoids

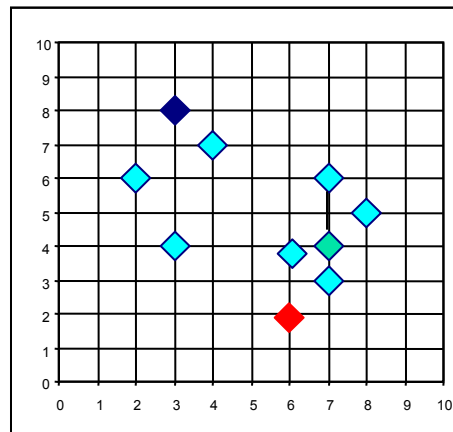


K=2

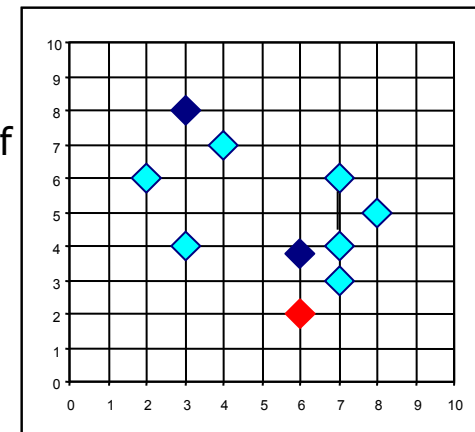
**Do loop
Until no
change**

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

Total Cost = 26



Compute
total cost of
swapping



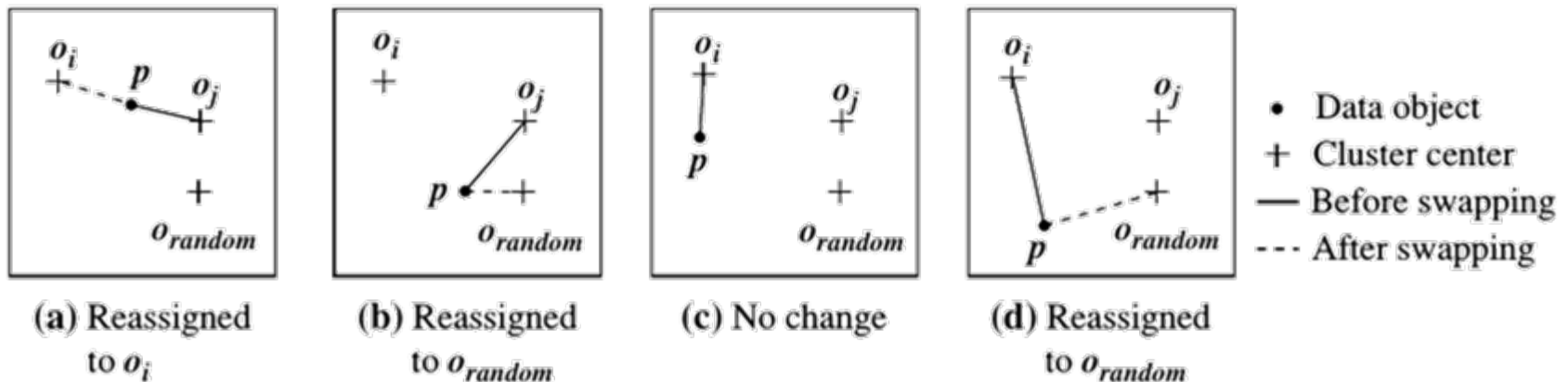
K-Medoids: k 中心点方法

- 基本思想
 - 首先随意选择初始代表对象
 - 用非代表对象替换代表对象，只要能提高聚类质量，就重复这一过程
 - 聚类质量用代价函数度量

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

K-Medoids: k 中心点方法

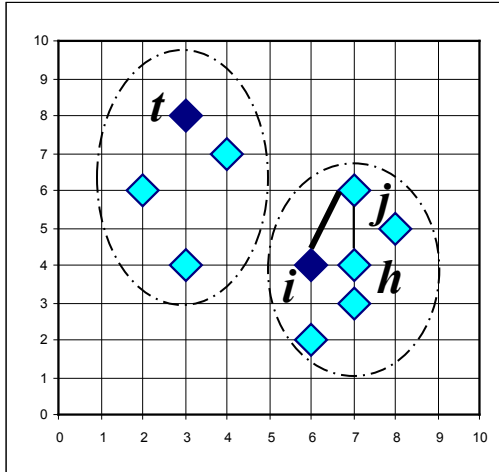
- 非代表对象 o_{random} 代替代表对象 o_j 后, 对于每个非代表对象 p , 可能存在以下四种情况之一
 - p 当前隶属于对象 o_j , 但 o_j 被 o_{random} 代替, 并且 p 离其它代表对象 $o_i (i \neq j)$ 最近, 则 p 重新分配给 o_i
 - p 当前隶属于对象 o_j , 但 o_j 被 o_{random} 代替, 并且 p 离 o_{random} 最近, 则 p 重新分配给 o_{random}
 - p 当前隶属于代表对象 o_i , $i \neq j$, o_j 被 o_{random} 代替, 并且 p 仍然离 o_i 最近, 则对象 p 的隶属关系不变
 - p 当前隶属于代表对象 o_i , $i \neq j$, o_j 被 o_{random} 代替, 并且 p 离 o_{random} 最近, 则 p 重新分配给 o_{random}



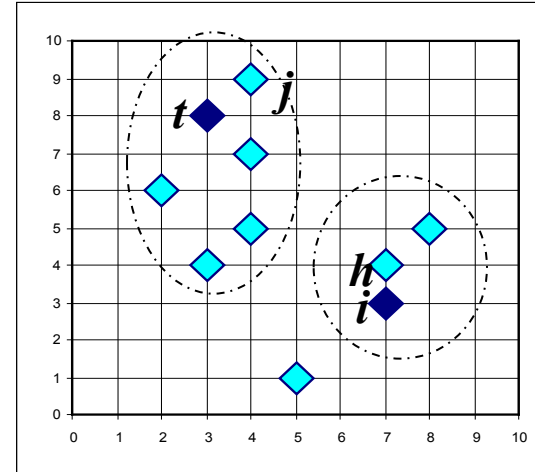
K-Medoids: k 中心点方法

- PAM: Partitioning Around Medoids
- 算法: k 中心点: PAM
- 输入: k : 聚类的数目
 D : 包含 n 个对象的数据集
- 方法:
 - 从 D 中任意选择 k 个对象作为初始的代表对象
 - repeat
 - 将其余的每个对象指派到最近的代表对象所在簇
 - 随机选一个非代表对象 o_{random}
 - 计算用 o_{random} 交换代表对象 o_j 造成的 E 的增量 S
 - if $S < 0$, then 用 o_{random} 替换 o_j , 形成新的 k 个代表对象
 - until 不发生变化

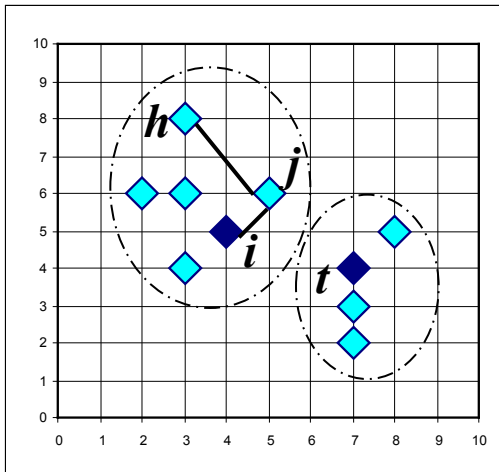
PAM Clustering: Total swapping cost $TC_{ih} = \sum_j C_{jih}$



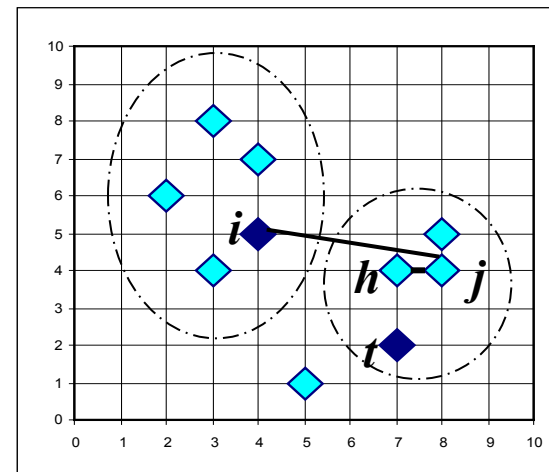
$$C_{jih} = d(j, h) - d(j, i)$$



$$C_{jih} = 0$$



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



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

What Is the Problem with PAM?

- PAM is more robust than k-means in the presence of noise and outliers because a medoid is less influenced by outliers or other extreme values than a mean
- PAM works efficiently for small data sets but does **not scale well** for large data sets.
 - $O(k(n-k)^2)$ for each iteration
where n is # of data, k is # of clusters
- Efficiency improvement on PAM
 - **CLARA** (Kaufmann & Rousseeuw, 1990): PAM on samples
 - **CLARANS** (Ng & Han, 1994): Randomized resampling

CLARA (Clustering Large Applications) (1990)

- **CLARA** (Kaufmann and Rousseeuw in 1990)
 - Built in statistical analysis packages, such as S+
- It draws *multiple samples* of the data set, applies *PAM* on each sample, and gives the best clustering as the output
- Strength: deals with larger data sets than *PAM*
- Weakness:
 - Efficiency depends on the sample size
 - A good clustering based on samples will not necessarily represent a good clustering of the whole data set if the sample is biased

CLARANS ("Randomized" CLARA) (1994)

- *CLARANS* (A Clustering Algorithm based on Randomized Search) (Ng and Han'94)
- CLARANS draws sample of neighbors dynamically
- The clustering process can be presented as searching a graph where every node is a potential solution, that is, a set of k medoids


CLARANS ("Randomized" CLARA) (1994)

- randomly select k objects in the data set as the current medoids
- randomly selects a current medoid x and a non-medoid object y , replace x by y if the replacement can improve the absolute-error criterion
- repeat such a randomized search l times, then the set of the current medoids is considered a local optimum
- repeat this randomized process m times and return the best local optimal as the final result

CLARANS ("Randomized" CLARA) (1994)

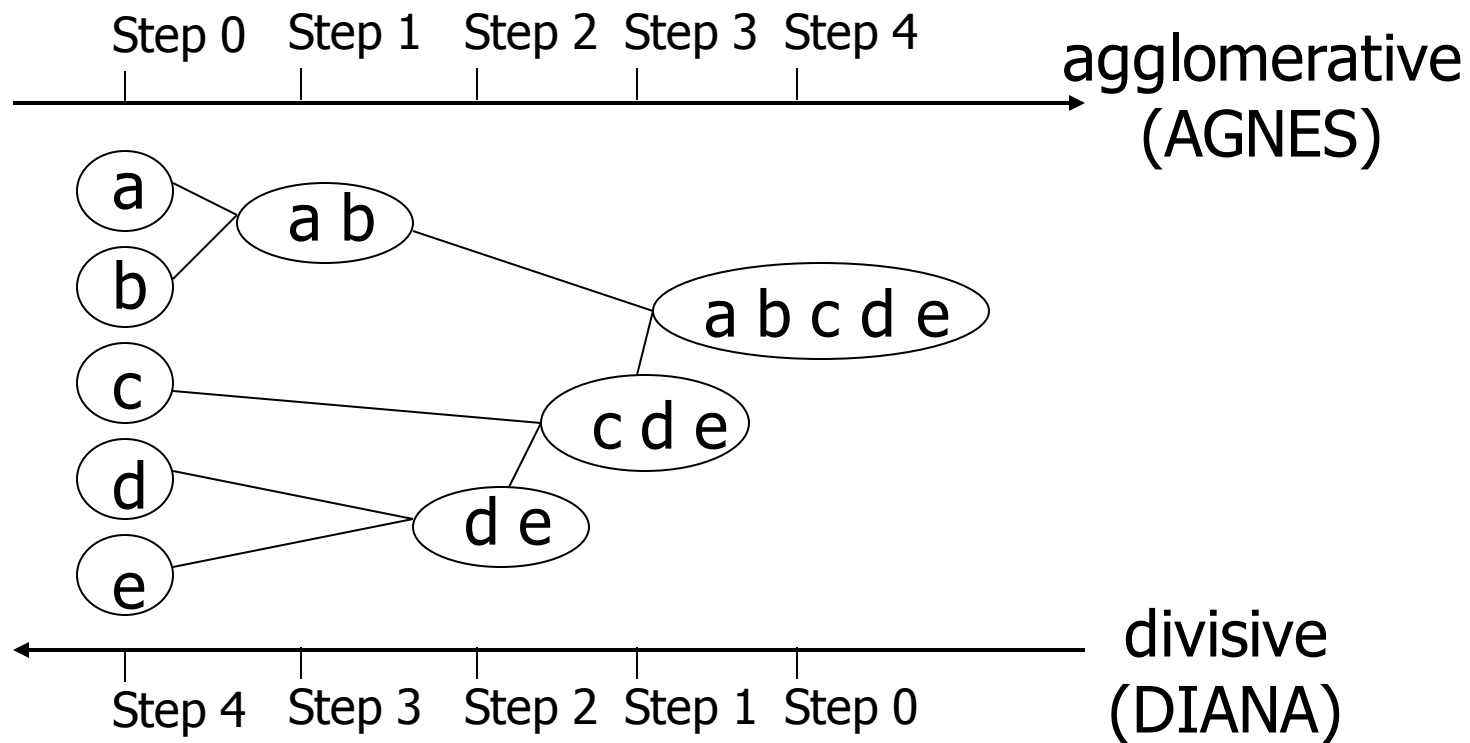
- It is more efficient and scalable than both *PAM* and *CLARA*
- Focusing techniques and spatial access structures may further improve its performance (Ester et al.'95)

Chapter 10. Cluster Analysis: Basic Concepts and Methods

- Cluster Analysis: Basic Concepts
- Partitioning Methods
-  ■ Hierarchical Methods
- Density-Based Methods
- Grid-Based Methods
- Evaluation of Clustering
- Summary

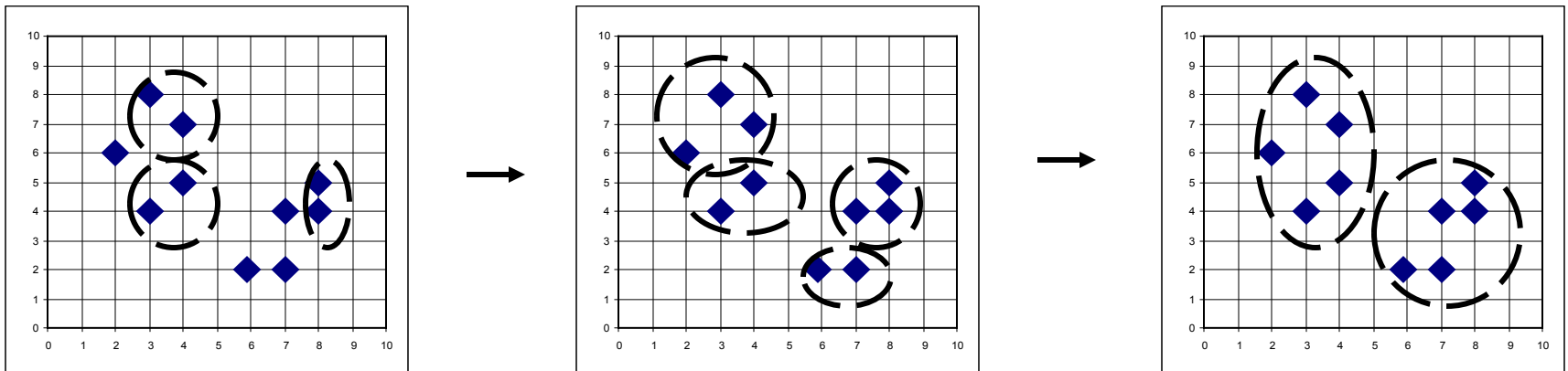
Hierarchical Clustering

- Use distance matrix as clustering criteria. This method does not require the number of clusters k as an input, but needs a termination condition



AGNES (Agglomerative Nesting)

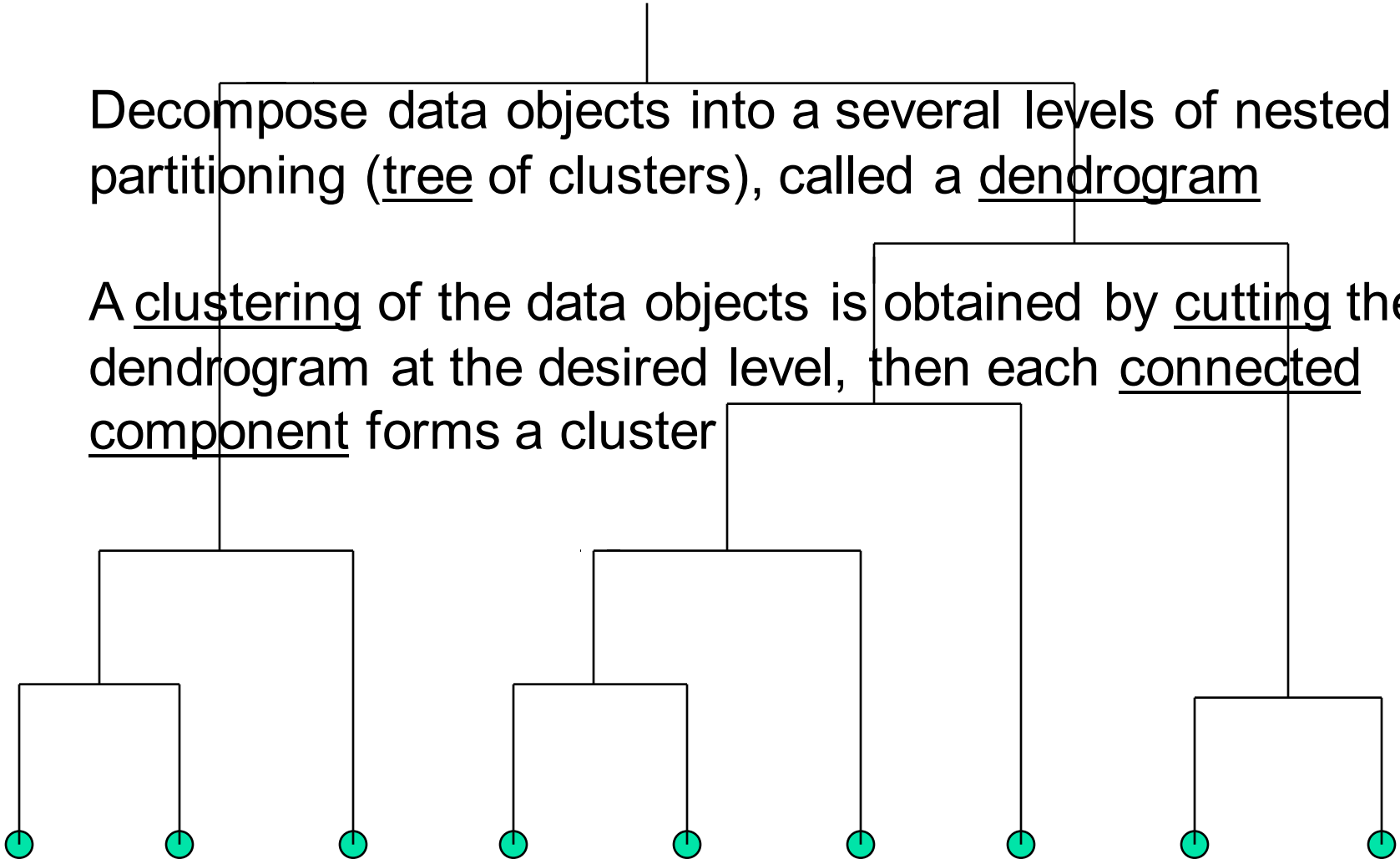
- Introduced in Kaufmann and Rousseeuw (1990)
- Implemented in statistical packages, e.g., Splus
- Use the **single-link** method and the dissimilarity matrix
- Merge nodes that have the least dissimilarity
- Go on in a non-descending fashion
- Eventually all nodes belong to the same cluster



Dendrogram: Shows How Clusters are Merged

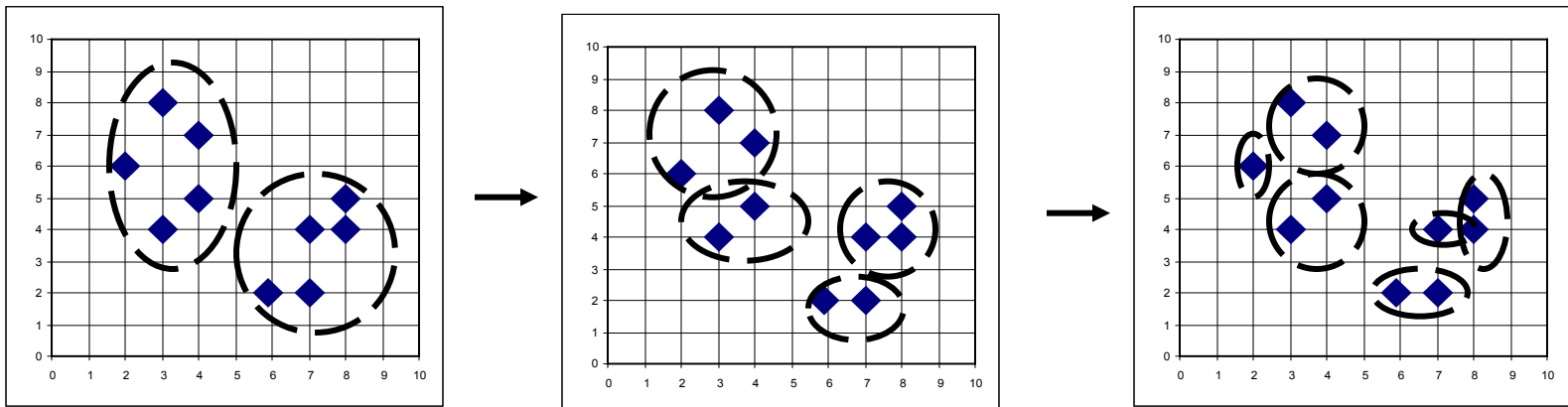
Decompose data objects into a several levels of nested partitioning (tree of clusters), called a dendrogram

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

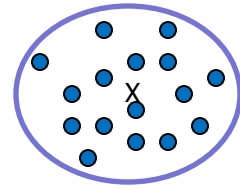
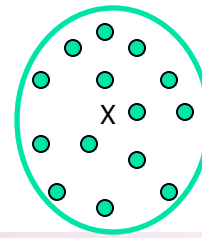


DIANA (Divisive Analysis)

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



Distance between Clusters



- **Single link:** smallest distance between an element in one cluster and an element in the other, i.e., $\text{dist}(K_i, K_j) = \min(t_{ip}, t_{jq})$
- **Complete link:** largest distance between an element in one cluster and an element in the other, i.e., $\text{dist}(K_i, K_j) = \max(t_{ip}, t_{jq})$
- **Average:** avg distance between an element in one cluster and an element in the other, i.e., $\text{dist}(K_i, K_j) = \text{avg}(t_{ip}, t_{jq})$
- **Centroid:** distance between the centroids of two clusters, i.e., $\text{dist}(K_i, K_j) = \text{dist}(C_i, C_j)$
- **Medoid:** distance between the medoids of two clusters, i.e., $\text{dist}(K_i, K_j) = \text{dist}(M_i, M_j)$
 - Medoid: a chosen, centrally located object in the cluster

Centroid, Radius and Diameter of a Cluster (for numerical data sets)

- Centroid: the "middle" of a cluster

$$x_0 = \frac{\sum_{i=1}^N x_i}{N}$$

- Radius: square root of average distance from any point of the cluster to its centroid

$$R_m = \sqrt{\frac{\sum_{i=1}^N (x_i - x_0)^2}{N}}$$

- Diameter: square root of average mean squared distance between all pairs of points in the cluster

$$D_m = \sqrt{\frac{\sum_{i=1}^N \sum_{j=1}^N (x_i - x_j)^2}{N(N-1)}}$$

Extensions to Hierarchical Clustering

- Major weakness of agglomerative 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
- Integration of hierarchical & distance-based clustering
 - BIRCH (1996): uses CF-tree and incrementally adjusts the quality of sub-clusters
 - CHAMELEON (1999): hierarchical clustering using dynamic modeling

BIRCH (Balanced Iterative Reducing and Clustering Using Hierarchies)

- 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
- *Scales linearly*: finds a good clustering with a single scan and improves the quality with a few additional scans
- *Weakness*: handles only numeric data, and sensitive to the order of the data record

The Birch Algorithm

- Centroid Euclidian distance
- Centroid Manhattan distance
- Average inter-cluster distance
- Average intra-cluster distance
- Variance increase distance

$$D_0 = \left(\left(\overrightarrow{X0_1} - \overrightarrow{X0_2} \right) \right)^{\frac{1}{2}}$$

$$D_1 = \left| \overrightarrow{X0_1} - \overrightarrow{X0_2} \right| = \sum_{i=1}^d \left| \overrightarrow{X0_1^{(i)}} - \overrightarrow{X0_2^{(i)}} \right|$$

$$D_2 = \left(\frac{\sum_{i=1}^{N_1} \sum_{j=N_1+1}^{N_1+N_2} \left(\overrightarrow{X_i} - \overrightarrow{X_j} \right)^2}{N_1 N_2} \right)^{\frac{1}{2}}$$

$$D_3 = \left(\frac{\sum_{i=1}^{N_1+N_2} \sum_{j=1}^{N_1+N_2} \left(\overrightarrow{X_i} - \overrightarrow{X_j} \right)^2}{(N_1 + N_2)(N_1 + N_2 - 1)} \right)^{\frac{1}{2}}$$

$$D_4 = \sum_{k=1}^{N_1+N_2} \left(\overrightarrow{X_k} - \frac{\sum_{l=1}^{N_1+N_2} \overrightarrow{X_l}}{N_1 + N_2} \right)^2 - \sum_{i=1}^{N_1} \left(\overrightarrow{X_i} - \frac{\sum_{l=1}^{N_1} \overrightarrow{X_l}}{N_1} \right)^2 - \sum_{j=N_1+1}^{N_1+N_2} \left(\overrightarrow{X_j} - \frac{\sum_{l=N_1+1}^{N_1+N_2} \overrightarrow{X_l}}{N_2} \right)^2$$

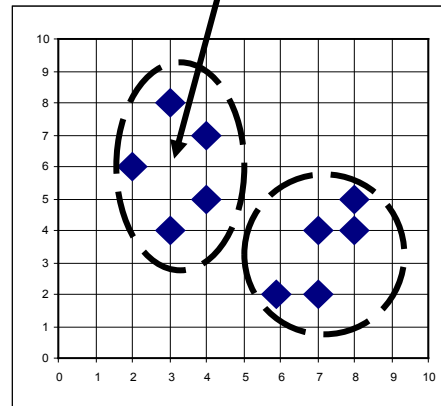
Clustering Feature Vector in BIRCH

Clustering Feature: $CF = (N, \vec{LS}, SS)$

N : Number of data points

$$LS: \sum_{i=1}^N \vec{X}_i$$

$$SS: \sum_{i=1}^N \vec{X}_i^2$$



$$CF = (5, (16, 30), (54, 190))$$

(3,4)

(2,6)

(4,5)

(4,7)

(3,8)

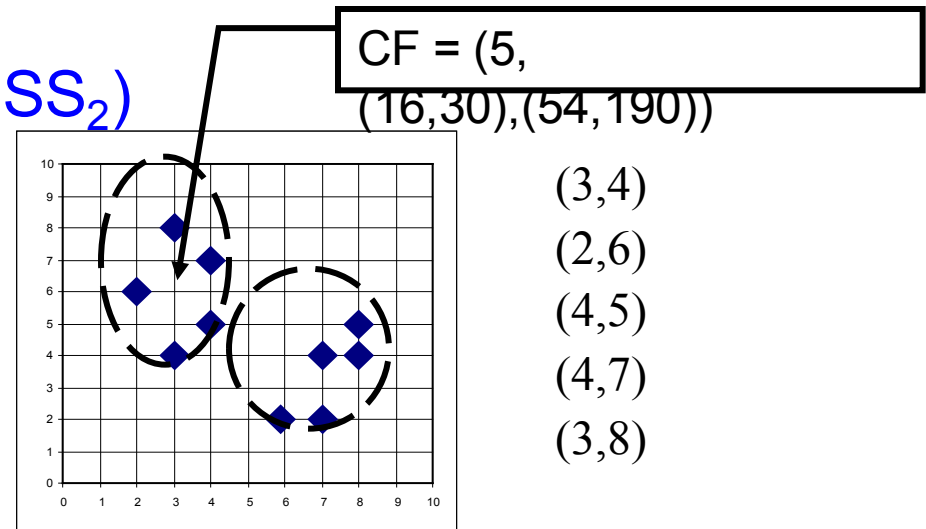
Clustering Feature Vector in BIRCH

Clustering Features can be added

- Two clusters with $CF_1=(N_1, \vec{LS}_1, SS_1)$, $CF_2=(N_2, \vec{LS}_2, SS_2)$
- Merge these two clusters into one, whose clustering Feature can be calculated as

$$CF = CF_1 + CF_2$$
$$= (N_1 + N_2, \vec{LS}_1 + \vec{LS}_2, SS_1 + SS_2)$$

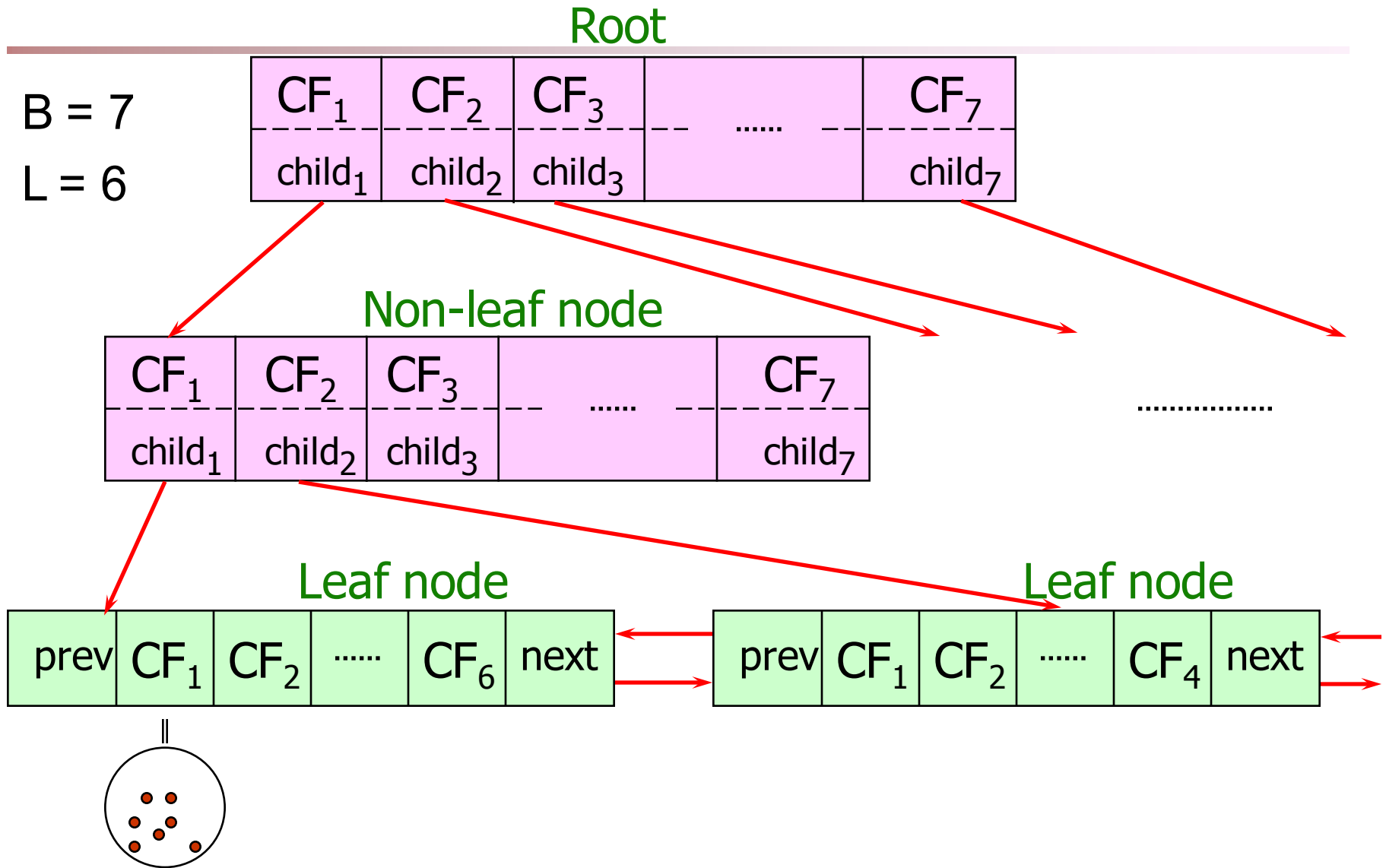
- After merge, new cluster's \vec{x}_0 , R , D , D_0 , D_1 , D_2 , D_3 , D_4 can be calculated easily



CF-Tree in BIRCH

- Clustering feature:
 - Summary of the statistics for a given subcluster: the 0-th, 1st, and 2nd moments of the subcluster from the statistical point of view
 - Registers crucial measurements for computing cluster and utilizes storage efficiently
- A CF tree is a height-balanced tree that stores the clustering features for a hierarchical clustering
 - A nonleaf node in a tree has descendants or “children”
 - The nonleaf nodes store sums of the CFs of their children
- A CF tree has two parameters
 - Branching factor: max # of children
 - Threshold: max diameter of sub-clusters stored at the leaf nodes

The CF Tree Structure



CF Tree construction

- The CF tree is built dynamically as new data objects are inserted
- The way of inserting a new data object into a CF tree is just the same as the insertion into a B+-tree
- Steps
 - Identify the appropriate leaf
 - Start with CF list at root node, find the closest cluster (by using CF values and distance measures D_0 , D_1 , D_2 , D_3 , and D_4)
 - Check all the children of the cluster, find the closest
 - And so on, until reach a leaf node
 - Modify the leaf
 - Find the closest leaf entry and test whether it can absorb new entry without violating threshold condition or not
 - If not, add new entry to leaf
 - Leaves have a max size, may need to be split
 - Modify the path
 - once the point has been added, must update the CF of all ancestors

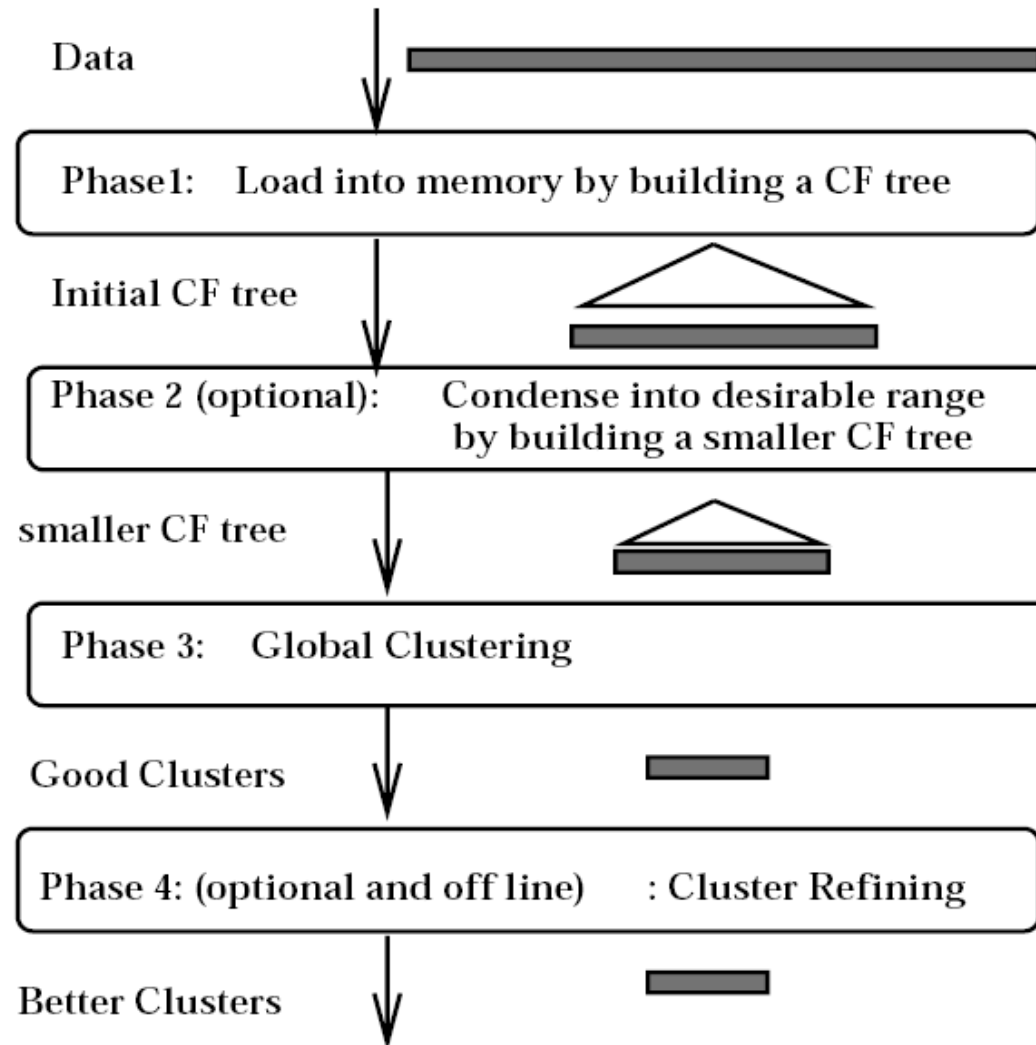
The Birch Algorithm

- Cluster Diameter

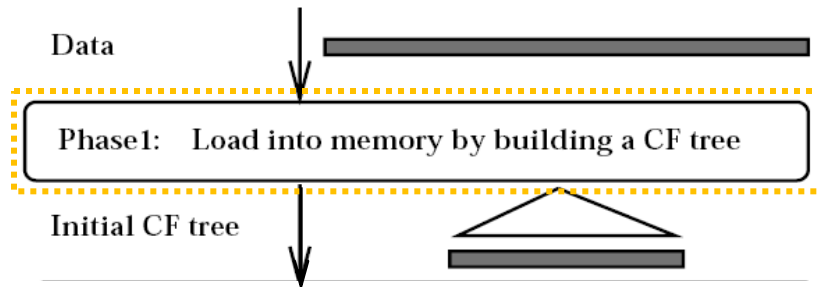
$$\sqrt{\frac{1}{n(n-1)} \sum (x_i - x_j)^2}$$

- For each point in the input
 - Find closest leaf entry
 - Add point to leaf entry and update CF
 - If entry diameter > max_diameter, then split leaf, and possibly parents

Phases of BIRCH algorithm

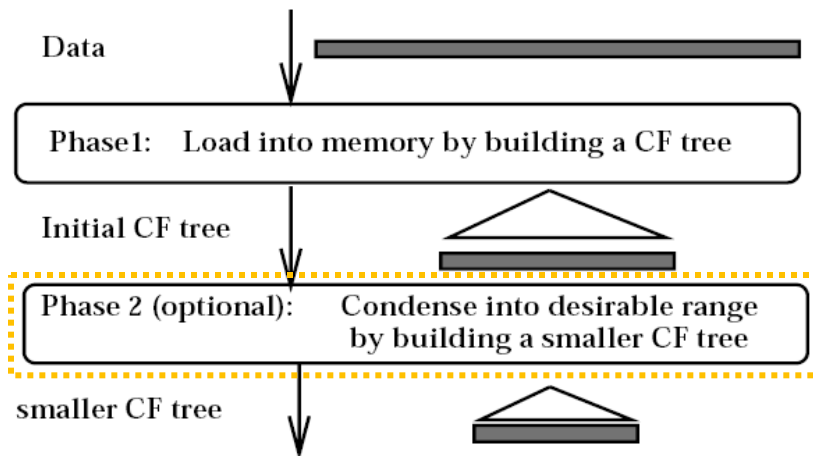


Phases of BIRCH algorithm



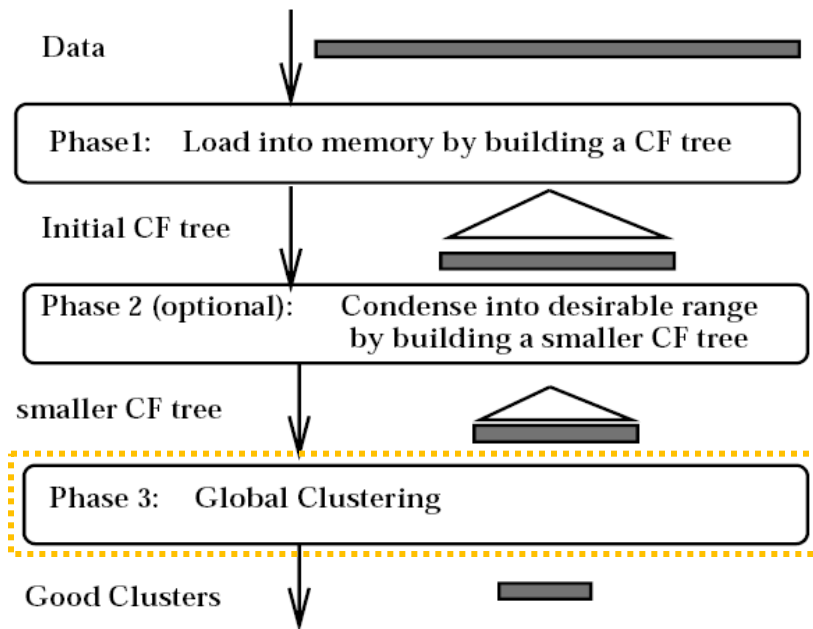
- Start with initial threshold T and insert points into tree
- The tree is built in the way of B+-tree construction alike
- If it runs out of memory, increase T and rebuild the tree
 - Reinsert leaf entries from old tree into new tree
 - remove outliers
- After phase 1:
 - data "reduced" to fit in memory
 - subsequent processing occurs entirely in memory (no I/O)

Phases of BIRCH algorithm



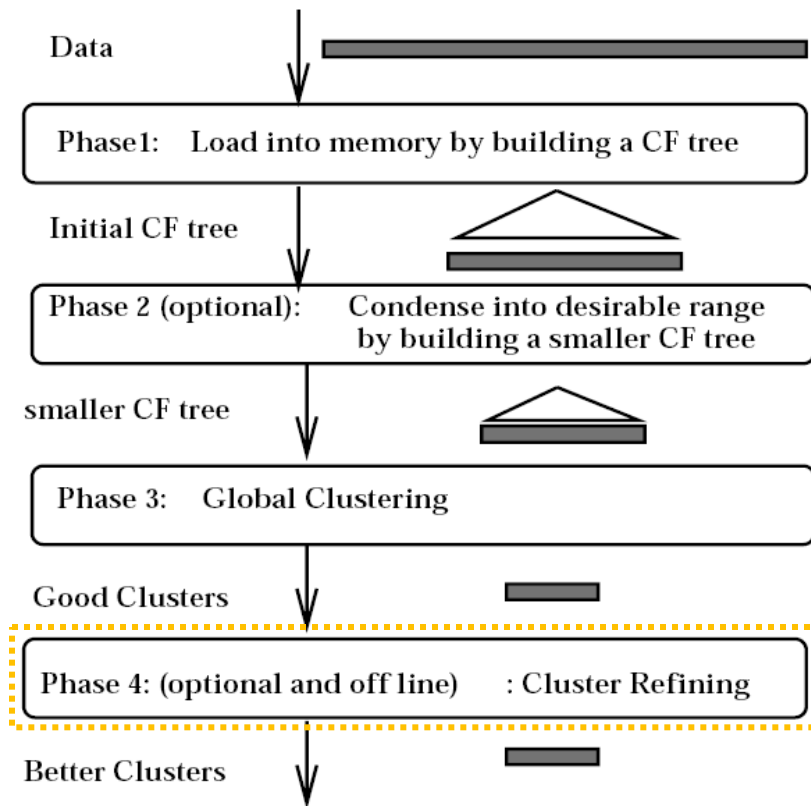
- Optional
- # of clusters produced in Phase 1 may be not be suitable for algorithms used in Phase 3
- Shrink tree as necessary
 - scan the leaf entries in the initial CF tree to rebuild a smaller CF tree
 - remove more outliers
 - crowded subclusters are merged

Phases of BIRCH algorithm



- Problems after Phase 1
 - Input order affect results
 - splitting triggered
 - With the CF known
 - treat each subcluster as a single data point by calculating the centroid as its representative
- treat a subcluster of n data points as its centroid repeating n times, modify an existing algorithm slightly to taking the counting information into account
 - apply an existing algorithm directly to the subclusters using the information of CF vectors
 - $D_0, D_1, D_2, D_3, D_4, \dots$
 - clustering algorithm: K-Means, K-Medoids, ...

Phases of BIRCH algorithm



- Optional
- Scan through data again and assign each data point to a cluster
 - choose cluster whose centroid is closest
- This redistributes data points amongst clusters in more accurate fashion than original CF cluster
- Can be repeated for improved refinement of clusters

The Birch Algorithm

- Algorithm is $O(n)$
- Concerns
 - Sensitive to insertion order of data points
 - Since we fix the size of leaf nodes, so clusters may not be so natural
 - Clusters tend to be spherical given the radius and diameter measures

Clustering Categorical Data: The ROCK Algorithm

- ROCK: RObust Clustering using linkS
 - S. Guha, R. Rastogi & K. Shim, ICDE'99
- Major ideas
 - Use links to measure similarity/proximity
 - Not distance-based
 - Computational complexity: $O(n^2 + nm_m m_a + n^2 \log n)$
- Algorithm: sampling-based clustering
 - Draw random sample
 - Cluster with links
 - Label data in disk
- Experiments
 - Congressional voting, mushroom data

Problem of partitional clustering when categorical data

- partitional clustering algorithms try to optimize the criterion function

$$E = \sum_{i=1}^k \sum_{\vec{x} \in C_i} d(\vec{x}, \vec{m}_i)$$

where \vec{m}_i is the centroid of cluster C_i

- for numerical data, it works well
- for categorical data, the distance between the cluster centroids may lead to unsatisfactory results

Example

- market basket database
 - the number of items — the number of attributes in the database is very large while the size of an average transaction is much smaller
 - customers with similar buying patterns and belonging to a single cluster, may buy a small subset of items from a much larger set that defines the cluster
 - a pair of transactions in a cluster may have few items in common
 - when the set of items that define clusters have not uniform sizes, the above situation is further exacerbated
 - it tends to split large clusters

Problem of hierarchical method

- Consider the centroid-based agglomerative hierarchical clustering algorithm as an example
- For categorical attributes, distances between centroids of clusters is a poor estimate of the similarity between them
- Example
 - Consider a market basket database containing the following 4 transactions over items 1, 2, 3, 4, 5 and 6
a: {1, 2, 3, 5}, b: {2, 3, 4, 5}, c: {1, 4}, and d: {6}.
The transactions can be viewed as points with 0 1 attributes corresponding to the items 1, 2, 3, 4, 5 and 6. The four points thus become a: {1,1,1,0,1,0}, b: {0,1,1,1,1,0}, c: {1,0,0,1,0,0} and d: {0,0,0,0,0,1}.

Problem of hierarchical method

$$d(a, b) = \sqrt{2} \quad d(a, c) = \sqrt{3} \quad d(a, d) = \sqrt{6}$$

$$d(b, c) = \sqrt{5} \quad d(b, d) = \sqrt{5} \quad d(c, d) = \sqrt{3}$$

- **e=a ∪ b**, its centroid: {0.5,1,1,0.5,1,0}

$$d(c, e) = \sqrt{3.5} \quad d(d, e) = \sqrt{4.5} \quad d(c, d) = \sqrt{3}$$

- **f=c ∪ d, it is not rational**
- As the cluster size grows, # of attributes appearing in the mean go up, and their value in the mean decreases
- It's difficult to distinguish the difference between two points that differ on few attributes, or two points that differ on every attribute by small amounts

Problem of hierarchical method

- Example

- $m_1: \{1/3, 1/3, 1/3, 0, 0, 0\}$ $m_2: \{0, 0, 0, 1/3, 1/3, 1/3\}$

- $p: \{1, 1, 1, 0, 0, 0\}$

$$d(m_1, m_2) = \sqrt{2/3}$$

$$d(p, m_1) = \sqrt{4/3}$$

$$d(p, m_2) = \sqrt{10/3}$$

- merge m_1 with m_2 , its centroid:

$$\{1/6, 1/6, 1/6, 1/6, 1/6, 1/6\}$$

Similarity Measure in ROCK

- Set theoretic similarity measures such as the Jaccard coefficient have often been used

$$J(T_i, T_j) = \frac{|T_i \cap T_j|}{|T_i \cup T_j|}$$

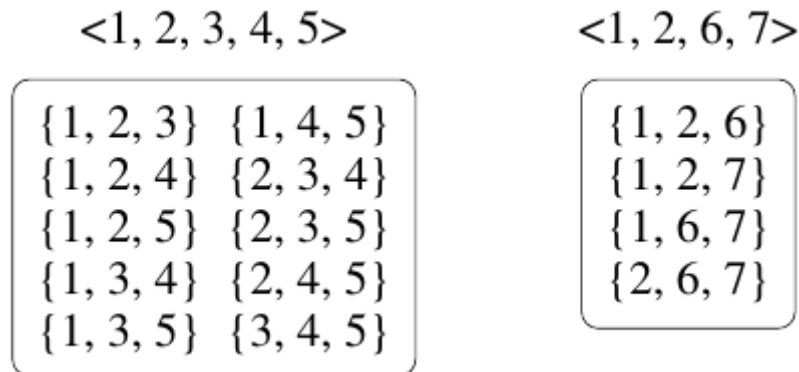
- With the Jaccard coefficient as the distance measure between clusters, centroid-based hierarchical clustering schemes cannot be used since the similarity measure is non-metric, and defined for only points in the cluster and not for its centroid.
- Thus, we have to use either the **minimum spanning tree MST** hierarchical clustering algorithm or hierarchical clustering with **group average**
- Take MST as an example: MST is fragile

Similarity Measure in ROCK

- the Jaccard coefficient is a measure of the similarity between only the two points, it thus does not reflect the properties of the neighborhood of the points
- Consequently, the Jaccard coefficient fails to capture the natural clustering of not so "well-separated" data sets with categorical attributes

Similarity Measure in ROCK

- Example: Two groups (clusters) of transactions



- $\text{sim}(\{1, 2, 3\}, \{3, 4, 5\}) = 0.2$, $\{1, 2, 3\}$ and $\{3, 4, 5\}$ are in the same cluster
- $\text{sim}(\{1, 2, 3\}, \{1, 2, 6\}) = 0.5$, but $\{1, 2, 3\}$ and $\{1, 2, 6\}$ are in the different clusters

Link Measure in ROCK

- Neighbors:

- a pair of points p_i, p_j are defined to be neighbors if

$$\text{sim}(p_i, p_j) \geq \theta$$

where θ is a user-given threshold, and **sim()** can be any similarity metrics, or even be non-metric similarity function provided by expert

Link Measure in ROCK

- Categorical data handling
 - Categorical data typically is of fixed dimension
 - Corresponding to every attribute A and value v in its domain, introduce an item $A.v$
 - A transaction T_i for a record contains $A.v$ if and only if the value of attribute A in the record is v
 - If the value for an attribute is missing in the record, then the corresponding transaction does not contain items for the attribute, that is equivalent to ignore the missing value

Link Measure in ROCK

- **Links:** # of common neighbors
 - $C_1 \langle a, b, c, d, e \rangle$: $\{a, b, c\}$, $\{a, b, d\}$, $\{a, b, e\}$, $\{a, c, d\}$, $\{a, c, e\}$, $\{a, d, e\}$, $\{b, c, d\}$, $\{b, c, e\}$, $\{b, d, e\}$, $\{c, d, e\}$
 - $C_2 \langle a, b, f, g \rangle$: $\{a, b, f\}$, $\{a, b, g\}$, $\{a, f, g\}$, $\{b, f, g\}$
- Let $T_1 = \{a, b, c\}$, $T_2 = \{c, d, e\}$, $T_3 = \{a, b, f\}$, $\theta=0.5$
 - $\text{link}(T_1, T_2) = 4$, since they have 4 common neighbors
 - $\{a, c, d\}$, $\{a, c, e\}$, $\{b, c, d\}$, $\{b, c, e\}$
 - $\text{link}(T_1, T_3) = 3$, since they have 3 common neighbors
 - $\{a, b, d\}$, $\{a, b, e\}$, $\{a, b, g\}$
- Thus link is a better measure than Jaccard coefficient

Link Measure in ROCK

- Criterion function

- maximize the sum of $\text{link}(p_q, p_r)$ for data point pairs p_q, p_r belonging to a single cluster and at the same time, minimize the sum of $\text{link}(p_q, p_s)$ for p_q, p_s in different clusters

- Criterion function

$$E_l = \sum_{i=1}^k n_i \times \sum_{p_q, p_r \in C_i} \frac{\text{link}(p_q, p_r)}{n_i^{1+2f(\theta)}}$$

where $n_i^{1+2f(\theta)}$ is the expected total number of links in C_i ,
and $f(\theta) = \frac{1 - \theta}{1 + \theta}$.

Link Measure in ROCK

- Goodness measure

$$g(C_i, C_j) = \frac{\text{link}(C_i, C_j)}{(n_i + n_j)^{1+2f(\theta)} - n_i^{1+2f(\theta)} - n_j^{1+2f(\theta)}}$$

- Rock Algorithm

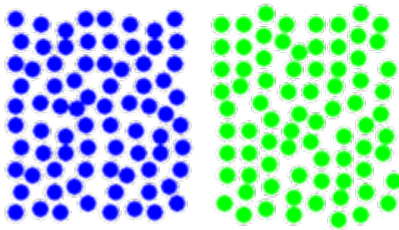


CHAMELEON: Hierarchical Clustering Using Dynamic Modeling (1999)

- CHAMELEON: 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 highly relative to the internal interconnectivity of the clusters and closeness of items within the clusters
- Graph-based, and a 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

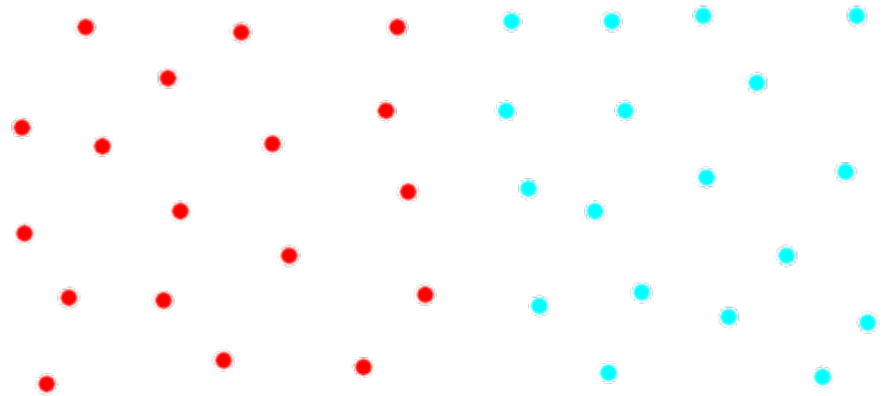
CHAMELEON: Hierarchical Clustering Using Dynamic Modeling (1999)

- Limitations of Existing Hierarchical Schemes
- CURE: **closeness**



(a)

(b)



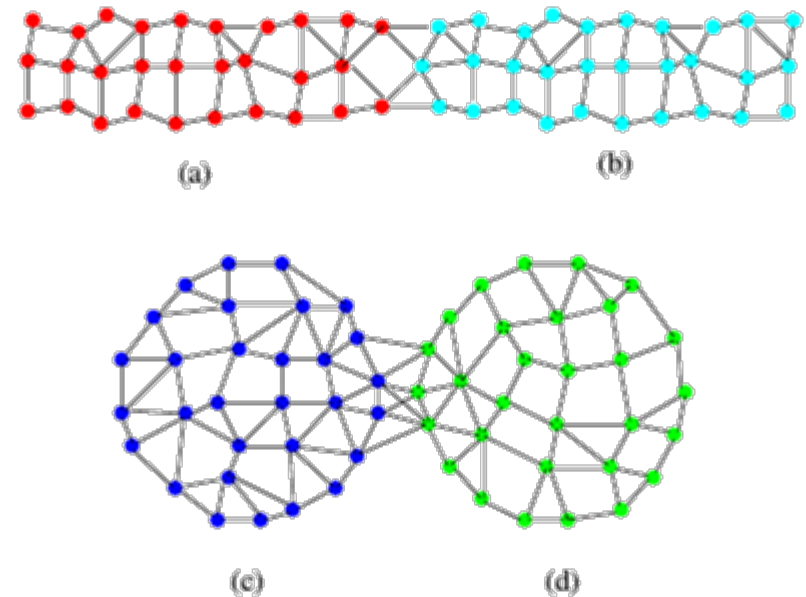
(c)

(d)

- Which pair of clusters will be merged first?
 - **a-b**
 - **c-d**

CHAMELEON: Hierarchical Clustering Using Dynamic Modeling (1999)

- Limitations of Existing Hierarchical Schemes
- ROCK: **inter-connectivity**
grouping average
- Which pair of clusters will be merged first?
 - a-b
 - c-d

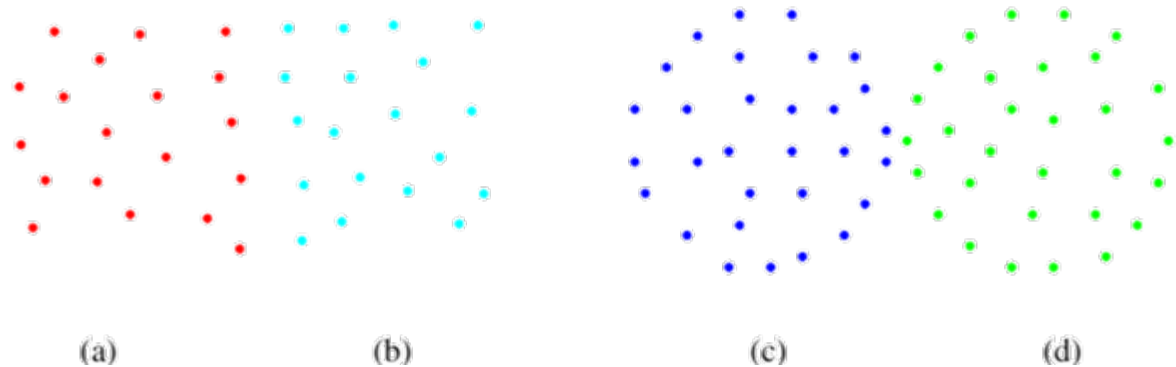


CHAMELEON: Hierarchical Clustering Using Dynamic Modeling (1999)

- Limitations of Existing Hierarchical Schemes

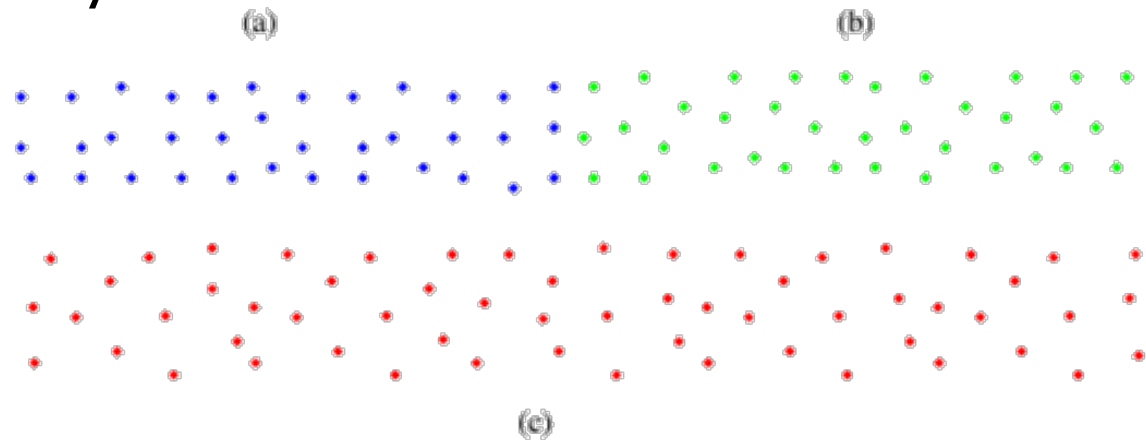
- Closeness

c - d

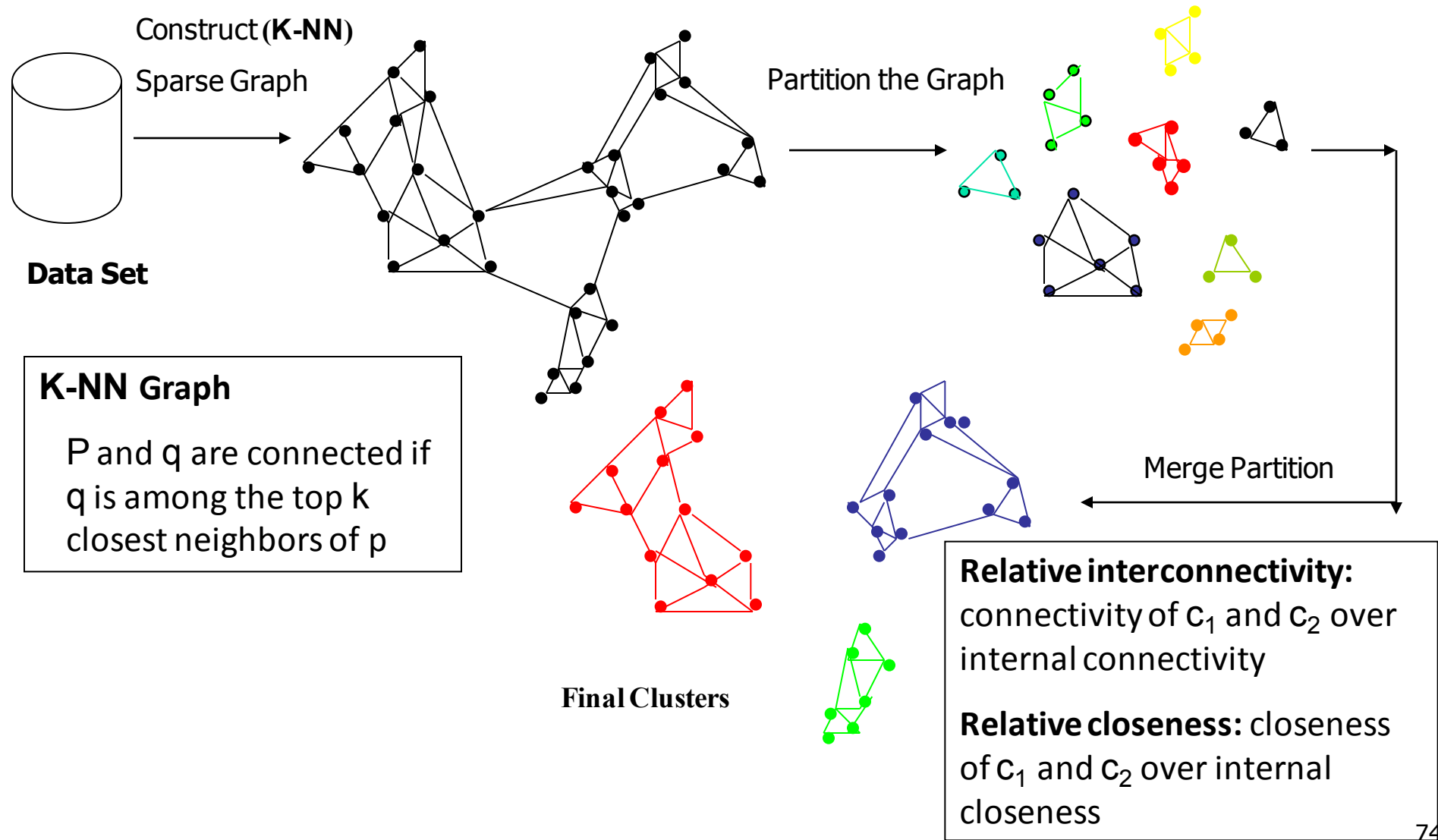


- inter-connectivity

a - c

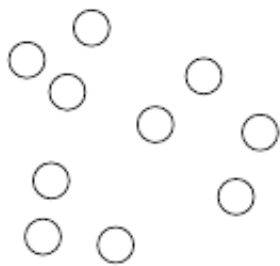


Overall Framework of CHAMELEON

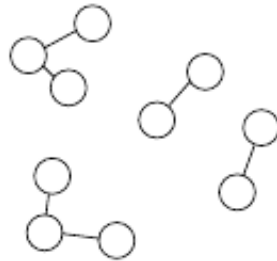


K-NN Graph and Graph Partitioning

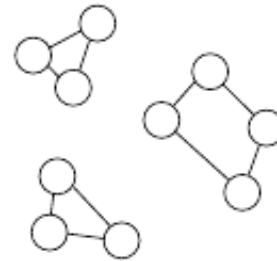
■ K-NN Graph



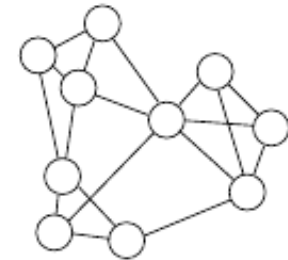
(a) Original Data in 2D



(b) 1-nearest neighbor graph



(c) 2-nearest neighbor graph



(d) 3-nearest neighbor graph

■ Graph Partitioning

- Multilevel graph partitioning algorithm
- Chameleon utilizes the partitioning algorithm that is part of the hMeTiS library

CHAMELEON: Hierarchical Clustering Using Dynamic Modeling (1999)

- Modeling the Cluster Similarity
- Chameleon determines the similarity between each pair of clusters C_i and C_j by looking at their **relative inter-connectivity** $RI(C_i, C_j)$ and their **relative closeness** $RC(C_i, C_j)$

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

$$RI(C_i, C_j) \geq T_{RI}$$

$$RC(C_i, C_j) \geq T_{RC}$$

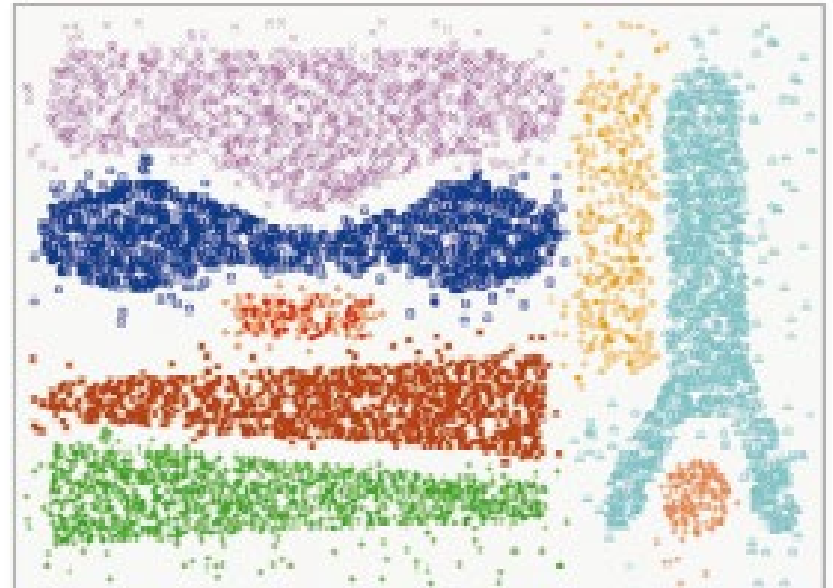
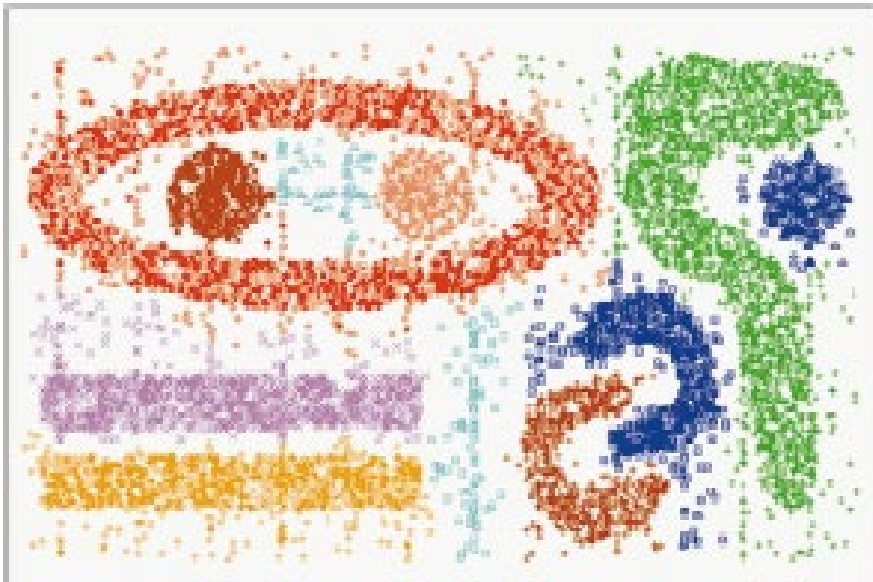
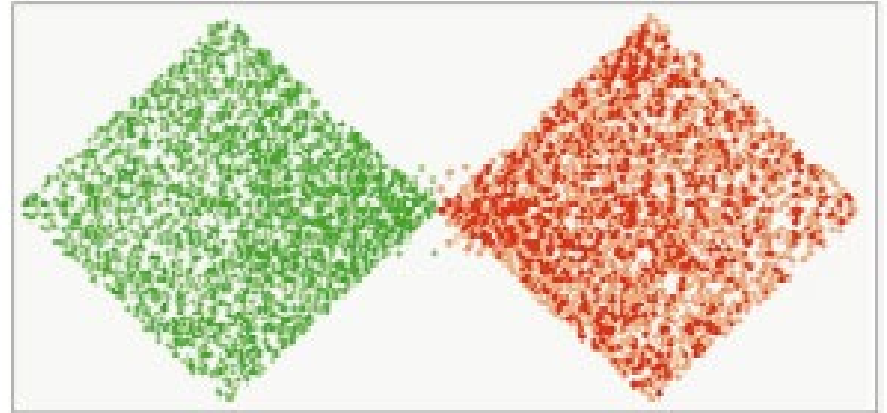
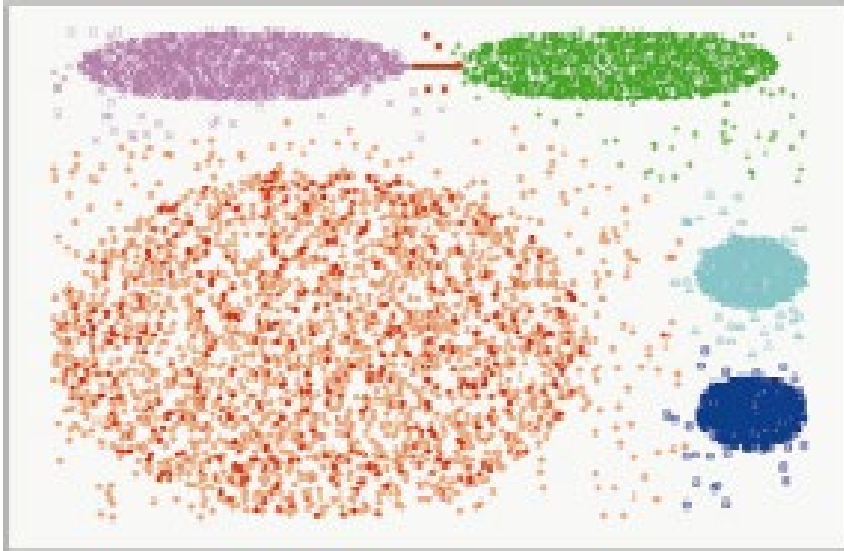
or

$$RI(C_i, C_j) \times RC(C_i, C_j)$$

$$RI(C_i, C_j) \times RC(C_i, C_j)^\alpha$$

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

CHAMELEON (Clustering Complex Objects)



Probabilistic Hierarchical Clustering

- Algorithmic hierarchical clustering
 - Nontrivial to choose a good distance measure
 - Hard to handle missing attribute values
 - Optimization goal not clear: heuristic, local search
- Probabilistic hierarchical clustering
 - Use probabilistic models to measure distances between clusters
 - Generative model: Regard the set of data objects to be clustered as a sample of the underlying data generation mechanism to be analyzed
 - Easy to understand, same efficiency as algorithmic agglomerative clustering method, can handle partially observed data
- In practice, assume the generative models adopt common distributions functions, e.g., Gaussian distribution or Bernoulli distribution, governed by parameters

Generative Model

- Given a set of 1-D points $X = \{x_1, \dots, x_n\}$ for clustering analysis & assuming they are generated by a Gaussian distribution:

$$\mathcal{N}(\mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

- The probability that a point $x_i \in X$ is generated by the model

$$P(x_i | \mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x_i - \mu)^2}{2\sigma^2}}$$

- The likelihood that X is generated by the model:

$$L(\mathcal{N}(\mu, \sigma^2) : X) = P(X | \mu, \sigma^2) = \prod_{i=1}^n \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x_i - \mu)^2}{2\sigma^2}}$$

- The task of learning the generative model: find the parameters μ and σ^2 such that

$$\mathcal{N}(\mu_0, \sigma_0^2) = \arg \max \{ L(\mathcal{N}(\mu, \sigma^2) : X) \}$$

the maximum likelihood

A Probabilistic Hierarchical Clustering Algorithm

- For a set of objects partitioned into m clusters C_1, \dots, C_m , the quality can be measured by $Q(\{C_1, \dots, C_m\}) = \prod_{i=1}^m P(C_i)$

where $P()$ is the maximum likelihood

$$\begin{aligned} & Q\left(\left(\{C_1, \dots, C_m\} - \{C_{j_1}, C_{j_2}\}\right) \cup \{C_{j_1} \cup C_{j_2}\}\right) - Q\left(\{C_1, \dots, C_m\}\right) \\ &= \frac{\prod_{i=1}^m P(C_i) \cdot P(C_{j_1} \cup C_{j_2})}{P(C_{j_1})P(C_{j_2})} - \prod_{i=1}^m P(C_i) \\ &= \prod_{i=1}^m P(C_i) \left(\frac{P(C_{j_1} \cup C_{j_2})}{P(C_{j_1})P(C_{j_2})} - 1 \right) \end{aligned}$$

- Distance between clusters C_1 and C_2 :

$$\text{dist}(C_i, C_j) = -\log \frac{P(C_i \cup C_j)}{P(C_i)P(C_j)}$$

A Probabilistic Hierarchical Clustering Algorithm

- Algorithm: Progressively merge points and clusters

Input: $D = \{o_1, \dots, o_n\}$: a data set containing n objects

Output: A hierarchy of clusters

Method

- (1) Create a cluster for each object $C_i = \{o_i\}$, $1 \leq i \leq n$;
- (2) For $i = 1$ to n {
- (3) Find pair of clusters C_i and C_j such that
- (4) **$C_i, C_j = \operatorname{argmax}_{i \neq j} \{\log(P(C_i \cup C_j)/(P(C_i)P(C_j)))\}$;**
- (5) If **$\log(P(C_i \cup C_j)/(P(C_i)P(C_j))) > 0$** then merge C_i and C_j
- (6) }

Chapter 10. Cluster Analysis: Basic Concepts and Methods

- Cluster Analysis: Basic Concepts
- Partitioning Methods
- Hierarchical Methods
- Density-Based Methods
- Grid-Based Methods
- Evaluation of Clustering
- Summary



Density-Based Clustering Methods

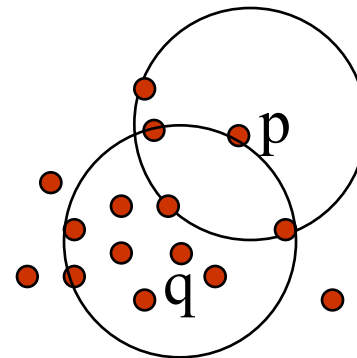
- Clustering based on density (local cluster criterion), such as density-connected points
- Major features
 - Discover clusters of arbitrary shape
 - Handle noise
 - One scan
 - Need density parameters as termination condition
- Several interesting studies
 - [DBSCAN](#): Ester, et al. (KDD'96)
 - [OPTICS](#): Ankerst, et al (SIGMOD'99).
 - [DENCLUE](#): Hinneburg & D. Keim (KDD'98)
 - [CLIQUE](#): Agrawal, et al. (SIGMOD'98) (more grid-based)
 - [CFSFDP](#): Alex Rodriguez and Alessandro Laio. Science 2014

Density-Based Clustering: Basic Concepts

- Two parameters:
 - **Eps**: Maximum radius of the neighbourhood
 - **MinPts**: Minimum number of points in an Eps-neighbourhood of that point
- $N_{Eps}(p): \{q \in D \mid \text{dist}(p, q) \leq Eps\}$
- **Directly density-reachable**: A point p is directly density-reachable from a point q w.r.t. Eps, MinPts if

- p belongs to $N_{Eps}(q)$
- **core point** condition:

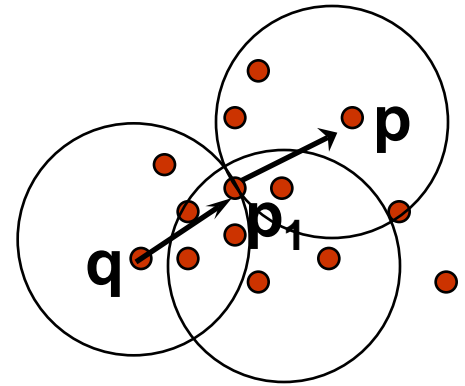
$$|N_{Eps}(q)| \geq \text{MinPts}$$



Density-Reachable and Density-Connected

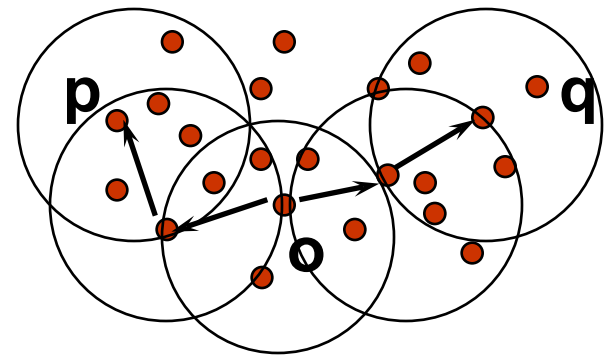
- **Density-reachable:**

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



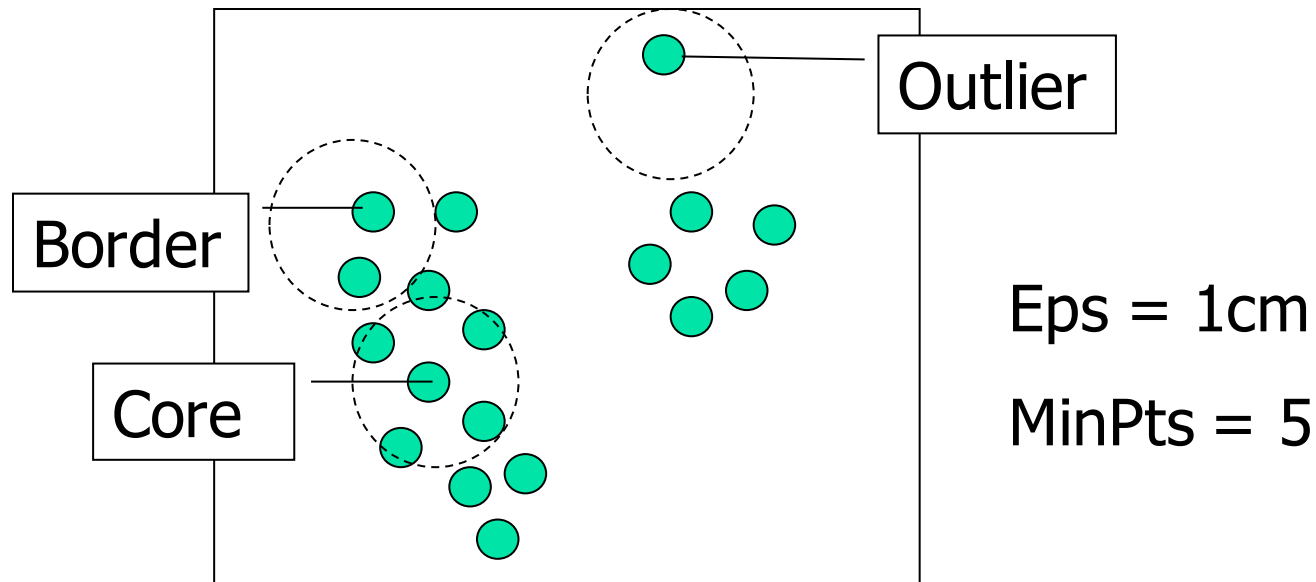
- **Density-connected**

- 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



DBSCAN: Density-Based Spatial Clustering of Applications with Noise

- Relies on a **density-based** notion of cluster: A *cluster* is defined as a maximal set of density-connected points
- Discovers clusters of arbitrary shape in spatial databases with noise



DBSCAN: The Algorithm

- Arbitrary 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

DBSCAN: The Algorithm

Algorithm: DBSCAN: a density-based clustering algorithm.

Input:

- D : a data set containing n objects,
- ϵ : the radius parameter, and
- $MinPts$: the neighborhood density threshold.

Output: A set of density-based clusters.

DBSCAN: The Algorithm

Method:

- (1) mark all objects as **unvisited**;
- (2) **do**
- (3) randomly select an unvisited object p ;
- (4) mark p as **visited**;
- (5) **if** the ϵ -neighborhood of p has at least $MinPts$ objects
- (6) create a new cluster C , and add p to C ;
- (7) let N be the set of objects in the ϵ -neighborhood of p ;
- (8) **for** each point p' in N
- (9) **if** p' is **unvisited**
- (10) mark p' as **visited**;
- (11) **if** the ϵ -neighborhood of p' has at least $MinPts$ points,
 add those points to N ;
- (12) **if** p' is not yet a member of any cluster, add p' to C ;
- (13) **end for**
- (14) output C ;
- (15) **else** mark p as **noise**;
- (16) **until** no object is **unvisited**;

DBSCAN: Sensitive to Parameters

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

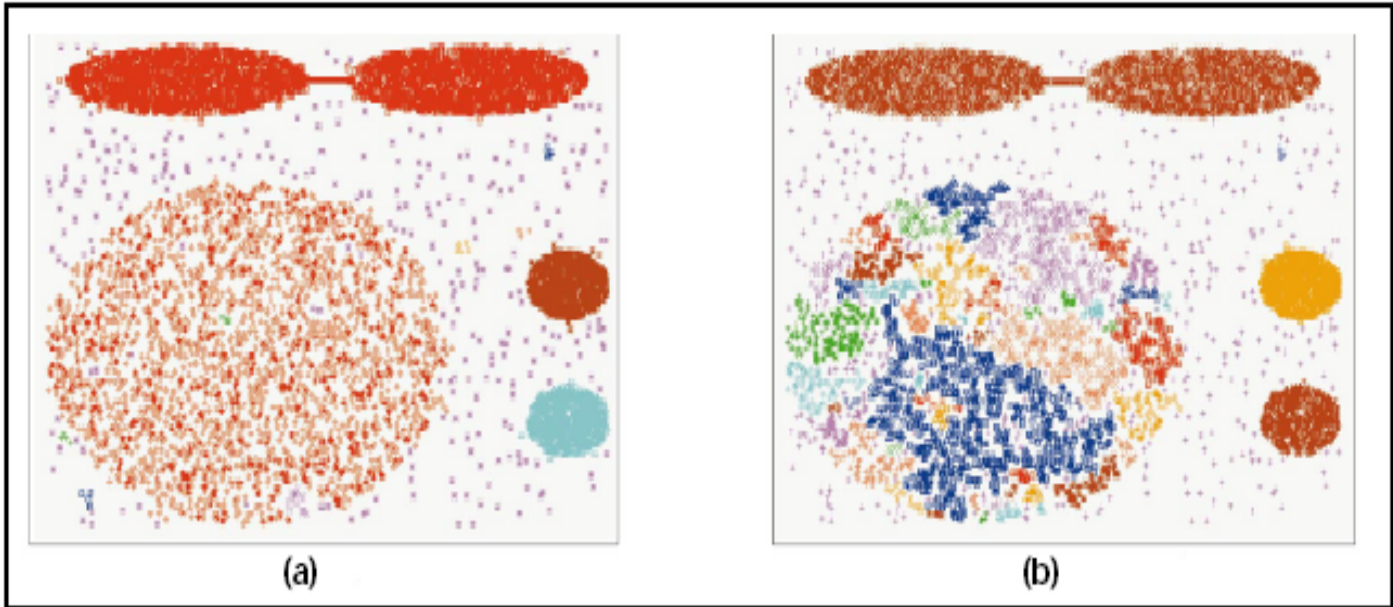
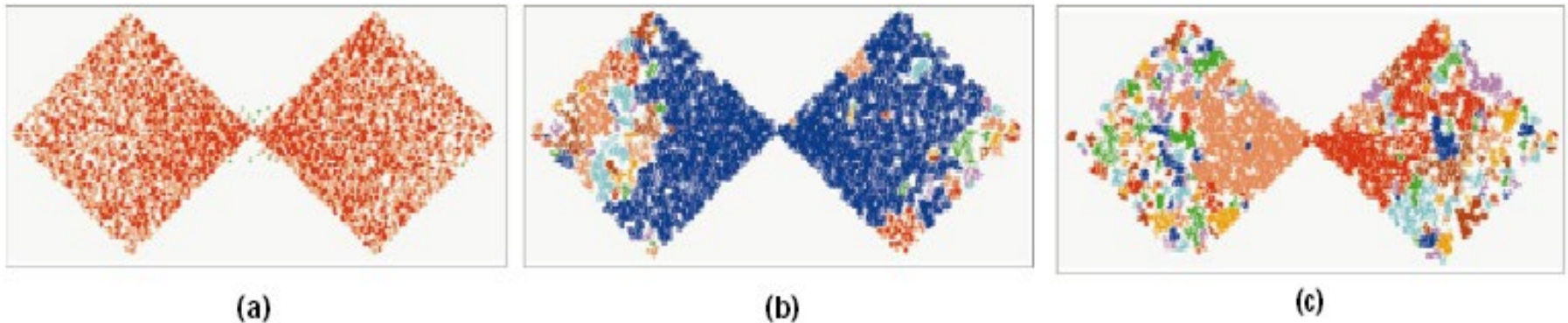


Figure 9. DBScan results for DS2 with MinPts at 4 and Eps at (a) 5.0, (b) 3.5, and (c) 3.0.

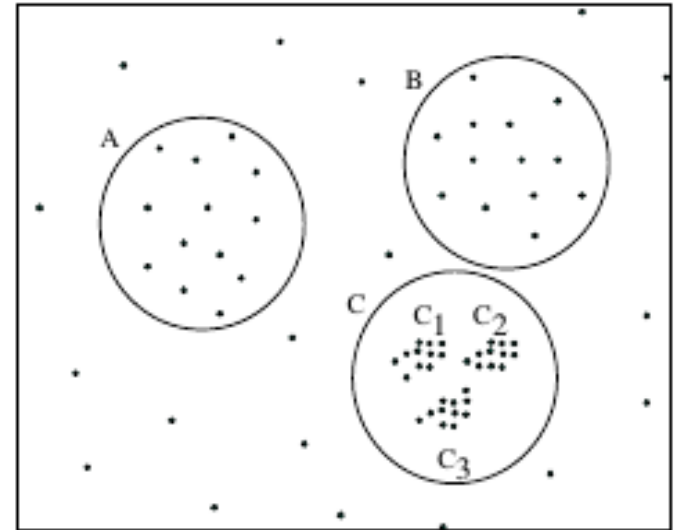


OPTICS: A Cluster-Ordering Method (1999)

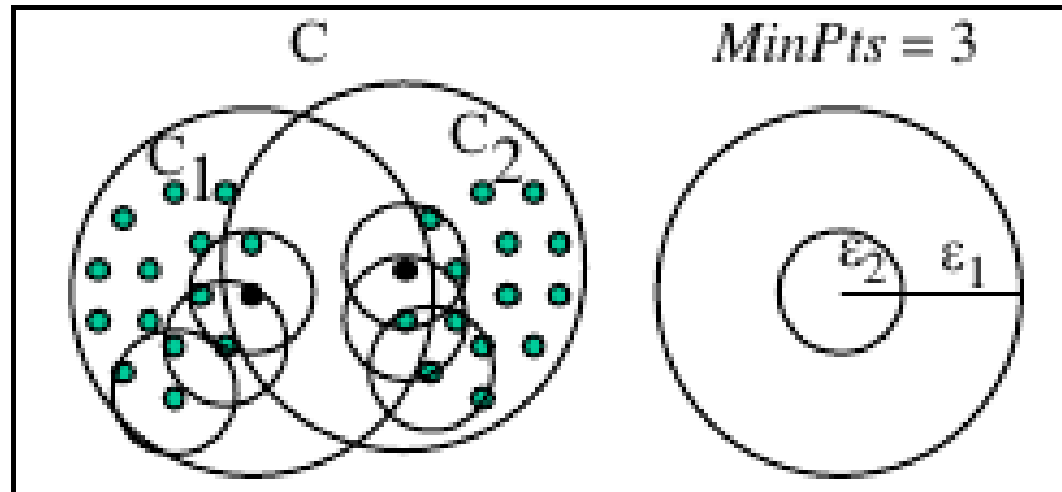
- OPTICS: Ordering Points To Identify the Clustering Structure
 - Ankerst, Breunig, Kriegel, and Sander (SIGMOD'99)
 - Produces a special order of the database wrt its density-based clustering structure
 - This cluster-ordering contains info equiv to the density-based clusterings corresponding to a broad range of parameter settings
 - Good for both automatic and interactive cluster analysis, including finding intrinsic clustering structure
 - Can be represented graphically or using visualization techniques

OPTICS: A Cluster-Ordering Method (1999)

- it is not possible to detect the clusters A, B, C₁, C₂, and C₃ simultaneously using one global density parameter
- A global density-based decomposition would consist only of the clusters A, B, and C, or C₁, C₂, and C₃



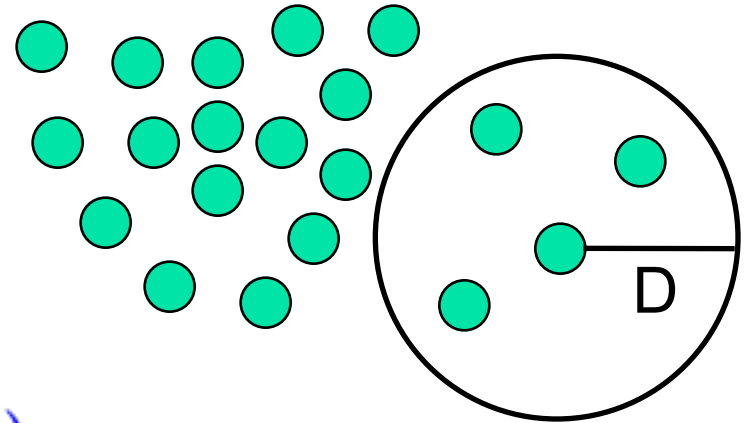
OPTICS: A Cluster-Ordering Method (1999)



- C_1 and C_2 are density-based clusters with respect to ϵ_2
- C is a density-based cluster with respect to ϵ_1
- $\epsilon_2 < \epsilon_1$
- C_1 and C_2 are completely contained in C

OPTICS: Some Extension from DBSCAN

- Core Distance:
 - min eps s.t. point is core



$$\begin{aligned} & \text{core-distance}_{\epsilon, \text{MinPts}}(p) \\ &= \begin{cases} \text{UNDEFINED}, & \text{if } |N_{\epsilon}(p)| < \text{MinPts} \\ \text{MinPts} - \text{distance}(p), & \text{otherwise} \end{cases} \end{aligned}$$

OPTICS: Some Extension from DBSCAN

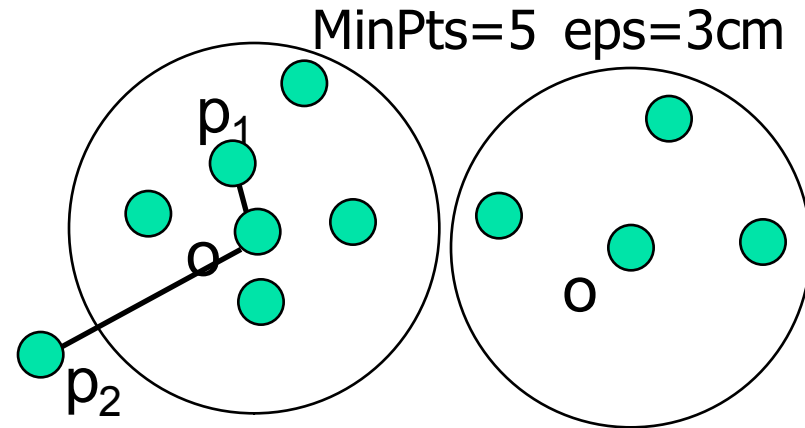
- Reachability Distance

$$\text{reachability} - \text{distance}_{\epsilon, \text{MinPts}}(p, o)$$

$$= \begin{cases} \text{UNDEFINED}, & \text{if } |N_{\epsilon}(o)| < \text{MinPts} \\ \max(\text{core-distance}(o), \text{distance}(o, p)), & \text{otherwise} \end{cases}$$

$$r(p_1, o) = 2.8\text{cm} \quad r(p_2, o) = 4\text{cm}$$

- $r(p, o)$ depends on the density of the space around p , **the larger the density, the smaller $r(p, o)$**



- OPTICS uses a list of seed points sorted in asc. ordering of reach. dist. to store the points to be expanded, to locate the dense area of points

基本思想

输入：数据集D, ε , MinPts

步骤：

1. 初始化所有点的可达距离和核心距离为 ∞
2. 建立两个队列：
 - 有序队列：存储核心点及该核心点的直接密度可达点
 - 结果队列：存储样本输出及处理次序
3. 如D中的数据全部处理完，则算法结束；否则，从D中选择一个未处理的核心对象放入结果队列，并将其直接密度可达对象放入有序队列，直接密度可达对象按可达距离升序排列
4. 如有序队列为空，则返回步骤3；否则，取出有序队列的第一个对象
5. 如该对象为核心对象，将其放入结果队列，并将其直接密度可达对象放入有序队列，如已在有序队列，则更新其可达距离，有序队列中的对象按可达距离升序排列
6. 重复步骤4，直到有序队列为空

基本思想：提取簇结构

步骤：

1. 从结果队列中按顺序取出对象，如该对象的可达距离不大于给定的半径 ϵ ，则该对象属于当前的簇，否则转步骤2；
2. 如该对象的核心距离大于给定半径 ϵ ，则该对象为噪声，将其忽略；否则，该对象属于一个新簇，回步骤1；
3. 重复上述过程，指导结果队列遍历完毕，算法结束

Main loop of OPTICS

OPTICS(SetofObject, ε , MinPts, OrderedFile)

OrderedFile.open();

FOR i FROM 1 TO SetOfObjects.size **DO**

Object := SetOfObjects.get(i);

IF NOT Object.Processed **THEN**

ExpandClusterOrder(SetOfObjects, Object, ε , MinPts,
OrderedFile);

OrderedFile.close();

END // OPTICS

ExpandClusterOrder(SetOfObjects, Object, ϵ , MinPts, OrderedFile)

neighbors := SetOfObjects.neighbors(Object, ϵ);

Object.Processed := True;

Object.reachability_distance := UNDEFINED;

Object.setCoreDistance(neighbors, ϵ , MinPts);

OrderedFile.write(Object);

IF Object.core_distance \neq UNDEFINED **THEN**

OrderSeeds.update(neighbors, Object);

WHILE NOT OrderSeeds.empty() **DO**

currentObject := OrderSeeds.next();

neighbors := SetOfObjects.neighbors(currentObject, ϵ);

currentObject.Processed := TRUE;

currentObject.setCoreDistance(neighbors, ϵ , MinPts);

OrderedFile.write(currentObject);

IF currentObject.core_distance \neq UNDEFINED **THEN**

OrderSeeds.update(neighbors, currentObject);

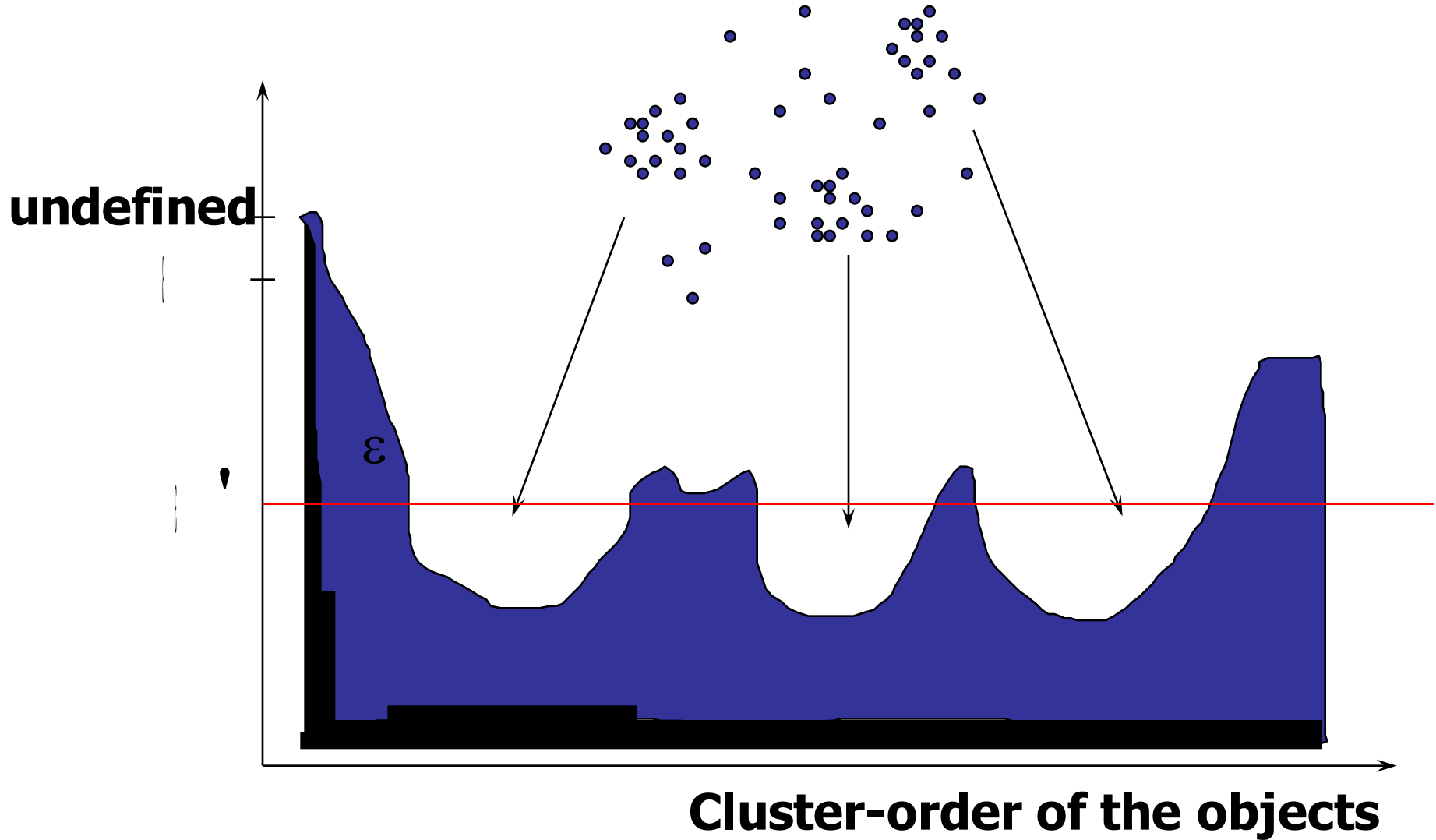
END; // ExpandClusterOrder

```
OrderSeeds::update(neighbors, CenterObject);  
  c_dist := CenterObject.core_distance;  
  FOR ALL 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
```

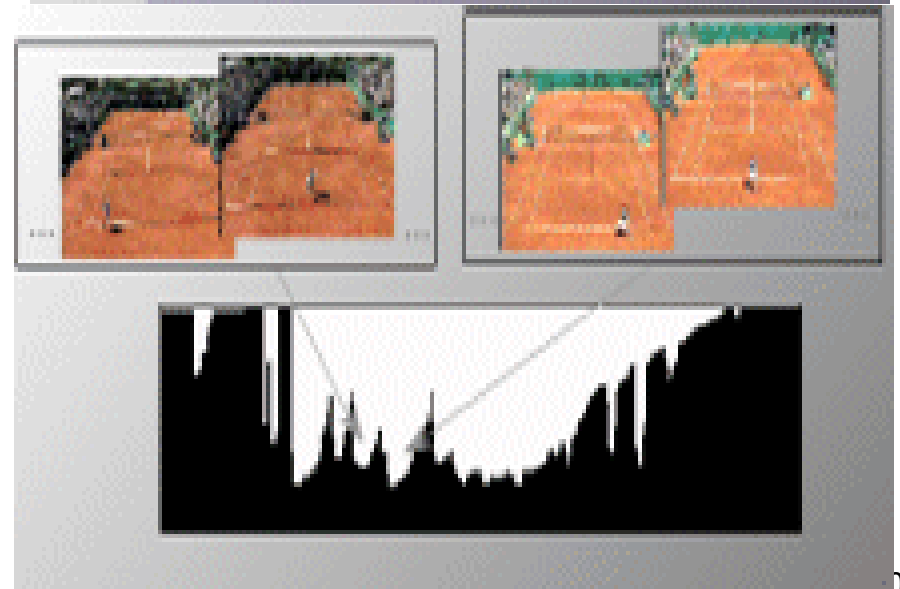
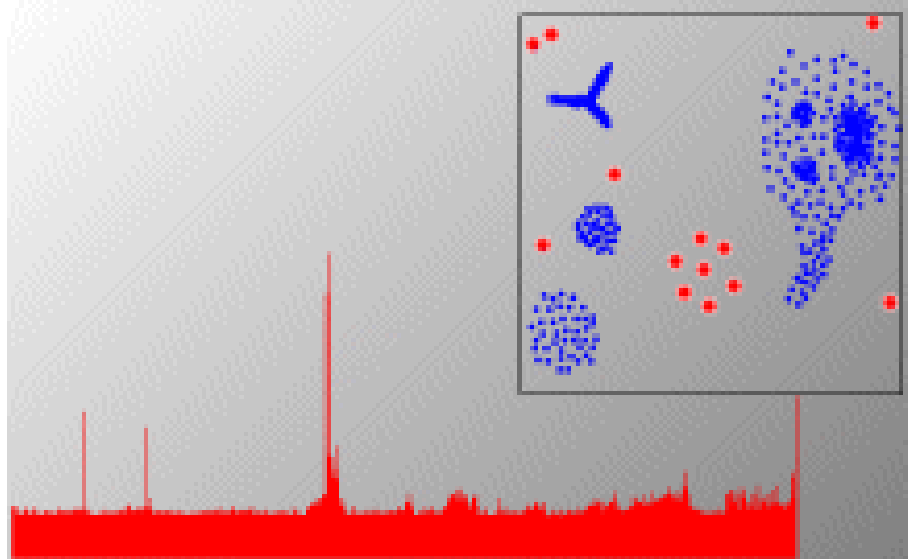
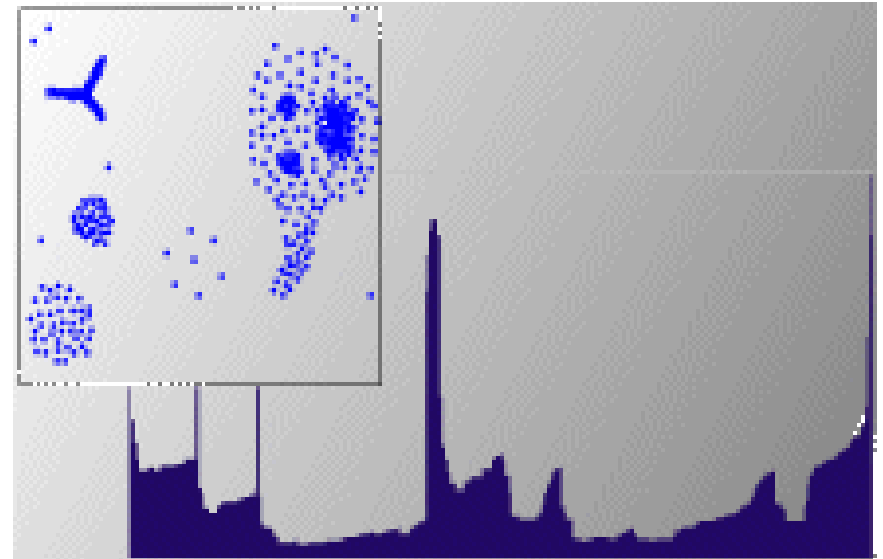
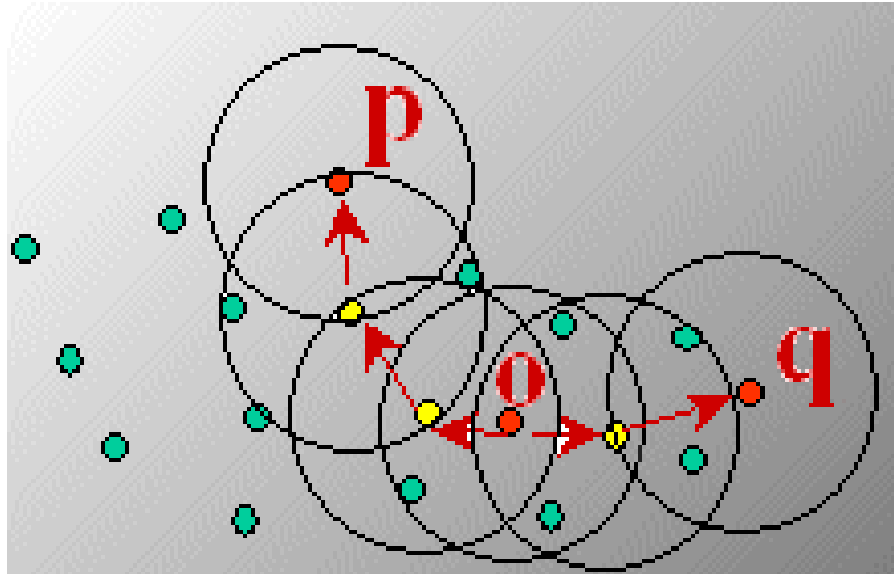
Extract Clustering using Ordered Objects

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

Reachability-distance



Density-Based Clustering: OPTICS & Its Applications



DENCLUE: Using Statistical Density Functions

- DENSity-based CLUstEring by Hinneburg & Keim (KDD'98)
- Using statistical density functions:

$$f_{\text{Gaussian}}(x, y) = e^{-\frac{d(x, y)^2}{2\sigma^2}}$$

influence of y
on x

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

total influence
on x

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

gradient of x in
the direction of x_i

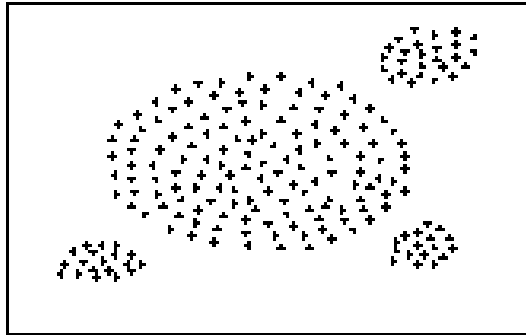
- Major features

- Solid mathematical foundation
- Good for data sets with large amounts of noise
- Allows a compact mathematical description of arbitrarily shaped clusters in high-dimensional data sets
- Significant faster than existing algorithm (e.g., DBSCAN)
- But needs a large number of parameters

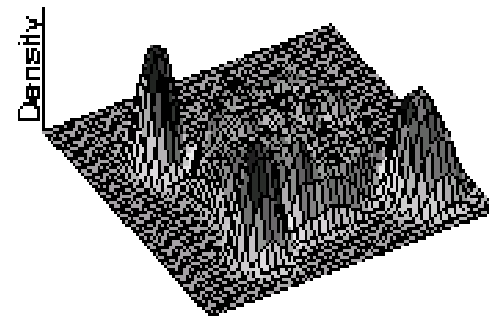
Dendclue: Technical Essence

- Uses grid cells but only keeps information about grid cells that do actually contain data points and manages these cells in a tree-based access structure
- Influence function: describes the impact of a data point within its neighborhood
- Overall density of the data space can be calculated as the sum of the influence function of all data points
- Clusters can be determined mathematically by identifying density attractors
- Density attractors are local maximal of the overall density function
- Center defined clusters: assign to each density attractor the points density attracted to it
- Arbitrary shaped cluster: merge density attractors that are connected through paths of high density ($>$ threshold)

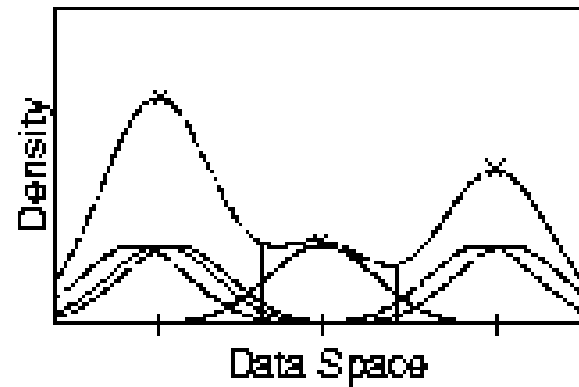
Density Attractor



(a) Data Set



(c) Gaussian



Center-Defined and Arbitrary

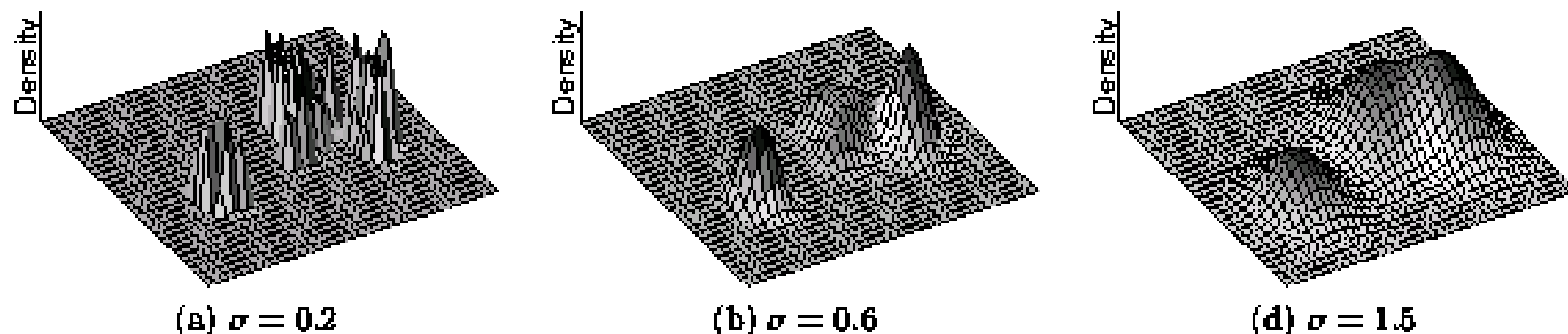


Figure 3: Example of Center-Defined Clusters for different σ

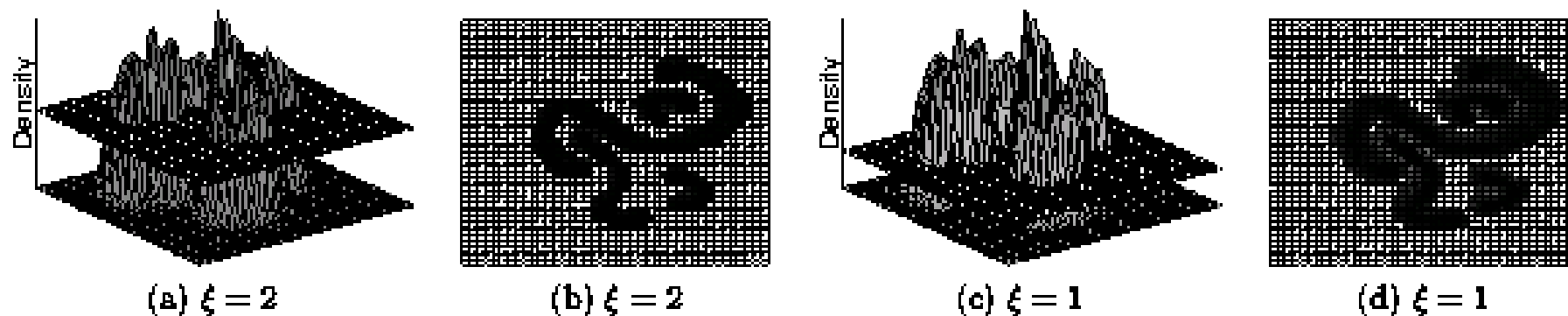


Figure 4: Example of Arbitrary-Shape Clusters for different ξ

Clustering by Fast Search and Find of Density Peaks

- This algorithm aims at determining the centers of clusters
- The authors think the centers of clusters have two features
 - They are surrounded by neighbors with lower local density
 - They are at a relatively large distance from any points with a higher local density
- For each data point i , the algorithm compute two quantities
 - its local density ρ_i
 - its distance δ_i from points of higher density

Clustering by Fast Search and Find of Density Peaks

- Local density ρ_i

- Cut-off kernel

$$\rho_i = \sum_j \chi(d_{ij} - d_c)$$

where

$$\chi(x) = \begin{cases} 1, & x < 0 \\ 0, & x \geq 0 \end{cases}$$

ρ_i is equal to the number of points that are closer than d_c to point i

- Gaussian kernel

$$\rho_i = \sum_j e^{-\left(\frac{d_{ij}}{d_c}\right)^2}$$

Gaussian kernel can yield continuous ρ_i , it is fit for small data set, especially

Clustering by Fast Search and Find of Density Peaks

- distance δ_i

$$\delta_i = \min_{j: \rho_j > \rho_i} (d_{ij})$$

- δ_i is the minimum distance between point **i** and any other point with higher density
- For the point with the highest density

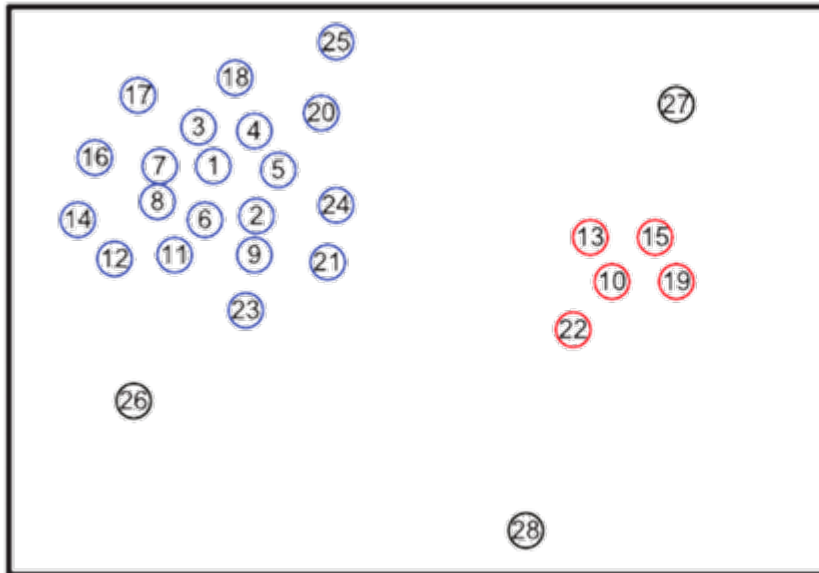
$$\delta_i = \max_j (d_{ij})$$

that is the largest distance between point **i** and any other point

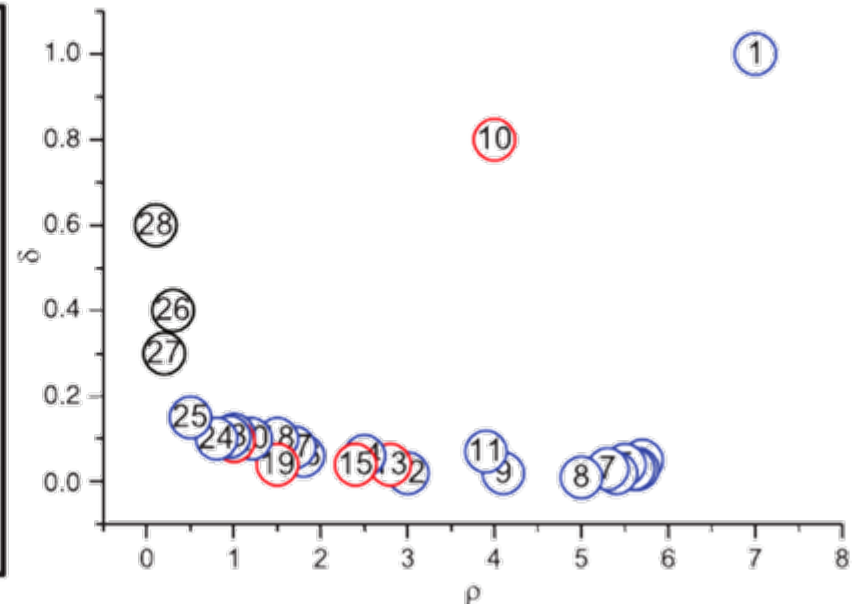
Clustering by Fast Search and Find of Density Peaks

- Decision graph
 - A plot of δ_i as a function of ρ_i in a 2-dim. space for each point
 - From which, the centers of clusters and their numbers can be determined visually

A

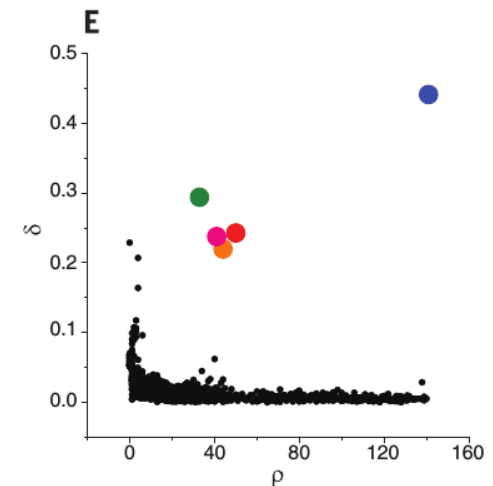
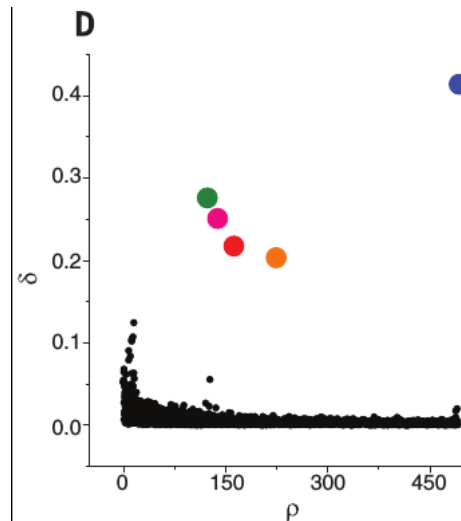
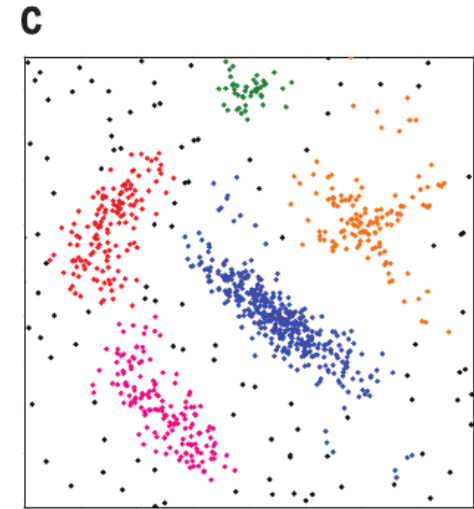
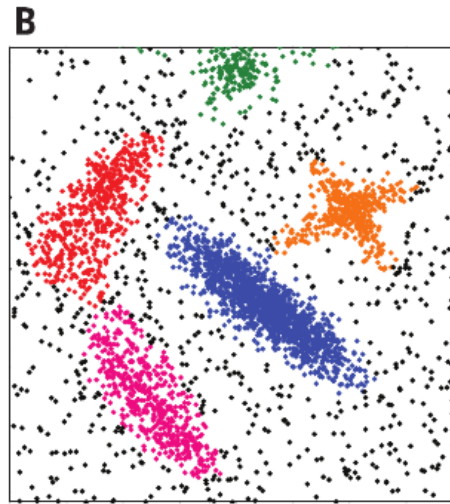
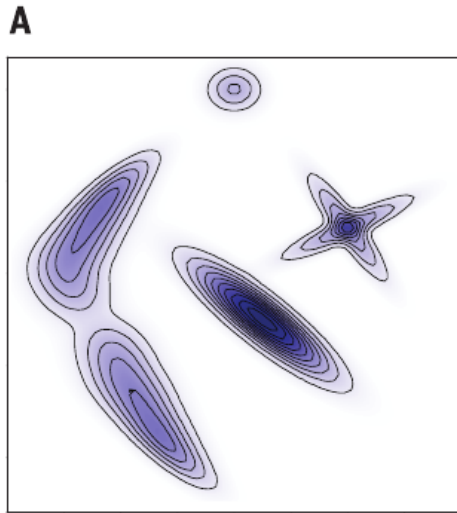


B



Clustering by Fast Search and Find of Density Peaks

■ Decision graph



A: prob. distrib.

B: 4000 samples

C: 1000 samples

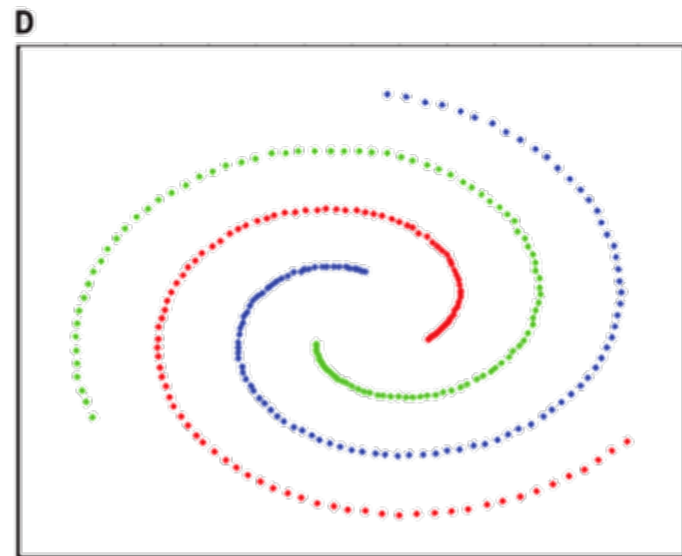
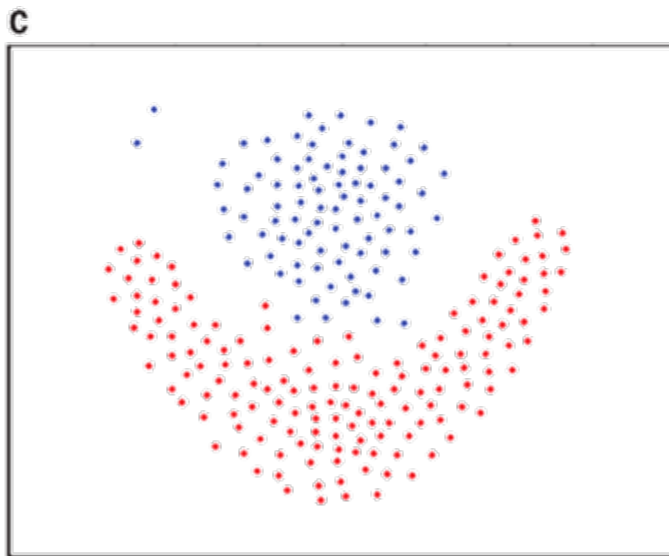
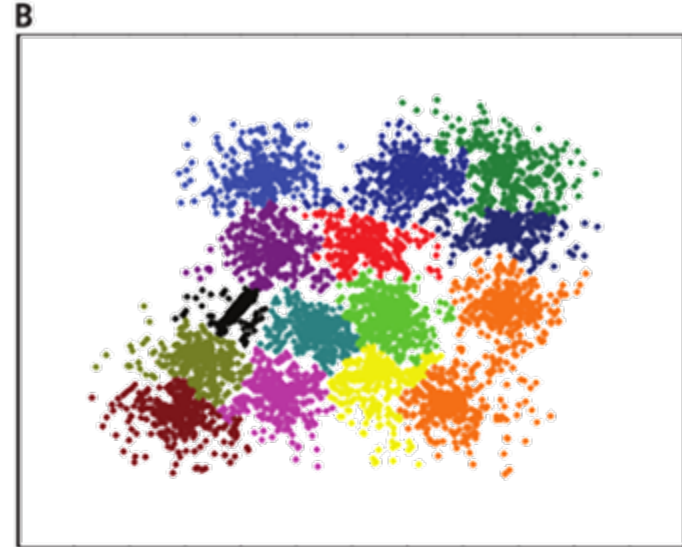
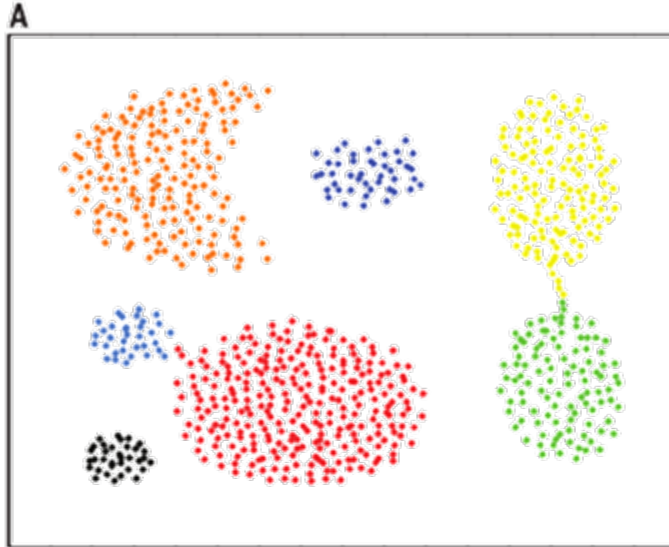
D: decision graph of B

E: decision graph of C

Clustering by Fast Search and Find of Density Peaks

- How to get the resulting clusters
 - After the cluster centers have been found, each remaining point is assigned to the same cluster as its nearest neighbor of higher density
 - The cluster assignment is performed in a single step, in contrast with other clustering algorithms where an objective function is optimized iteratively

Clustering by Fast Search and Find of Density Peaks



Clustering by Fast Search and Find of Density Peaks



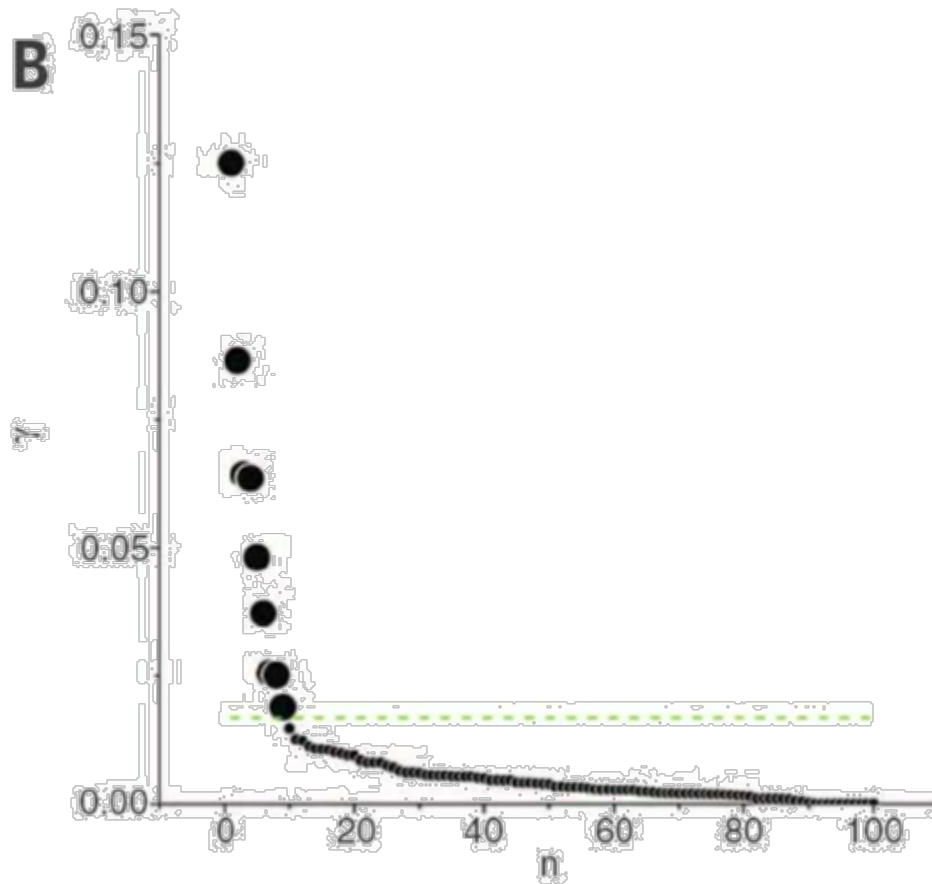
Clustering by Fast Search and Find of Density Peaks

- How to determine the number of cluster centers

$$\gamma_i = \rho_i \delta_i$$

- γ_i provide an indicator of number of cluster centers
 - Sort all γ_i 's in descending order
 - Select **TOP-K** value of γ_i 's
- What value should **K** be?
 - plot γ_i in a 2-dim. space
 - distribution of all γ_i 's of non-centers are smooth
 - from non-center points to center points, there exists a dramatically gap

Clustering by Fast Search and Find of Density Peaks



Chapter 10. Cluster Analysis: Basic Concepts and Methods

- Cluster Analysis: Basic Concepts
- Partitioning Methods
- Hierarchical Methods
- Density-Based Methods
- Grid-Based Methods
- Evaluation of Clustering
- Summary



Grid-Based Clustering Method

- Using multi-resolution grid data structure
- Several interesting methods
 - **STING** (a STatistical INformation Grid approach) by Wang, Yang and Muntz (1997)
 - **WaveCluster** by Sheikholeslami, Chatterjee, and Zhang (VLDB'98)
 - A multi-resolution clustering approach using wavelet method
 - **CLIQUE**: Agrawal, et al. (SIGMOD'98)
 - Both grid-based and subspace clustering

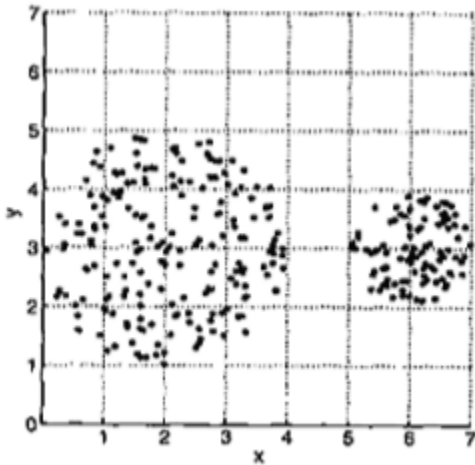
Grid-Based Clustering Method

- 用不同的网格划分方法，将数据空间划分成为有限个单元（cell）的网格结构，并对网格数据结构进行了不同的处理
- 相同的核心步骤
 - 划分网格
 - 使用网格单元内数据的统计信息对数据进行压缩表达
 - 基于这些统计信息判断高密度网格单元
 - 最后将相连的高密度网格单元识别为簇

Grid-Based Clustering Method

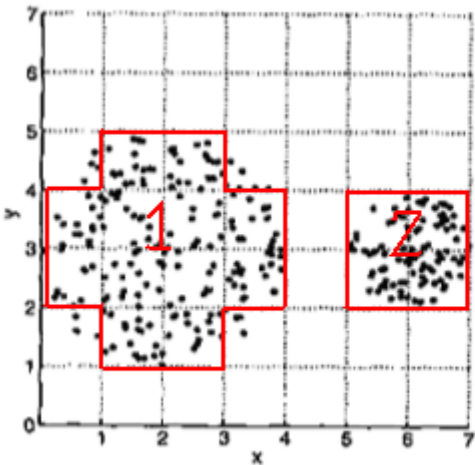
- 将每个属性的可能值分割成许多相邻的区间，创建网格单元的集合(类似于对各维特征进行等宽或等深分箱)
- 将每个样本映射到网格单元中，同时统计每个网格单元的信息，如样本数，均值，中位数，最大值，最小值，数据的分布类型等，之后所有处理的基本单位都是网格单元
- 删除密度低于某个指定阈值的网格单元.
- 由稠密的单元组形成簇

Grid-Based Clustering Method



0	0	0	0	0	0	0
0	0	0	0	0	0	0
4	17	18	6	0	0	0
14	14	13	13	0	18	27
11	18	10	21	0	24	31
3	20	14	4	0	0	0
0	0	0	0	0	0	0

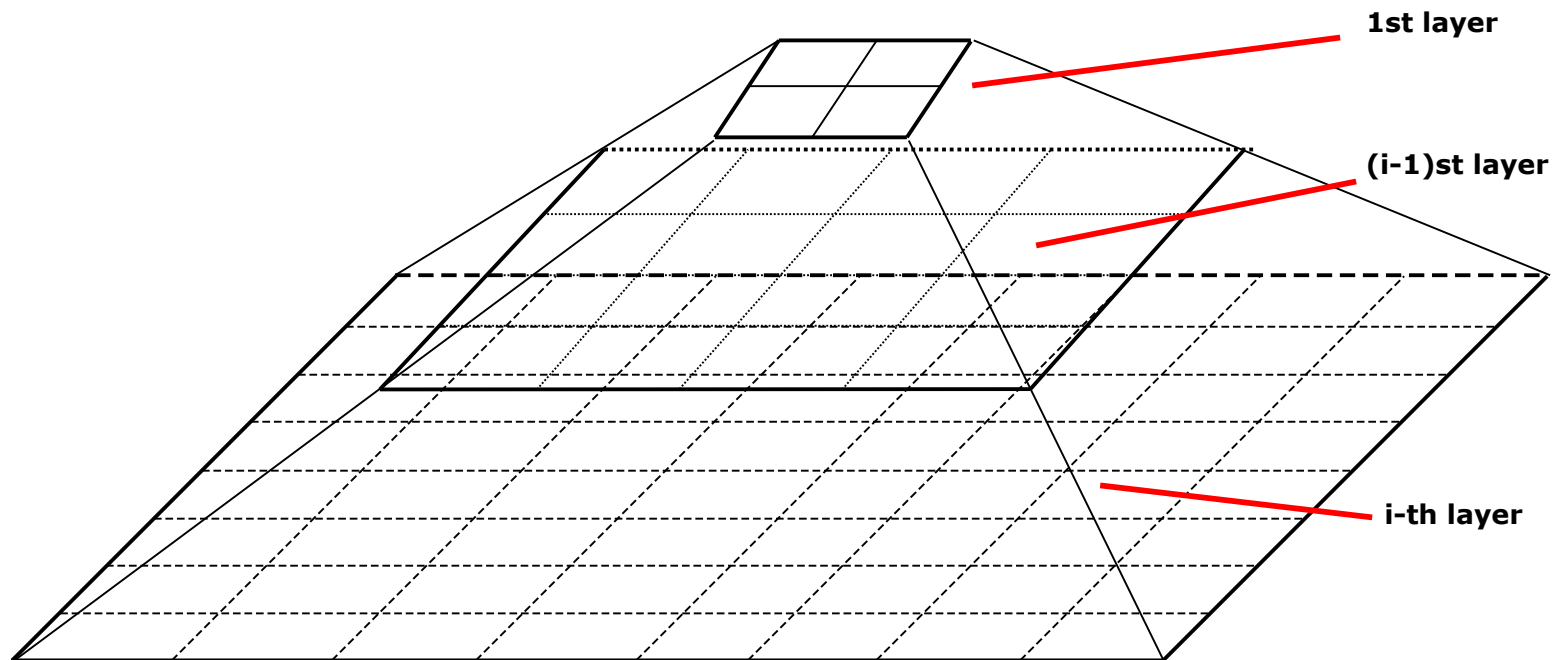
- 若閾值取为8



0	0	0	0	0	0	0
0	0	0	0	0	0	0
4	17	18	6	0	0	0
14	14	13	13	0	18	27
11	18	10	21	0	24	31
3	20	14	4	0	0	0
0	0	0	0	0	0	0

STING: A Statistical Information Grid Approach

- Wang, Yang and Muntz (VLDB'97)
- The spatial area is divided into rectangular cells
- There are several levels of cells corresponding to different levels of resolution



The STING Clustering Method

- Each cell at a high level is partitioned into a number of smaller cells in the next lower level
- Statistical info of each cell is calculated and stored beforehand and is used to answer queries
- Parameters of higher level cells can be easily calculated from parameters of lower level cell
 - **count, mean, s, min, max**
 - **type of distribution—normal, uniform, etc.**
- Use a top-down approach to answer spatial data queries
- Start from a pre-selected layer—typically with a small number of cells
- For each cell in the current level compute the confidence interval

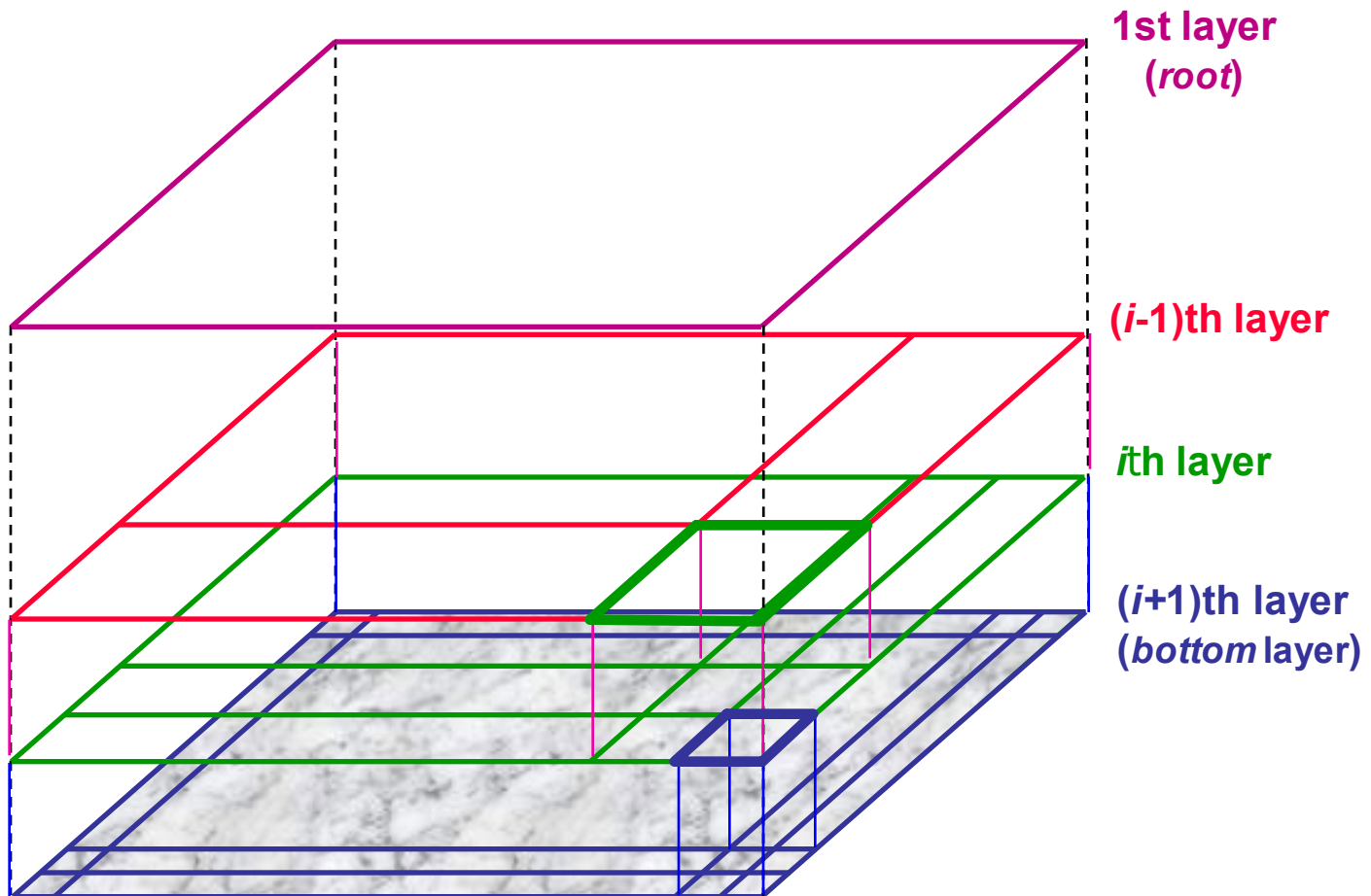
Motivation

- Example: a house map of California State
 - Query 1: Find the **expensive areas** in California State.
 - **expensive**: **price > \$300K**
 - Traditional Method
 - **Step 1**: Scan all the houses and select out the **expensive** one.
 - **Step 2**: Do clustering analysis on those houses selected out.
 - **Step 3**: Form the region that each cluster occupies.
 - **Step 4**: Return those regions larger than 4 square mile.
 - It is **not efficient** because the database is very large (> 10 millions).
 - If "**expensive**" is defined as: **80% houses have price > \$300K**
 - The previous method can not even answer the query.

Motivation

- **STING**: (**ST**atistical **IN**formation **G**rid)
 - a hierarchical statistical information grid based approach
 - Preprocess data
 - capture the **statistical** information
 - **STING outperforms the best previous approach by at least an order of magnitude.**

Hierarchical Structure



STING网格建立

- 数据载入数据库时以自底向上的方式建立网格
- 根据属性的取值划分一些层次，按层次划分网格
- 计算最底层单位网格的统计信息
 - n : 网格中对象数目，与属性无关
 - m : 网格中所有值的平均值
 - s : 网格中属性值的标准偏差
 - \min : 网格中属性值的最小值
 - \max : 网格中属性值的最大值
 - distribution : 网格中属性值符合的分布类型，如正态分布，均匀分布，指数分布
- 最底层的单元参数直接由数据计算，父单元格统计信息由其对应的子单元格计算

STING网格建立

- 父单元格的计算公式

$$n = \sum_i n_i$$

$$max = \max_i(max_i)$$

$$min = \min_i(min_i)$$

$$m = \frac{\sum_i m_i n_i}{n}$$

$$s = \sqrt{\frac{\sum_i (s_i^2 + m_i^2) n_i}{n} - m^2}$$

- 父单元格分布的计算：设dist为对应子单元格多数的分布类型，计算confl

若 $dist_i \neq dist, m_i \approx m, s_i \approx s$, $confl = confl + n_i$

若 $dist_i \neq dist, m_i \not\approx m$ 或 $s_i \not\approx s$, 则 $confl = n$

若 $dist_i = dist, m_i \approx m, s_i \approx s$, 则 $confl = confl + 0$

若 $dist_i = dist, m_i \not\approx m$ 或 $s_i \not\approx s$, 则 $confl = n$

如果 $\frac{confl}{n} > t$ (阈值, 设为 0.05), $dist = NONE$, 否则 $dist = dist$

- 从最底层逐层计算上一层每个父单元格的统计信息，直到最顶层，同时根据密度阈值标记稠密网格

Queries Type

- **Region Query Example:**

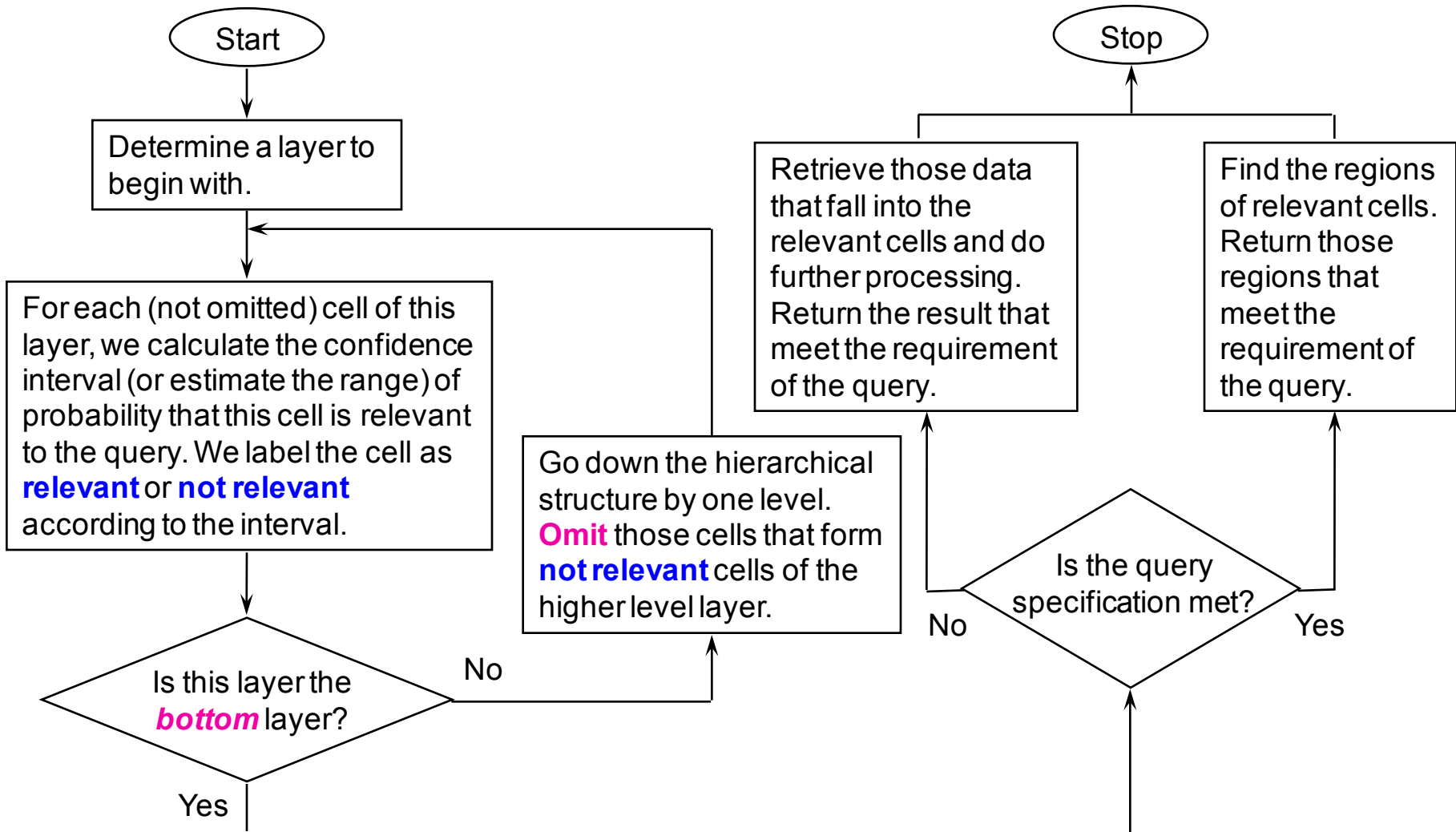
- Select the maximal regions that have at least 100 houses per unit area and at least 70% of the house prices are above \$400K and with total area at least 100 units with 90% confidence.

```
SELECT REGION  
FROM house-map  
WHERE DENSITY IN (100,  $\infty$ )  
AND price RANGE (400000,  $\infty$ )  
    WITH PERCENT (0.7, 1)  
AND AREA (100,  $\infty$ )  
AND WITH CONFIDENCE 0.9
```

Algorithm

- Take advantage of the **statistical information** captured.
- Only go through **relevant** cells at each level.
 - Root is **relevant**.
 - For each **relevant** cell, we exam its children at next level by **statistical test** and label them as **relevant** or **not relevant**.
 - Form regions from **relevant** bottom level cells.
- Do not need to access full database.
- It is very **efficient**.

Algorithm



Sting聚类算法

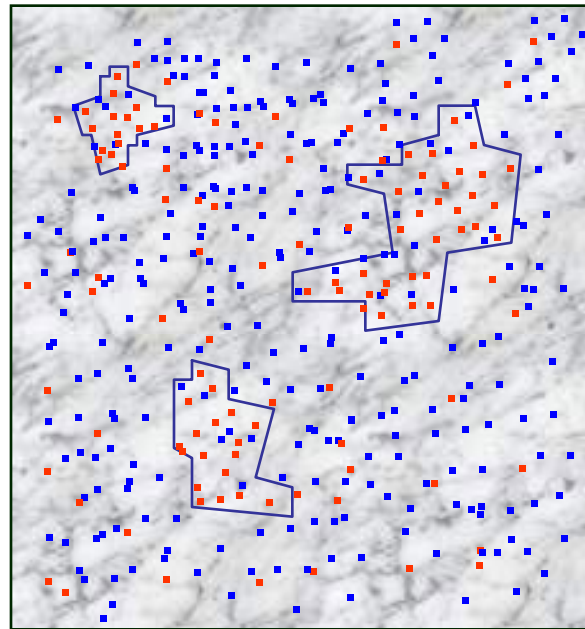
- 将STING查询算法稍做修改即可将其用于聚类
 - 首先，在层次结构中选定一层作为聚类的开始层
 - 对当前层次的每个单元，根据其统计信息(如样本数，样本数百分比等)与设定的阈值，考察该单元是否为噪声单元
 - 噪声单元内所有点认为是噪声点，不参与聚类，低一层的处理就只检查剩余的非噪声单元
 - 反复进行这一过程，直到达到最底层，最底层稠密的单元组形成簇

Performance

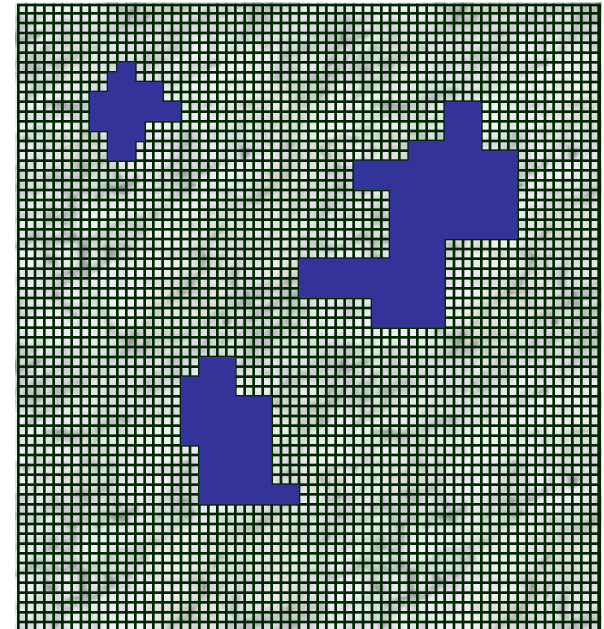
9.8 seconds to generate the hierarchical structure

0.2 second to answer the query (SPARC 10, 192 MB Memory)

- expensive house
- non-expensive house



Expected Result



STING's Result

As the granularity of the leaf cells approaches zero, the regions returned by STING better approximate the expected result.

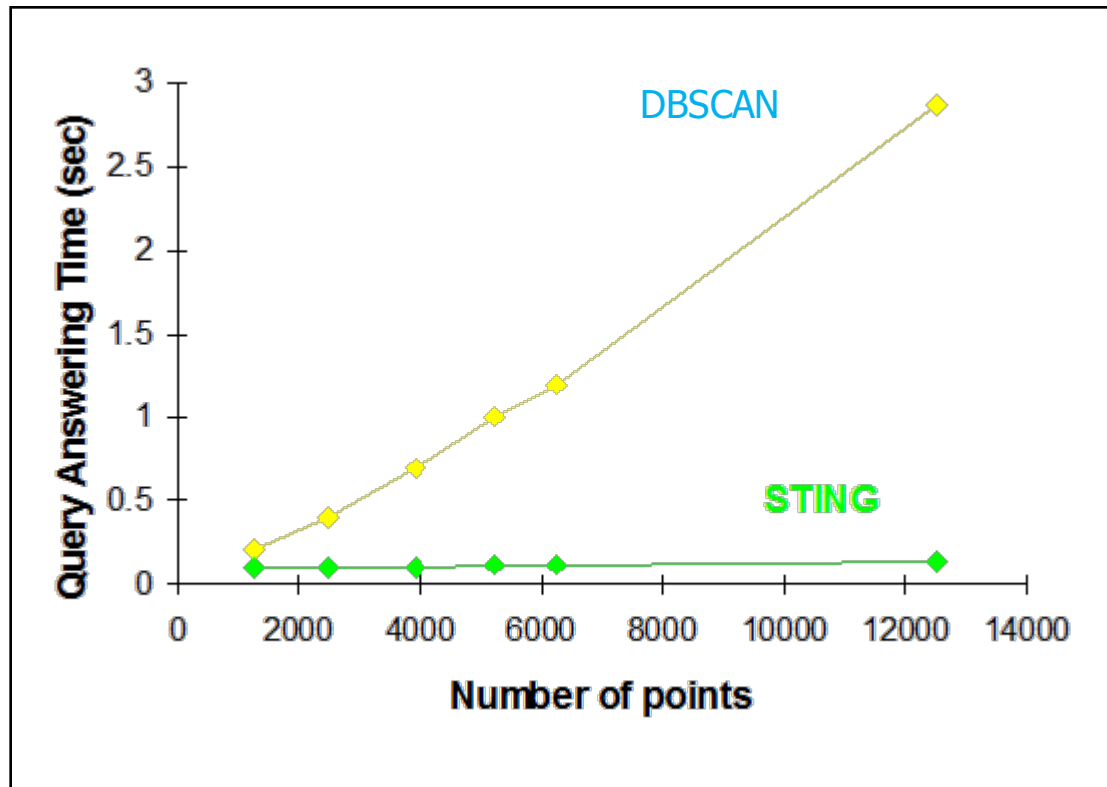
Performance

- The computational complexity of STING is **linearly proportional** to the number of leaf cells.
- We used the SEQUOIA 2000 benchmark as the data set to compare the performance of STING with other approaches.
- All times are taken in seconds.

Number of Points		1256	2503	3910	5213	6256	12512
Previous Approaches	CLARANS	49	200	457	785	1238	5538
	DBSCAN (projected)	0.2	0.4	0.7	1.0	1.2	2.86
STING	query answering	0.1	0.11	0.11	0.12	0.12	0.14
	structure generation	1.25	1.32	1.40	1.48	1.55	1.62

Performance

- STING outperforms the best previous algorithm (DBSCAN) by **at least an order of magnitude**, especially when the data set is large.



STING Algorithm and Its Analysis

- Remove the irrelevant cells from further consideration
- When finish examining the current layer, proceed to the next lower level
- Repeat this process until the bottom layer is reached
- Advantages:
 - Query-independent, easy to parallelize, incremental update
 - $O(K)$, where K is the number of grid cells at the lowest level
- Disadvantages:
 - All the cluster boundaries are either horizontal or vertical, and no diagonal boundary is detected

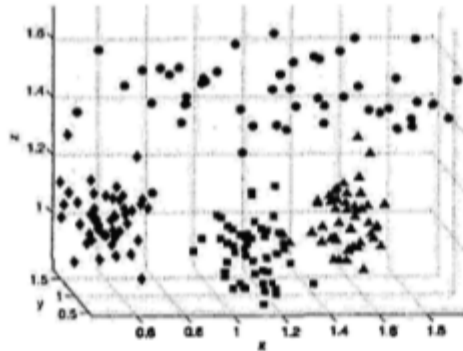
Conclusion

- STING is a **query-independent** approach.
 - The statistical information exists independently of queries.
- STING has **a much smaller response time** compared to other approaches.
 - The computational complexity is linearly proportional to number of leaves.
 - I/O cost is low.
- STING can support **different resolution** of query result.
- Regions returned by STING approach that returned by DBSCAN when the granularity approaches zero.
- The hierarchical structure of STING can be **maintained efficiently**.
 - incremental update

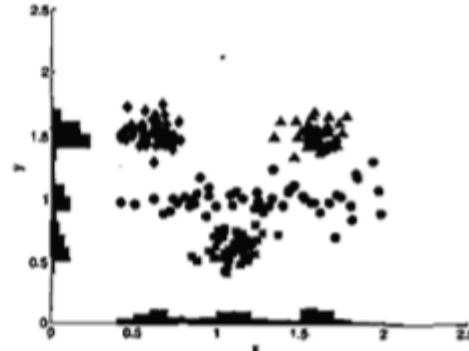
CLIQUE (Clustering In QUES)

- Agrawal, Gehrke, Gunopulos, Raghavan (SIGMOD'98)
- Automatically identifying subspaces of a high dimensional data space that allow better clustering than original space
- CLIQUE can be considered as both **density-based and grid-based**
 - It partitions each dimension into the same number of equal length interval
 - It partitions an m-dimensional data space into non-overlapping rectangular units
 - A unit is dense if the fraction of total data points contained in the unit exceeds the input model parameter
 - A cluster is a maximal set of connected dense units within a subspace

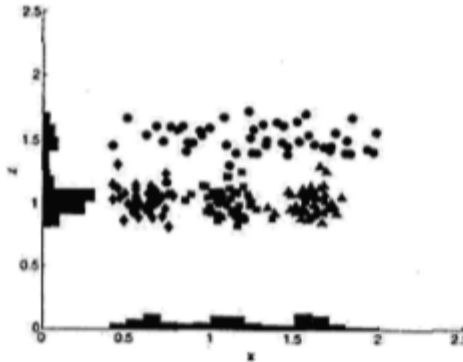
CLIQUE (Clustering In QUES)



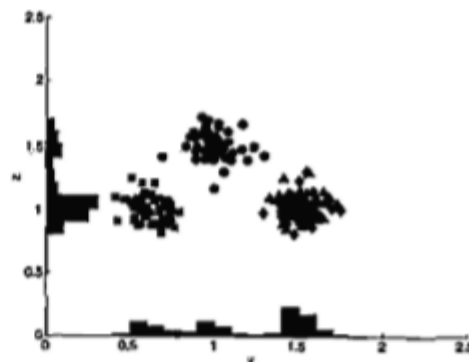
(a) 三维空间上的3个簇



(b) xy 平面上的视图



(c) xz 平面上的视图



(d) yz 平面上的视图

CLIQUE: The Major Steps

- Partition the data space and find the number of points that lie inside each cell of the partition.
- Identify the subspaces that contain clusters using the Apriori principle
- Identify clusters
 - Determine dense units in all subspaces of interests
 - Determine connected dense units in all subspaces of interests.
- Generate minimal description for the clusters
 - Determine maximal regions that cover a cluster of connected dense units for each cluster
 - Determination of minimal cover for each cluster

CLIQUE: The Major Steps

- Concepts and Notations
 - $A=\{A_1, A_2, \dots, A_d\}$ is a set of bounded, totally ordered domains, $S=A_1 \times A_2 \times \dots \times A_d$ is a d-dim. numerical space, we refer to A_1, \dots, A_d as the **dimensions (attributes)** of S
 - $D=\{v_1, v_2, \dots, v_m\}$, where $v_i=\langle v_{i1}, v_{i2}, \dots, v_{id} \rangle$
 - We partition the data space S into non-overlapping rectangular **units (cells)**. The units are obtained by partitioning every dim. into **ξ** intervals of equal length
 - Each unit u is the intersection of one interval from each attribute, it has the form $\{u_1, \dots, u_d\}$, where $u_i=[l_i, h_i)$
 - We say that a point $v=\langle v_1, \dots, v_d \rangle$ is **contained** in a unit $u=\{u_1, \dots, u_d\}$ if $l_i \leq v_i < h_i$ for all u_i

CLIQUE: The Major Steps

- Concepts and Notations
 - The **selectivity** of a unit is defined to be the fraction of total data points contained in the unit
 - We call a unit u dense if **selectivity**(u) $\geq \tau$, where τ is density threshold
 - Project the data set D into $A_{t_1} \times A_{t_2} \times \dots \times A_{t_k}$, where $k < d$ and $t_i < t_j$ if $i < j$, then a **unit** in the subspace is the intersection of an interval from each of the k attributes
 - A **cluster** is a **maximal set of connected dens units** in k -dim.
 - Two k -dim. units u_1, u_2 are **connected** if they **have a common face** or if there exists another k -dim. unit u_3 such that both u_1 and u_2 are connected to u_3

CLIQUE: The Major Steps

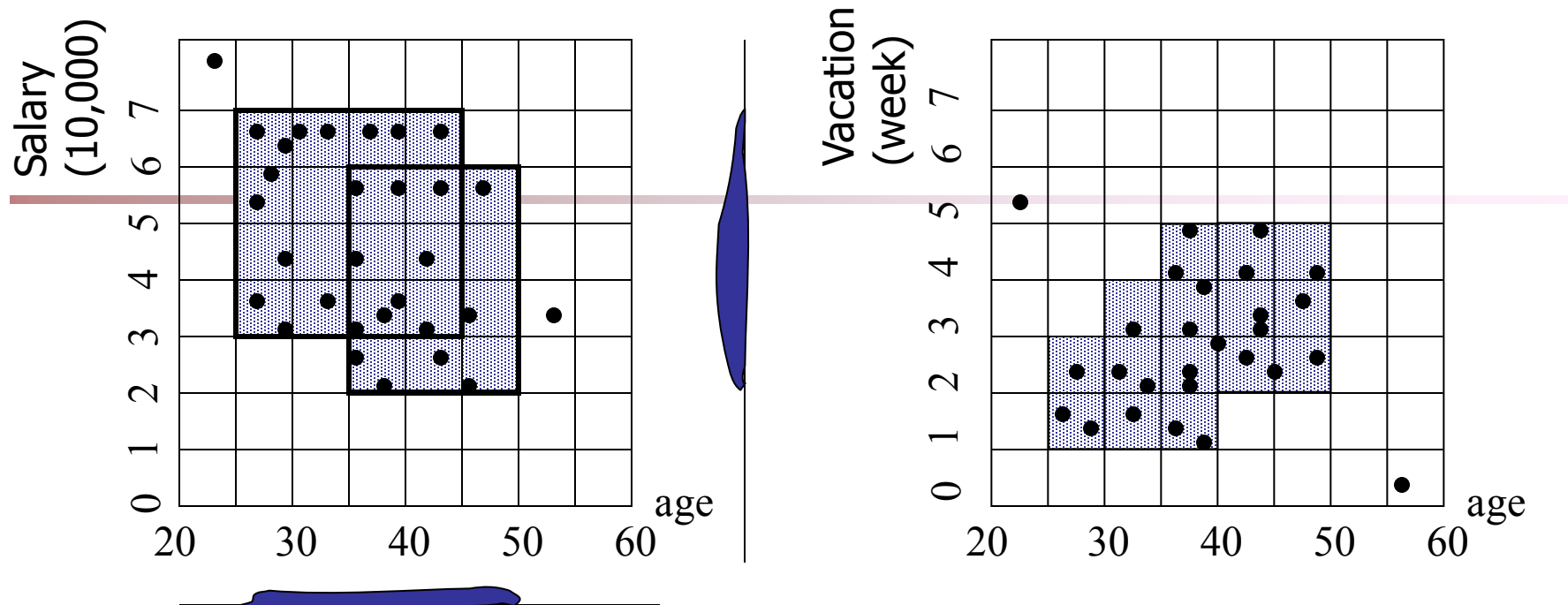
- Concepts and Notations
 - Two units $u_1=\{r_{t1},\dots,r_{tk}\}$ and $u_2=\{r_{t1}',\dots,r_{tk}'\}$ **have a common face** if there are $k-1$ dim., assume A_{t1},\dots,A_{tk-1} , such that $r_{tj}=r_{tj}'$ and either $h_{tk}=l_{tk}'$ or $h_{tk}'=l_{tk}$
 - A **region** in k dim. is an axis-parallel rectangular k -dim. set. We are only interested in those regions that can be expressed as unions of units.

CLIQUE: The Major Steps

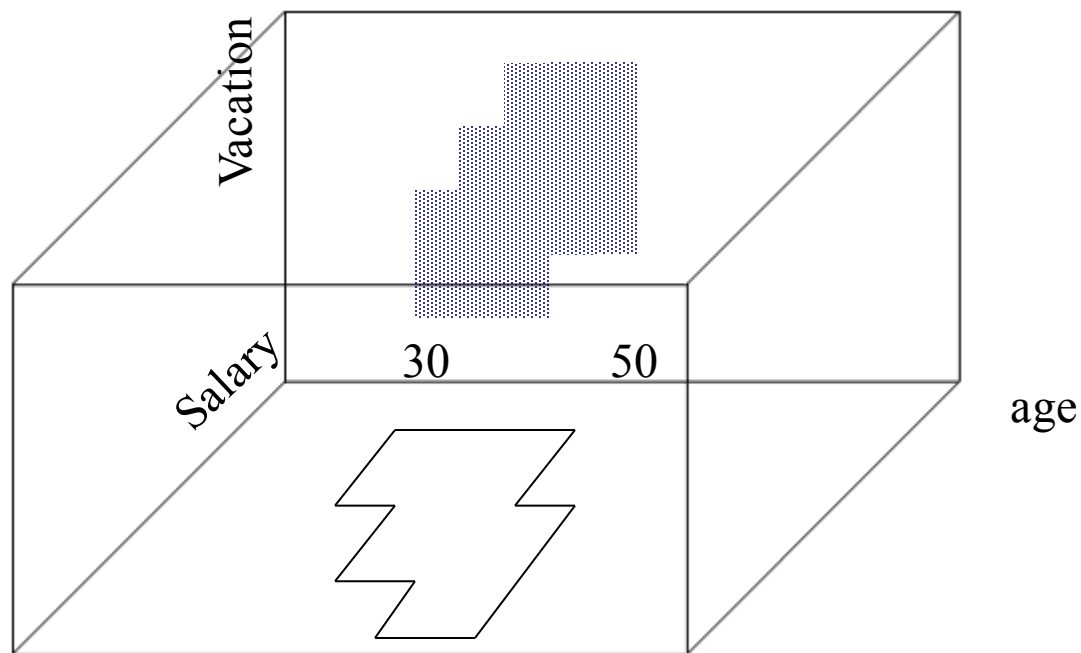
- Identification of subspaces that contain clusters
 - CLIQUE uses a bottom-up algorithm exploits the monotonicity of the clustering criterion with respect to dimensionality to prune the search space
 - **Monotonicity**: if a collection of points S is a cluster in a k -dim. space, then S is also part of a cluster in any $(k-1)$ -dimensional projections of this space.

CLIQUE: The Major Steps

- Identification of subspaces that contain clusters
 - Determine 1-dim. dense units by making a pass over the data to find those candidate dense units
 - The candidate k-dim. units are determined using the method alike what is used in Apriori algorithm from (k-1)-dim. dense units
 - Takes D_{k-1} , all (k-1)-dim. dense units as argument
 - **Slef-join** D_{k-1} , the join condition is that units share the first k-2 dimensions; and then pruning
- ```
insert into C_k
select $u_1.[l_1, h_1), u_1.[l_2, h_2), \dots, u_1.[l_{k-1}, h_{k-1}), u_2.[l_{k-1}, h_{k-1})$
from $D_{k-1} \ u_1, D_{k-1} \ u_2$
where $u_1.dim_1 = u_2.dim_1, u_1.l_1 = u_2.l_1, u_1.h_1 = u_2.h_1,$
 $u_1.dim_2 = u_2.dim_2, u_1.l_2 = u_2.l_2, u_1.h_2 = u_2.h_2, \dots,$
 $u_1.dim_{k-2} = u_2.dim_{k-2}, u_1.l_{k-2} = u_2.l_{k-2}, u_1.h_{k-2} = u_2.h_{k-2},$
 $u_1.dim_{k-1} < u_2.dim_{k-1}$
```
- This procedure terminates when no more candidates are generated



$\tau = 3$



# CLIQUE: The Major Steps

---

- Find clusters
  - **Input:**  $D$ , a set of dense units, all in the same  $k$ -dim. space  $S$
  - **Output:** a partition of  $D$  into  $D_1, \dots, D_q$ , s.t. all units in  $D_i$  are connected and no two units  $u_i \in D_i$ ,  $u_j \in D_j$  with  $i \neq j$
  - **Method**
    - **Construct a graph** for the dense units in  $D$ 
      - Each dense unit in  $D$  is represented as a vertex
      - When two dense units have a common face, connect a edge between the corresponding vertices
    - Use a **depth-first search** to find the **connected components** of the graph, each connected component is a cluster

# Strength and Weakness of *CLIQUE*

---

## ■ Strength

- automatically finds subspaces of the highest dimensionality such that high density clusters exist in those subspaces
- insensitive to the order of records in input and does not presume some canonical data distribution
- scales linearly with the size of input and has good scalability as the number of dimensions in the data increases

## ■ Weakness

- The accuracy of the clustering result may be degraded at the expense of simplicity of the method

# Chapter 10. Cluster Analysis: Basic Concepts and Methods

---

- Cluster Analysis: Basic Concepts
- Partitioning Methods
- Hierarchical Methods
- Density-Based Methods
- Grid-Based Methods
- Evaluation of Clustering
- Summary



# Evaluation of Clustering

---

- **Assessing clustering tendency:** for a given data set, we assess whether a nonrandom structure exists in the data.
- **Determining the number of clusters in a data set:** estimating of the number of clusters before a clustering algorithm is used to derive detailed clusters
- **Measuring clustering quality:** assessing how good the resulting clusters are after applying a clustering method on a data set.

# Assessing Clustering Tendency

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

# Determine the Number of Clusters

---

## ■ Empirical method

- # of clusters  $\approx \sqrt{n}/2$  for a dataset of  $n$  points. In expectation, each cluster has  $\sqrt{2n}$  points

## ■ Elbow method

- Observation
  - Increasing the number of clusters can help to reduce the sum of within-cluster variance of each cluster
  - If too many clusters are formed, the marginal effect of reducing the sum of within-cluster variances may drop
- Use the turning point in the curve of sum of within cluster variance w.r.t the # of clusters

# Determine the Number of Clusters

---

- Cross validation method

- Divide a given data set into  $m$  parts
- Use  $m - 1$  parts to obtain a clustering model
- Use the remaining part to test the quality of the clustering
  - E.g., For each point in the test set, find the closest centroid, and use the sum of squared distance between all points in the test set and the closest centroids to measure how well the model fits the test set
- For any  $k > 0$ , repeat it  $m$  times, compare the overall quality measure w.r.t. different  $k$ 's, and find # of clusters that fits the data the best

# Measuring Clustering Quality

---

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

# Measuring Clustering Quality: Extrinsic Methods

---

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

# Measuring Clustering Quality: Extrinsic Methods

- **BCubed Precision** and **Recall**

- $D=\{o_1, \dots, o_n\}$ ,  $C$  is a clustering on  $D$ ,  $L(o_i)$  is the category of  $o_i$  in the ground truth,  $C(o_i)$  is the cluster\_id of  $o_i$  in  $C$

- For two objects,  $o_i$  and  $o_j$

$$\text{Correctness}(o_i, o_j) = \begin{cases} 1 & \text{if } L(o_i) = L(o_j) \Leftrightarrow C(o_i) = C(o_j) \\ 0 & \text{otherwise} \end{cases}$$

- **Bcubed Precision**

$$\text{Precision BCubed} = \frac{\sum_{i=1}^n \frac{\sum_{o_j: i \neq j, C(o_i)=C(o_j)} \text{Correctness}(o_i, o_j)}{\left\| \{o_j | i \neq j, C(o_i) = C(o_j)\} \right\|}}{n}$$

- **BCubed Recall**

$$\text{Recall BCubed} = \frac{\sum_{i=1}^n \frac{\sum_{o_j: i \neq j, L(o_i)=L(o_j)} \text{Correctness}(o_i, o_j)}{\left\| \{o_j | i \neq j, L(o_i) = L(o_j)\} \right\|}}{n}$$

# Measuring Clustering Quality: Intrinsic Methods

## ■ Silhouette Coefficient

- D is a data set with n objects, which is partitioned into k clusters,  $C_1, \dots, C_k$ .
- For each object  $o \in D$ ,  $a(o)$  is the average distance between o and all other objects in the cluster to which o belongs,  $b(o)$  is the minimum average distance from o to all clusters to which o does not belong

$$a(o) = \frac{\sum_{o' \in C_i, o' \neq o} \text{dist}(o, o')}{|C_i| - 1}$$

$$b(o) = \min_{C_j: 1 \leq j \leq k, j \neq i} \left\{ \frac{\sum_{o' \in C_j} \text{dist}(o, o')}{|C_j|} \right\}$$

- Silhouette Coefficient

$$s(o) = \frac{b(o) - a(o)}{\max\{a(o), b(o)\}}$$

compactness

degree to which o is separated from other clusters

# Chapter 10. Cluster Analysis: Basic Concepts and Methods

---

- Cluster Analysis: Basic Concepts
- Partitioning Methods
- Hierarchical Methods
- Density-Based Methods
- Grid-Based Methods
- Evaluation of Clustering
- Summary



# Summary

---

- **Cluster analysis** groups objects based on their **similarity** and has wide applications
- Measure of similarity can be computed for **various types of data**
- Clustering algorithms can be **categorized** into partitioning methods, hierarchical methods, density-based methods, grid-based methods, and model-based methods
- **K-means** and **K-medoids** algorithms are popular partitioning-based clustering algorithms
- **Birch** and **Chameleon** are interesting hierarchical clustering algorithms, and there are also probabilistic hierarchical clustering algorithms
- **DBSCAN**, **OPTICS**, and **DENCLU** are interesting density-based algorithms
- **STING** and **CLIQUE** are grid-based methods, where CLIQUE is also a subspace clustering algorithm
- Quality of clustering results can be evaluated in various ways

# CS512-Spring 2011: An Introduction

---

- Coverage
  - Cluster Analysis: Chapter 11
  - Outlier Detection: Chapter 12
  - Mining Sequence Data: BK2: Chapter 8
  - Mining Graphs Data: BK2: Chapter 9
  - Social and Information Network Analysis
    - BK2: Chapter 9
    - Partial coverage: Mark Newman: "Networks: An Introduction", Oxford U., 2010
    - Scattered coverage: Easley and Kleinberg, "Networks, Crowds, and Markets: Reasoning About a Highly Connected World", Cambridge U., 2010
    - Recent research papers
  - Mining Data Streams: BK2: Chapter 8
- Requirements
  - One research project
  - One class presentation (15 minutes)
  - Two homeworks (no programming assignment)
  - Two midterm exams (no final exam)

# References (1)

---

- R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan. Automatic subspace clustering of high dimensional data for data mining applications. SIGMOD'98
- M. R. Anderberg. Cluster Analysis for Applications. Academic Press, 1973.
- M. Ankerst, M. Breunig, H.-P. Kriegel, and J. Sander. Optics: Ordering points to identify the clustering structure, SIGMOD'99.
- Beil F., Ester M., Xu X.: "Frequent Term-Based Text Clustering", KDD'02
- M. M. Breunig, H.-P. Kriegel, R. Ng, J. Sander. LOF: Identifying Density-Based Local Outliers. SIGMOD 2000.
- M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases. KDD'96.
- M. Ester, H.-P. Kriegel, and X. Xu. Knowledge discovery in large spatial databases: Focusing techniques for efficient class identification. SSD'95.
- D. Fisher. Knowledge acquisition via incremental conceptual clustering. Machine Learning, 2:139-172, 1987.
- D. Gibson, J. Kleinberg, and P. Raghavan. Clustering categorical data: An approach based on dynamic systems. VLDB'98.
- V. Ganti, J. Gehrke, R. Ramakrishan. CACTUS Clustering Categorical Data Using Summaries. KDD'99.

## References (2)

---

- D. Gibson, J. Kleinberg, and P. Raghavan. Clustering categorical data: An approach based on dynamic systems. In Proc. VLDB'98.
- S. Guha, R. Rastogi, and K. Shim. Cure: An efficient clustering algorithm for large databases. SIGMOD'98.
- S. Guha, R. Rastogi, and K. Shim. ROCK: A robust clustering algorithm for categorical attributes. In *ICDE'99*, pp. 512-521, Sydney, Australia, March 1999.
- A. Hinneburg, D. I. A. Keim: An Efficient Approach to Clustering in Large Multimedia Databases with Noise. KDD'98.
- A. K. Jain and R. C. Dubes. Algorithms for Clustering Data. Printice Hall, 1988.
- G. Karypis, E.-H. Han, and V. Kumar. CHAMELEON: A Hierarchical Clustering Algorithm Using Dynamic Modeling. *COMPUTER*, 32(8): 68-75, 1999.
- L. Kaufman and P. J. Rousseeuw. Finding Groups in Data: an Introduction to Cluster Analysis. John Wiley & Sons, 1990.
- E. Knorr and R. Ng. Algorithms for mining distance-based outliers in large datasets. VLDB'98.

# References (3)

---

- G. J. McLachlan and K.E. Bkasford. Mixture Models: Inference and Applications to Clustering. John Wiley and Sons, 1988.
- R. Ng and J. Han. Efficient and effective clustering method for spatial data mining. VLDB'94.
- L. Parsons, E. Haque and H. Liu, Subspace Clustering for High Dimensional Data: A Review, SIGKDD Explorations, 6(1), June 2004
- E. Schikuta. Grid clustering: An efficient hierarchical clustering method for very large data sets. Proc. 1996 Int. Conf. on Pattern Recognition
- G. Sheikholeslami, S. Chatterjee, and A. Zhang. WaveCluster: A multi-resolution clustering approach for very large spatial databases. VLDB'98.
- A. K. H. Tung, J. Han, L. V. S. Lakshmanan, and R. T. Ng. Constraint-Based Clustering in Large Databases, ICDT'01.
- A. K. H. Tung, J. Hou, and J. Han. Spatial Clustering in the Presence of Obstacles, ICDE'01
- H. Wang, W. Wang, J. Yang, and P.S. Yu. Clustering by pattern similarity in large data sets, SIGMOD'02
- W. Wang, Yang, R. Muntz, STING: A Statistical Information grid Approach to Spatial Data Mining, VLDB'97
- T. Zhang, R. Ramakrishnan, and M. Livny. BIRCH : An efficient data clustering method for very large databases. SIGMOD'96
- X. Yin, J. Han, and P. S. Yu, "LinkClus: Efficient Clustering via Heterogeneous Semantic Links", VLDB'06

---

Slides unused in class  
chjj

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

---

### Input:

**k**: the number of clusters;    **D**: a data set containing  $n$  objects.

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

### Method:

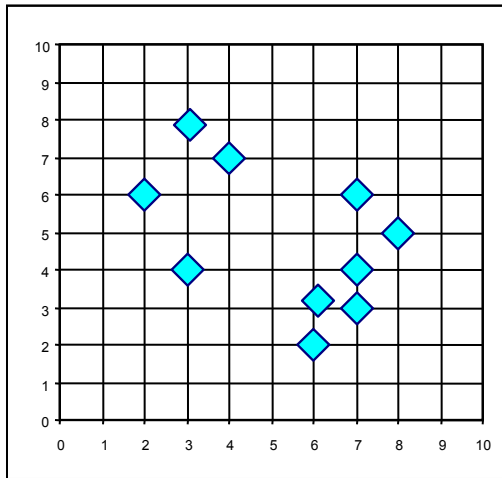
- (1) arbitrarily choose  $k$  objects in  $D$  as the initial representative objects or seeds;
- (2) **repeat**
- (3)    assign each remaining object to the cluster with the nearest representative obj.;
- (4)    randomly select a non-representative object,  $\mathbf{o}_{\text{random}}$  ;
- (5)    compute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

---

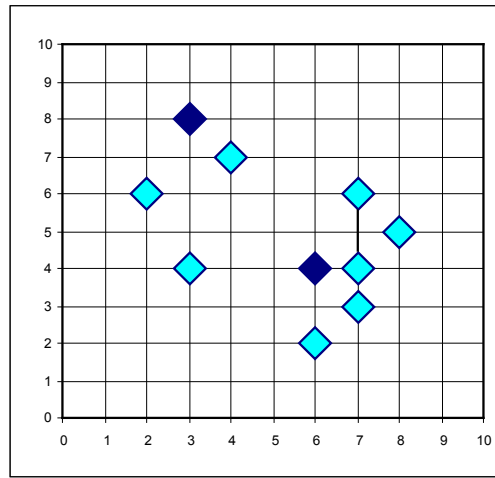
Slides unused in class

# A Typical K-Medoids Algorithm (PAM)

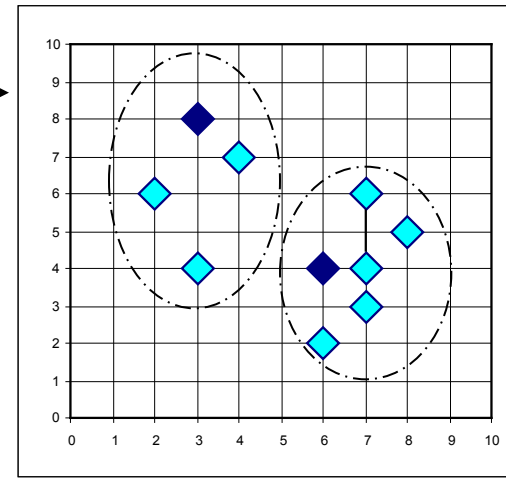
Total Cost = 20



Arbitrary  
choose  $k$   
object as  
initial  
medoids



Assign  
each  
remainin  
g object  
to  
nearest  
medoids

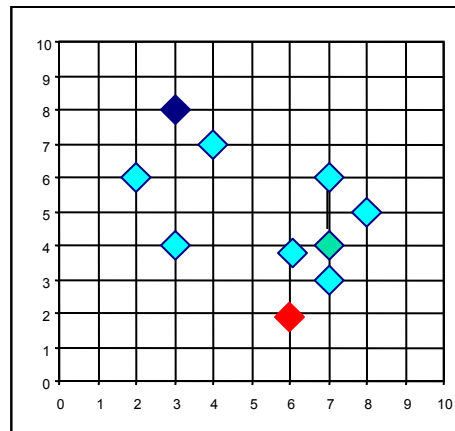


$K=2$

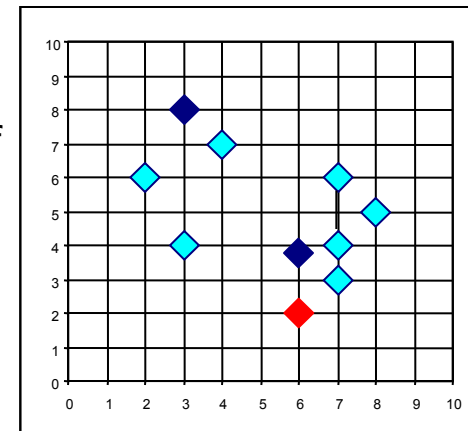
**Do loop  
Until no  
change**

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

Total Cost = 26



Compute  
total cost of  
swapping



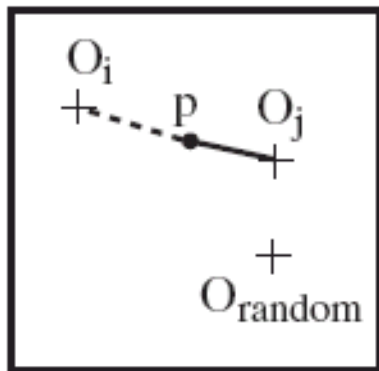
# PAM (Partitioning Around Medoids) (1987)

---

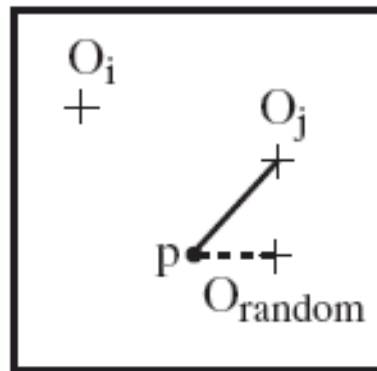
- PAM (Kaufman and Rousseeuw, 1987), built in Splus
- Use real object to represent the cluster
  - Select  $k$  representative objects arbitrarily
  - For each pair of non-selected object  $h$  and selected object  $i$ , calculate the total swapping cost  $TC_{ih}$
  - For each pair of  $i$  and  $h$ ,
    - If  $TC_{ih} < 0$ ,  $i$  is replaced by  $h$
    - Then assign each non-selected object to the most similar representative object
  - repeat steps 2-3 until there is no change

# PAM Clustering: Finding the Best Cluster Center

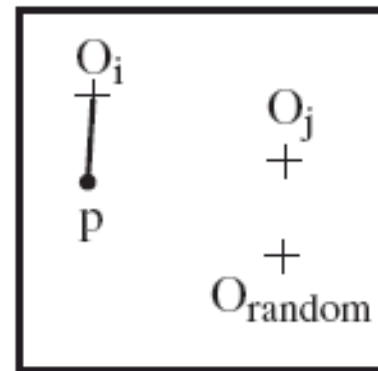
- Case 1:  $p$  currently belongs to  $o_j$ . If  $o_j$  is replaced by  $o_{\text{random}}$  as a representative object and  $p$  is the closest to one of the other representative object  $o_i$ , then  $p$  is reassigned to  $o_i$



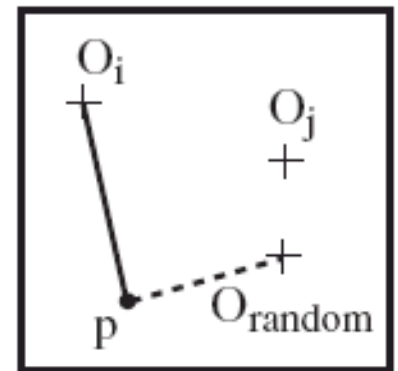
1. Reassigned to  $O_i$



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



3. No change



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

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

# What Is the Problem with PAM?

---

- Pam is more robust than k-means in the presence of noise and outliers because a medoid is less influenced by outliers or other extreme values than a mean
- Pam works efficiently for small data sets but does not **scale well** for large data sets.
  - $O(k(n-k)^2)$  for each iteration

where  $n$  is # of data,  $k$  is # of clusters

➔ Sampling-based method

CLARA(Clustering LARge Applications)

# CLARA (Clustering Large Applications) (1990)

---

- CLARA (Kaufmann and Rousseeuw in 1990)
  - Built in statistical analysis packages, such as SPlus
  - It draws *multiple samples* of the data set, applies *PAM* on each sample, and gives the best clustering as the output
- Strength: deals with larger data sets than *PAM*
- Weakness:
  - Efficiency depends on the sample size
  - A good clustering based on samples will not necessarily represent a good clustering of the whole data set if the sample is biased

# *CLARANS* ("Randomized" CLARA) (1994)

---

- *CLARANS* (A Clustering Algorithm based on Randomized Search) (Ng and Han'94)
  - Draws sample of neighbors dynamically
  - The clustering process can be presented as searching a graph where every node is a potential solution, that is, a set of  $k$  medoids
  - If the local optimum is found, *it* starts with new randomly selected node in search for a new local optimum
- Advantages: More efficient and scalable than both *PAM* and *CLARA*
- Further improvement: Focusing techniques and spatial access structures (Ester et al.'95)

# ROCK: Clustering Categorical Data

---

- ROCK: RObust Clustering using linkS
  - S. Guha, R. Rastogi & K. Shim, ICDE'99
- Major ideas
  - Use links to measure similarity/proximity
  - Not distance-based
- Algorithm: sampling-based clustering
  - Draw random sample
  - Cluster with links
  - Label data in disk
- Experiments
  - Congressional voting, mushroom data

# Similarity Measure in ROCK

- Traditional measures for categorical data may not work well, e.g., Jaccard coefficient
- Example: Two groups (clusters) of transactions
  - $C_1$ .  $\langle a, b, c, d, e \rangle$ :  $\{a, b, c\}, \{a, b, d\}, \{a, b, e\}, \{a, c, d\}, \{a, c, e\}, \{a, d, e\}, \{b, c, d\}, \{b, c, e\}, \{b, d, e\}, \{c, d, e\}$
  - $C_2$ .  $\langle a, b, f, g \rangle$ :  $\{a, b, f\}, \{a, b, g\}, \{a, f, g\}, \{b, f, g\}$
- Jaccard co-efficient may lead to wrong clustering result
  - $C_1$ : 0.2 ( $\{a, b, c\}, \{b, d, e\}$ ) to 0.5 ( $\{a, b, c\}, \{a, b, d\}$ )
  - $C_1$  &  $C_2$ : could be as high as 0.5 ( $\{a, b, c\}, \{a, b, f\}$ )
- Jaccard co-efficient-based similarity function:

$$Sim(T_1, T_2) = \frac{|T_1 \cap T_2|}{|T_1 \cup T_2|}$$

- Ex. Let  $T_1 = \{a, b, c\}$ ,  $T_2 = \{c, d, e\}$

$$Sim(T_1, T_2) = \frac{|\{c\}|}{|\{a, b, c, d, e\}|} = \frac{1}{5} = 0.2$$

# Link Measure in ROCK

- Clusters

- $C_1: \langle a, b, c, d, e \rangle: \{a, b, c\}, \{a, b, d\}, \{a, b, e\}, \{a, c, d\}, \{a, c, e\}, \{a, d, e\}, \{b, c, d\}, \{b, c, e\}, \{b, d, e\}, \{c, d, e\}$
- $C_2: \langle a, b, f, g \rangle: \{a, b, f\}, \{a, b, g\}, \{a, f, g\}, \{b, f, g\}$

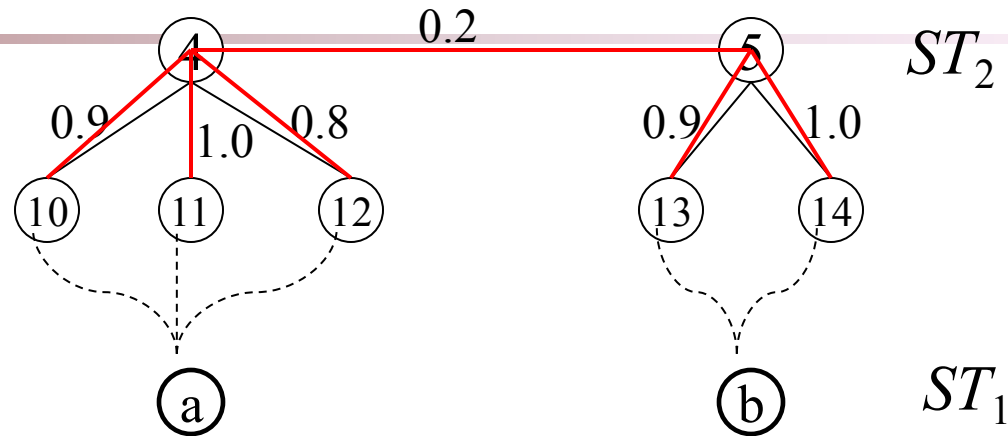
- Neighbors

- Two transactions are neighbors if  $\text{sim}(T_1, T_2) > \text{threshold}$
- Let  $T_1 = \{a, b, c\}$ ,  $T_2 = \{c, d, e\}$ ,  $T_3 = \{a, b, f\}$ 
  - $T_1$  connected to:  $\{a, b, d\}, \{a, b, e\}, \{a, c, d\}, \{a, c, e\}, \{b, c, d\}, \{b, c, e\}, \{a, b, f\}, \{a, b, g\}$
  - $T_2$  connected to:  $\{a, c, d\}, \{a, c, e\}, \{a, d, e\}, \{b, c, e\}, \{b, d, e\}, \{b, c, d\}$
  - $T_3$  connected to:  $\{a, b, c\}, \{a, b, d\}, \{a, b, e\}, \{a, b, g\}, \{a, f, g\}, \{b, f, g\}$

- Link Similarity

- Link similarity between two transactions is the # of common neighbors
- $\text{link}(T_1, T_2) = 4$ , *since they have 4 common neighbors*
  - $\{a, c, d\}, \{a, c, e\}, \{b, c, d\}, \{b, c, e\}$
- $\text{link}(T_1, T_3) = 3$ , *since they have 3 common neighbors*
  - $\{a, b, d\}, \{a, b, e\}, \{a, b, g\}$

# Aggregation-Based Similarity Computation



For each node  $n_k \in \{n_{10}, n_{11}, n_{12}\}$  and  $n_l \in \{n_{13}, n_{14}\}$ , their path-based similarity  $sim_p(n_k, n_l) = s(n_k, n_4) \cdot s(n_4, n_5) \cdot s(n_5, n_l)$ .

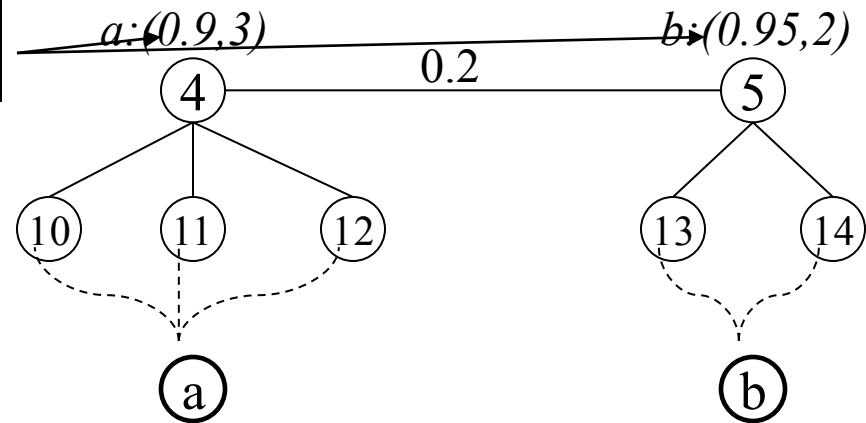
$$sim(n_a, n_b) = \frac{\sum_{k=10}^{12} s(n_k, n_4)}{3} \cdot s(n_4, n_5) \cdot \frac{\sum_{l=13}^{14} s(n_l, n_5)}{2} = 0.171$$

takes  $O(3+2)$  time

After aggregation, we reduce quadratic time computation to linear time computation.

# Computing Similarity with Aggregation

Average similarity  
and total weight



$sim(n_a, n_b)$  can be computed  
from aggregated similarities


$$sim(n_a, n_b) = avg\_sim(n_a, n_4) \times s(n_4, n_5) \times avg\_sim(n_b, n_5) \\ = 0.9 \times 0.2 \times 0.95 = 0.171$$

To compute  $sim(n_a, n_b)$ :

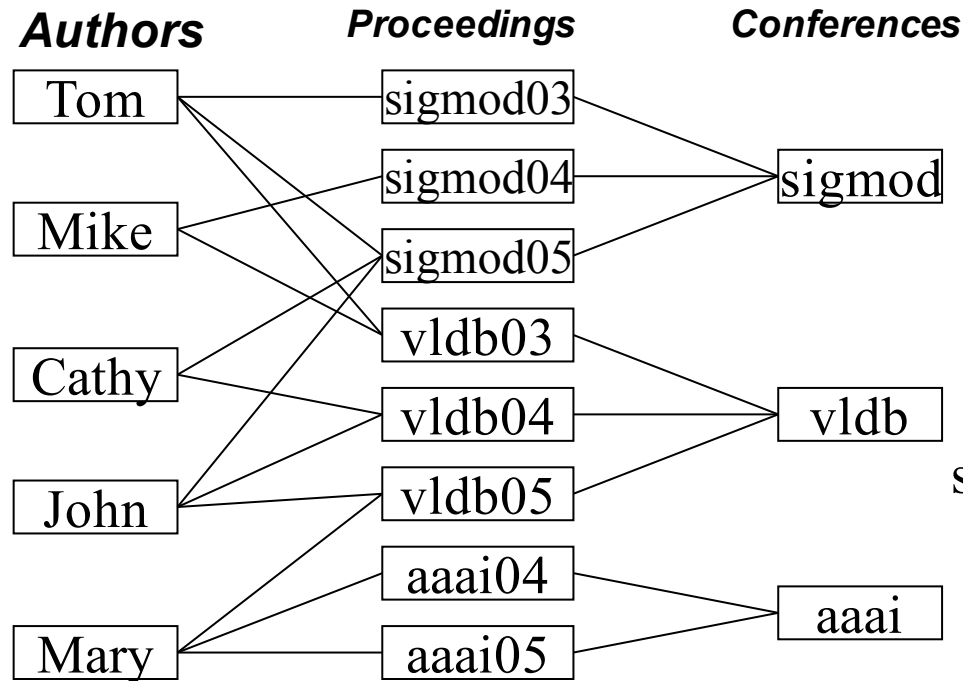
- Find all pairs of sibling nodes  $n_i$  and  $n_{j'}$  so that  $n_a$  linked with  $n_i$  and  $n_b$  with  $n_{j'}$ .
- Calculate similarity (and weight) between  $n_a$  and  $n_b$  w.r.t.  $n_i$  and  $n_{j'}$ .
- Calculate weighted average similarity between  $n_a$  and  $n_b$  w.r.t. all such pairs.

# Chapter 10. Cluster Analysis: Basic Concepts and Methods

---

- Cluster Analysis: Basic Concepts
- Overview of Clustering Methods
- Partitioning Methods
- Hierarchical Methods
- Density-Based Methods 
- Grid-Based Methods
- Summary

# Link-Based Clustering: Calculate Similarities Based On Links



- The similarity between two objects  $x$  and  $y$  is defined as the average similarity between objects linked with  $x$  and those with  $y$ :

$$\text{sim}(a, b) = \frac{C}{|I(a)||I(b)|} \sum_{i=1}^{|I(a)|} \sum_{j=1}^{|I(b)|} \text{sim}(I_i(a), I_j(b))$$

- Issue: Expensive to compute:
  - For a dataset of  $N$  objects and  $M$  links, it takes  $O(N^2)$  space and  $O(M^2)$  time to compute all similarities.

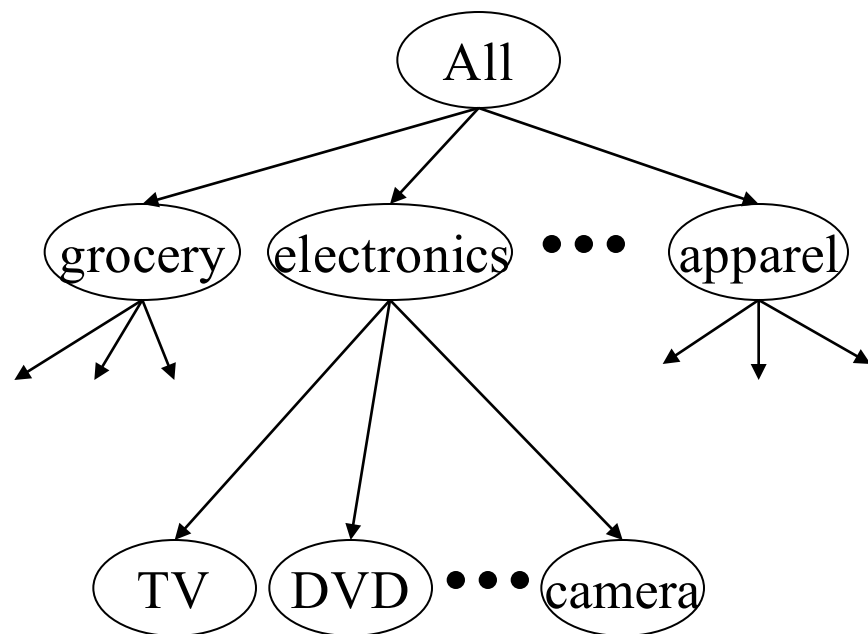
Jeh & Widom, KDD'2002: *SimRank*

Two objects are similar if they are linked with the same or similar objects

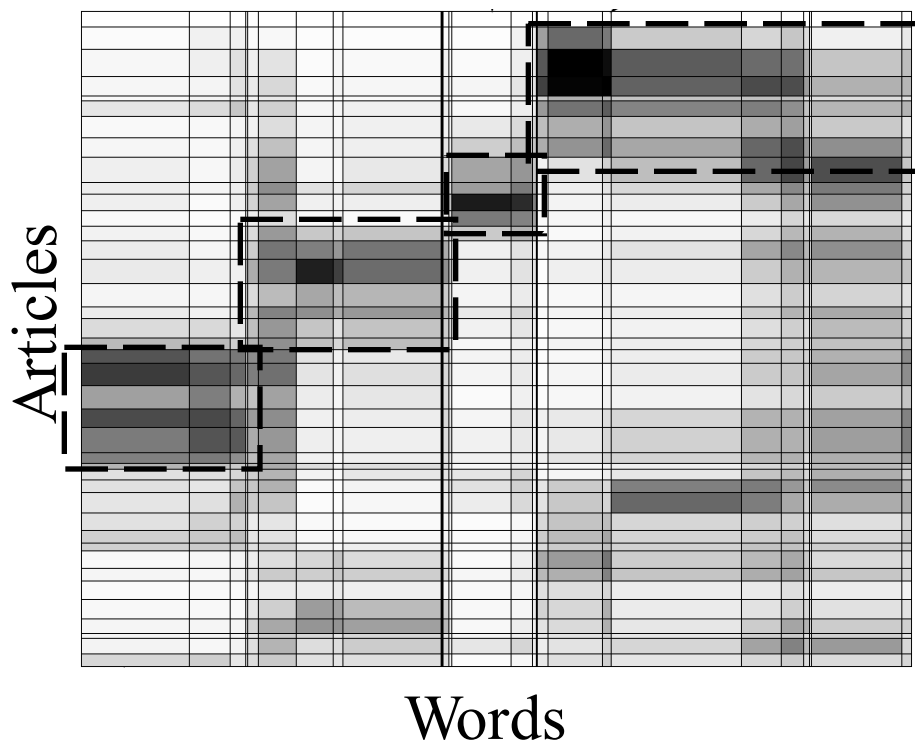
# Observation 1: Hierarchical Structures

- Hierarchical structures often exist naturally among objects (e.g., taxonomy of animals)

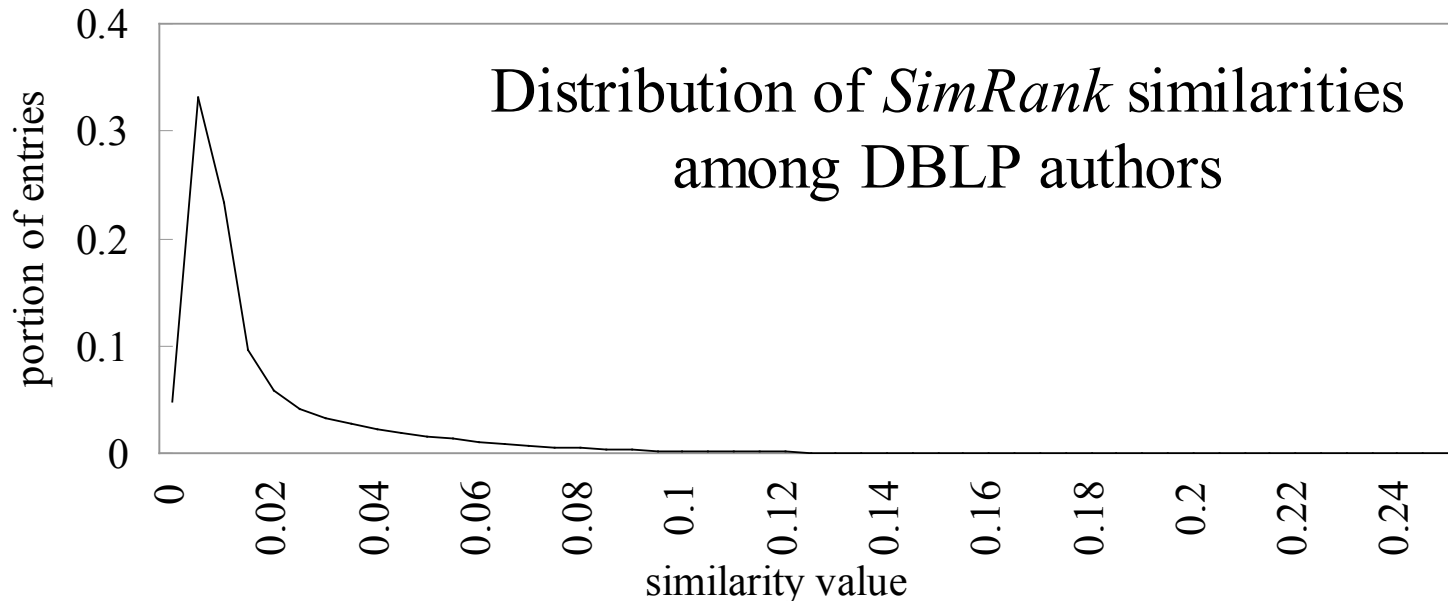
A hierarchical structure of products in Walmart



Relationships between articles and words (Chakrabarti, Papadimitriou, Modha, Faloutsos, 2004)

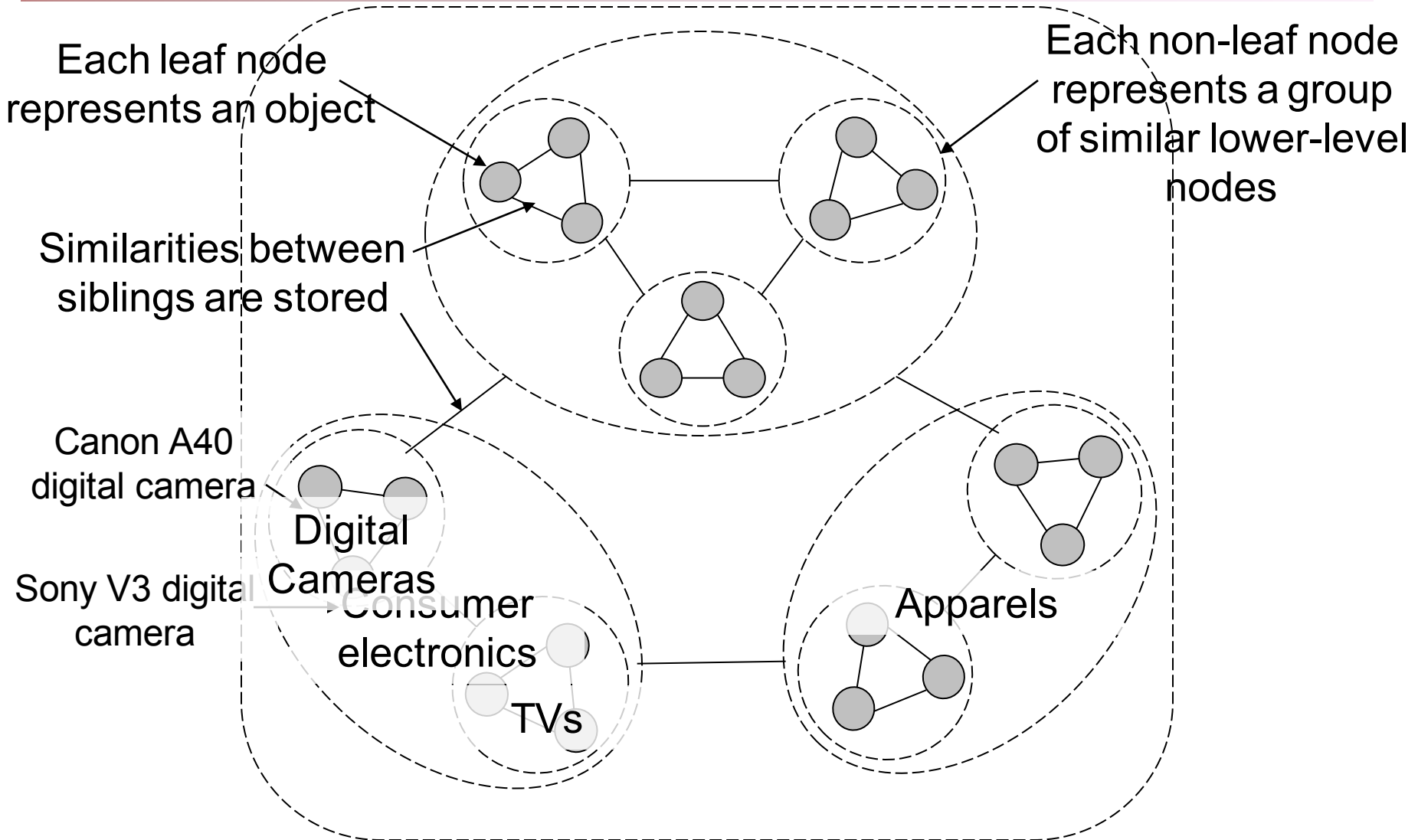


## Observation 2: Distribution of Similarity

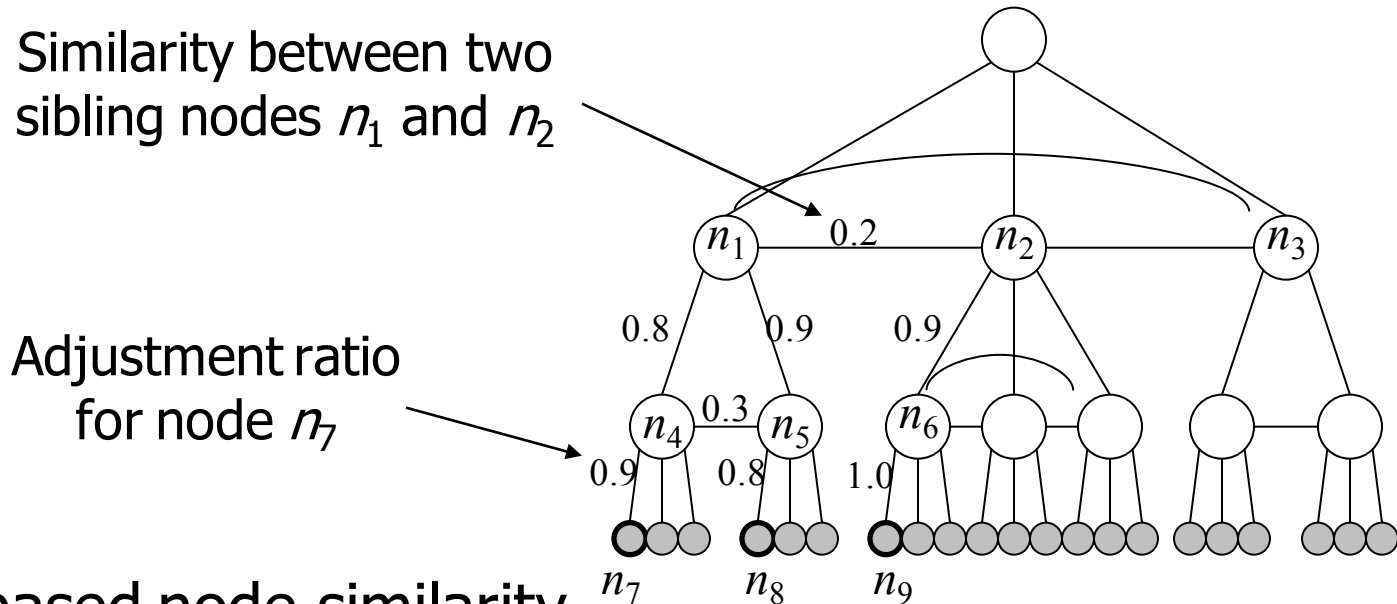


- Power law distribution exists in similarities
  - 56% of similarity entries are in  $[0.005, 0.015]$
  - 1.4% of similarity entries are larger than 0.1
  - Can we design a data structure that stores the significant similarities and compresses insignificant ones?

# A Novel Data Structure: SimTree



# Similarity Defined by SimTree



- Path-based node similarity

- $sim_p(n_7, n_8) = s(n_7, n_4) \times s(n_4, n_5) \times s(n_5, n_8)$

- Similarity between two nodes is the average similarity between objects linked with them in other SimTrees

- Adjust/ ratio for  $x = \frac{\text{Average similarity between } x \text{ and all other nodes}}{\text{Average similarity between } x\text{'s parent and all other nodes}}$

# LinkClus: Efficient Clustering via Heterogeneous Semantic Links

---

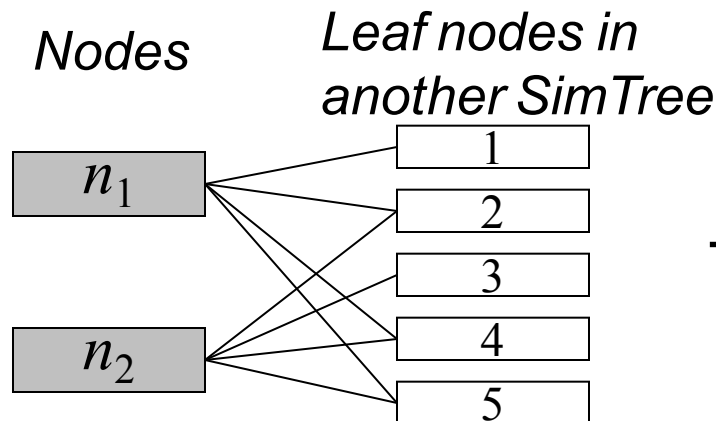
## Method

- Initialize a SimTree for objects of each type
- Repeat until stable
  - For each SimTree, update the similarities between its nodes using similarities in other SimTrees
    - Similarity between two nodes  $x$  and  $y$  is the average similarity between objects linked with them
  - Adjust the structure of each SimTree
    - Assign each node to the parent node that it is most similar to

For details: X. Yin, J. Han, and P. S. Yu, "LinkClus: Efficient Clustering via Heterogeneous Semantic Links", VLDB'06

# Initialization of SimTrees

- Initializing a SimTree
  - Repeatedly find groups of tightly related nodes, which are merged into a higher-level node
- Tightness of a group of nodes
  - For a group of nodes  $\{n_1, \dots, n_k\}$ , its tightness is defined as the number of leaf nodes in other SimTrees that are connected to all of  $\{n_1, \dots, n_k\}$

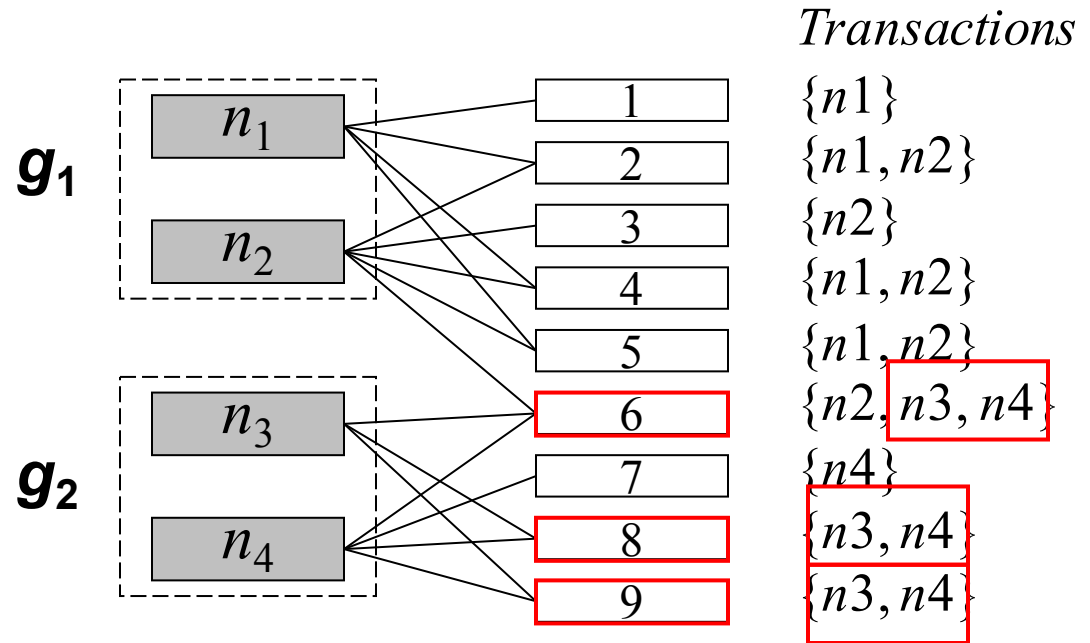


The tightness of  $\{n_1, n_2\}$  is 3

# Finding Tight Groups by Freq. Pattern Mining

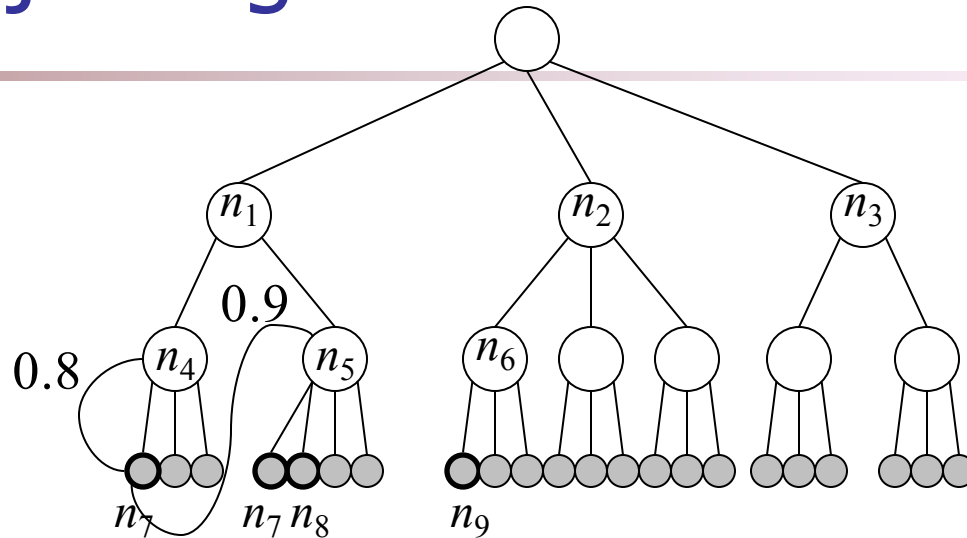
- Finding tight groups  $\longrightarrow$  Frequent pattern mining  
*Reduced to*

The tightness of a group of nodes is the support of a frequent pattern



- Procedure of initializing a tree
  - Start from leaf nodes (level-0)
  - At each level  $l$ , find non-overlapping groups of similar nodes with frequent pattern mining

# Adjusting SimTree Structures



- After similarity changes, the tree structure also needs to be changed
  - If a node is more similar to its parent's sibling, then move it to be a child of that sibling
  - Try to move each node to its parent's sibling that it is most similar to, under the constraint that each parent node can have at most  $c$  children

# Complexity

---

For two types of objects,  $N$  in each, and  $M$  linkages between them.

|                           | Time             | Space    |
|---------------------------|------------------|----------|
| Updating similarities     | $O(M(\log N)^2)$ | $O(M+N)$ |
| Adjusting tree structures | $O(N)$           | $O(N)$   |
|                           |                  |          |
| <i>LinkClus</i>           | $O(M(\log N)^2)$ | $O(M+N)$ |
| <i>SimRank</i>            | $O(M^2)$         | $O(N^2)$ |

# Experiment: Email Dataset

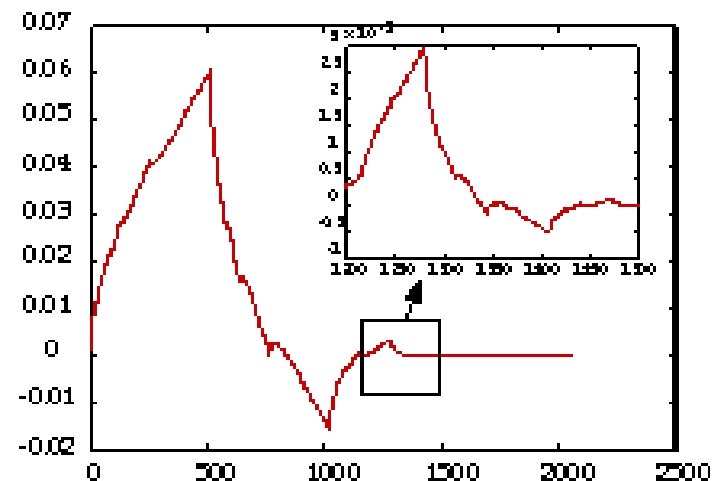
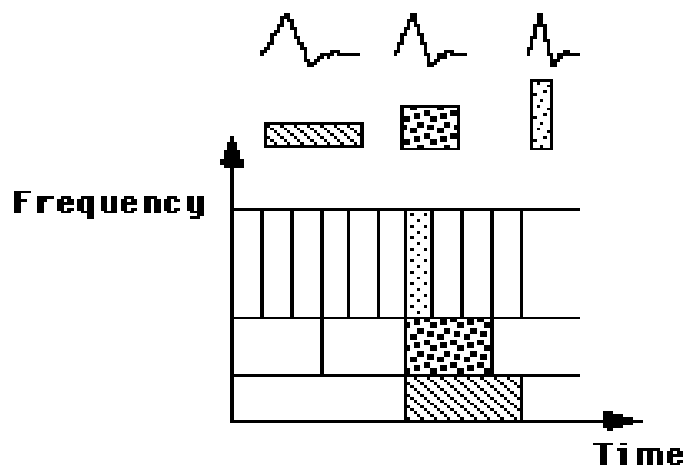
- F. Nielsen. Email dataset.  
[www.imm.dtu.dk/~rem/data/Email-1431.zip](http://www.imm.dtu.dk/~rem/data/Email-1431.zip)
- 370 emails on conferences, 272 on jobs, and 789 spam emails
- Accuracy: measured by manually labeled data
- Accuracy of clustering: % of pairs of objects in the same cluster that share common label

| <i>Approach</i> | <i>Accuracy</i> | <i>time (s)</i> |
|-----------------|-----------------|-----------------|
| LinkClus        | 0.8026          | 1579.6          |
| SimRank         | 0.7965          | 39160           |
| ReCom           | 0.5711          | 74.6            |
| F-SimRank       | 0.3688          | 479.7           |
| CLARANS         | 0.4768          | 8.55            |

- Approaches compared:
  - SimRank (Jeh & Widom, KDD 2002): Computing pair-wise similarities
  - SimRank with FingerPrints (F-SimRank): Fogaras & R'acz, WWW 2005
    - pre-computes a large sample of random paths from each object and uses samples of two objects to estimate SimRank similarity
  - ReCom (Wang et al. SIGIR 2003)
    - Iteratively clustering objects using cluster labels of linked objects

# WaveCluster: Clustering by Wavelet Analysis (1998)

- Sheikholeslami, Chatterjee, and Zhang (VLDB'98)
- A multi-resolution clustering approach which applies wavelet transform to the feature space; both grid-based and density-based
- Wavelet transform: A signal processing technique that decomposes a signal into different frequency sub-band
  - Data are transformed to preserve relative distance between objects at different levels of resolution
  - Allows natural clusters to become more distinguishable



# The WaveCluster Algorithm

---

- How to apply wavelet transform to find clusters
  - Summarizes the data by imposing a multidimensional grid structure onto data space
  - These multidimensional spatial data objects are represented in a n-dimensional feature space
  - Apply wavelet transform on feature space to find the dense regions in the feature space
  - Apply wavelet transform multiple times which result in clusters at different scales from fine to coarse
- Major features:
  - Complexity  $O(N)$
  - Detect arbitrary shaped clusters at different scales
  - Not sensitive to noise, not sensitive to input order
  - Only applicable to low dimensional data

# Quantization & Transformation

- Quantize data into m-D grid structure then wavelet transform
  - a) scale 1: high resolution
  - b) scale 2: medium resolution
  - c) scale 3: low resolution

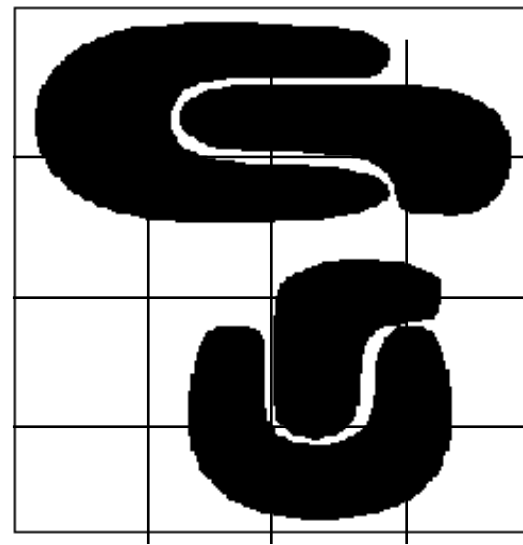
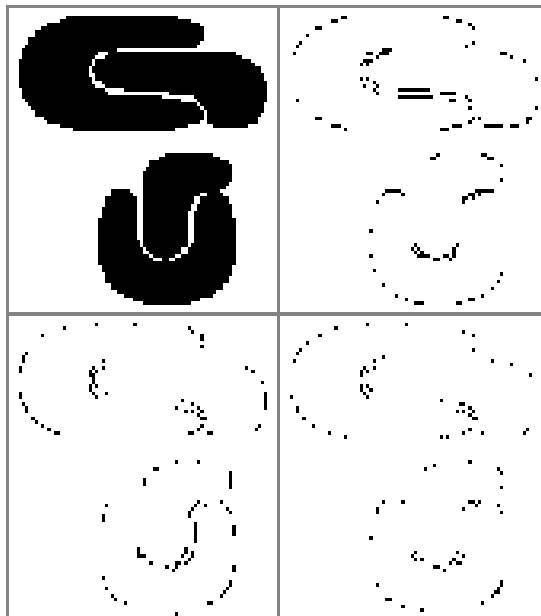


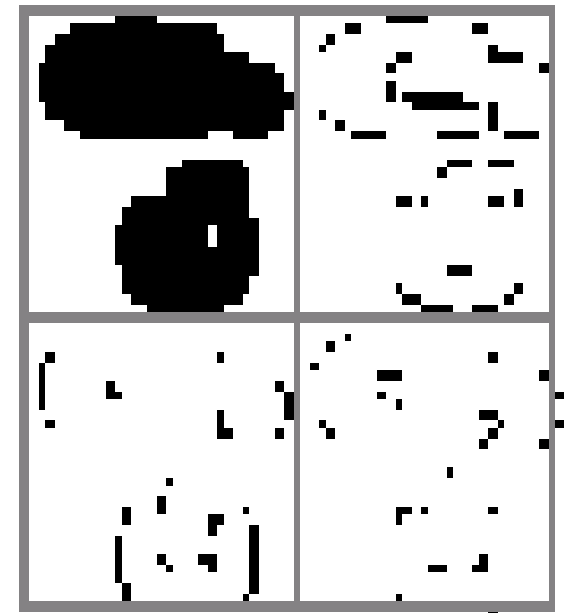
Figure 1: A sample 2-dimensional feature space.



a)



b)



c)