Introduction to Data Mining

Lecture6 Clustering

Jun Huang

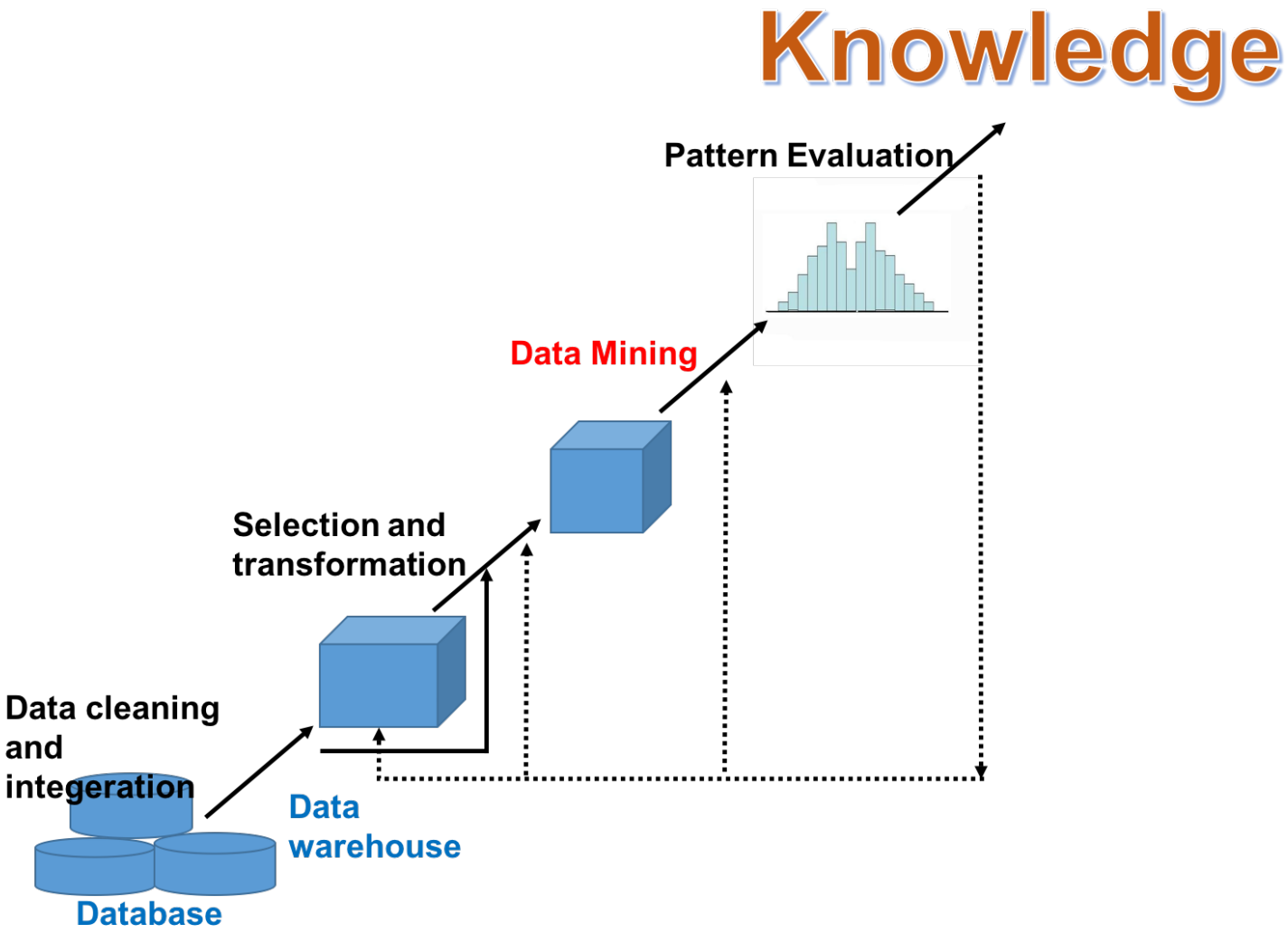
Anhui University of Technology

Spring 2018

huangjun_cs@163.com

KDD Process

Data Mining-Core of Knowledge discovery process



Cluster Analysis

- What is Cluster Analysis?
- Types of Data in Cluster Analysis
- A categorization of major clustering methods
- Partitioning methods
- Hierarchical methods
- Density-based methods
- Grid-based methods
- Outlier analysis
- Summary

What is Cluster Analysis?

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

Examples of Clustering Applications

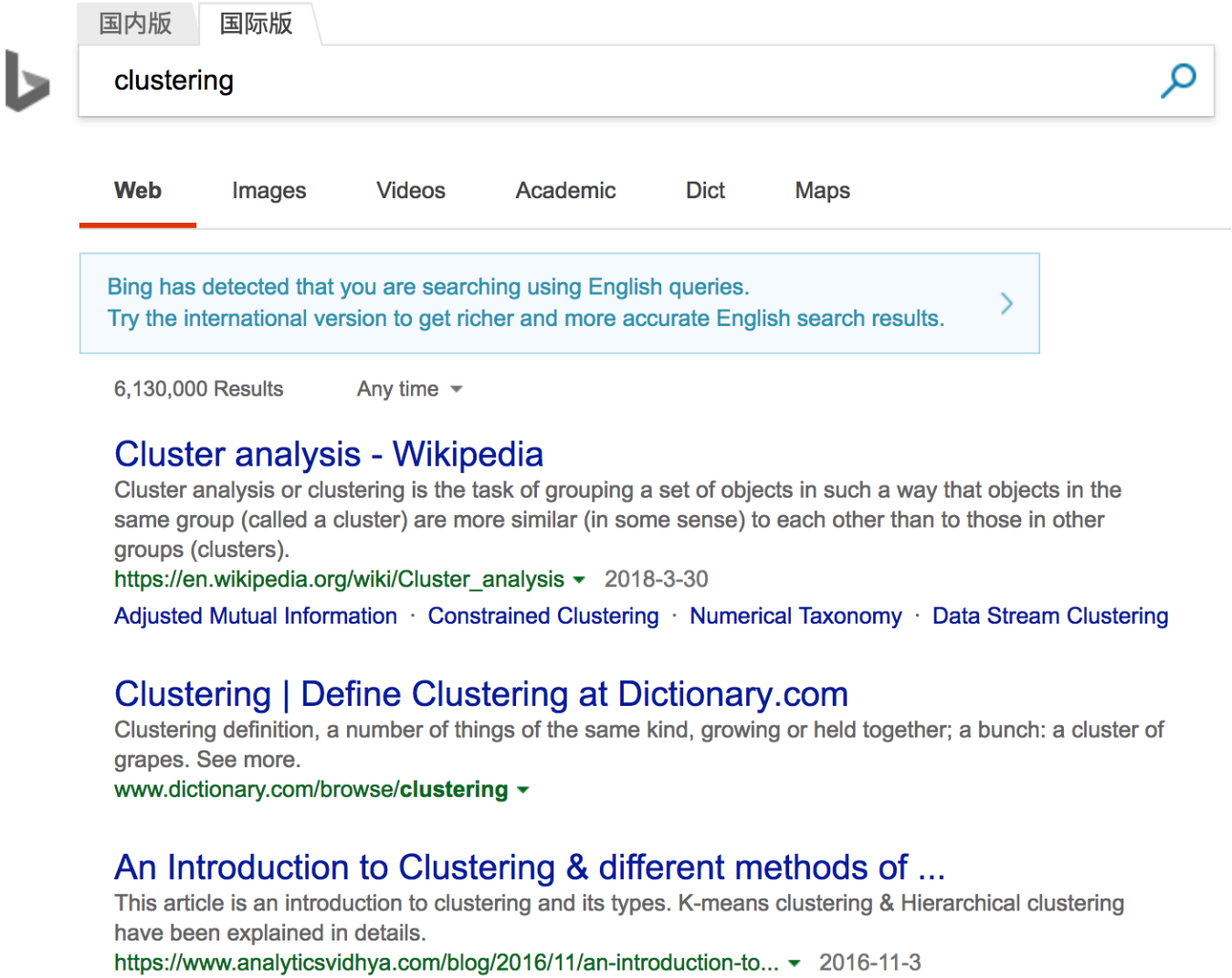
- Marketing: help marketers discover distinct groups according to their customer databases, and then use this knowledge to develop targeted marketing programes
- Land use:Identification of areas of similar land use in an earth observation database
- Insurance: identifying groups of motor insurance policy holders with a high average claim cost
- City-planning: identifying groups of houses according to their house type, value, and geographical location

Examples of Clustering Applications

- Earth-quake studies: observed earth quake epicenters should be clustered along continent faults
- Biology: categorize genes with similar functionality
- WWW
  - Document classification
  - Cluster weblog data to discover groups of similar accessing patterns
  - Topic detection
  - Community detection

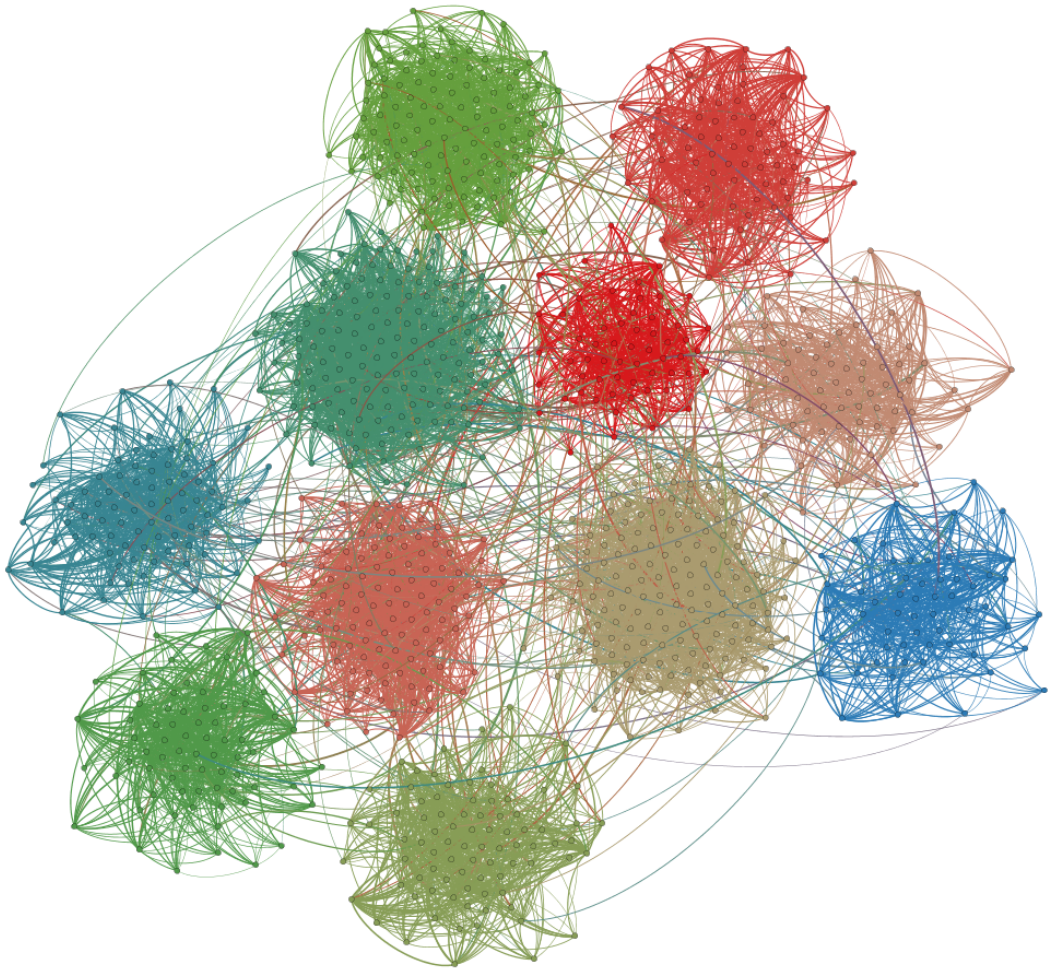
Examples of Clustering Applications

Information Retrieval



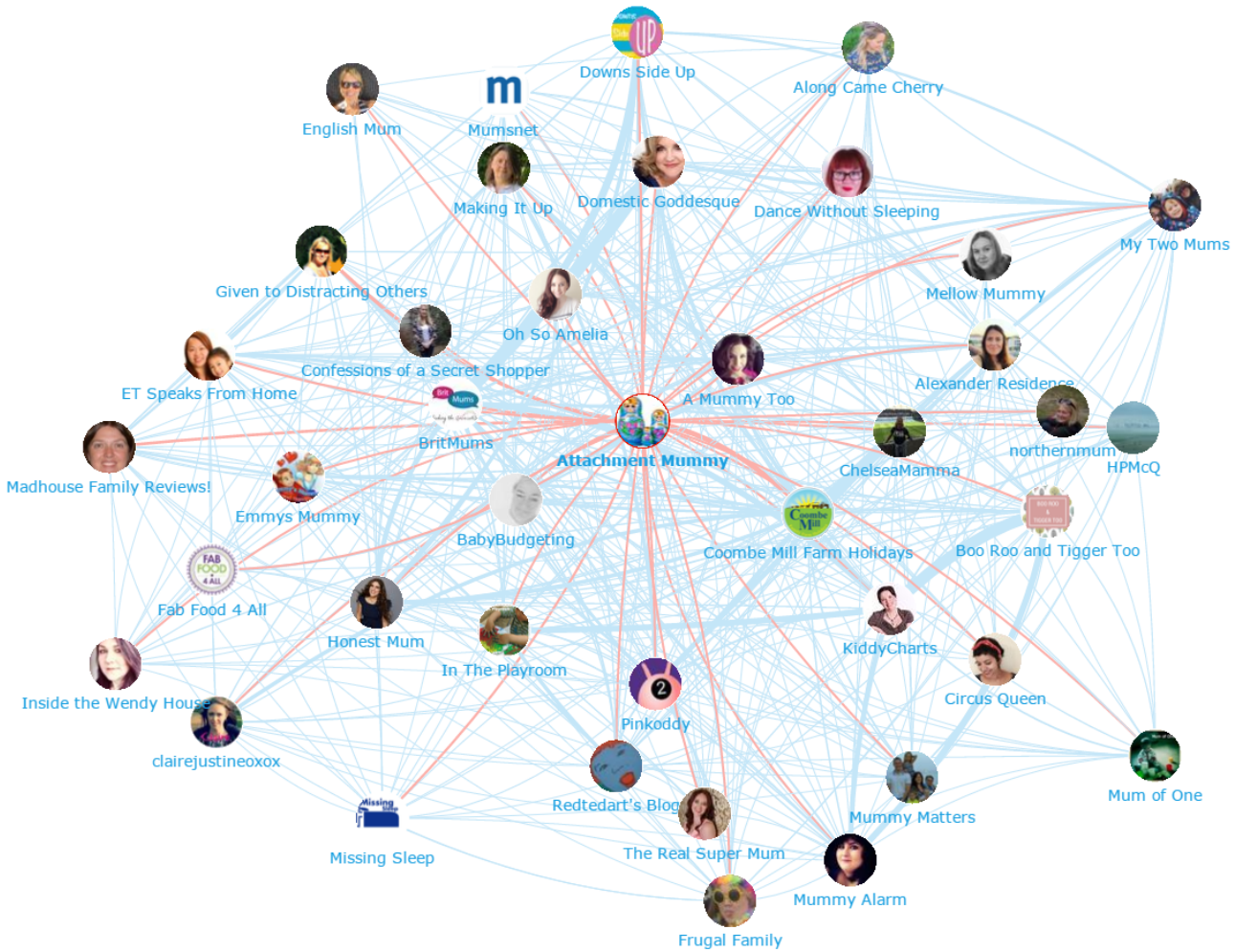
Examples of Clustering Applications
Topic/Event Detection

Examples of Clustering Applications
Community Detection



Examples of Clustering Applications

Community Detection



Clustering: Rich Applications and Multidisciplinary Efforts

- Patter Recognition
- GIS
  - Create thematic maps in GIS by clustering feature spaces
  - Detect spatial clusters or for other spatial mining tasks
- Image Processing
- Economic Science (especially marketing research)
- Software package:S-plus, SPSS, SAS, R, Python, Matlab

What is Good Clustering?

- A good clustering method will produce high quality clusters with
  - high intra-class similarity
  - low inter-class similarity
- The quality of a clustering result depends on both the similarity measure used by the method and its implementation
- The quality of a clustering method is also measured by its ability to discover some or all of the hidden patterns

Requirements of Clustering

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

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 very different for **interval-scaled, boolean, categorical, ordinal ratio, and vector** variables
- Weights may be assigned to different variables based on applications and data semantics
- It is hard to define "similar enough" or "good enough"
- The answer is typically highly **subjective**

Data Structures

- Data matrix

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

- Dissimilarity matrix

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

Types of Data in Cluster Analysis

- Interval-scaled variables
- Binary variables
- Nominal/Categorical variables
- Ordinal variables
- Ratio-scaled variables
- Variables of mixed types

Interval-valued Variables

- Standardize data
- Calculate the mean absolute deviation:

$$s_f = \frac{1}{n}(|x_{1f} - m_f| + |x_{2f} - m_f| + ... + |x_{nf} - m_f|)$$

- where $m_f = \frac{1}{n}(x_{1f} + x_{2f} + x_{nf})$
- Calculate the standardized measurement (z-score)

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

- Using mean absolute deviation is more robust than using standard deviation

Similarity and Dissimilarity Between Objects

- Distances are normally used to measure the similarity or dissimilarity between two data objects
- Some popular ones include: Minikowski distance

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

- where $i = (x_{i1}, x_{i2}, ..., x_{ip})$ and $j = (x_{j1}, x_{j2}, ..., x_{jp})$ are two p-dimensional data objects, and q is a positive integer
- If $q = 1$, d is Manhattan distance

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

Similarity and Dissimilarity Between Objects

- If $q = 2$, d is Euclidean distance

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

- Properties
  - $d(i,j) \geq 0$
  - $d(i,i) = 0$
  - $d(i,j) = d(j,i)$
  - $d(i,j) \leq d(i,k) + d(k,j)$
- Also, one can use weighted distance

Binary Variables

Ojbect	attr1	attr2	attr3	attr4
object1	1	1	0	1
object2	0	1	1	1

Binary Variables

- A contingency table for binary data

		object j		
		1	0	sum
object i	1	a	b	$a+b$
	0	c	d	$c+d$
	sum	$a+c$	$b+d$	p

- Distance measure for symmetric binary variables:

$$d(i,j) = \frac{b+c}{a+b+c+d}$$

- Distance measure for asymmetric binary variables:

$$d(i,j) = \frac{b+c}{a+b+c}$$

Dissimilarity between Binary Variables

- Example

Name	Gender	Fever	Cough	Test1	Test2	Test3	Test4
Jack	M	Y(1)	N(0)	P(1)	N(0)	N(0)	N(0)
Mary	F	Y(1)	N(0)	P(1)	N(0)	P(1)	N(0)
Jim	M	Y(1)	P(1)	N(0)	N(0)	N(0)	N(0)

- gender is a symmetric attribute
- the remaining attributes are asymmetric binary
- let the values Y and P be set 1, and the value N be set to 0
- $d(jack, mary) = \frac{0+1}{2+0+1} = 0.33$
- $d(jack, jim) = \frac{1+1}{1+1+1} = 0.67$
- $d(jim, mary) = \frac{1+2}{1+1+2} = 0.75$

Nominal/Categorical Variables

- A generalization of binary variable in that it can take more than 2 states, e.g., Code-A, Code-B, Code-C

ID	Test1 (分类的)	Test2 (序数的)	Test3 (比例标度度)
1	Code-A	优秀	445
2	Code-B	一般	22
3	Code-C	好	164
4	Code-A	优秀	1210

Nominal/Categorical Variables

- A generalization of binary variable in that it can take more than 2 states, e.g., Code-A, Code-B, Code-C
- Method 1: Simple matching
- m: # of matches, p: total # of nominal variables

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

- $d(i,j)$ is caluclated based on attribute Test1

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

Nominal/Categorical Variables

- A generalization of binary variable in that it can take more than 2 states, e.g., Code-A, Code-B, Code-C
- Method 2: use a large number of binary variables
  - creating a new binary variable for each of the M nominal states
  - e.g., attribute Test1 has three states, we can define 3 new binary variables

ID	Test1 (分类的)	b1	b2	b3
1	Code-A	1	0	0
2	Code-B	0	1	0
3	Code-C	0	0	1
4	Code-A	1	0	0

Ordinal Variables

- An ordianl variable can be discrete or continuous
- Order is important, e.g, rank

ID	Test1 (分类的)	Test2 (序数的)	Test3 (比例标度度)
1	Code-A	优秀	445
2	Code-B	一般	22
3	Code-C	好	164
4	Code-A	优秀	1210

Ordinal Variables

- An ordianl variable can be **discrete** or **continuous**
- **Order** is important, e.g, rank
- Can be treated like interval-scaled
  - replace x_{if} by their rank, $r_{if} \in \{1, ..., M_f\}$
  - map the range of each variable onto [0,1] by replacing the i-th object in the f-th variable by

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

  - compute the dissimilarity using methods for interval-scaled variables

Ordinal Variables

ID	Test1 (分类的)	Test2 (序数的)	Test3 (比例标度度)
1	Code-A	优秀	445
2	Code-B	一般	22
3	Code-C	好	164
4	Code-A	优秀	1210

- Test2 has three states
- Thus, $M_f = 3$, then replace x_{if} by their **rank**, r_{if}, i.e., $3, 1, 2, 3$, and r_{if} will be $1, 0, 0.5, 1$

- compute the dissimilarity by **Euclidean distance**

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

Ratio-Scaled Variables

- Ratio-scaled variable: a positive measurement on a **nonlinear scale**, approximately at **exponential scale**, such as Ae^{Bt} or Ae^{-Bt}
- e.g., Test3

ID	Test1 (分类的)	Test2 (序数的)	Test3 (比例标度度)
1	Code-A	优秀	445
2	Code-B	一般	22
3	Code-C	好	164
4	Code-A	优秀	1210

Ratio-Scaled Variables

- Ratio-scaled variable: a positive measurement on a nonlinear scale, approximately at exponential scale, such as Ae^{Bt} or Ae^{-Bt}
- Methods:
  - treat them like **interval-scaled** variables: not a good choice
  - apply logarithmic transformation

$$y_{if} = \log(x_{if})$$

  - replace x_{if} by y_{if}
  - treat them as **continuous ordianl data** and treat their rank as **interval-scaled**

Variables of Mixed Tyes

- A database may contain**all the six types of variables**
  - sysmetric binay, asymmetric binay, nominal, ordinal, interval and ratio

ID	Test1 (分类的)	Test2 (序数的)	Test3 (比例标度度)
1	Code-A	优秀	445
2	Code-B	一般	22
3	Code-C	好	164
4	Code-A	优秀	1210

Variables of Mixed Tyes

- A database may contain all the six types of variables
- One may use **a weighted formula to combine their effects**

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

- $\delta_{ij}^{(f)} = 0$ if x_{if} or x_{jf} is missing, or $x_{if} = x_{jf} = 0$ and f is asymmetric attribute; otherwise, $\delta_{ij}^{(f)} = 1$
  - f is binary or nomial: $d_{ij}^{f} = 0$ if $x_{if} = x_{jf}$, or $d_{ij}^{f} = 1$ otherwise
  - f is interval-based: use the normalized distance $d_{ij}^{f} = \frac{|x_{if} - x_{jf}|}{\max_h x_{hf} - \min_h x_{hf}}$
  - f is ordinal: compute ranks r_{if} and treat $z_{if} = \frac{r_{if}-1}{M_f - 1}$ as interval-scaled
  - f is ratio-scaled: transform f and treat f as interval-scaled

Exercise

- Please compute the dissimilarity matrix for the data set

ID	Test1 (categorical)	Test2 (ordinal)	Test3 (ratio-scaled)
1	A	excellent	445
2	B	fair	22
3	C	good	164
4	A	excellent	1,210

Solution

- For test1, use simple matching

0			
d(2,1)	0		
d(3,1)	d(3,2)	0	
d(4,1)	d(4,2)	d(4,3)	0

=

0			
1	0		
1	1	0	
0	1	1	0

- For test2

0			
1	0		
0.5	0.5	0	
0	1	0.5	0

Solution

- For test3, use log transformation
  - convert test3 to 2.65, 1.34, 2.21, 3.08
  - normalize to 0.75, 0, 0.5, 1

0			
0.75	0		
0.25	0.5	0	
0.25	1	0.5	0

- Dissimilarity matrix, $\delta_{ij}^{(f)} = 1$

0			
0.92	0		
0.58	0.67	0	
0.08	1	0.67	0

Other methods for similarity calculation

- In many applications, e.g., information retrieval, document categorization..., there are many attributes
- We may use **cosine similarity**

$$s(\mathbf{x}, \mathbf{y}) = \frac{\mathbf{x}^T \mathbf{y}}{\|\mathbf{x}\|_2 \|\mathbf{y}\|_2}$$

- e.g., $\mathbf{x} = [1, 1, 0, 0]$ and $\mathbf{y} = [0, 1, 1, 0]$, then

$$s(\mathbf{x}, \mathbf{y}) = \frac{0 + 1 + 0 + 0}{\sqrt{2}\sqrt{2}} = 0.5$$

- other measures, e.g., **Hamming distance, Chebyshev distance, Jaccard similarity coefficient,...**

Major Clustering Approaches (1)

- 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, ROCK, CHAMELEON
- Density-based approach:
  - Based on connectivity and density functions
  - Typical methods: DBSCAN, OPTICS, DenClue

Major Clustering Approaches (2)

- Grid-based approach:
  - based on a multiple-level granularity structure
  - Typical methods: STING, WaveCluster, CLIQUE
- Probabilistic Model-Based approach
  - Typical methods: EM

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

- **Centroid**: the middle of a cluster

$$C = \frac{\sum_{i=1}^{N} t_i}{N}$$

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

$$R = \sqrt{\frac{\sum_{i=1}^{N} (t_i - C)^2}{N}}$$

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

$$D = \sqrt{\frac{\sum_{i=1}^{N} \sum_{j=1}^{N} (t_i - t_j)^2}{N(N-1)}}$$

78

Partitioning Algorithms: Basic Concept

- **Partitioning method**: construct a partition of a database D of n objects into a set of k clusters, s.t., min sum of squared distance

$$\sum_{m=1}^{k} \sum_{t_{mi} \in K_m} (C_m - t_{mi})^2$$

- Given a k, find a partition of k clusters that optimizes the chosen partitioning criterion
  - k-means (MacQueen'67): 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

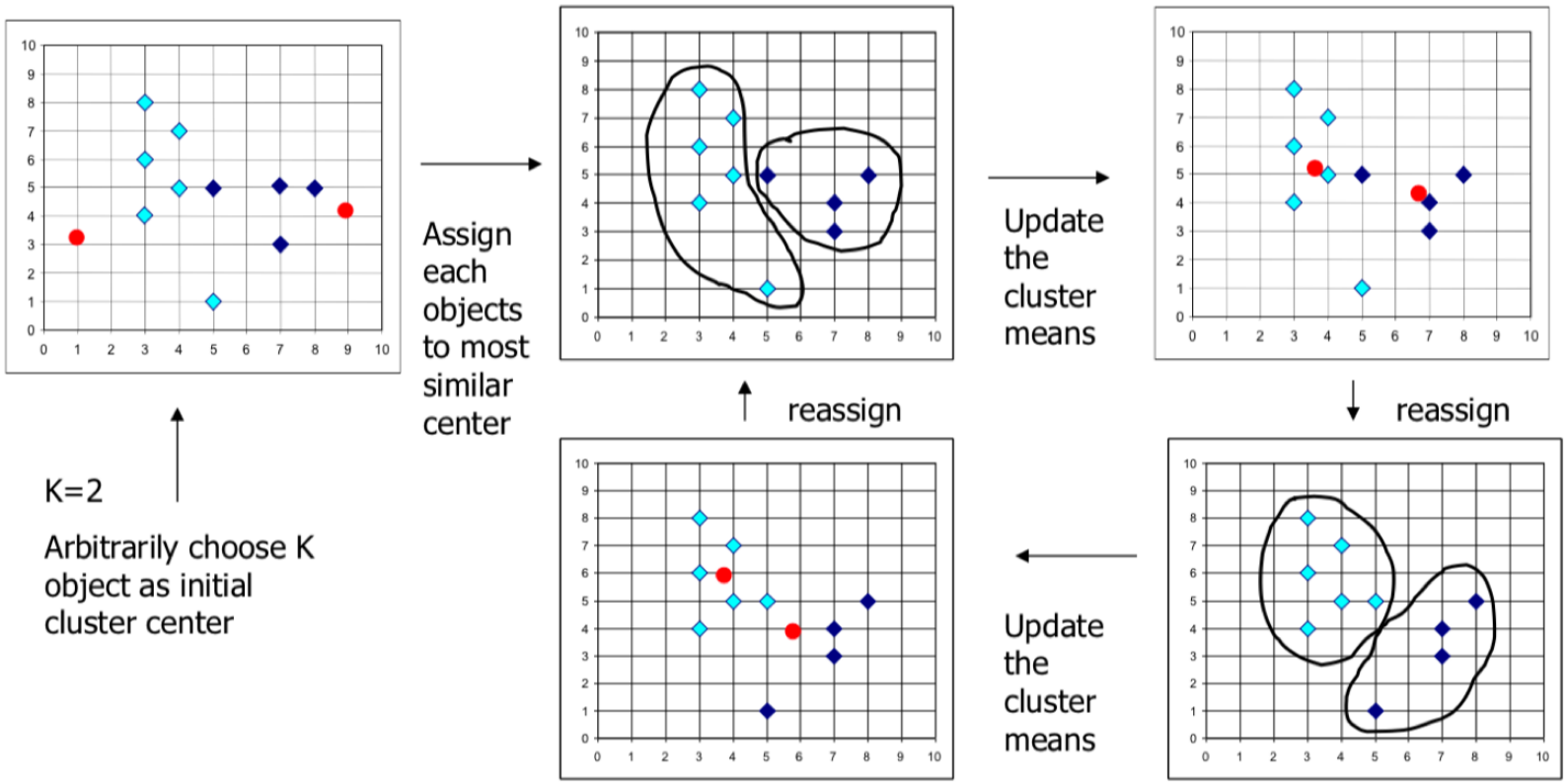
The k-Means Clustering Method

- Given a k, the k-means algorithm is implemented in four steps:

1. Given k random seeds as the initial centroids
2. Compute the centroid of each cluster of the current partition (the centroid is the center, i.e., mean point $C_m, 1 \leq m \leq k$)
3. For each object p, compute its distance to the centroids $C_m, 1 \leq m \leq k$, and assign it to the cluster with the nearest centroid $\arg\min_m \|p - C_m\|^2$
4. Go back to Step 2, stop when no more new assignment

The k-Means Clustering Method

- Example



Comments on the k-Means Method

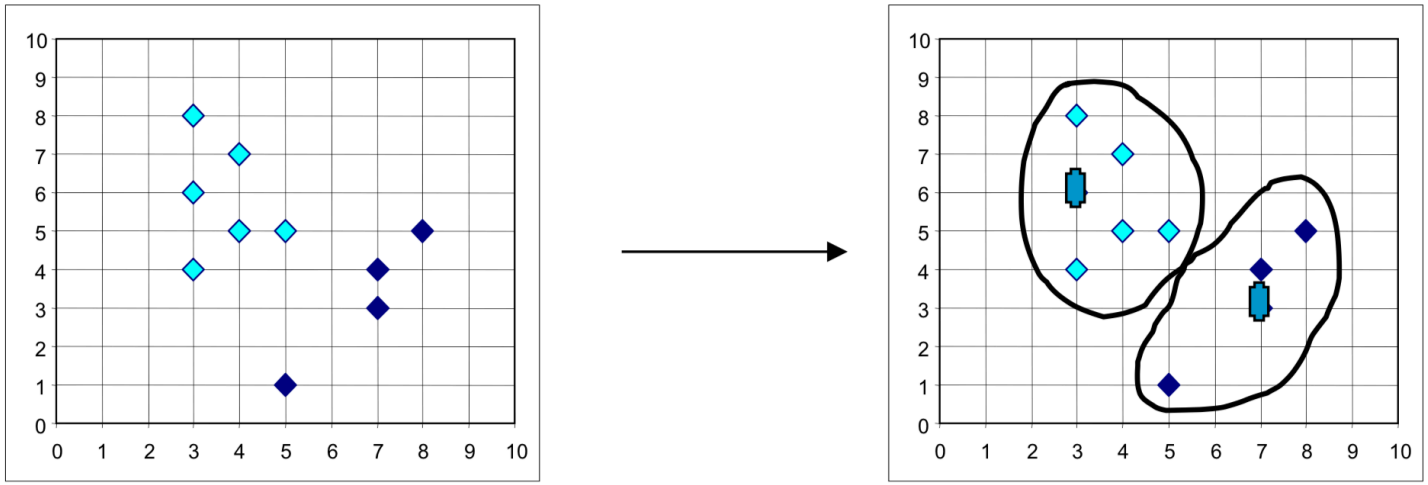
- Strength: Relatively efficient: $O(tkn)$, where n is # of objects, k is # of clusters, and t is # of iterations. Normally, $k, t \ll n$
- Comment: often terminates at a local optimum. The global optimum may found using techniques such as: deterministic annealing and genetic algorithms
- Weakness
  - Applicable only when *mean* is defined, then what about categorical data?
  - Need to specify k, the number of clusters, in advance
  - Unable to handle noisy data and outliers
  - Not suitable to discover clusters with non-convex shapes

Variations of the k-Means Method

- Handling categorical data: k-modes
  - Replacing means of clusters with modes
  - Using new dissimilarity measures to deal with categorical objects , e.g., Jaccard coefficient
  - Using a frequency-based method to update modes of clusters
  - A mixture of categorical and numerical data: k-prototype method
- Expectation Maximization: an extensin to k-means
  - Assign each object to a cluster according to a weight (prob.)
  - New means are computed based on weighted measures

What is the Problem of the k-Means Method?

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



k-medoids: PAM (Partitioning Around Medoids) Method

- k-Medoids: Instead of taking the **mean** value of the objects in a cluster as a reference point, **medoids** can be used, which is the **most centrally located** object in a cluster
- Partitioning method: construct a partition of a database D of n objects into a set of k clusters, s.t., min sum of absolute distance

$$E = \sum_{m=1}^{k} \sum_{p_i \in K_m} |p_i - o_m|$$

k-medoids: PAM (Partitioning Around Medoids) Method

Four situations of changing representative object:

- p belongs to o_j. If o_j is repalced by o_{random} and p is near to $o_i, (i \neq j)$, then p belongs to o_i
- p belongs to o_j. If o_j is repalced by o_{random} and p is near to o_{random}), then p belongs to o_{random}
- p belongs to $o_i, i \neq j$. If o_j is repalced by o_{random} and p is still near to o_i), then nothing changed
- p belongs to $o_i, i \neq j$. If o_j is repalced by o_{random} and p is near to o_{random}), then p belongs to o_{random}

Cost: $S = E_{t+1} - E_t$

k-medoids: PAM (Partitioning Around Medoids) Method

The k-mediods PAM algorithm is implemented as follows:

- **Input**: k and $\mathcal{D}$
- **Output**: k partitions of $\mathcal{D}$
- Given k random seeds as the initial representative objects
- **repeat**
- **For each** object, compute its distance to the representative objects, and assign it to the cluster with the nearest representative object
- **For each** object in the rest $n-k$ non-representative objects
- Choose a new representative object o_{random}
- Compute the cost S of replacing o_j by o_{random}
- **if** $S<0$, then replace replacing o_j by o_{random}
- **End**
- **untill convergence**

k-medoids: PAM (Partitioning Around Medoids) Method

- Time complexity: $\mathcal{O}(k(n-k)^2)$ in each iteration
- Computational cost is higher than k-means
- It will be efficient on clustering small-scale data
- More robust to outliers than k-means algorithm

CLARA and CLARANS Methods

- Two extentions of PAM to model large-scale data sets
  - CLARA: Clustering LARge Applications
  - CLARANS: Clustering LARge Applications based upon RANdomized Search

CLARA Method

CLARA: Clustering LARge Applications

- **Input**: k, v, s, and $\mathcal{D}$
- **Output**: k partitions of $\mathcal{D}$
- **for** $i = 1$ to v
  - Sampling s examples from $\mathcal{D}$
  - Call PAM to find the k optimal medoids
  - For each non-representative example p in $\mathcal{D}$, assign it to the nearest representative object o_j
  - Compute the cost E_t
  - if $E_t < E_{min}$, then $E_{min} = E_t$, and replace replacing the k representative objects with the new ones
- **end**

CLARA Method

CLARA: Clustering LARge Applications

- The effectiveness of CLARA is dependent on the size s of the subset data set
- The results of each iteration may be local-optimal
  - e.g., o_j is one of the best representative objects of the whole data set $\mathcal{D}$, while it doesn't be sampled in each iteration
- Time complexity: $k(s-k)^2 + k(n-k)$ (each iteration)

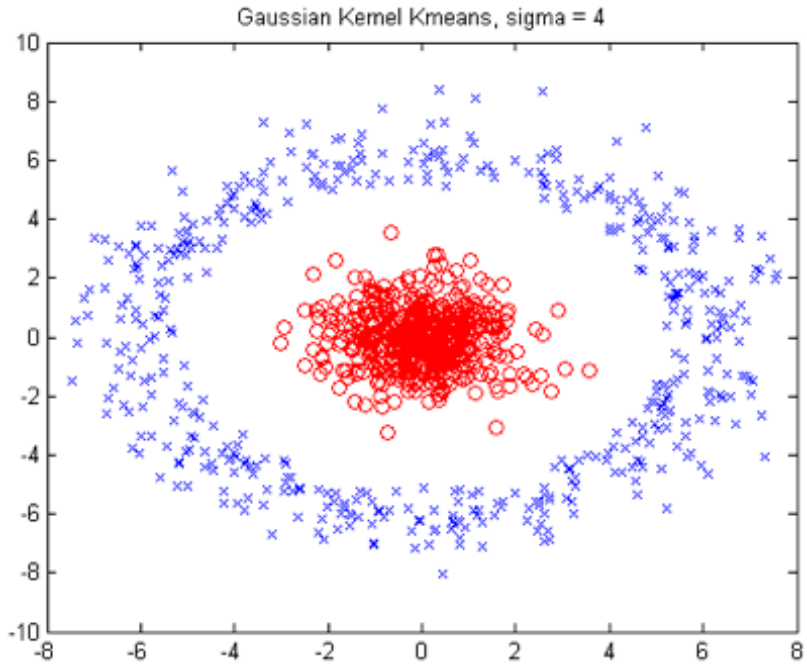
CLARANS Method

CLARANS: Clustering LARge Applications based upon RANdomized Search

- **Input**: k, v, m, and $\mathcal{D}$
- **Output**: k partitions of $\mathcal{D}$
- **for** $i = 1$ to v
- Randomly sampling k examples from $\mathcal{D}$ as the centers C
- **for** $j = 1$ to m
- Randomly sample an example o from $n - k$ rest data set
- Compute the cost E_t with replacing any representative object p in C by o
- if $E_t < E_{min}$, then $E_{min} = E_t$, replace p by o in C, and $j = 1$
- else $j = j + 1$
- **end**
- **end**

Kernel k-Means Clustering

- Kernel k-Means can be used to detect non-convex clusters
- A region is convex if it contains all the line segments connecting any pair of its points. Otherwise, it is concave
- k-Means can only detect clusters that are **linearly separable**



Kernel k-Means Clustering

- **Idea**: Project data onto the high-dimensional kernel space, and then perform k-Means clustering
- Map data points in the input space onto a high-dimensional feature space using the **kernel function**
- Perform k-Means on the mapped feature space
- Computational complexity is higher than k-Means
- Need to compute and store $n \times n$ kernel matrix generated from the kernel function on the original data, where n is the number of points
- **Spectral clustering** can be considered as a variant of Kernel k-Means clustering

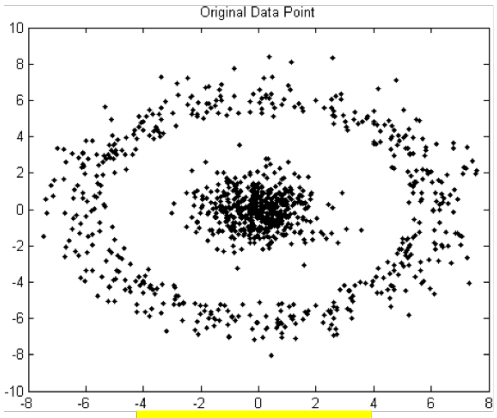
Kernel Functions and Kernel k-Means Clustering

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

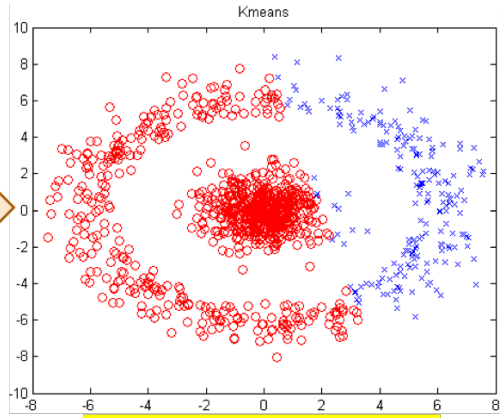
Example: Kernel Functions and Kernel k-Means Clustering

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

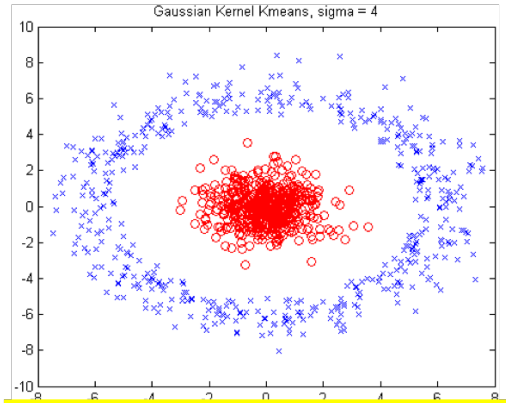
Example: Kernel k-Means Clustering



The original data set



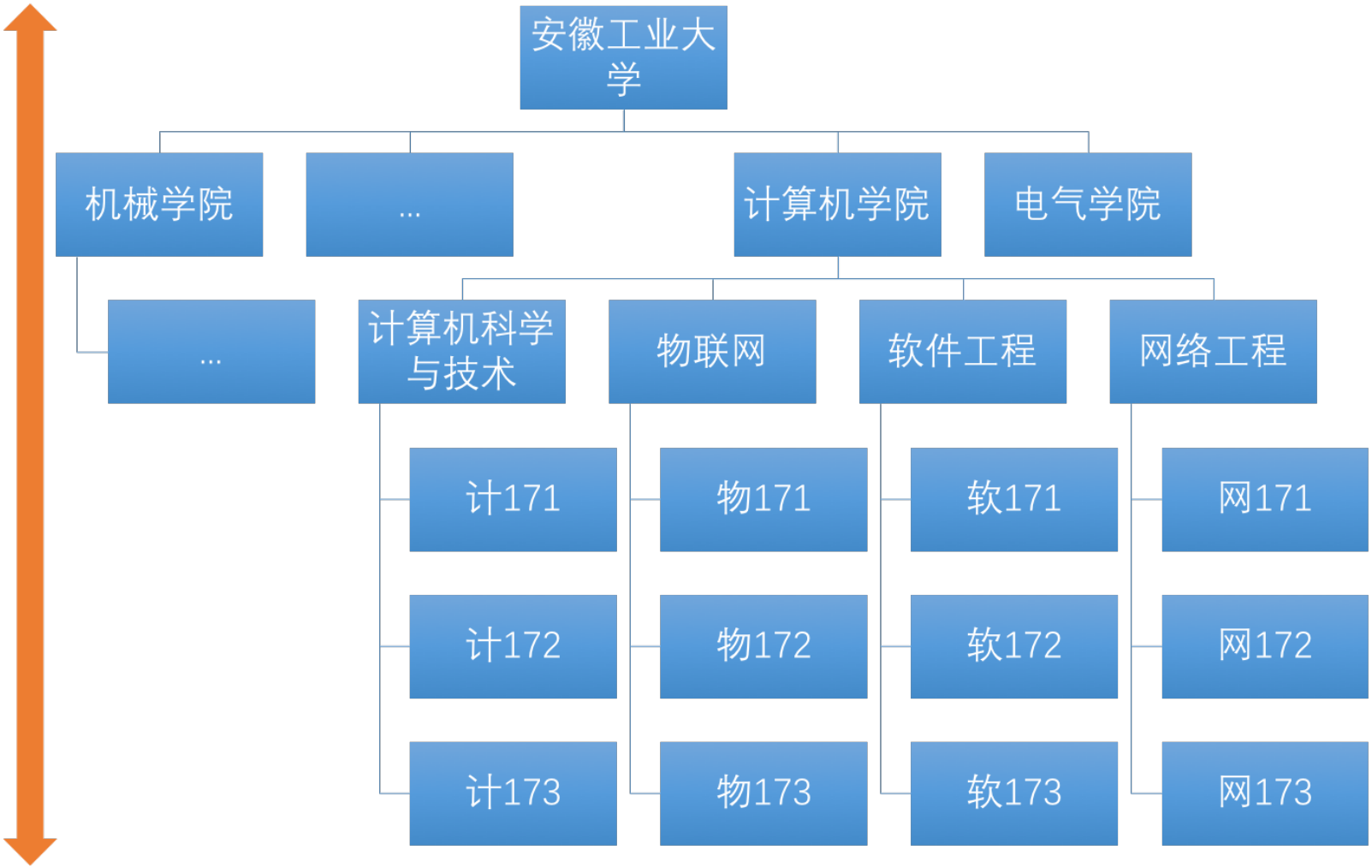
The result of K-Means clustering



The result of Gaussian Kernel K-Means clustering

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

Hierarchical Clustering



Distances between Clusters

- minimum distance:

$$d_{\min}(C_i, C_j) = \min_{p \in C_i, p' \in C_j} |p - p'|$$

- maximum distance:

$$d_{\max}(C_i, C_j) = \max_{p \in C_i, p' \in C_j} |p - p'|$$

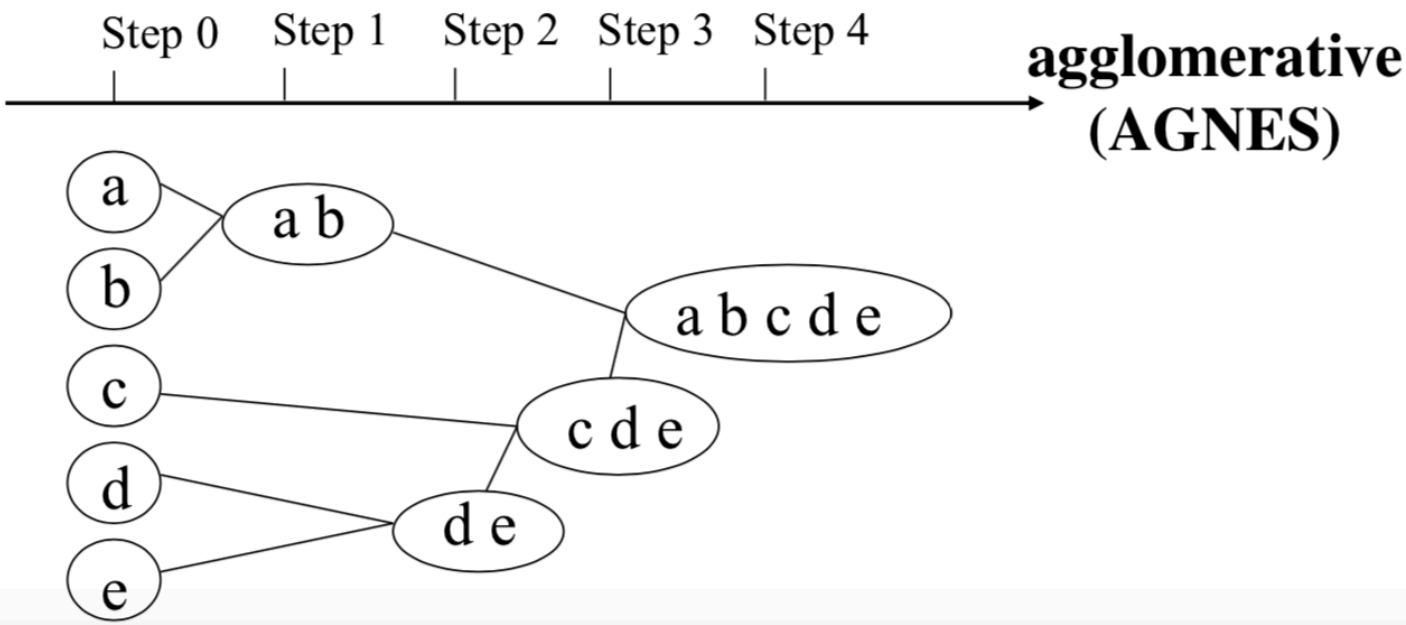
- mean distance:

$$d_{\text{mean}}(C_i, C_j) = |m_i - m_j|, m_i = \frac{1}{|C_i|} \sum_{p \in C_i} p$$

- average distance:

$$d_{\text{avg}}(C_i, C_j) = \frac{1}{|C_i||C_j|} \sum_{p \in C_i} \sum_{p' \in C_j} |p - p'|$$

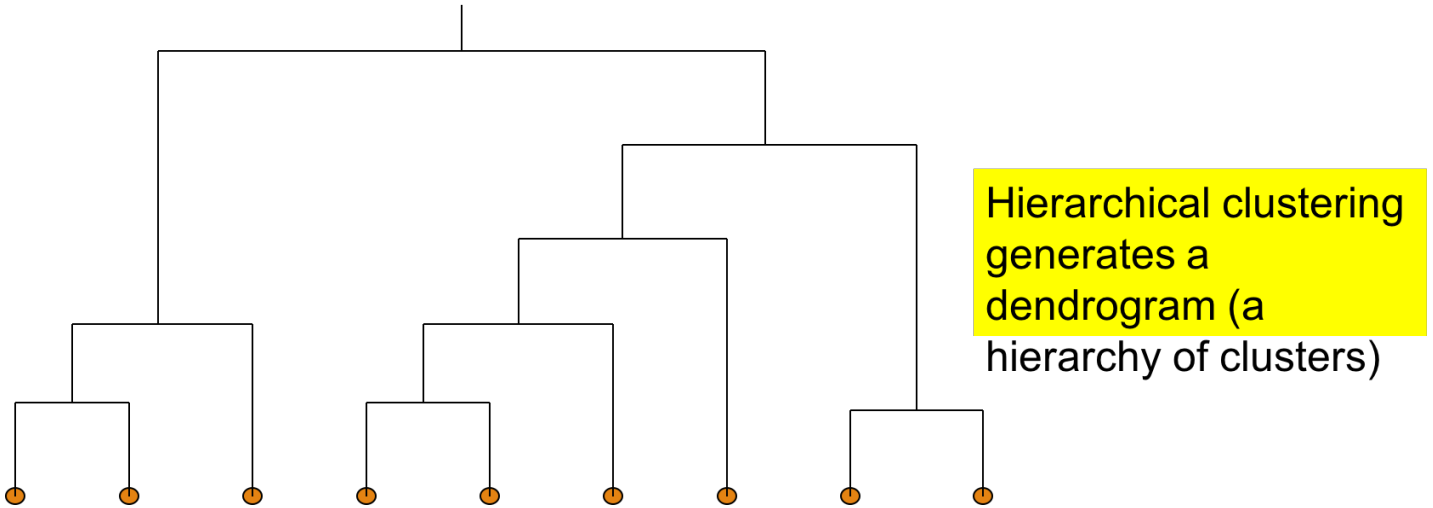
- This method does not require the number of clusters k as an input, but needs a termination condition



Hierarchical Clustering

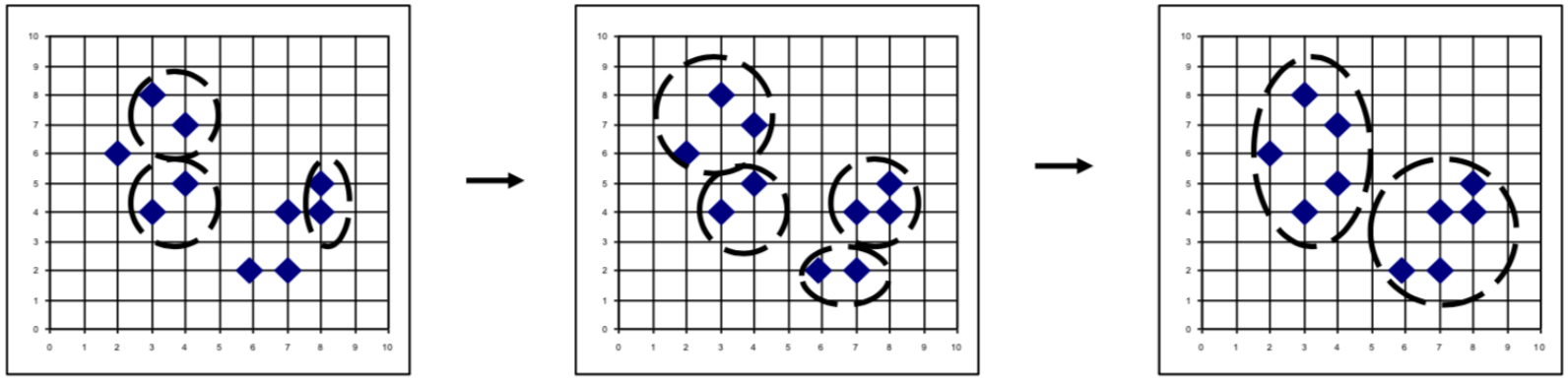
Dendrogram: Shows How Clusters are Merged

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



AGNES (Agglomerative Nesting)

- Introduced by Kaufmann and Rousseeuw (1990)
- 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



AGNES (Agglomerative Nesting)

- **Input**: Data set $\mathcal{D}$, stop criterion st
- **Output**: hierarchical partitions of $\mathcal{D}$
- treat each sample in $\mathcal{D}$ as a cluster
- **repeat**
  - compute the distance between clusters, and find two clusters with the shortest distance
  - merge the two clusters that have the least dissimilarity into a new cluster
- **untill stop criterion** st meet

AGNES (Agglomerative Nesting)

Stop criterion:

- constraint the number of clusters, e.g., k. When the number clusters reaches k, then stop clustering
- set a minimum distance, e.g., d_t. When the shortest distance between two clusters is bigger than d_t, then stop clustering

AGNES (Agglomerative Nesting)

Example: clustering the data by AGNES method

Table: Data set

ID	attr1	attr2
1	1	1
2	1	2
3	2	1
4	2	2
5	3	4
6	3	5
7	4	4
8	4	5

AGNES (Agglomerative Nesting)

Example: clustering the data by AGNES method

Table: Data set

ID	attr1	attr2
1	1	1
2	1	2
3	2	1
4	2	2
5	3	4
6	3	5
7	4	4
8	4	5

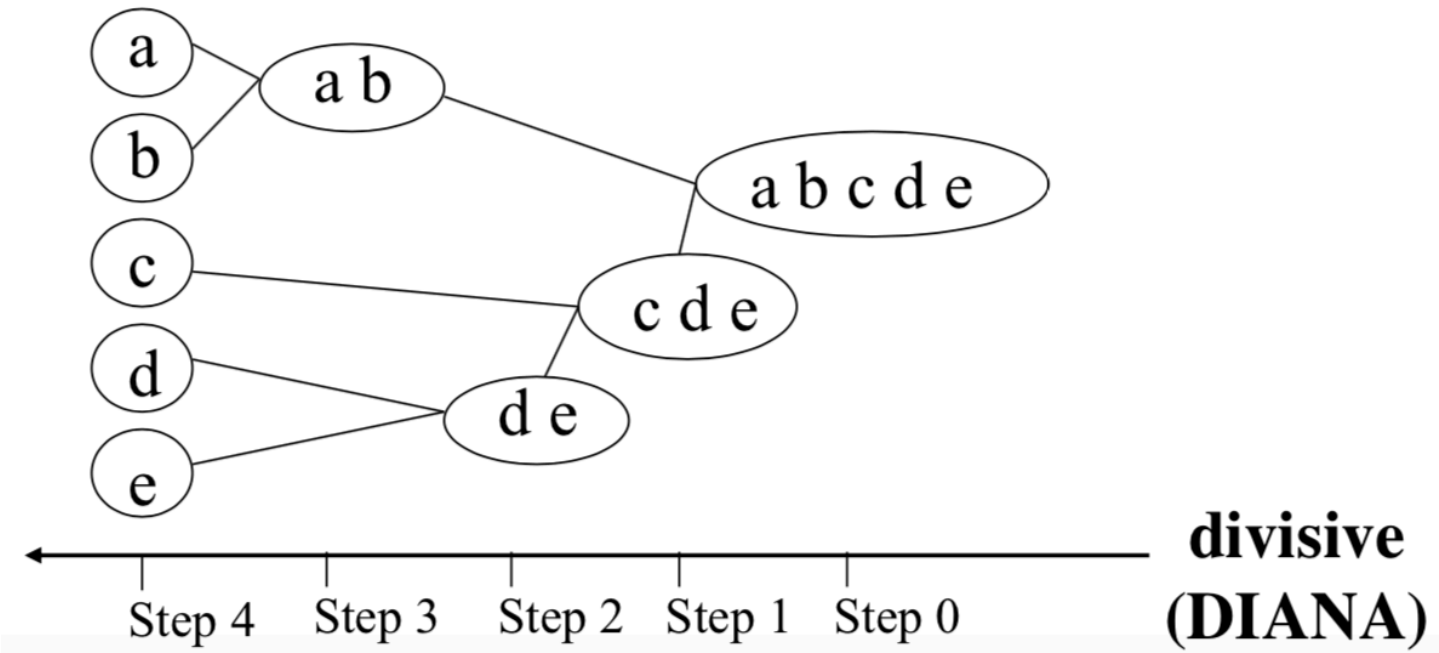
Table: Results of AGNES

Step	NearCluster	New Merged Clusters
1	{1},{2}	{1,2},{3},{4},{5},{6},{7},{8}
2	{3},{4}	{1,2},{3,4},{5},{6},{7},{8}
3	{5},{6}	{1,2},{3,4},{5,6},{7},{8}
4	{7},{8}	{1,2},{3,4},{5,6},{7,8}
5	{1,2},{3,4}	{1,2,3,4},{5,6},{7,8}
6	{5,6},{7,8}	{1,2,3,4},{5,6,7,8}

Comment on AGNES

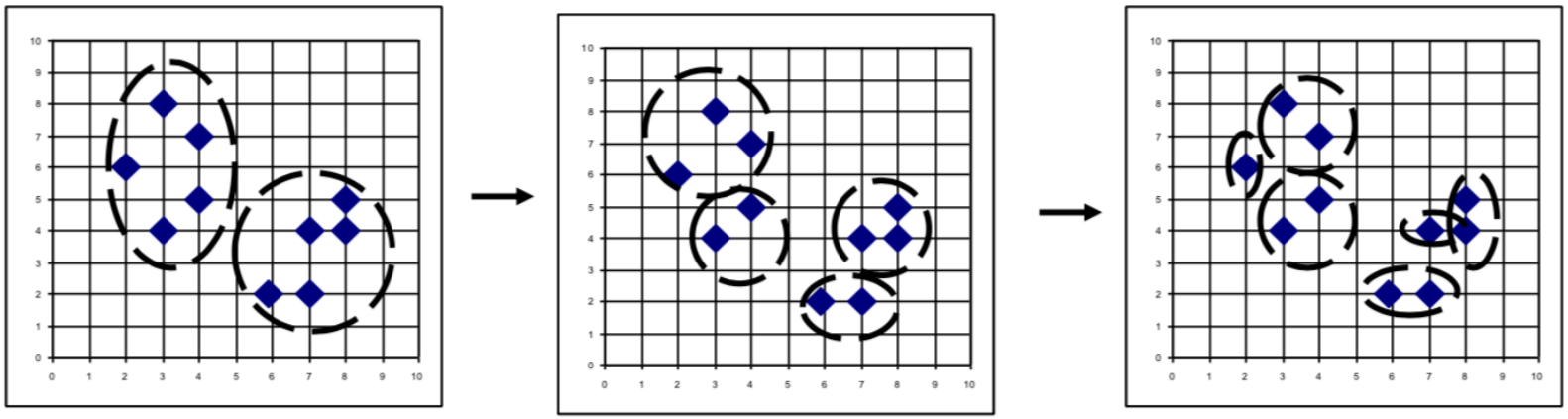
- It is a bottom-up strategy
- Simple
- Once one example is merged, it can not be withdraw
- Time complexity: $\mathcal{O}(n^2)$, it is not applicable for large-scale data sets

- This method does not require the number of clusters k as an input, but needs a termination condition



DIANA (Divisive Analysis)

- Introduced by Kaufmann and Rousseeuw (1990)
- Inverse order of AGNES
- Eventually each node forms a cluster on its own



DIANA (Divisive Analysis)

- **Input**: Data set $\mathcal{D}$ and number of clusters k
- **Output**: k partitions of $\mathcal{D}$
- treat all the samples in $\mathcal{D}$ as a cluster
- **for** $(i = 1, i \neq k; i++)$
- find the cluster C with biggest **diameter** from all the clusters
- find an example p with the biggest average dissimilarity from C, and put it into splinter group and the rest examples into old party
- **repeat**
- find an example from old party with smaller (or equal) distance to splinter group than to old party, and add it to splinter group
- **untill** no examples adding to splinter group from old party
- splinter group and old party are two new splited clusters
- **end**

DIANA (Divisive Analysis)

Example: clustering the data by DIANA method

Table: Data set

ID	attr1	attr2
1	1	1
2	1	2
3	2	1
4	2	2
5	3	4
6	3	5
7	4	4
8	4	5

DIANA (Divisive Analysis)

Example: clustering the data by DIANA method

Table: Data set

ID	attr1	attr2
1	1	1
2	1	2
3	2	1
4	2	2
5	3	4
6	3	5
7	4	4
8	4	5

1. find the cluster C with biggest diameter from all the clusters
2. avgdist(1) = (1+1+1.414+3.6+4.24+4.47 + 5)/7=2.96; similarly, avgdist(2) = 2.526, avgdist(3) = 2.68, avgdist(4) = 2.18, avgdist(5) = 2.18 avgdist(6) = 2.68, avgdist(7) = 2.526, avgdist(8) = 2.96
3. assign point 1 to splinter group, and the rest to old party
4. find an example from old party with smaller (or equal) distance to splinter group than to old party, and add it to splinter group, e.g, point 2
5. repeat step 4

DIANA (Divisive Analysis)

Table: Results of DIANA

Step	selectedCluster	splinter group	old party
1	{1,2,3,4,5,6,7,8}	{1}	{2,3,4,5,6,7,8}
2	{1,2,3,4,5,6,7,8}	{1,2}	{3,4,5,6,7,8}
3	{1,2,3,4,5,6,7,8}	{1,2,3}	{4,5,6,7,8}
4	{1,2,3,4,5,6,7,8}	{1,2,3,4}	{5,6,7,8}
5	{1,2,3,4,5,6,7,8}	{1,2,3,4}	{5,6,7,8} **stop**

Recent Hierarchical Clustering Methods

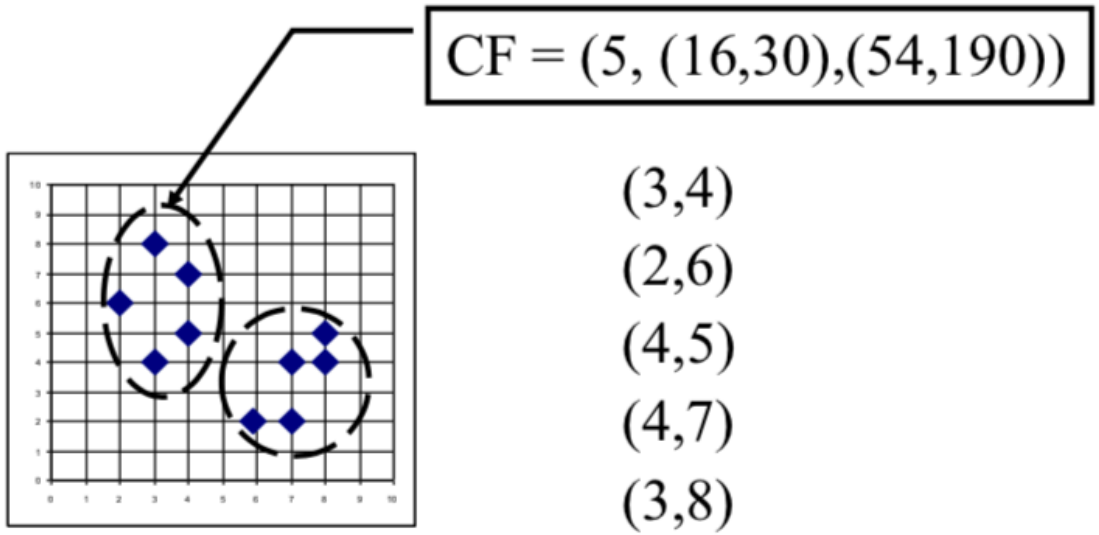
- Major weakness of hierarchical clustering methods
  - do not scale well: time complexity of at least $\mathcal{O}(n^2)$, where n is the number of total objects
  - can **never undo** what was done previously
- Integration of hierarchical with distance-based clustering
  - BIRCH (1996): uses CF-tree and incrementally adjusts the quality of sub-clusters
  - ROCK (1999): clustering categorical data by neighbor and link analysis
  - CHAMELEON (1999): hierarchical clustering using dynamic modeling

BIRCH (1996)

- BIRCH: integrated hierarchical clustering
- **Clustering feature**, **Clustering feature tree**
- 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 a clustering algorithm to cluster the leaf nodes of the CF-tree

Clustering Feature Vector in BIRCH

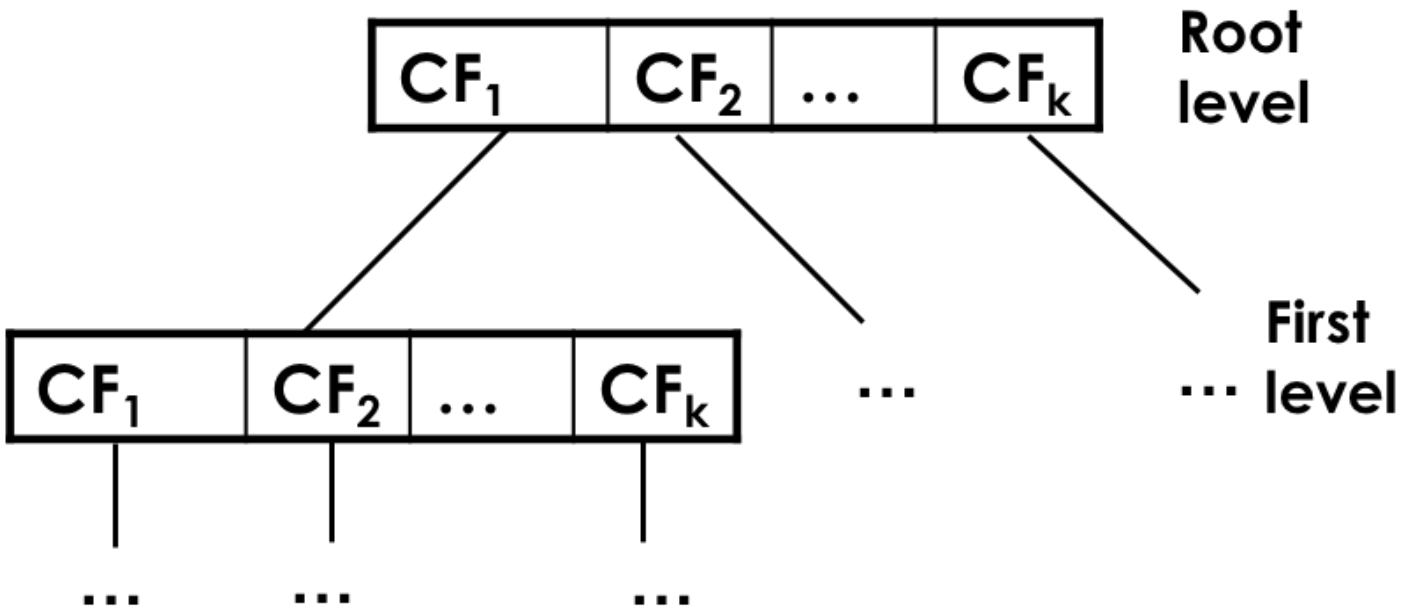
- **Clustering Feature**: $CF = (N, \overrightarrow{LS}, \overrightarrow{SS})$, summarize the cluster members
- N: number of data points
- $\overrightarrow{LS}: \sum_{i=1}^{N} \overrightarrow{\mathbf{x}_i}$, linear sum of N points
- $\overrightarrow{SS}: \sum_{i=1}^{N} \overrightarrow{\mathbf{x}_i^2}$, square sum of N points



CF-Tree in BIRCH

- **Clustering feature**:
  - summary of the statistics for a given subcluster
  - registers crucial measurements for computing cluster and utilizes storage efficiently
  - A representation of the cluster
  - A CF entry has sufficient information to calculate the **centroid**, **radius**, **diameter** and many other distance measures (39)

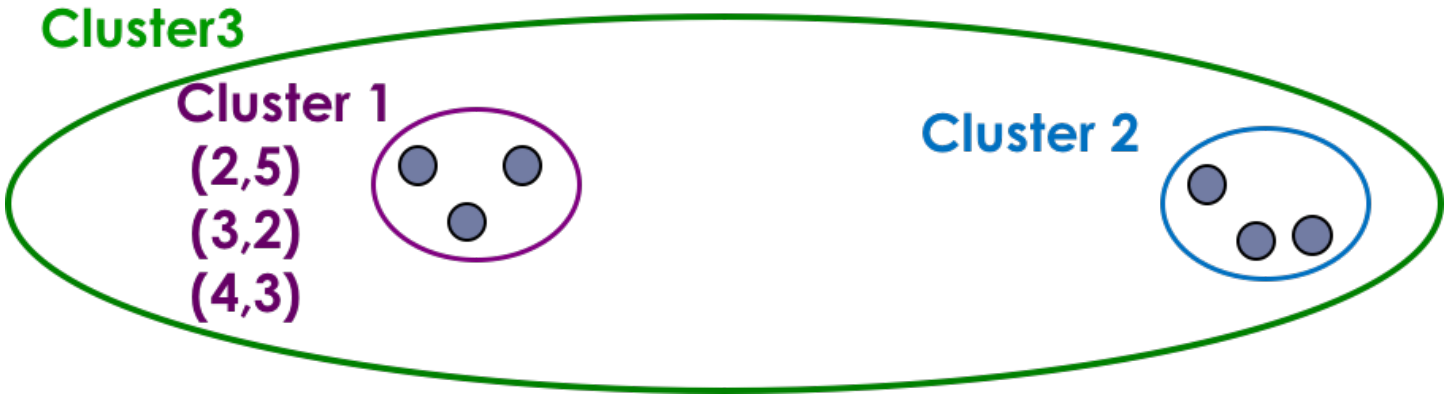
The CF Tree Structure



- A CF tree has two parameters
  - **Branching factor**: specify the maximum number of children
  - **Threshold**: max diameter of sub-clusters stored at the leaf nodes

CF-Tree in BIRCH

- 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
  - **CF1** = (3, (2+3+4 , 5+2+3), (22+32+42 , 52+22+32)) = (3, (9,10), (29 ,38))
  - **CF2** = (3, (35,36), (417 ,440))
  - **CF3** = **CF1** + **CF2** = (3+3, (9+35, 10+36), (29+417 , 38+440)) = (6, (44,46), (446 ,478))



BIRCH

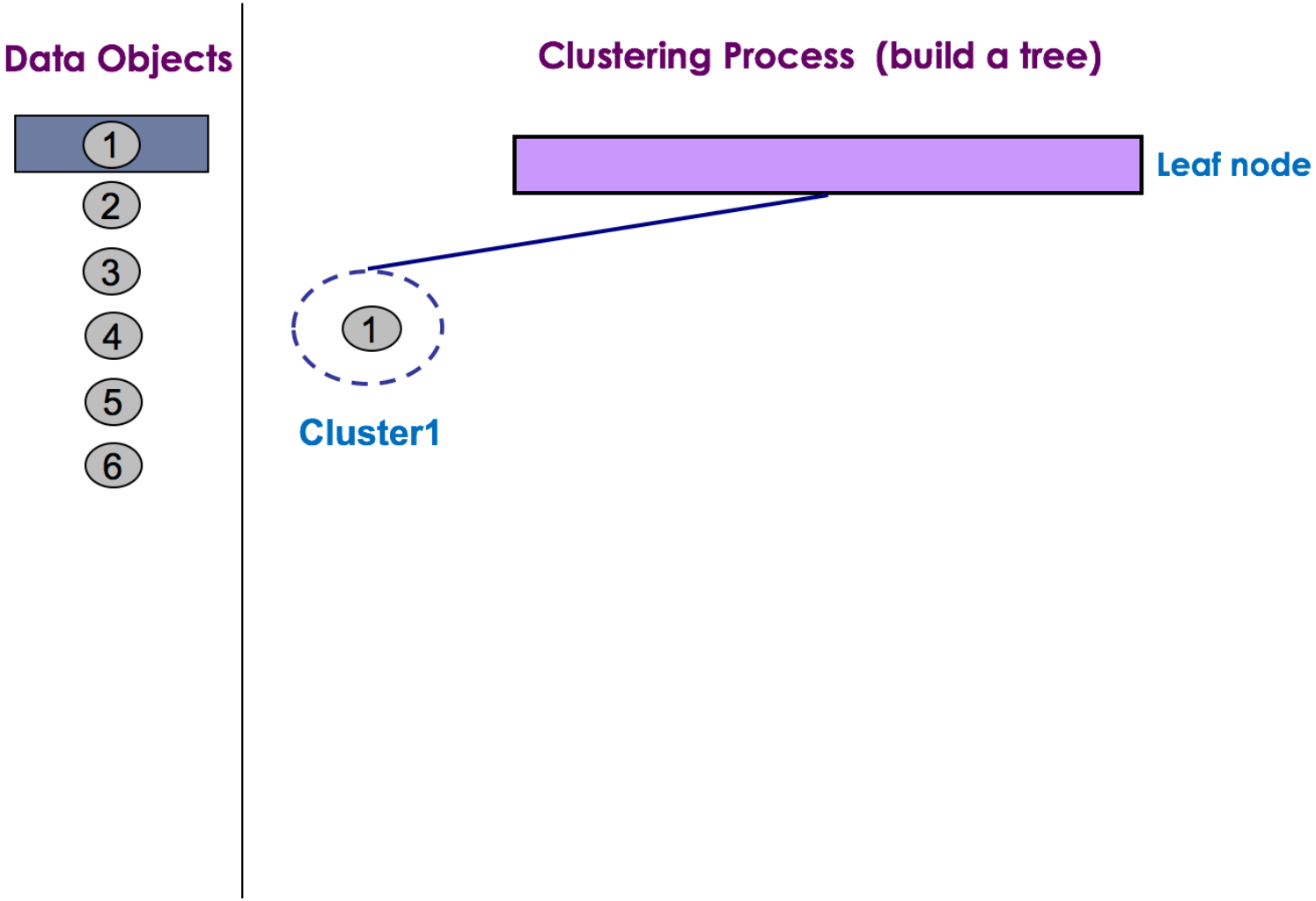
1. Insert each object to its closest leaf entry
2. If the diameter of a leaf is larger than a threshold, the leaf will be split
3. Update the CF and its ancestor's CF
4. If the size of the CF tree is too big, re-build the tree from the leaf node, no re-scan the original objects
5. Two parameters (branching factor, threshold), control the size of the tree

BIRCH
CF Tree Insertion

- Start with the root
- Find the CF entry in the root closest to the data point, move to that child and repeat the process until a closest leaf entry is found.
- At the leaf
  - If the point can be accommodated in the cluster, update the entry
  - If this addition violates the threshold T, split the entry, if this violates the limit imposed by L, split the leaf. If its parent node too is full, split that and so on
- Update the CF entries from the root to the leaf to accommodate this point

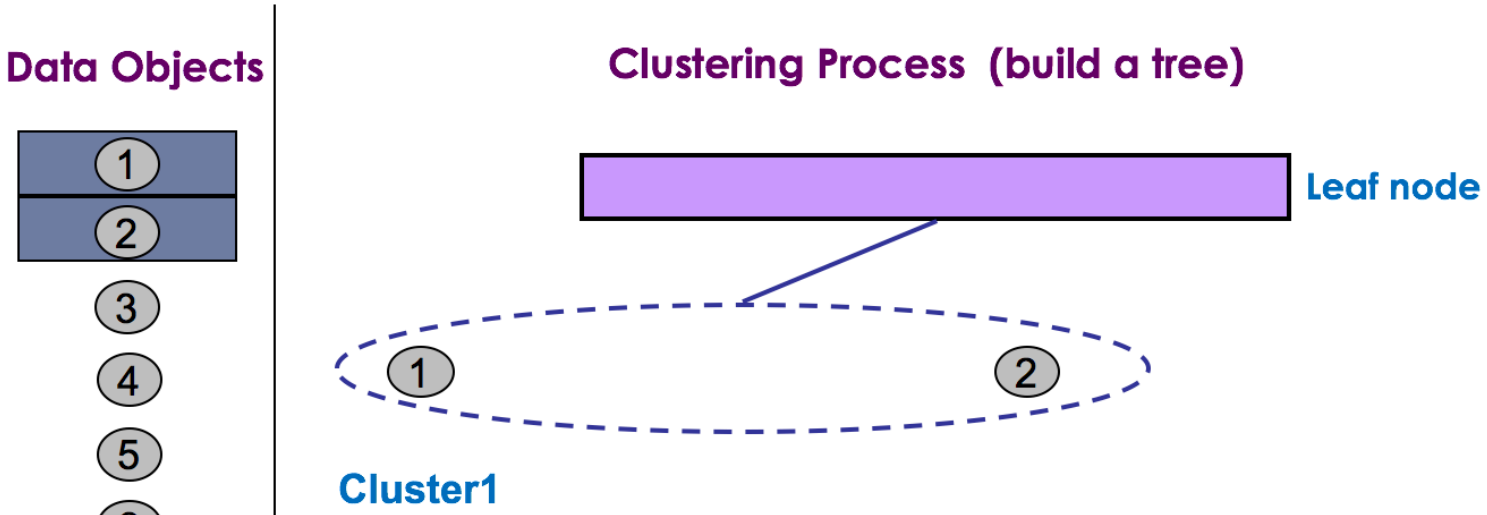
CF-Tree in BIRCH

Example



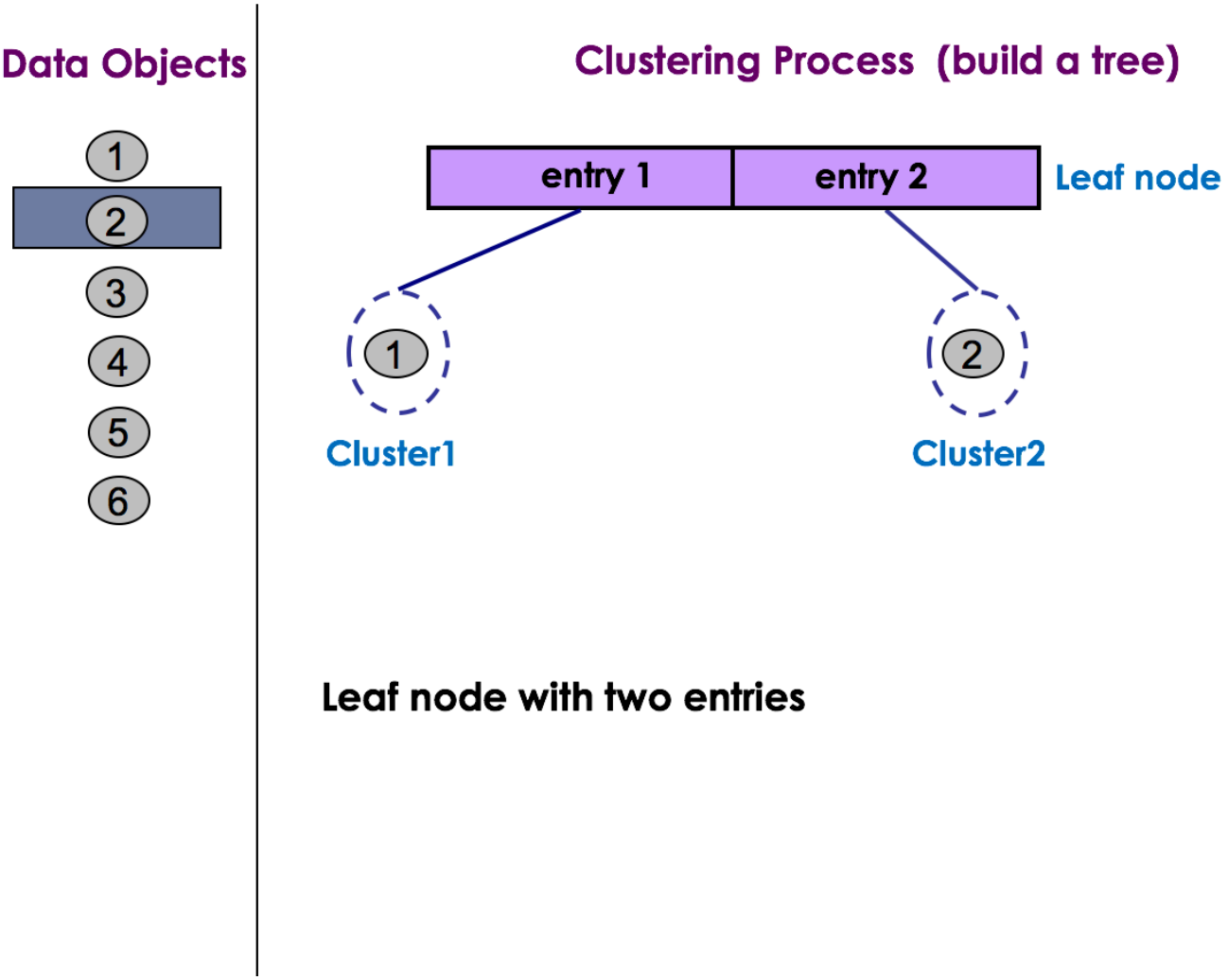
CF-Tree in BIRCH

Example



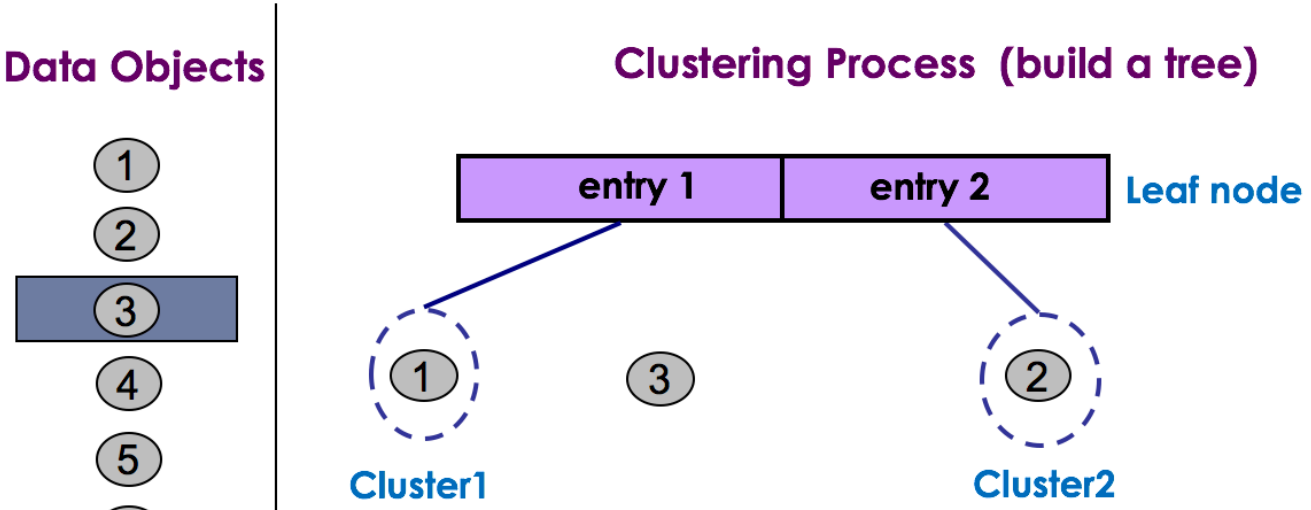
If cluster 1 becomes too large (not compact) by adding object 2, then split the cluster

CF-Tree in BIRCH
Example



CF-Tree in BIRCH

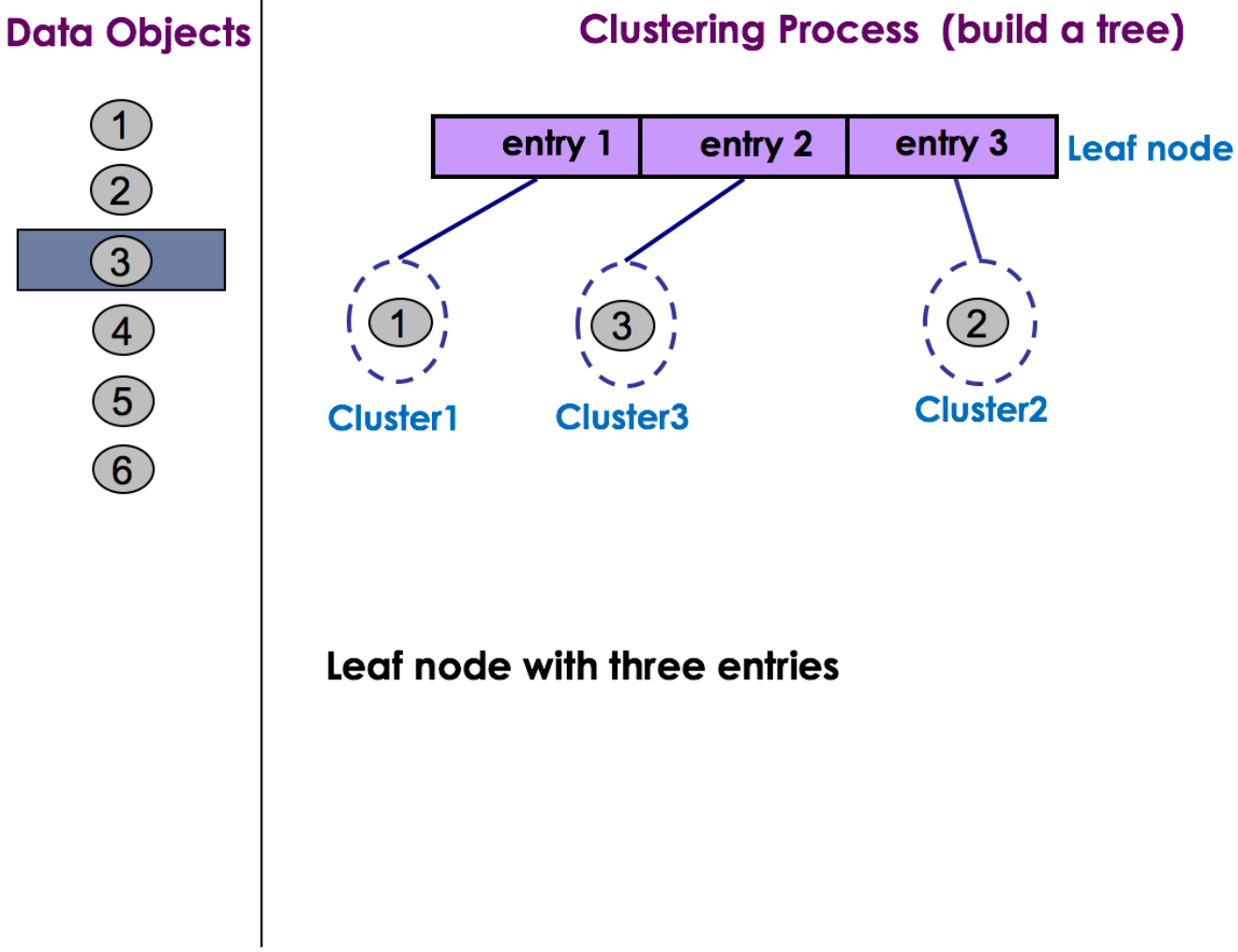
Example



entry1 is the closest to object 3

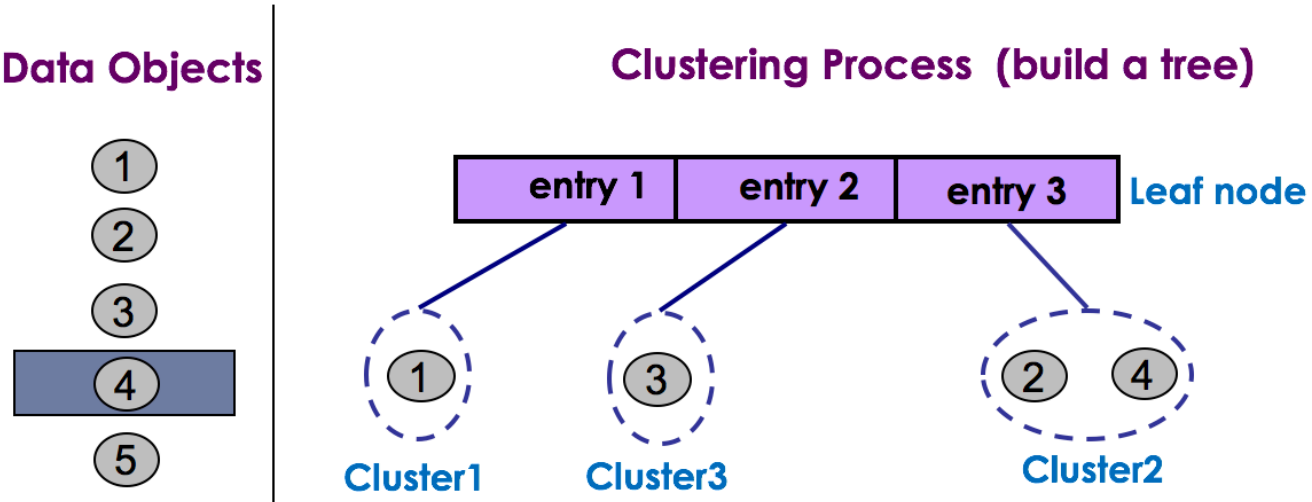
If cluster 1 becomes too large by adding object 3, then split the cluster

CF-Tree in BIRCH
Example



CF-Tree in BIRCH

Example

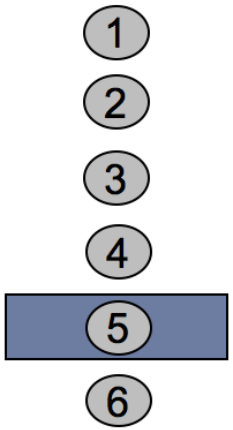


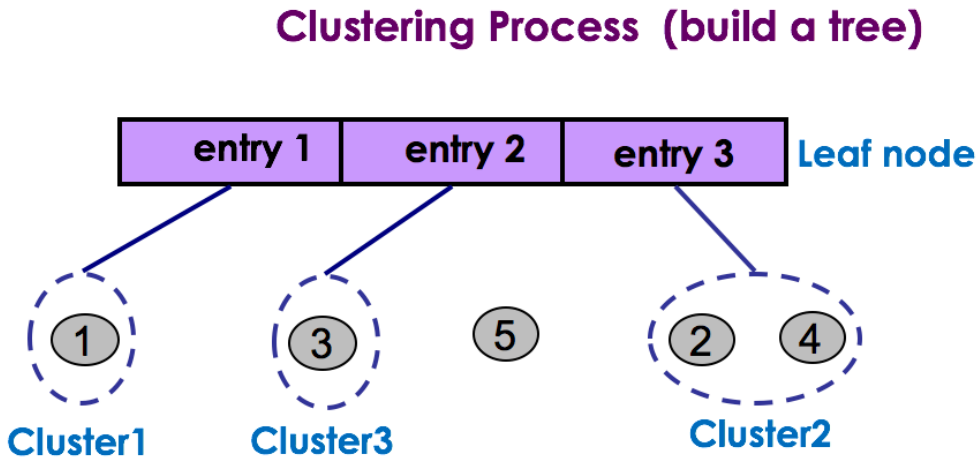
entry3 is the closest to object 4

Cluster 2 remains compact when adding object 4 then add object 4 to cluster 2

CF-Tree in BIRCH

Example





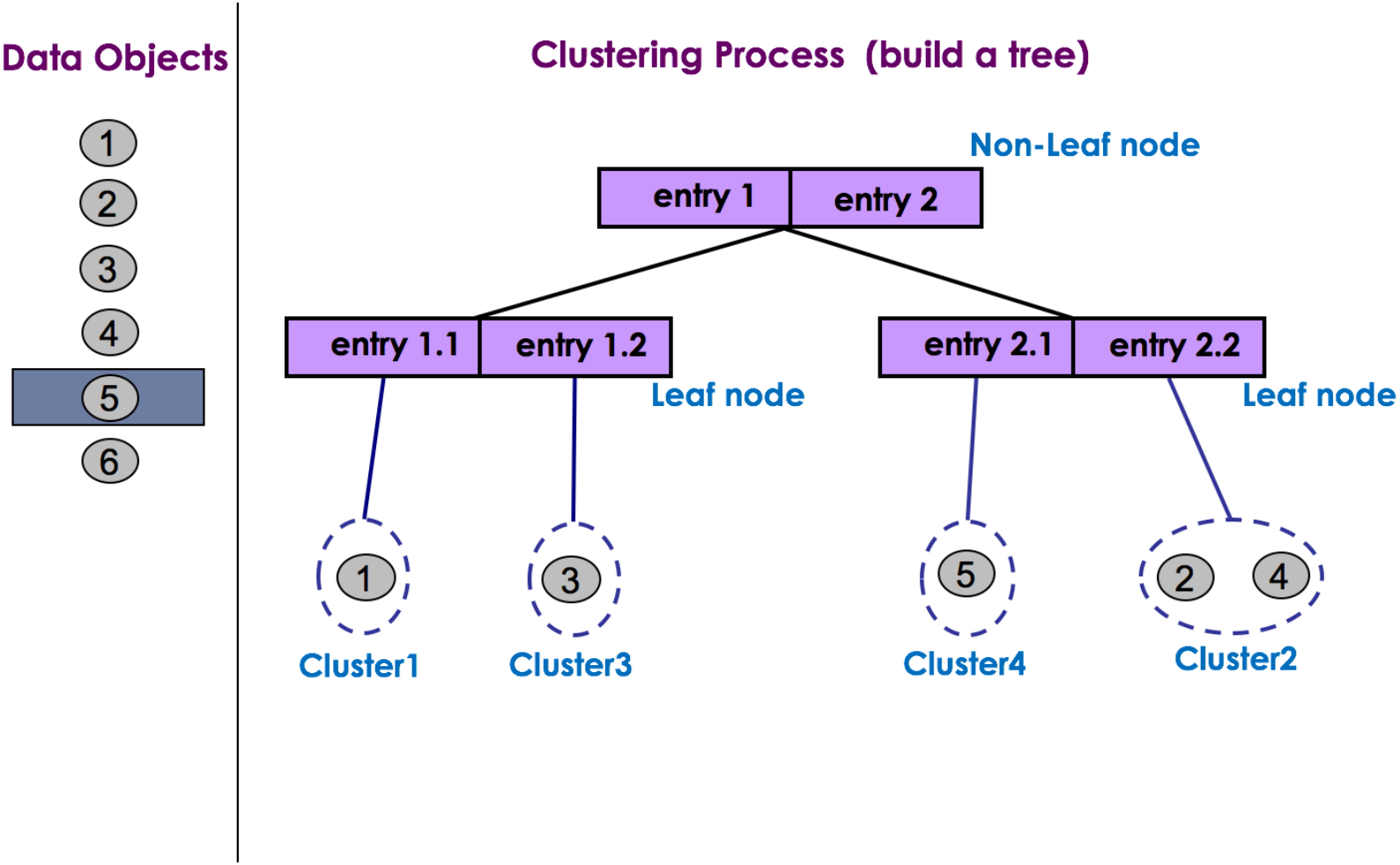
entry2 is the closest to object 5

Cluster 3 becomes too large by adding object 5 then split cluster 3?

BUT there is a limit to the number of entries a node can have Thus, split the node

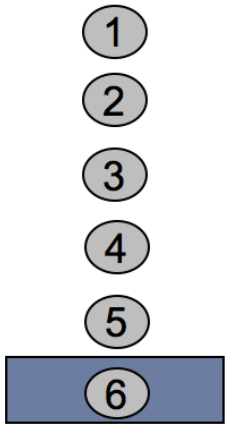
CF-Tree in BIRCH

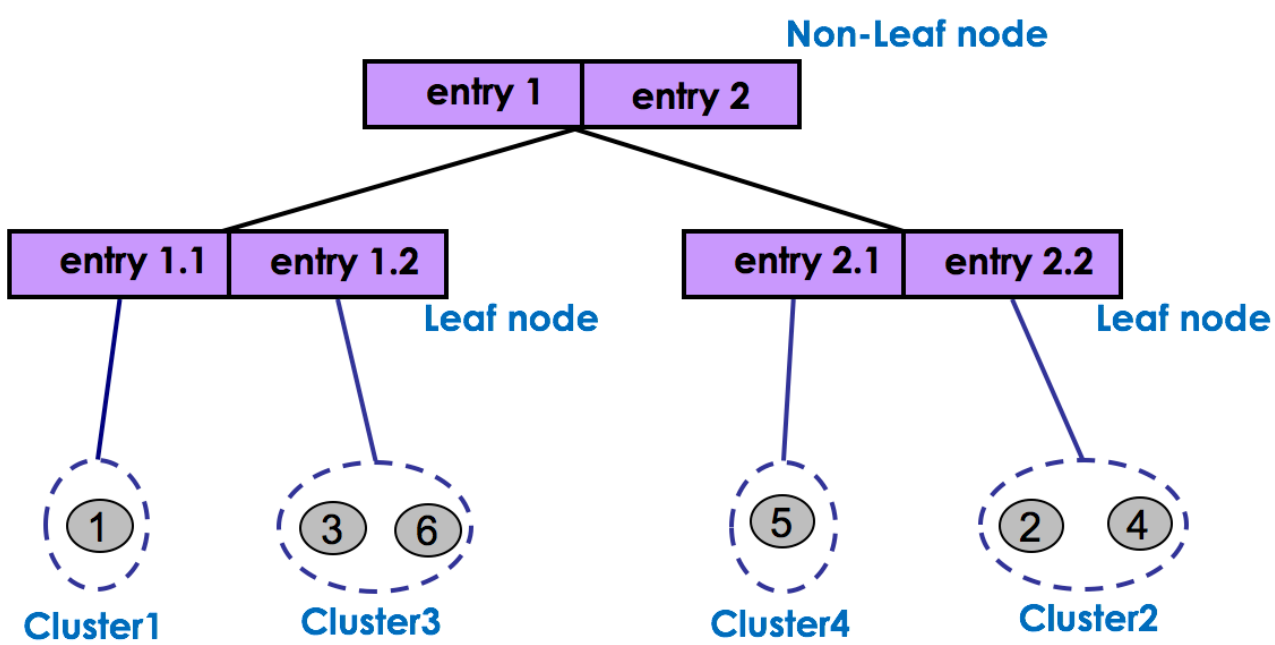
Example



CF-Tree in BIRCH

Example





entry1.2 is the closest to object 6

Cluster 3 remains compact when adding object 6 then add object 6 to cluster 3

BIRCH

- Scales linearly
  - Complexity: $\mathcal{O}(n)$
  - Scalable for large database
  - Incremental clustering
  - Finds a good clustering with a single scan, I/O cost small
- Weakness
  - Handles only numeric data, and sensitive to the order of the data record
  - Not good at arbitrary shaped cluster

ROCK: for Categorical Data

ROCK: RObust Clustering using linKs

- Experiments show that distance functions do not lead to high quality clusters when clustering **categorical data**
- Most clustering techniques assess the similarity between points to create clusters
- At each step, points that are similar are merged into a single cluster
- Localized approach prone to errors
- ROCK: used links instead of distances

ROCK: for Categorical Data

Example: Compute Jaccard Coefficient

- Compute **Jaccard coefficient** between transactions
- $\text{Sim}(T_i, T_j) = \frac{|T_i \cap T_j|}{|T_i \cup T_j|}$, e.g., Sim({a,b,c}, {b,d,e}) $= 1/5 = 0.2$
- Jaccard coefficient between transactions of Cluster1 ranges from 0.2 to 0.5
- Jaccard coefficient between transactions belonging to different clusters can also reach 0.5
- e.g., **Sim({a,b,c},{a,b,f})=2/4=0.5**

Two clusters of transactions

Cluster1. <a, b, c, d, e>	Cluster2. <a, b, f, g>
{a, b, c}	{a, b, f}
{a, b, d}	{a, b, g}
{a, b, e}	{a, f, g}
{a, c, d}	{b, f, g}
{a, c, e}	
{a, d, e}	
{b, c, d}	
{b, c, e}	
{b, d, e}	
{c, d, e}	

ROCK: for Categorical Data

Example: Using Links

- The number of links between T_i and T_j is **the number of common neighbors**
- T_i and T_j are neighbors if Sim$(T_i, T_j) > \theta$, e.g.,$\theta = 0.5$
- Link({a,b,f}, {a,b,g}) = 5 (common neighbors)
- Link({a,b,f},{a,b,c}) = 3 (common neighbors)
- **Link is a better measure than Jaccard coefficient**

Two clusters of transactions

Cluster1. <a, b, c, d, e>

{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}

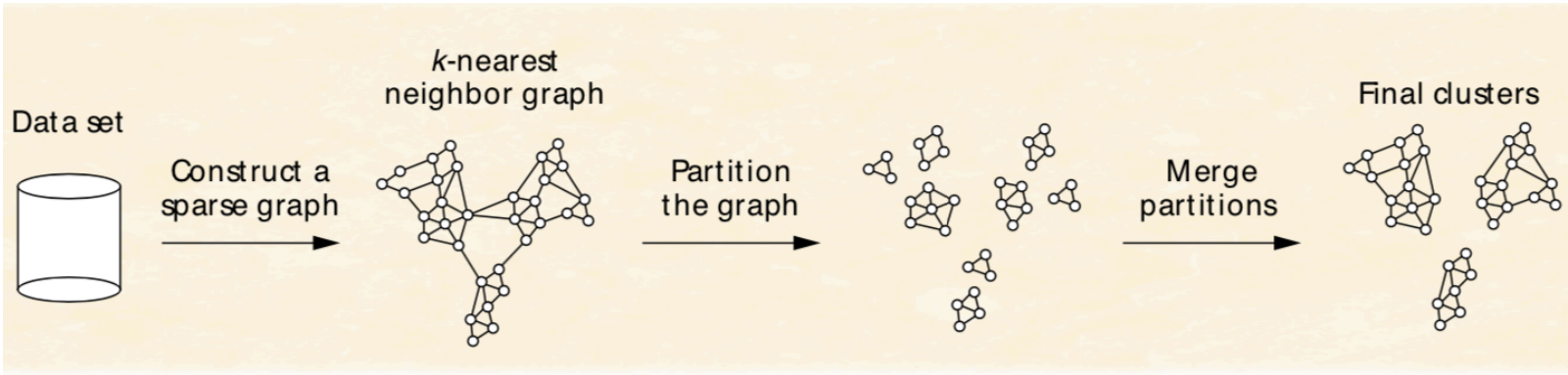
Cluster2. <a, b, f, g>

{a, b, f}
{a, b, g}
{a, f, g}
{b, f, g}

ROCK: for Categorical Data

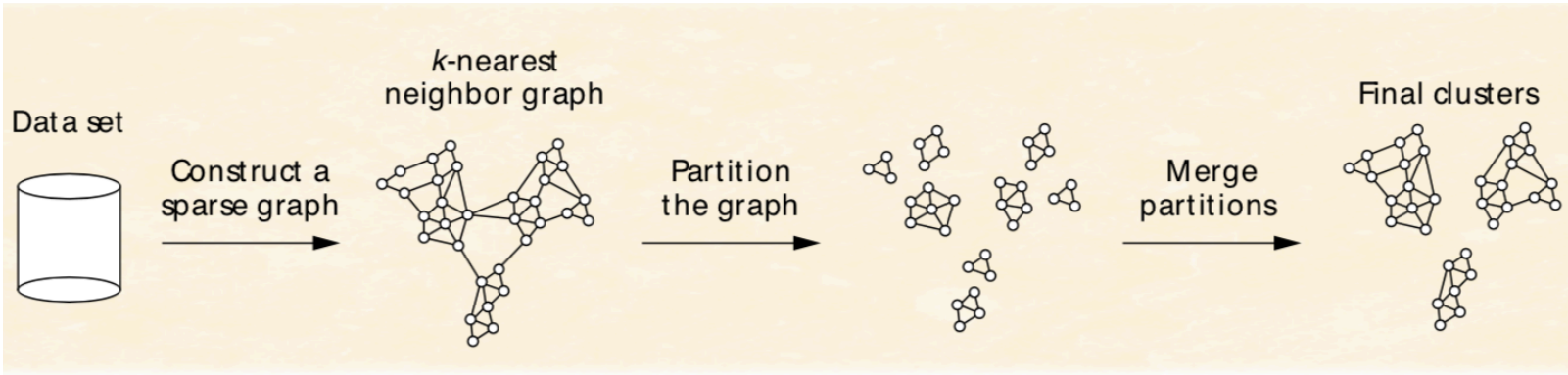
- ROCK: RObust Clustering using linKs
- Major Ideas
  - Use links to measure similarity/proximity
  - Not distance-based
  - Computational complexity: $O(n^2 + nm_m m_a + n^2 \log n)$
    - n: number of objects
    - m_a: average number of neighbors
    - m_m: maximum number of neighborse
- Algorithm
  - Sampling-based clustering
  - Draw random sample
  - Cluster with links
  - Label data in disk

Chameleon Method



- Combines initial partition of data with hierarchical clustering techniques it modifies clusters dynamically
- **Step1**:
  - Generate a ***k*NN graph**
  - because it's local, it reduces influence of noise and outliers
  - provides automatic adjustment for densities
- **Step2**:
  - use METIS: a graph partitioning algorithm
  - get equally-sized groups of well-connected vertices
  - this produces "sub-clusters" - something that is a part of true clusters

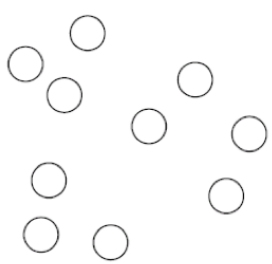
Chameleon Method



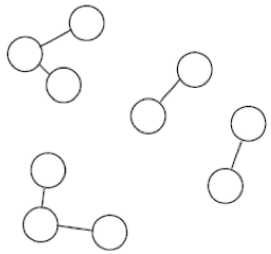
- **Step3**:
  - recombine sub-clusters
  - combine two clusters if
    - they are relatively close
    - they are relatively interconnected
  - so they are merged only if the new cluster will be similar to the original ones
  - i.e. when "self-similarity" is preserved (similar to the join operation in Scatter/Gather)

Chameleon Method
KNN Graphs

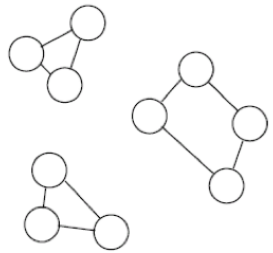
- The **k-nearest neighbor graph (k-NNG)** is a graph in which two vertices p and q are connected by an edge, if the distance between p and q is among the k-th smallest distances from p to other objects from P.
- K-nearest neighbor (KNN) graphs from an original data in 2D:



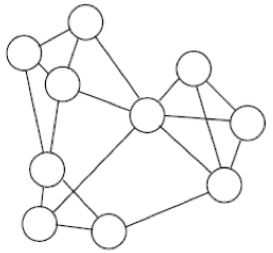
(a) Original Data in 2D



(b) 1-nearest neighbor graph



(c) 2-nearest neighbor graph



(d) 3-nearest neighbor graph

Issues to Chameleon Method

- Curse of Dimensionality makes similarity functions behave poorly
- distances become more uniform as dimensionality grows
- and this makes clustering difficult
- Similarity between two point of high dimensionality can be misleading
- often points may be similar even though they should belong to different clusters

Ref: George Karypis, Eui-Hong Han, Vipin Kumar, *CHAMELEON: A Hierarchical Clustering Algorithm Using Dynamic Modeling*, IEEE Computer, 32(8): 68-75, 1999.

CURE: Clustering Using Representatives

- Start with each individual point as a separate cluster
- Merge closest clusters till each cluster contains more than c points
- For each cluster, use c scattered points as representatives
- If more than k clusters
  - Clusters with the closest pair of representative points are merged
  - Update the representative points of merged clusters

CURE: Clustering Using Representatives

- **Choose representatives**
  - the point farthest from the mean of the cluster
  - **for** 2 to c do
  - the point farthest from the previously chosen point
  - Shrink the scattered points toward the mean by a fraction
  - **for each** scattered point p do
  - representative $= p + \alpha * (mean–p)$
- **Merge**
  - Euclidian distance
  - Closest clusters - minimum distance between representative points from two clusters

$$dist(u, v) = \min_{p \in u.\text{rep}, q \in v.\text{rep}} dist(p, q)$$

CURE: Clustering Using Representatives

Comment

- Multiple representative points allow CURE to discover arbitrary shaped clusters
- Less sensitive to outliers
  - Shrink scattered points toward the mean, weaken the effects of outliers
- Time complexity $O(n^2 \log(n))$
- For large-scale database, do sampling and partitioning

CURE: Clustering Using Representatives

Large-scale database: Sampling

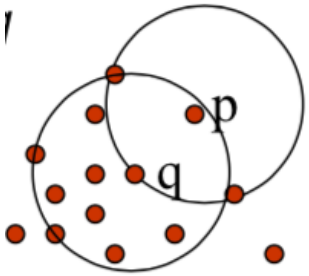
- Draw a random sampling S from original objects
- Cluster the sampled objects (basic CURE)
- Eliminate outliers
- Each unsampled original object is assigned to the cluster containing the closest representative point to it

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
  - Need density parameters as termination condition
  - Complexity is $O(n^2)$

Density-Based Clustering: Basic Concepts

- Two parameters:
  - ϵ-neighborhood: neighborhood within a radius ϵ of a point
  - MinPts: Min number of points in ϵ -neighborhood of a point
- Core object: If the number of points in ϵ-neighborhood of point p exceeds MinPts
- Directly density-reachable: A point p is directly density-reachable from a point q w.r.t., ϵ, MinPts if
  - p belongs to ϵ-neighborhood of q
  - q is core object

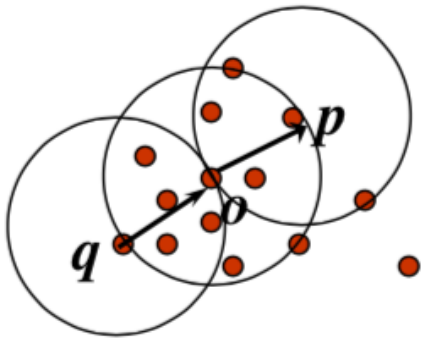


$MinPts = 5$

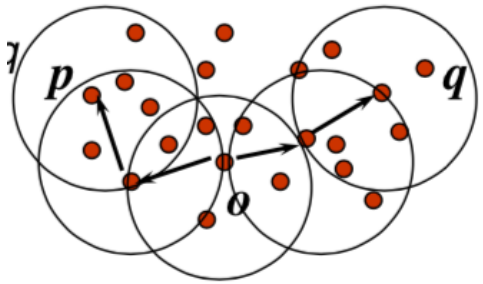
$\varepsilon = 1$ cm

Density-Reachable and Density-Connected

- Density-reachable:
  - A point p is density-reachable from a point q w.r.t. ϵ and MinPts, if there is a chain of points $p_1, ..., p_n$, $p_1 = q$, $p_n = p$ 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. ϵ and MinPts, if there is a point o such that both p and q are density-reachable from o w.r.t. ϵ and MinPts



(a) Density Reachable



(b) Density Connected

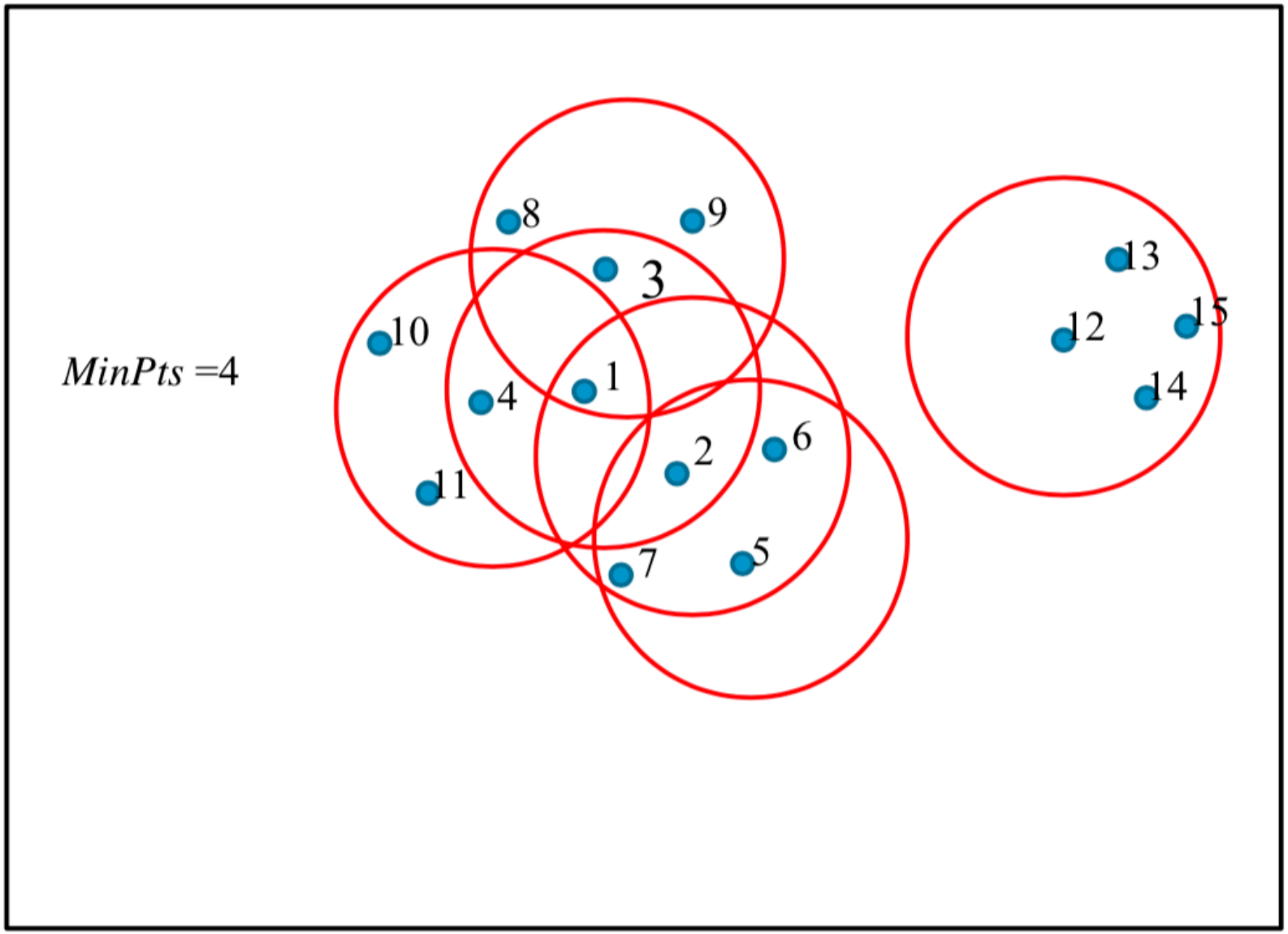
DBSCAN Algorithm

Density-Based Spatial Clustering of Applications with Noise

1. mark all the 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.

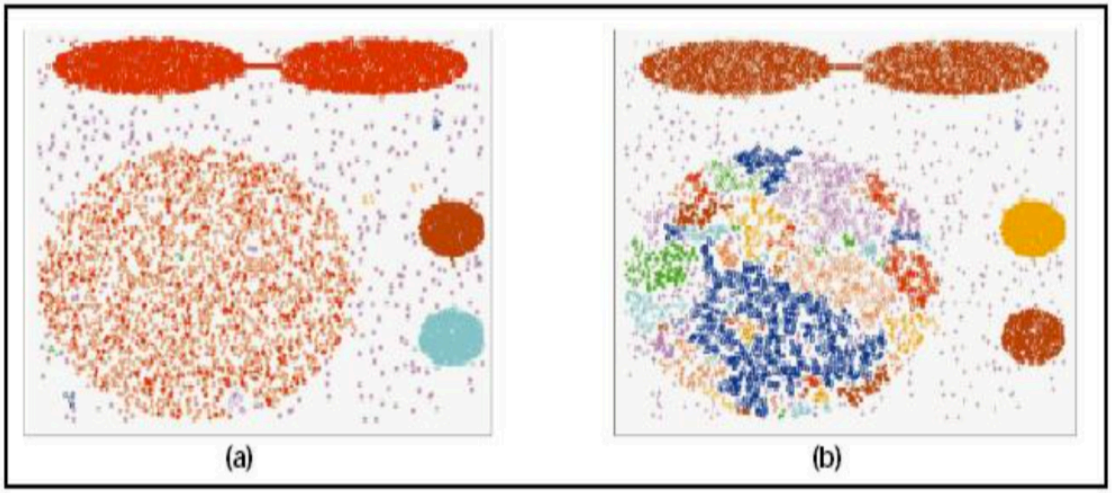
DBSCAN

Ester, et al. (KDD'96)

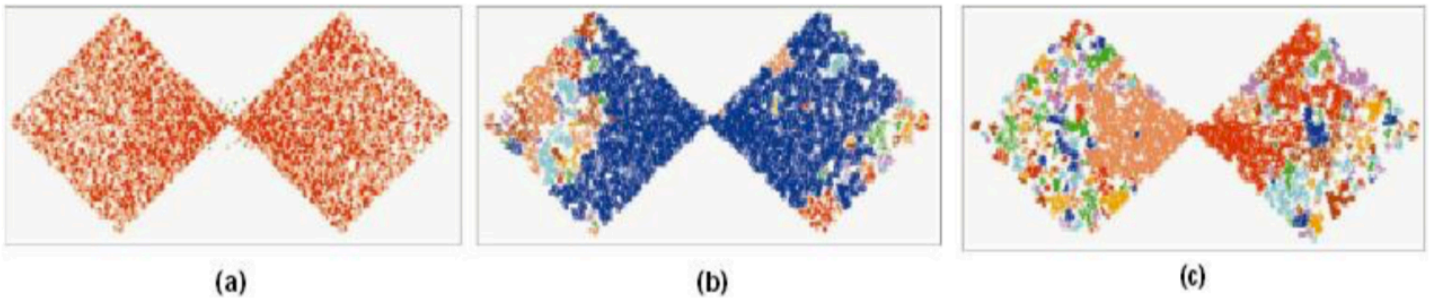


DBSCAN: Sensitive to Parameters

- DBSCAN results with MinPts $= 4$ and $\epsilon =$ (a) 0.5, (b) 0.4

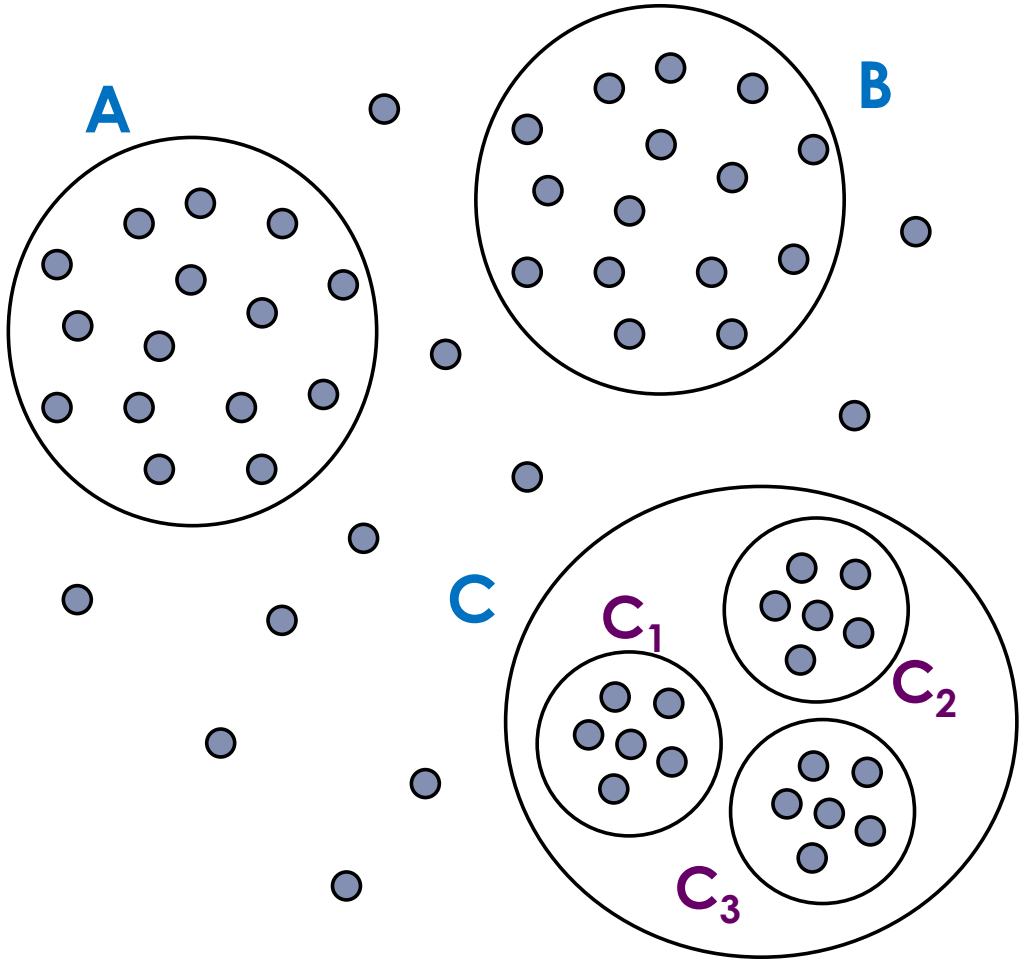


- DBSCAN results with MinPts $= 4$ and $\epsilon =$ (a) 5, (b) 4, (c) 3



OPTICS: Ankerst, et al (SIGMOD99)

Motivation



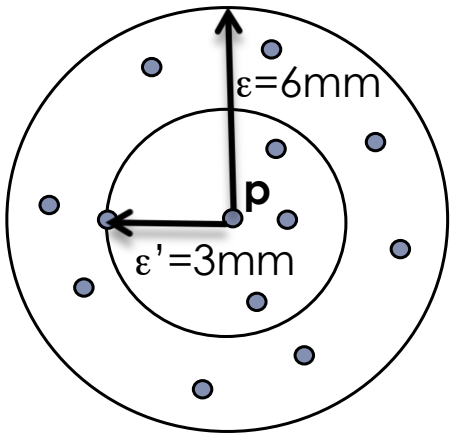
OPTICS: Ankerst, et al (SIGMOD99)

Ordering Points to Identify the Clustering Structure

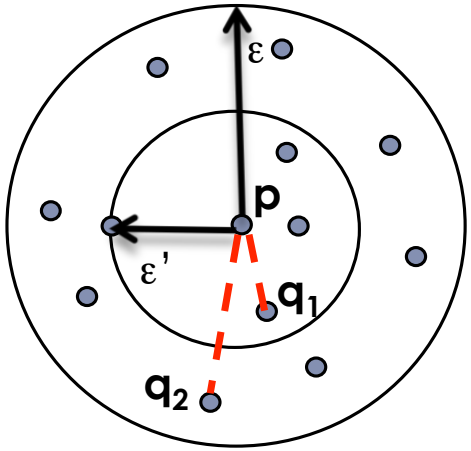
Motivation:

- Very different local densities may be needed to reveal clusters in different regions
- Clusters A,B,C_1,C_2, and C_3 cannot be detected using one global density parameter
- A global density parameter can detect either A,B,C or C_1,C_2,C_3
- **Solutions**
  - Use hierarchical clustering, but
    - Single link effect
    - Hard to interpret
  - Use OPTICS

- The **core-distance** of an object is smallest the ε' that makes p a core object
- If p is not a core object, the core distance of p is undefined
- Example (ε, MinPts=5)
  - ε' is the core distance of p
  - It is the distance between p and the fourth closest object



- The **reachability-distance** of an object q with respect to object to object p is:
  Max(core-distance(p), Euclidian(p,q))
- Example
  - Reachability-distance(q_1,p)=core-distance(p)=ε'
  - Reachability-distance(q_2,p)=Euclidian(q_2,p)



OPTICS

1. **OPTICS**(DB, ϵ, MinPts)
2. **for each** point p of DB
3. p.reachability-distance $=$ UNDEFINED
4. **for each** unprocessed point p of DB
5. $N =$ getNeighbors(p, ϵ)
6. mark p as processed
7. output p to the **ordered list**
8. **if** (core-distance(p, ϵ, Minpts) != UNDEFINED)
9. Seeds $=$ **empty priority queue**
10. **update**(N, p, Seeds, ϵ, Minpts)
11. **for each** next q in Seeds
12. $N' =$ getNeighbors(q, ϵ)
13. mark q as processed
14. output q to the ordered list
15. **if** (core-distance(q, ϵ, Minpts) != UNDEFINED)
16. **update**(N', q, Seeds, ϵ, Minpts)

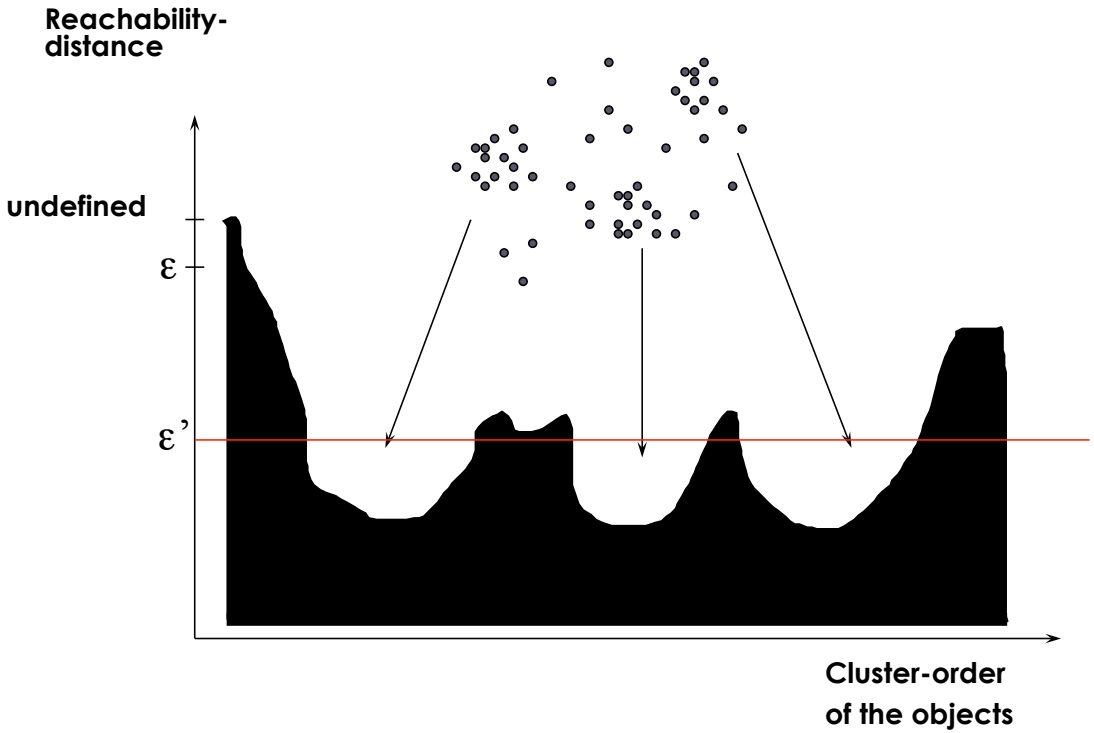
OPTICS

In update(), the priority queue Seeds is updated with the ε -neighborhood of p and q, respectively

1. **update**(N, p, Seeds, ϵ, Minpts)
2. coredist $=$ core-distance(p, ϵ, MinPts)
3. **for each** o in N
4. **if** (o is not processed)
5. new-reach-dist $=$ max(coredist, dist(p,o))
6. **if** (o.reachability-distance $==$ UNDEFINED) // o is not in Seeds
7. o.reachability-distance $=$ new-reach-dist
8. Seeds.insert(o, new-reach-dist)
9. **else** // o in Seeds, check for improvement
10. **if** (new-reach-dist $<$ o.reachability-distance)
11. o.reachability-distance $=$ new-reach-dist
12. Seeds.move-up(o, new-reach-dist)

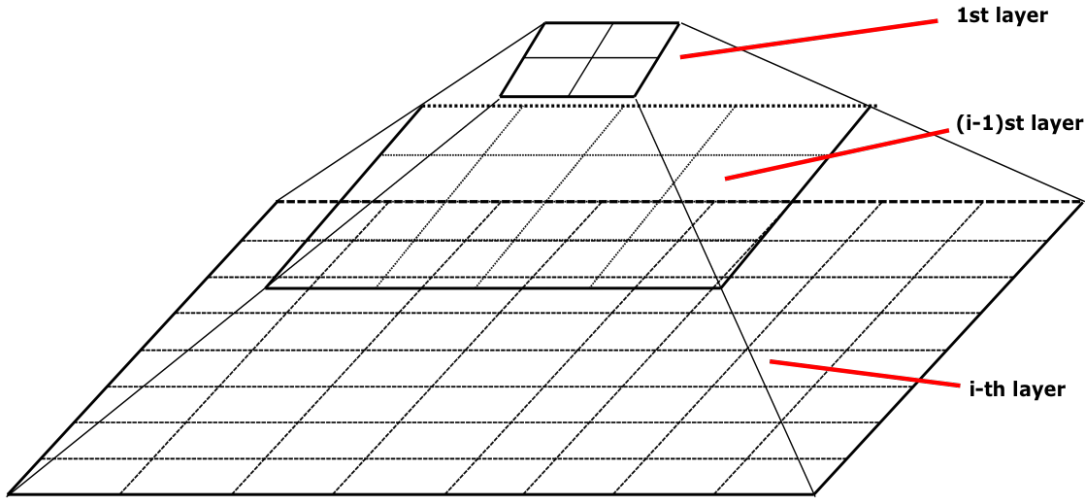
OPTICS
Output: Cluster-order

- OPTICS outputs the points in a particular ordering
- annotated with their smallest reachability distance (in the original algorithm, the core distance is also exported, but this is not required for further processing).



Grid-based Clustering Method

- The spatial area is divided into rectangular cells
- There are several levels of cells corresponding to different levels of resolution
- Each cell at a high level is partitioned into a number of smaller cells in the next lower level



Grid-based Clustering Method

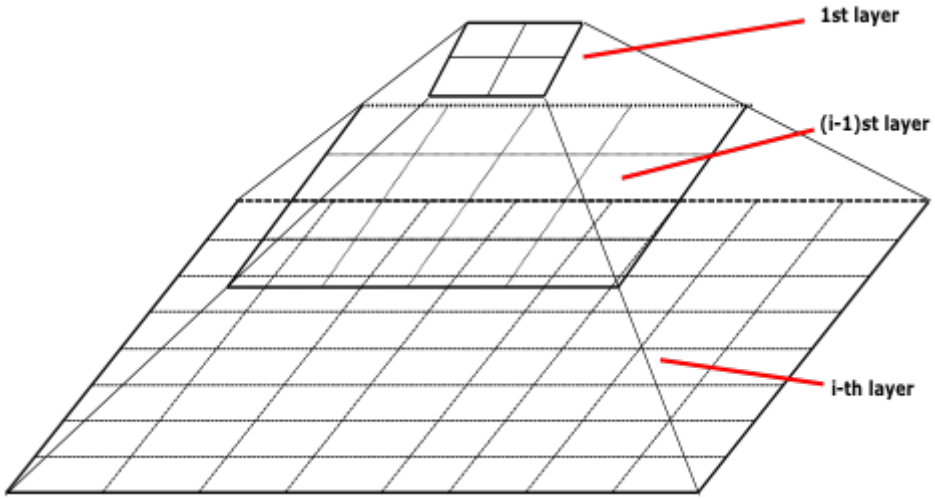
- Grid-Based Clustering: Explore multi-resolution grid data structure in clustering
  - Partition the data space into a finite number of cells to form a grid structure
  - Find clusters (dense regions) from the cells in the grid structure
- Features and challenges of a typical grid-based algorithm
  - Efficiency and scalability: # of cells $\ll$ # of data points
  - Uniformity: Uniform, hard to handle highly irregular data distributions
  - Locality: Limited by predefined cell sizes, borders, and the density threshold
  - Curse of dimensionality: Hard to cluster high-dimensional data

Grid-based Clustering Method

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

STING: A Statistical Information Grid Approach

- STING (Statistical Information Grid) (Wang, Yang and Muntz, VLDB' 97)
- The spatial area is divided into rectangular cells at different levels of resolution, and these cells form a tree structure
- A cell at a high level contains a number of smaller cells of the next lower level



The STING Clustering Method

- 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 cells
  - count, mean, stdev, min, max
  - type of distribution—normal, uniform, NONE, etc.
- Clusters are identified based on count, cell size, etc.

Region Query

- STING can be used to facilitate several kinds of spatial queries
- The most commonly asked query is **region query** which is to select regions that satisfy certain conditions
- E.g., 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, ∞) **AND** price RANGE (400000,∞) **WITH** PERCENT (0.7, 1)
  - **AND** AREA (100,∞)
  - **AND WITH** CONFIDENCE 0.9

The STING Query Method

To process a region query:

- 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
- Remove the irrelevant cells from further consideration
- When finish examining the current layer, proceed to the next lower level of the relevant cells
- Repeat this process until the bottom layer is reached

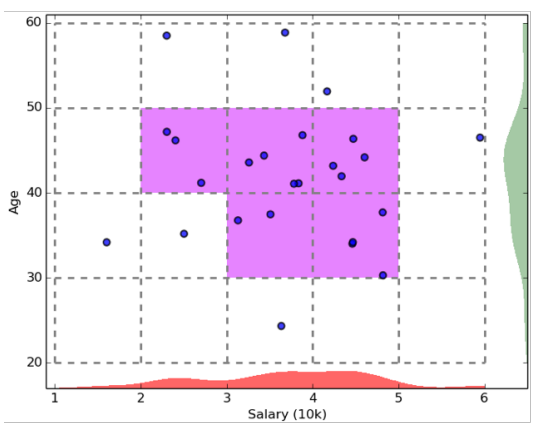
Comments on STING Clustering

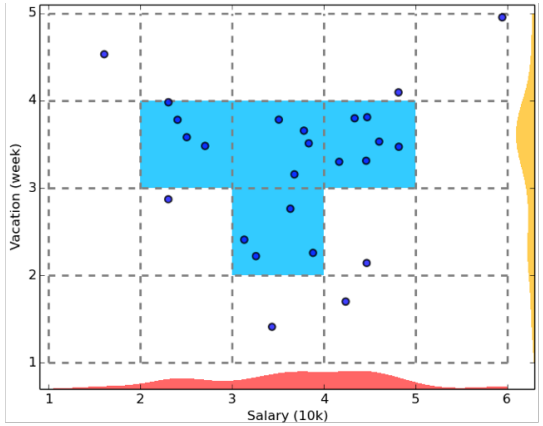
- Advantages:
  - Query-independent
  - incremental update
  - $O(K)$ for query, where K is the number of grid cells at the lowest level
  - $O(n)$ for generating clusters
- Disadvantages:
  - All the cluster boundaries are either horizontal or vertical, and no diagonal boundary is detected
  - Processing time depends on the size of each grid
  - Its probabilistic nature may imply a loss of accuracy in query processing

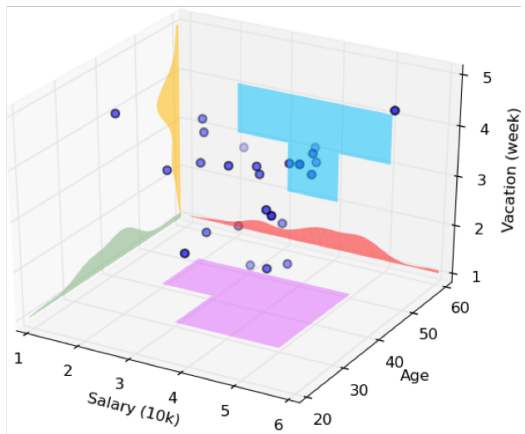
CLIQUE: Grid-based Subspace Clustering

- CLIQUE (Clustering In QUEst) (Agrawal, Gehrke, Gunopulos, Raghavan: SIGMOD' 98)
- CLIQUE is a density-based and grid-based subspace clustering algorithm
  - Grid-based: It discretizes the data space through a grid and estimates the density by counting the number of points in a grid cell
  - Density-based: A cluster is a maximal set of connected dense units in a subspace
    - A unit is dense if the fraction of total data points contained in the unit exceeds the input model parameter
  - Subspace clustering: A subspace cluster is a set of neighboring dense cells in an arbitrary subspace. It also discovers some minimal descriptions of the clusters
- It automatically identifies subspaces of a high dimensional data space that allow better clustering than original space using the Apriori principle

CLIQUE: SubSpace Clustering with Aprori Pruning







- Start at 1-D space and discretize numerical intervals in each axis into grid
- Find dense regions (clusters) in each subspace and generate their minimal descriptions
- Use the dense regions to find promising candidates in 2-D space based on the Apriori principle
- Repeat the above in level-wise manner in higher dimensional subspaces

Major Steps of the CLIQUE Algorithm

- Identify subspaces that contain clusters
  - 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 descriptions for the clusters
  - Determine maximal regions that cover a cluster of connected dense units for each cluster
  - Determine minimal cover for each cluster

Comments on CLIQUE

- Strengths
  - Automatically finds subspaces of the highest dimensionality as long as 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
- Weaknesses
  - As in all grid-based clustering approaches, the quality of the results crucially depends on the appropriate choice of the number and width of the partitions and grid cells

The Curse of Dimensionality

- Data in only one dimension is relatively packed
- Adding a dimension "stretch" the points across that dimension, making them further apart
- Adding more dimensions will make the points further apart —high dimensional data is extremely sparse
- Distance measure becomes meaningless —due to equi-distance

Handling High Dimensionality

- **Feature Transformation**
  - Transform the data into a small space while generally preserving the original relative distance between objects
  - Do not remove any of the original attributes
  - Irrelevant information may mask the real clusters
  - Difficult to interpret the resulting transformed attributes
- **Feature Subset Selection**
  - Remove irrelevant or redundant features
  - Find a subset of features that are relevant
  - Evaluate subsets of features using certain criteria

Attribute Subset Selection

- Some attributes can be irrelevant to the mining task
- Example
  - Classify customers whether or not they are likely to purchase a popular new CD
  - Attributes such customers's phone number are likely to be irrelevant unlike attributes such as age and music-taste
- First approach: Manual selection of attributes (by experts)
  - Difficult
  - Time consuming
  - The behavior of the data is not always well known
  - Leaving out relevant attributes and keeping irrelevant ones cause confusion
- Second approach: do attribute subset selection

Attribute Subset Selection

- Some attributes can be irrelevant to the mining task
- Find a minimum set of attributes such as the resulting probability distribution of the data classes is as close as possible to the original distribution
- Mining on a reduced set of attributes as an additional benefit
- It reduces the number of attributes appearing in discovered patterns
- Helps making the patterns easier to understand
- How can we find a "good" subset of attributes?

Attribute Subset Selection

- For n attributes, there are 2^n possible subsets
- Exhaustive search for optimal subset of attributes can be very expensive
- Heuristic methods are needed to reduce the search space
- Use greedy methods that looks for the best choice at the time
- "Best" and "Worse" attributes can be determined using
  - Statistical significance (assume independence between attributes)
  - Use evaluation measures such as information gain
  - ...

Basic Heuristic Methods

- **Stepwise forward selection**
  - Start with an empty set of attributes as the reduced set
  - The best of the original attributes is selected and added to the reduced set
  - Iterate until a stopping condition is satisfied
- **Stepwise backward elimination**
  - Start with the full set of attributes
  - At each step, remove the worst attribute remaining in the set
  - Iterate until a stopping condition is satisfied

Measuring Clustering Quality

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

Measuring Clustering Quality

Several references:

- Yanchi Liu, Zhongmou Li, Hui Xiong, Xuedong Gao, Junjie Wu, *Understanding of Internal Clustering Validation Measures*, 911-916, ICDM2010
- E.R. Dougherty, M. Brun, *A probabilistic theory of clustering*, Patter Recognition, 2004
- M. Brun, C. Sima, .et al, *Model-based evaluation of clustering measures*, Patter Recognition, 2007
- M. Halkidi, Y. Batistakis, and M. Vazirgiannis. *On Clustering Validation Techniques*. Journal of Intelligent Info. Systems, 17(2-3):107–145, 2001
- J. Han, M. Kamber, and J. Pei. *Data Mining: Concepts and Techniques*. Morgan Kaufmann, 3rd ed. , 2011
- H. Xiong and Z. Li. *Clustering Validation Measures*. in (Chapter 23) C. Aggarwal and C. K. Reddy (eds.), Data Clustering: Algorithms and Applications. CRC Press, 2014

What Is Outlier Discovery?

- What are outliers?
  - The set of objects are considerably dissimilar from the remaining of the data
  - Caused by
    - Measurement or execution errors
    - Result of inherent variability
- Mining outliers is valuable
- Applications:
  - Credit card fraud detection
  - Customer segmentation
  - Medical analysis

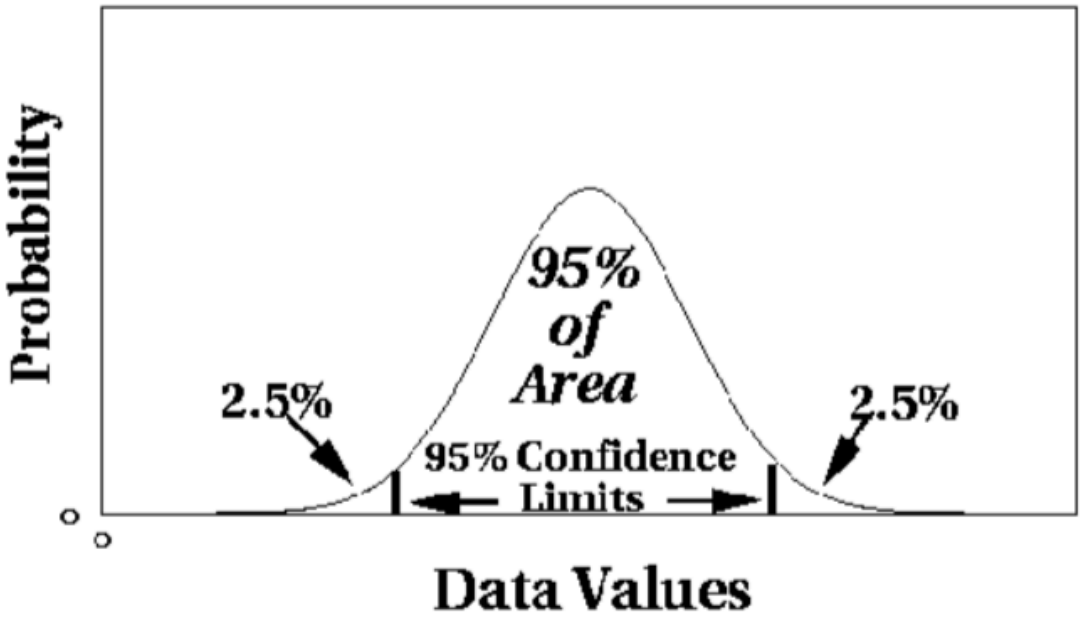
Outlier Detection

- Visualizaion
  - Weak in data with categorical data, high dimensional data
  - Good at numerical data of 2 or 3 dimensions
- Clustering
  - Byproduct of clustering may be outliers
- Computer-based methods
  - Statistical-based outlier detection
  - Distance-based outlier detection
  - Deviation-based outlier detection

Outlier Detection: Statistical Approaches

- Assume a distribution (e.g., normal distribution) for the data set and then use discordancy test to find outliers
- Discordancy tests depends on knowledge
  - data distribution
  - two hypothesis
  - distribution parameter (e.g., mean, variance)
  - number of expected outliers

Outlier Detection: Statistical Approaches



- Drawbacks
  - Most tests are for single attribute
  - In many cases, data distribution may not be known
  - Require input parameters

Outlier Detection: Distance-Based Approach

- Introduced to overcome the main limitations imposed by statistical methods
  - We need multi-dimensional analysis without knowing data distribution, no statistical test
- Distance-based outlier: A DB(p, d)-outlier is an object O in a dataset T such that at least a fraction p of the objects in T lies at a distance greater than d from O
- Algorithms for mining distance-based outliers
  - Index-based algorithm
  - Cell-based algorithm

Index-Based Algorithm

- Search for neighbors of each object O within radius d around the object
- Multi-dimensional index structure, e.g. kd tree
- Max number of objects within d-neighborhood of each outlier
- The worst case $O(kn^2)$: k dimensionality, n number of objects
- Drawbacks:
  - Tree building is computational intensive

Cell-based Algorithm

- Cell partition
  - Partition data space into cells, side length $d/2\sqrt{k}$
  - Each cell has two layers around it
    - First layer one cell thick
    - Second layer $(2\sqrt{k})$ cell thick
- Outlier detection
  - If count of the first layer $> M$, no outlier in this cell
  - If count of the second layer $<= M$, all objects are outliers
  - Otherwise, examine every object in the cell
- Good for large-scale data set

Outlier Detection: Deviation-Based Approach

- Identifies outliers by examining the main characteristics of objects in a group
- Objects that "deviate" from this description are considered outliers
- Sequential exception technique
  - Simulates the way in which humans can distinguish unusual objects from among a series of supposedly like objects
  - A sequence of subsets, $\{S_1, S_2, ..., S_m\}, S_{j-1} \subset S_j$
  - Calculate the dissimilarity difference between the current subset with the proceeding subset in the sequence

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
- Outlier detection and analysis are very useful for fraud detection, etc. and can be performed by statistical, distance-based or deviation-based approaches

Summary

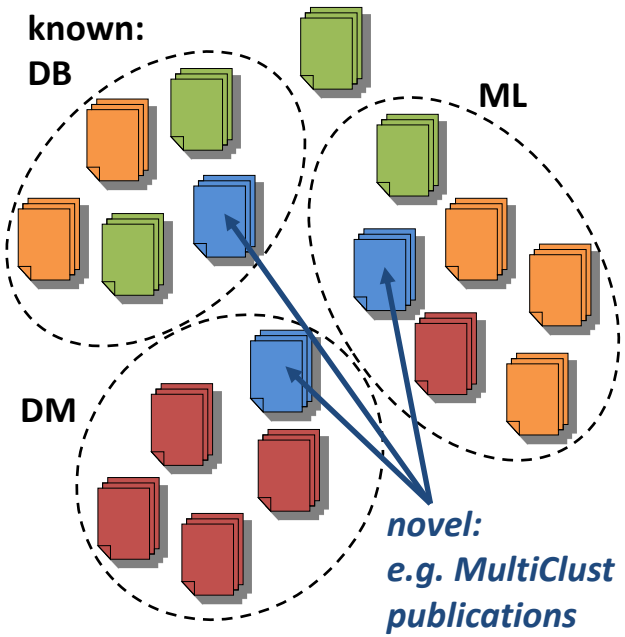
- Considerable progress has been made in scalable clustering methods
  - Partitioning: k-means, k-medoids, CLARANS
  - Hierarchical: BIRCH, ROCK, CHAMELEON
  - Density-based: DBSCAN, OPTICS, DenClue
  - Grid-based: STING, WaveCluster, CLIQUE
  - Model-based: EM, Fuzzy K-Means
  - Frequent pattern-based: pCluster
  - Constraint-based: COD, constrained-clustering

Traditional Cluster Detection

- Abstract cluster definition
  - Group similar objects in one group
  - separate dissimilar objects in different groups
- Several instances focus on:
  - different similarity functions, cluster characteristics, data types, . . .
  - Most definitions provide only a single clustering solution
- Aims at a **single partitioning** of the data: Each object is assigned to exactly one cluster
- Aims at **one clustering solution**: One set of K clusters forming the resulting groups of objects
- In contrast, we focus on **multiple clustering solutions...**

Text Analysis –Multiple Clusterings

- Detecting novel topics based on given knowledge, objects are text documents described by their content.
- **Aim**: Groups of documents on similar topic.
- **Challenge**:
- One document describes different topics simultaneously
- There are multiple alternative clustering solutions



Example Customer Analysis –Multiple Clusterings

object ID	age	income	blood pres.	sport activ.	profession
1	rich oldies				
2			healthy sporties		
3					
4			sport professionals		
5			unhealthy gamers		
6	average people				
7					
8	unemployed people				
9					

- Each object might be clustered by using multiple views
- For example, considering combinations of attributes
- For each object multiple clusters are detected
- Novel challenges in cluster definition, i.e. not only similarity of objects

Requirements for Multiple Clustering Solutions

- Informally, **Multiple Clustering Solutions** are...
  - Multiple sets of clusters providing more insights than only one solution
  - One given solution and a different grouping forming alternative solutions
- **Goals and objectives**:
  - Each object should be grouped in multiple clusters, representing different perspectives on the data.
  - The result should consist of many alternative solutions. Users may choose one or use multiple of these solutions.
  - Solutions should differ to a high extend, and thus, each of these solutions provides additional knowledge.
  - Overall, enhanced extraction of knowledge.
- Objectives are motivated by various application scenarios...
- Tutorial: http://dme.rwth-aachen.de/sites/default/files/public_files/dmcs-icml2013.pdf

References

- Jiawei Han, Micheline Kamber, and Jian Pei. Data Mining: Concepts and Techniques. Morgan Kaufmann, 3rd ed. , 2011 (Chapters 10 & 11)
- Charu Aggarwal and Chandran K. Reddy (eds.). Data Clustering: Algorithms and Applications. CRC Press, 2014
- Mohammed J. Zaki and Wagner Meira, Jr.. Data Mining and Analysis: Fundamental Concepts and Algorithms. Cambridge University Press, 2014
- L. Kaufman and P. J. Rousseeuw, Finding Groups in Data: An Introduction to Cluster Analysis, John Wiley & Sons, 1990
- Charu Aggarwal. An Introduction to Clustering Analysis. in Aggarwal and Reddy (eds.). Data Clustering: Algorithms and Applications (Chapter 1). CRC Press, 2014
- J. MacQueen. Some Methods for Classification and Analysis of Multivariate Observations. In Proc. of the 5th Berkeley Symp. on Mathematical Statistics and Probability, 1967
- S. Lloyd. Least Squares Quantization in PCM. IEEE Trans. on Information Theory, 28(2), 1982
- A. K. Jain and R. C. Dubes. Algorithms for Clustering Data. Prentice Hall, 1988
- R. Ng and J. Han. Efficient and Effective Clustering Method for Spatial Data Mining. VLDB'94

References

- B. Schölkopf, A. Smola, and K. R. Müller. Nonlinear Component Analysis as a Kernel Eigenvalue Problem. Neural computation, 10(5):1299–1319, 1998
- I. S. Dhillon, Y. Guan, and B. Kulis. Kernel K-Means: Spectral Clustering and Normalized Cuts. KDD' 04
- D. Arthur and S. Vassilvitskii. K-means++: The Advantages of Careful Seeding. SODA' 07
- C. K. Reddy and B. Vinzamuri. A Survey of Partitional and Hierarchical Clustering Algorithms, in (Chap. 4) Aggarwal and Reddy (eds.), Data Clustering: Algorithms and Applications. CRC Press, 2014
- A. K. Jain and R. C. Dubes. Algorithms for Clustering Data. Prentice Hall, 1988
- L. Kaufman and P. J. Rousseeuw. Finding Groups in Data: An Introduction to Cluster Analysis. John Wiley & Sons, 1990
- T. Zhang, R. Ramakrishnan, and M. Livny. BIRCH: An Efficient Data Clustering Method for Very Large Databases. SIGMOD'96
- S. Guha, R. Rastogi, and K. Shim. Cure: An Efficient Clustering Algorithm for Large Databases. SIGMOD' 98
- G. Karypis, E.-H. Han, and V. Kumar. CHAMELEON: A Hierarchical Clustering Algorithm Using Dynamic Modeling. COMPUTER, 32(8): 68-75, 1999.

References

- C. K. Reddy and B. Vinzamuri. A Survey of Partitional and Hierarchical Clustering Algorithms, in (Chap. 4) Aggarwal and Reddy (eds.), Data Clustering: Algorithms and Applications. CRC Press, 2014
- M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases. KDD'96
- W. Wang, J. Yang, R. Muntz, STING: A Statistical Information Grid Approach to Spatial Data Mining, VLDB' 97
- R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan. Automatic Subspace Clustering of High Dimensional Data for Data Mining Applications. SIGMOD' 98
- A. Hinneburg and D. A. Keim. An Efficient Approach to Clustering in Large Multimedia Databases with Noise. KDD' 98
- M. Ankerst, M. M. Breunig, H.-P. Kriegel, and J. Sander. Optics: Ordering Points to Identify the Clustering Structure. SIGMOD' 99
- M. Ester. Density-Based Clustering. In (Chapter 5) Aggarwal and Reddy (eds.), Data Clustering: Algorithms and Applications . CRC Press. 2014
- W. Cheng, W. Wang, and S. Batista. Grid-based Clustering. In (Chapter 6) Aggarwal and Reddy (eds.), Data Clustering: Algorithms and Applications. CRC Press. 2014

References

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