CS6220: DATA MINING TECHNIQUES

Matrix Data: Clustering: Part 1

Instructor: Yizhou Sun

yzsun@ccs.neu.edu

October 30, 2013

Announcement

- Homework 1 due next Monday (10/14)
- Course project proposal due next Wednesday (10/16)
  - Submit pdf file in blackboard
  - Sign-up for discussions on next Friday (15mins for each group)

Matrix Data: Clustering: Part 1

- Cluster Analysis: Basic Concepts
- Partitioning Methods
- Hierarchical Methods
- Density-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

Applications of Cluster Analysis

- Data reduction
  - Summarization: Preprocessing for regression, PCA, classification, and association analysis
  - Compression: Image processing: vector quantization
- Prediction based on groups
  - Cluster & find characteristics/patterns for each group
- 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

Clustering: Application Examples

- **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

Basic Steps to Develop a Clustering Task

- Feature selection
  - Select info concerning the task of interest
  - Minimal information redundancy
- Proximity measure
  - Similarity of two feature vectors
- Clustering criterion
  - Expressed via a cost function or some rules
- Clustering algorithms
  - Choice of algorithms
- Validation of the results
  - Validation test (also, *clustering tendency* test)
- Interpretation of the results
  - Integration with applications

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

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

Matrix Data: Clustering: Part 1

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

Partitioning Algorithms: Basic Concept

- Partitioning method: Partitioning a dataset ***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)

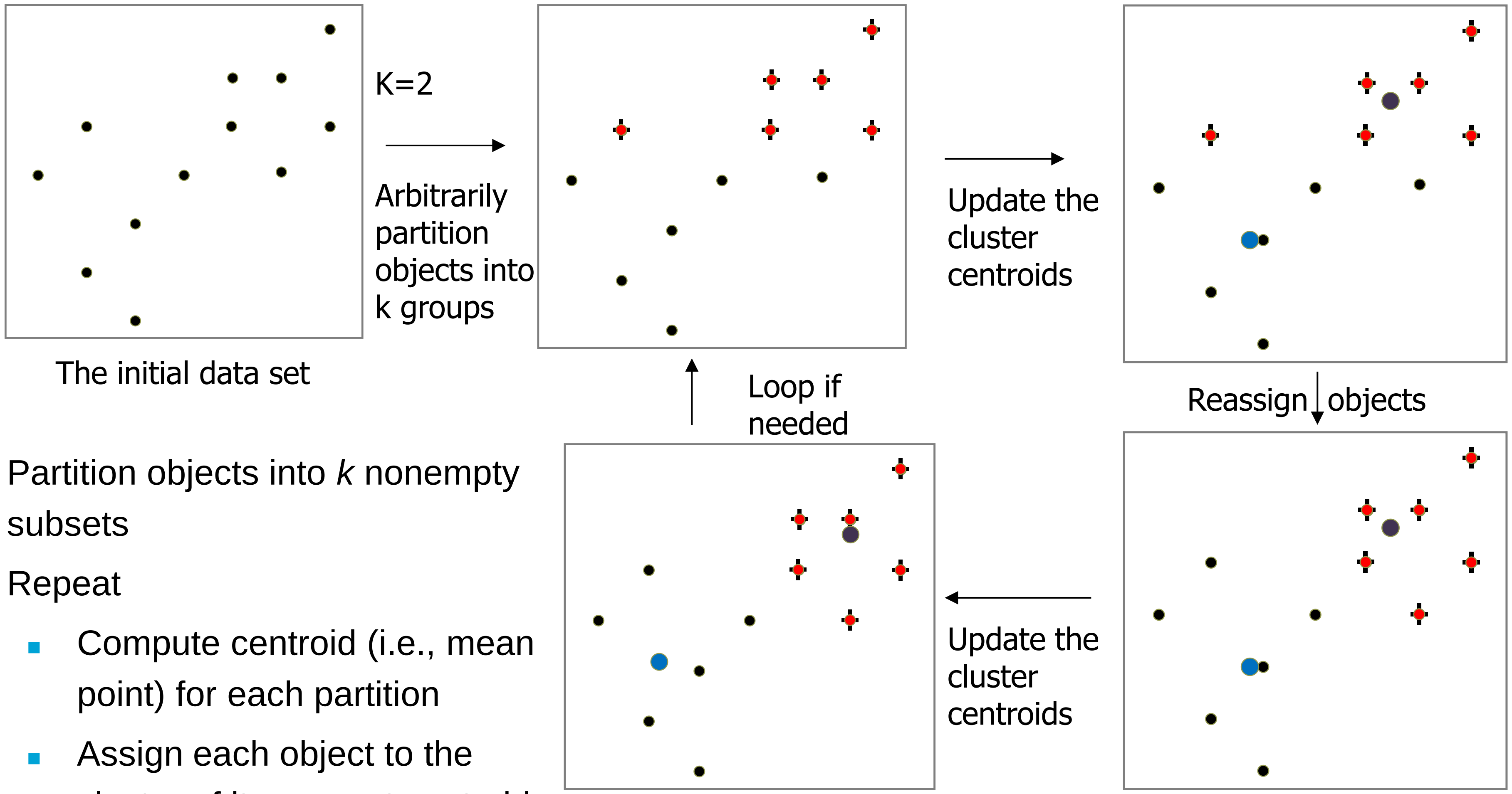
$$E = \Sigma_{i=1}^{k} \Sigma_{p \in C_i} (d(p, c_i))^2$$

- 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:
  - Partition objects into *k* nonempty subsets
  - 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)
  - Assign each object to the cluster with the nearest seed point
  - Go back to Step 2, stop when the assignment does not change

An Example of *K-Means* Clustering


K=2
Arbitrarily partition objects into k groups
The initial data set
Update the cluster centroids
Reassign objects
Update the cluster centroids
Loop if needed

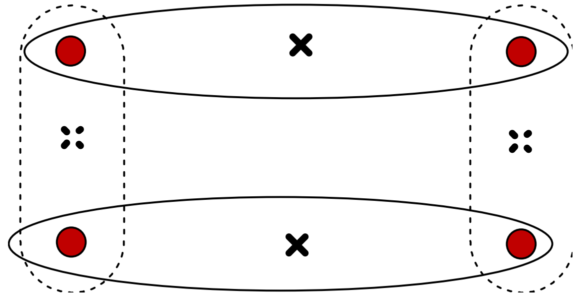
- Partition objects into *k* nonempty subsets
- Repeat
  - Compute centroid (i.e., mean point) for each partition
  - Assign each object to the cluster of its nearest centroid
- Until no change

Comments on the *K-Means* Method

- Strength: *Efficient*: $O(tkn)$, where n is # objects, k is # clusters, and t is # iterations. Normally, k, $t << n$.
- 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

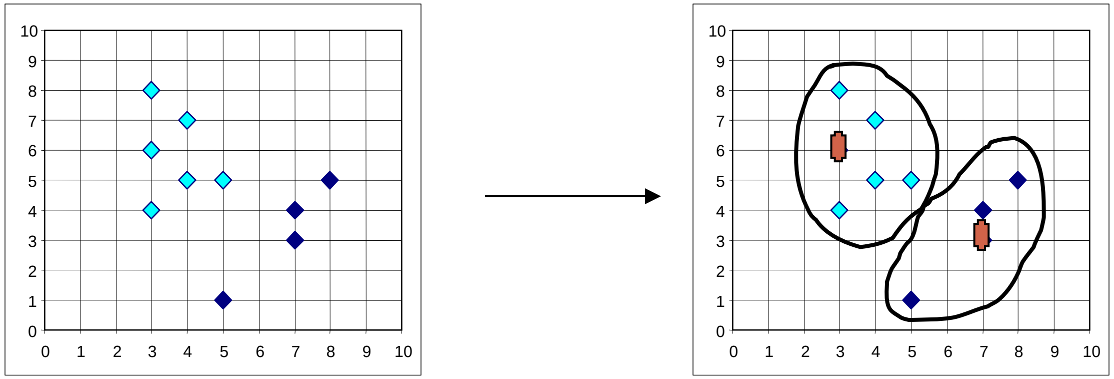
- Most of the variants of the *k-means* which differ in
  - Selection of the initial *k* means
  - Dissimilarity calculations
  - Strategies to calculate cluster means



- 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

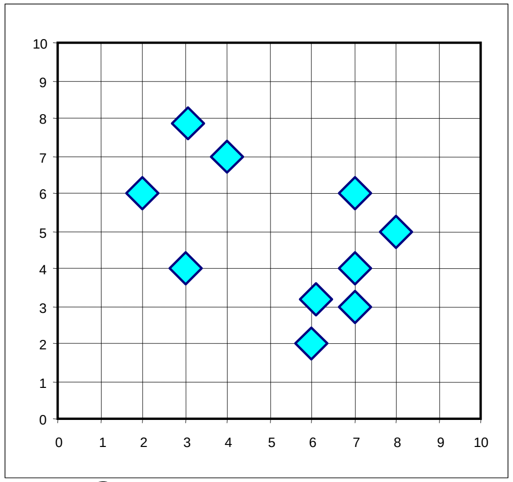
What Is the Problem of the K-Means Method?

- 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



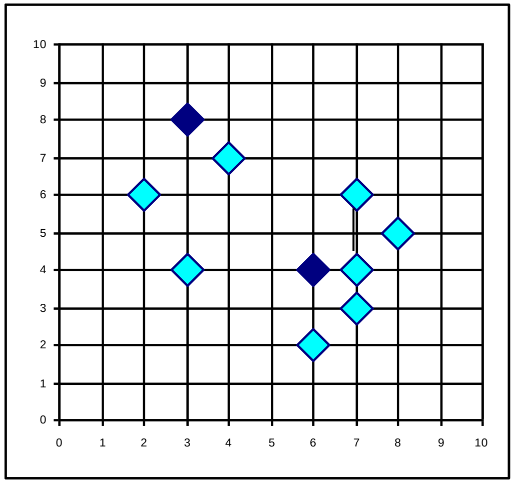
PAM: A Typical K-Medoids Algorithm

Total Cost = 20

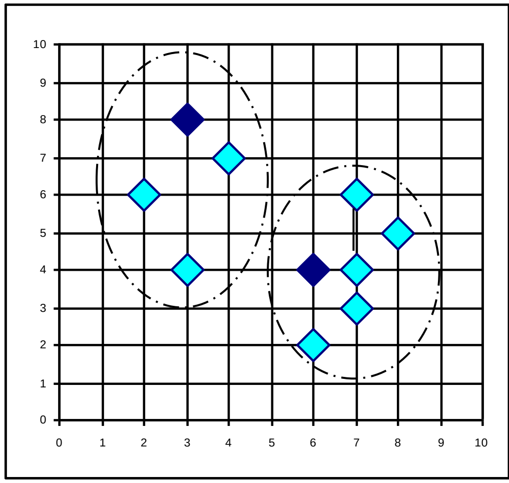


K=2

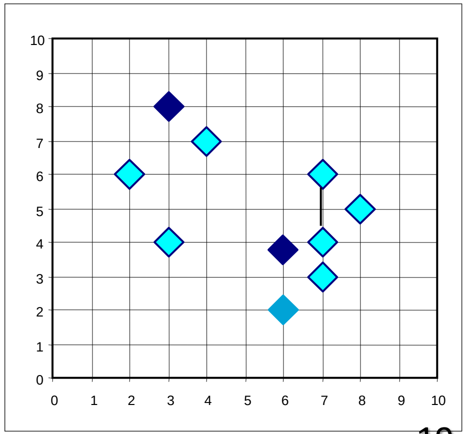
Arbitrary choose k object as initial medoids



Assign each remaining object to nearest medoids

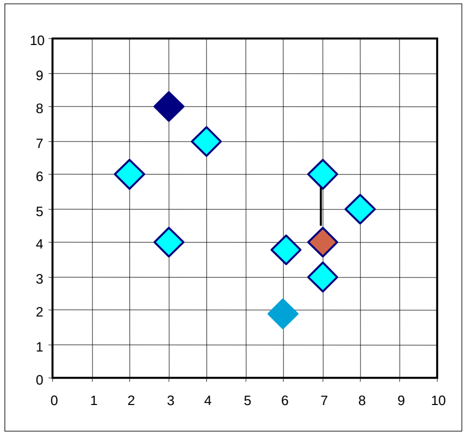


Randomly select a nonmedoid object,O_{ramdom}



Compute total cost of swapping

Total Cost = 26



Swapping O and O_{ramdom}

If quality is improved.

Do loop

Until no change

The K-Medoid Clustering Method

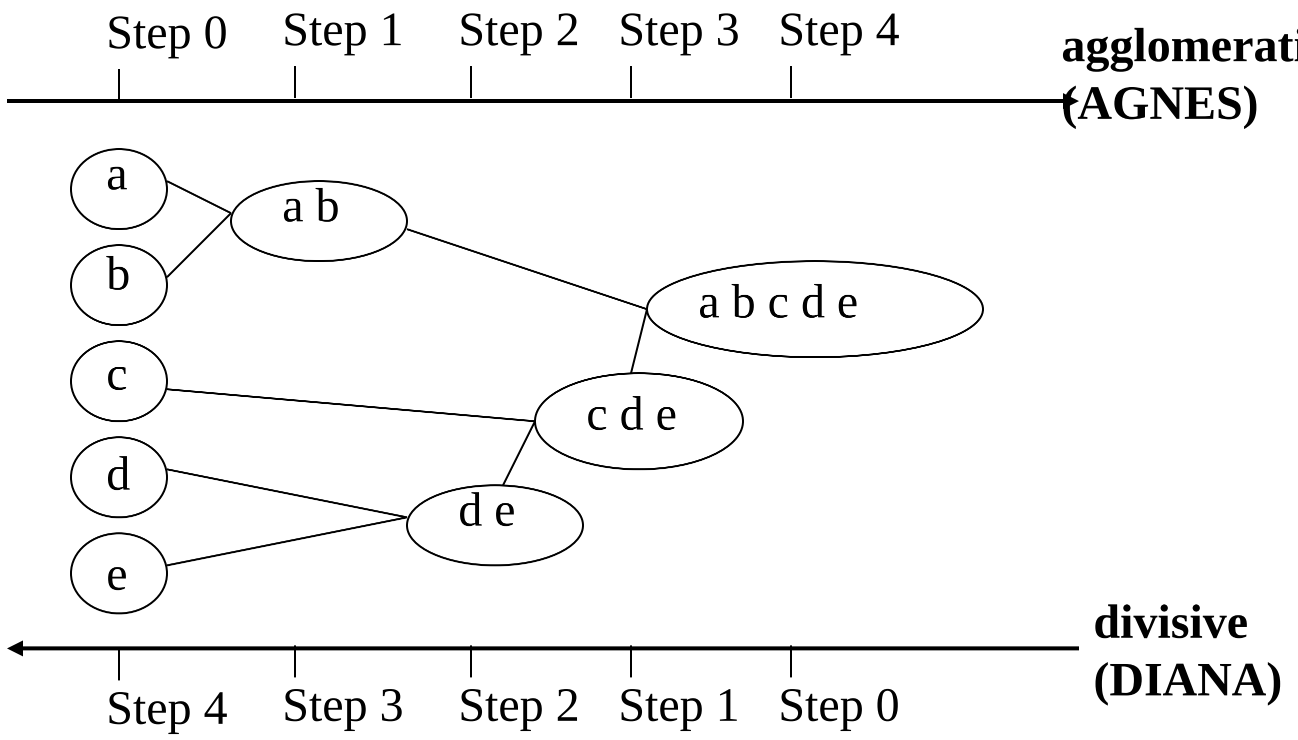
- *K-Medoids* Clustering: Find *representative* objects (medoids) in clusters
  - *PAM* (Partitioning Around Medoids, Kaufmann & Rousseeuw 1987)
    - Starts from an initial set of medoids and iteratively replaces one of the medoids by one of the non-medoids if it improves the total distance of the resulting clustering
    - *PAM* works effectively for small data sets, but does not scale well for large data sets (due to the computational complexity)
- Efficiency improvement on PAM
  - *CLARA* (Kaufmann & Rousseeuw, 1990): PAM on samples
  - *CLARANS* (Ng & Han, 1994): Randomized re-sampling

Matrix Data: Clustering: Part 1

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

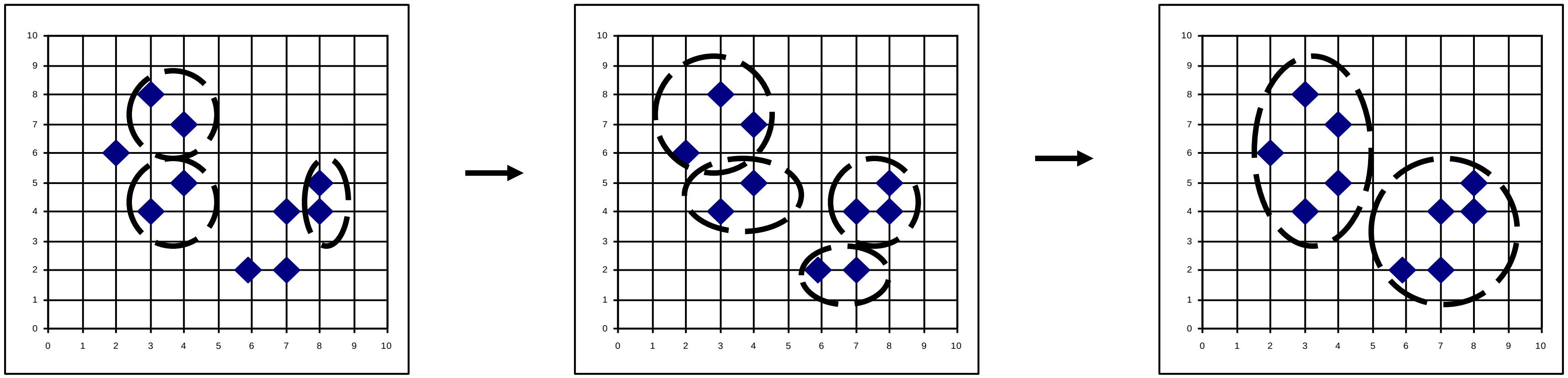
Hierarchical Clustering

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

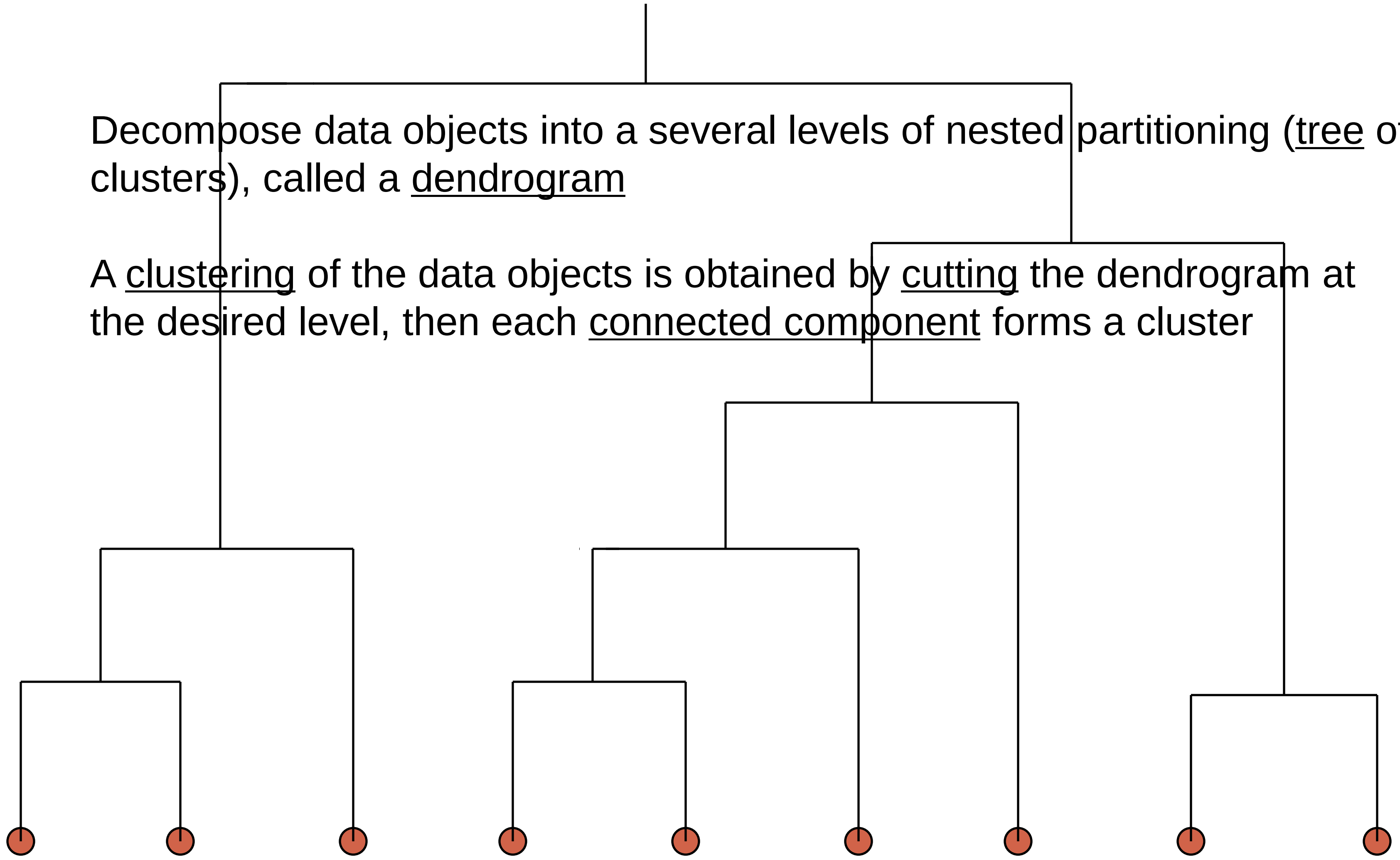


AGNES (Agglomerative Nesting)

- 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

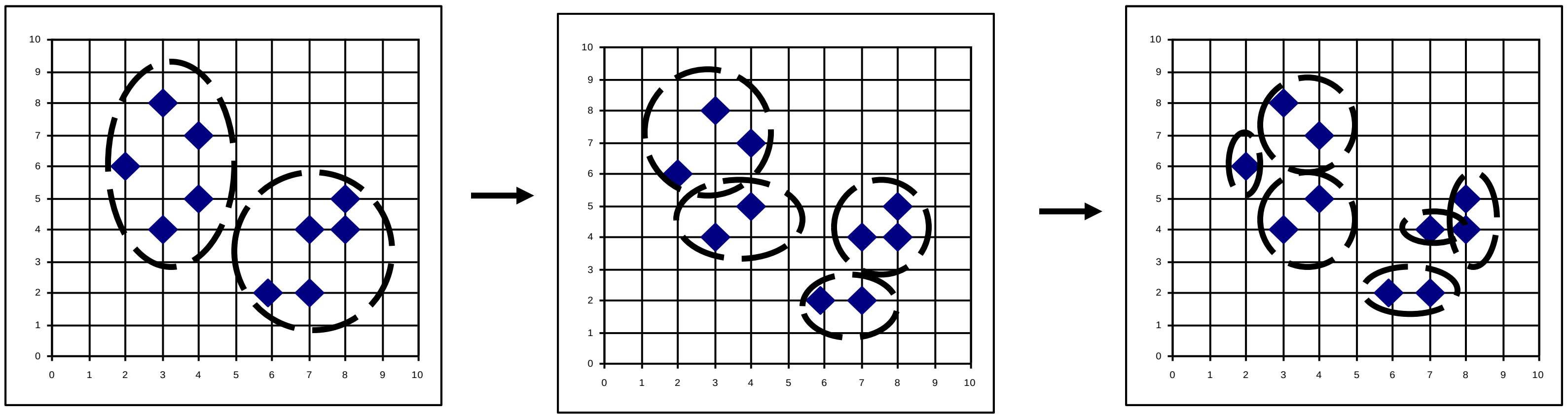


Decompose data objects into a several levels of nested partitioning (tree of clusters), called a dendrogram

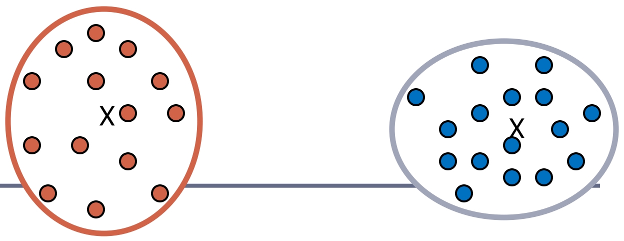
A clustering of the data objects is obtained by cutting the dendrogram at the desired level, then each connected component forms a cluster

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., 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., 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., dist(K_i, K_j) = avg(t_{ip}, t_{jq})
- Centroid: distance between the centroids of two clusters, i.e., dist(K_i, K_j) = dist(C_i, C_j)
- Medoid: distance between the medoids of two clusters, i.e., dist(K_i, K_j) = 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

$$C_m = \frac{\sum_{i=1}^{N} (t_{ip})}{N}$$

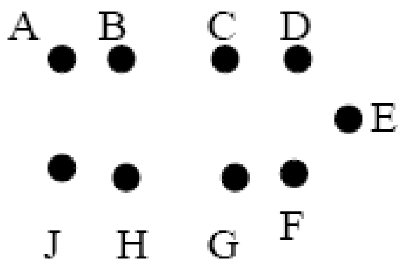
- Radius: square root of average distance from any point of the cluster to its centroid

$$R_m = \sqrt{\frac{\sum_{i=1}^{N} (t_{ip} - c_m)^2}{N}}$$

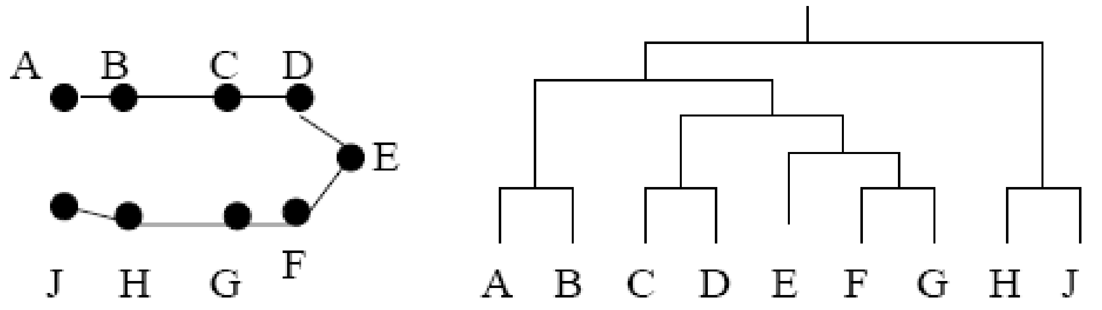
- Diameter: square root of average mean squared distance between all pairs of points in the cluster

$$D_m = \sqrt{\frac{\sum_{i=1}^{N} \sum_{i=1}^{N} (t_{ip} - t_{iq})^2}{N(N-1)}}$$

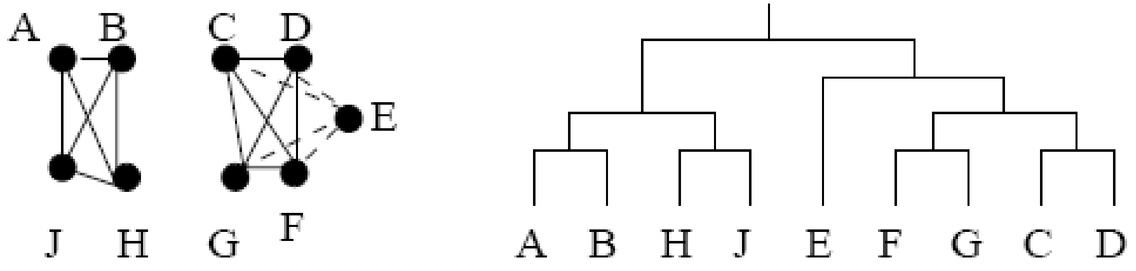
Example: Single Link vs. Complete Link



(a) Data set



(b) Clustering using single linkage



(c) Clustering using complete linkage

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

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

*A Probabilistic Hierarchical Clustering Algorithm

- For a set of objects partitioned into *m* clusters $C_1, \ldots, C_m$, the quality can be measured by,

$$Q(\{C_1, \ldots, C_m\}) = \prod_{i=1}^{m} P(C_i)$$

where *P*() is the maximum likelihood

- If we merge two clusters C_{j1} and C_{j2} into a cluster $C_{j1} \cup C_{j2}$, then, the change in quality of the overall clustering is

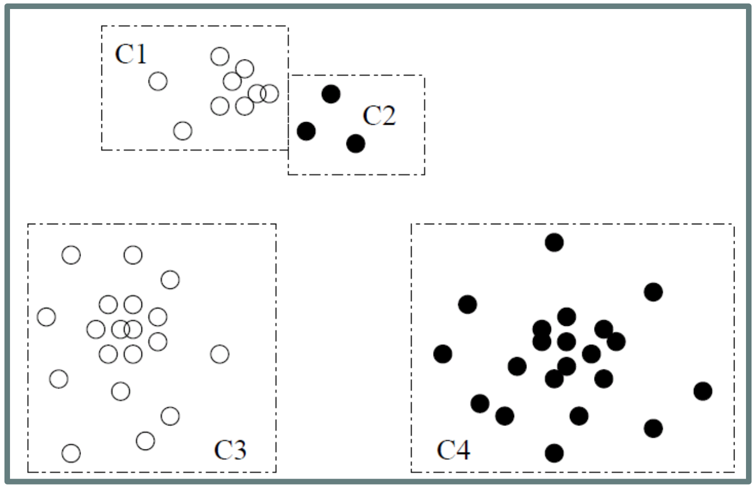
$$Q((\{C_1, \ldots, C_m\} - \{C_{j_1}, C_{j_2}\}) \cup \{C_{j_1} \cup C_{j_2}\}) - Q(\{C_1, \ldots, C_m\})$$

$$= \frac{\prod_{i=1}^{m} P(C_i) \cdot P(C_{j_1} \cup C_{j_2})}{P(C_{j_1}) P(C_{j_2})} - \prod_{i=1}^{m} P(C_i)$$

$$= \prod_{i=1}^{m} P(C_i) \left(\frac{P(C_{j_1} \cup C_{j_2})}{P(C_{j_1}) P(C_{j_2})} - 1 \right)$$

- Distance between clusters C_1 and C_2:

$$dist(C_i, C_j) = -\log \frac{P(C_1 \cup C_2)}{P(C_1) P(C_2)}$$



Matrix Data: Clustering: Part 1

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

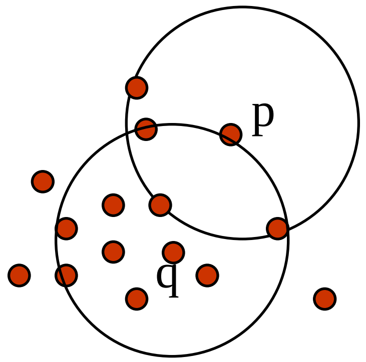
Density-Based Clustering Methods

- 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)

DBSCAN: Basic Concepts

- Two parameters*:*
    - *Eps*: Maximum radius of the neighborhood
    - *MinPts*: Minimum number of points in an Eps-neighborhood of that point
- $N_{Eps}(q)$: {p belongs to D | dist(p,q) ≤ 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:

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

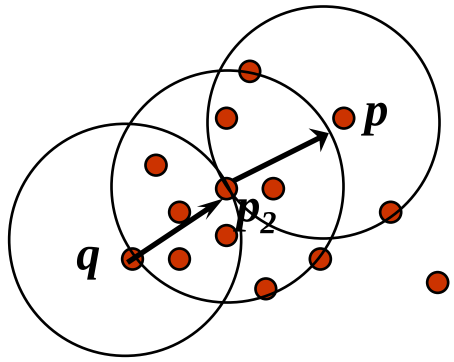


MinPts = 5

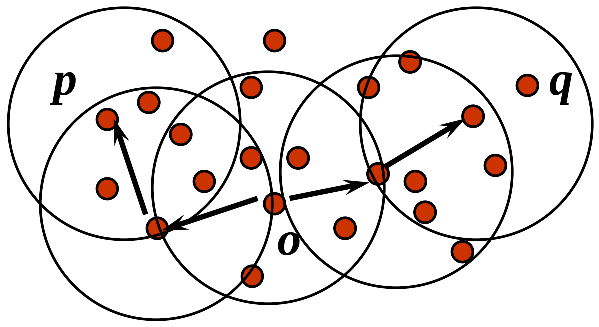
Eps = 1 cm

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 $p_1, \ldots, p_n$, $p_1 = q$, $p_n = p$ such that p_{i+1} is directly density-reachable from p_i

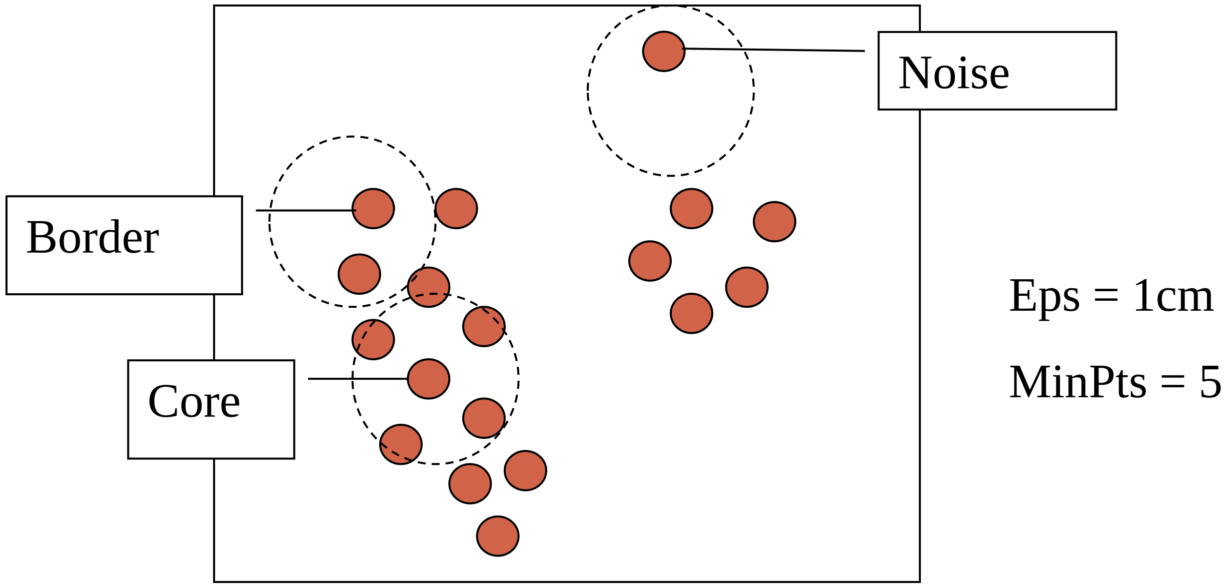


- Density-connected
  - A point p is density-connected to a point q w.r.t. *Eps*, *MinPts* if there is a point o such that both, p and q are density-reachable from o w.r.t. *Eps* and *MinPts*



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
- Noise: object not contained in any cluster is noise
- Discovers clusters of arbitrary shape in spatial databases with noise



DBSCAN: The Algorithm

```
(1)  mark all objects as unvisited;
(2)  do
(3)      randomly select an unvisited object $p$;
(4)      mark $p$ as visited;
(5)      if the $\epsilon$-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 $\epsilon$-neighborhood of $p$;
(8)          for each point $p'$ in $N$
(9)              if $p'$ is unvisited
(10)                 mark $p'$ as visited;
(11)                 if the $\epsilon$-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;
```

- *If a spatial index is used, the computational complexity of DBSCAN is O(nlogn), where n is the number of database objects. Otherwise, the complexity is O(n^2)*

DBSCAN: Sensitive to Parameters

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

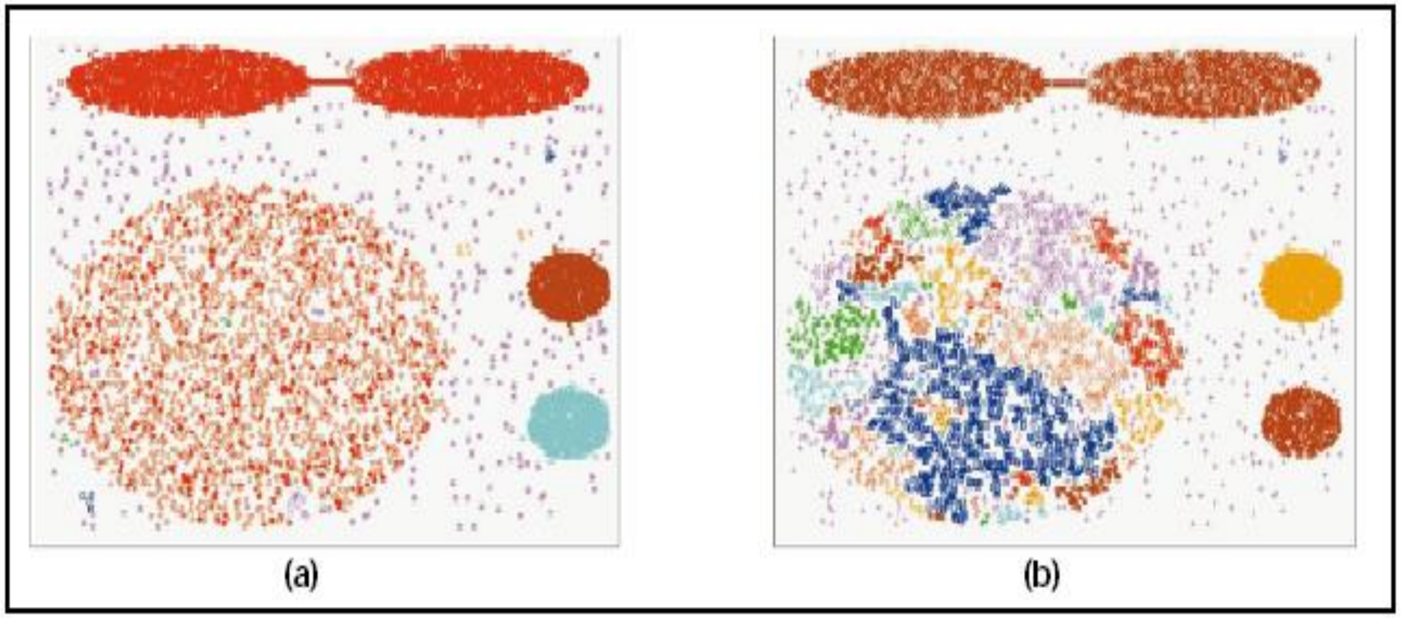
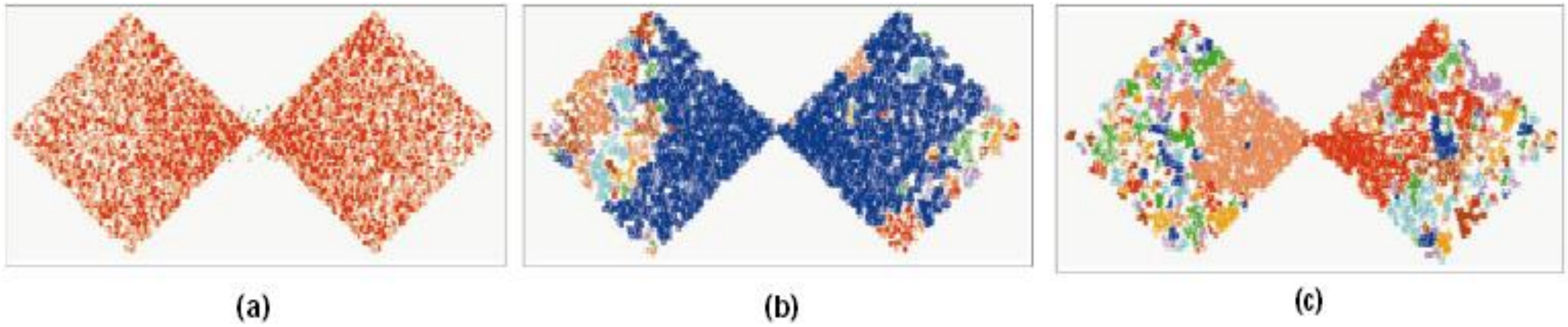


Figure 9. DBScan results for DS2 with MinPts at 4 and Eps at (a) 5.0, (b) 3.5, and (c) 3.0.



DBSCAN online Demo:

http://webdocs.cs.ualberta.ca/~yaling/Cluster/Applet/Code/Cluster.html

Questions about Parameters

- Fix Eps, increase MinPts, what will happen?
- Fix MinPts, decrease Eps, what will happen?

*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
  - Index-based time complexity: O(N*logN)

OPTICS: Some Extension from DBSCAN

- Core Distance of an object p: the smallest value ε' such that the ε-neighborhood of p has at least MinPts objects
  - Let $N_\varepsilon(p)$: ε-neighborhood of p, ε is a distance value; $card(N_\varepsilon(p))$: the size of set $N_\varepsilon(p)$
  - Let MinPts-distance(p): the distance from p to its MinPts' neighbor

$$\text{Core-distance}_{\varepsilon,\ MinPts}(p) = \begin{cases} \text{Undefined, if } card(N_\varepsilon(p)) < MinPts \\ \text{MinPts-distance}(p)\text{, otherwise} \end{cases}$$

- Reachability Distance of object p from core object q is the min radius value that makes p density-reachable from q
  - Let distance(q,p) be the Euclidean distance between q and p

$$\text{Reachability-distance}_{\varepsilon,\ \text{MinPts}}(p, q) = \begin{cases} \text{Undefined, if } q \text{ is not a core object} \\ \max(\text{core-distance}(q),\ \text{distance}(q, p)),\ \text{otherwise} \end{cases}$$

Core Distance & Reachability Distance

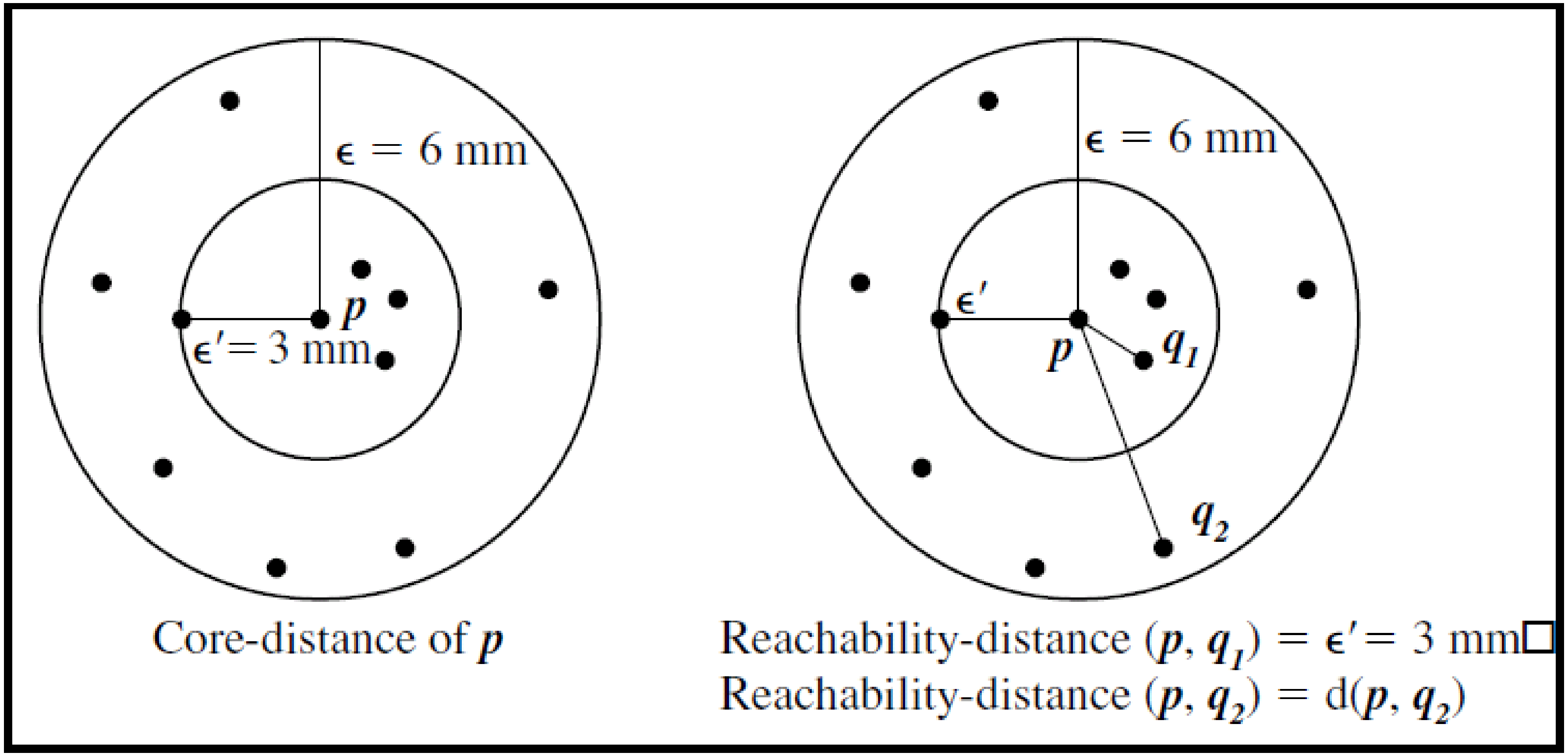
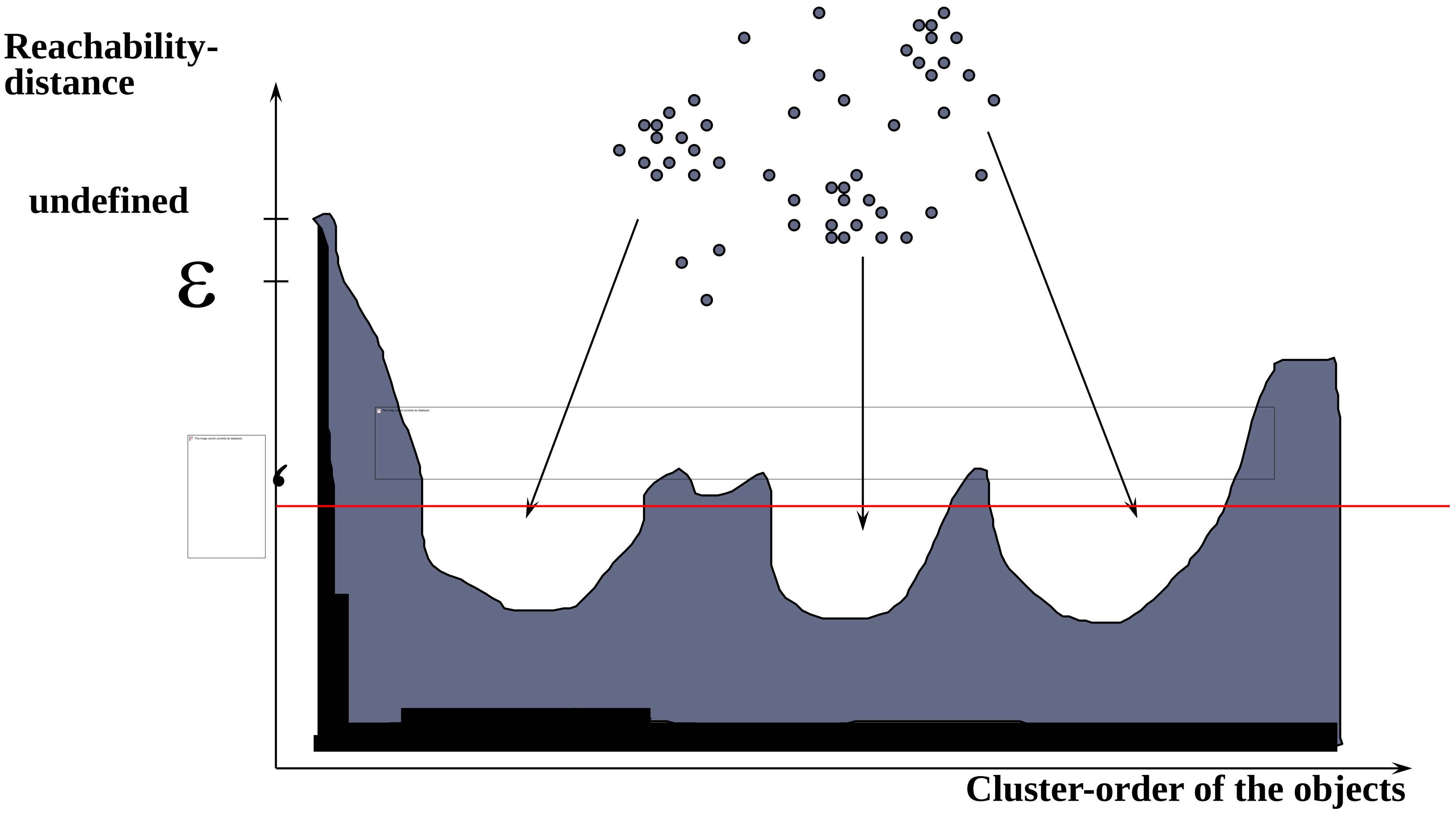


Figure 10.16: OPTICS terminology. Based on [ABKS99].

$$\varepsilon = 6mm, MinPts = 5$$

Output of OPTICS: cluster-ordering



Effects of Parameter Setting

ent.

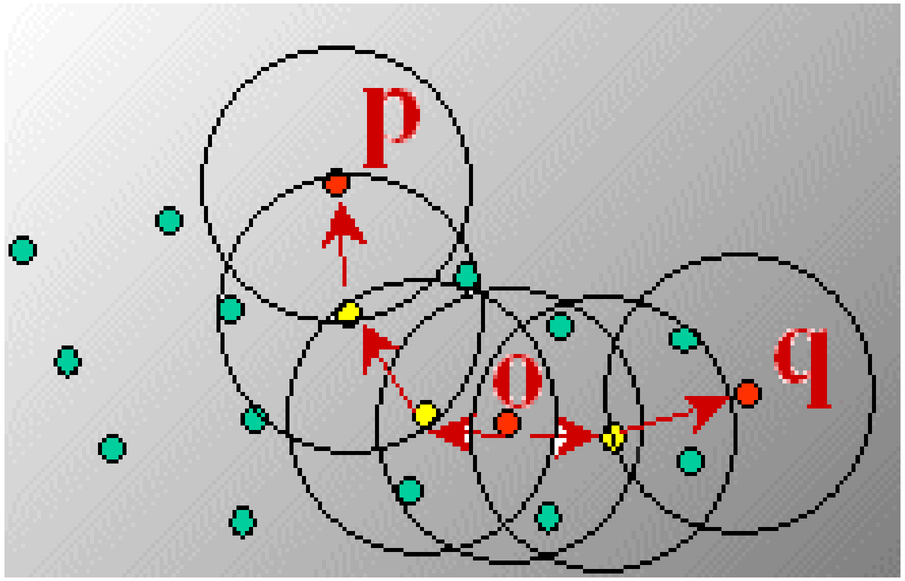
This image cannot currently be displayed.

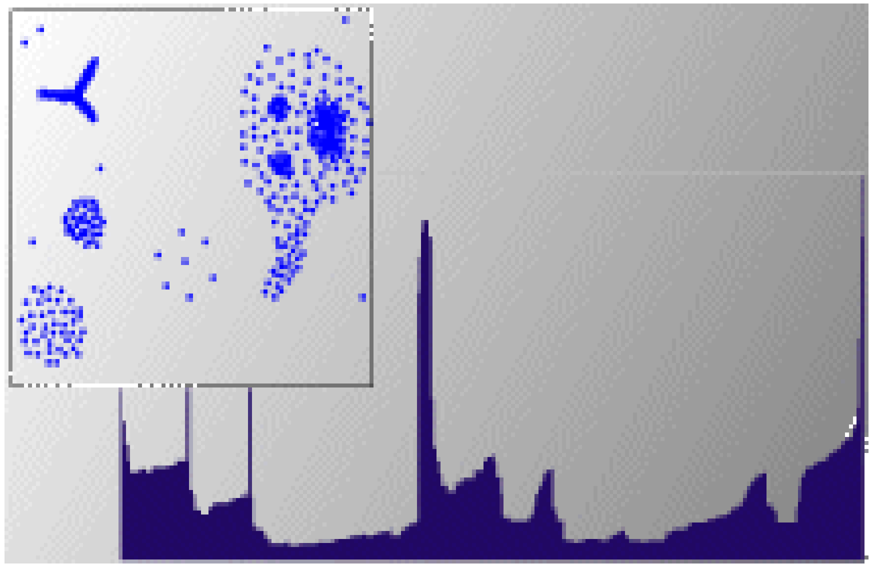
Extract DBSCAN-Clusters

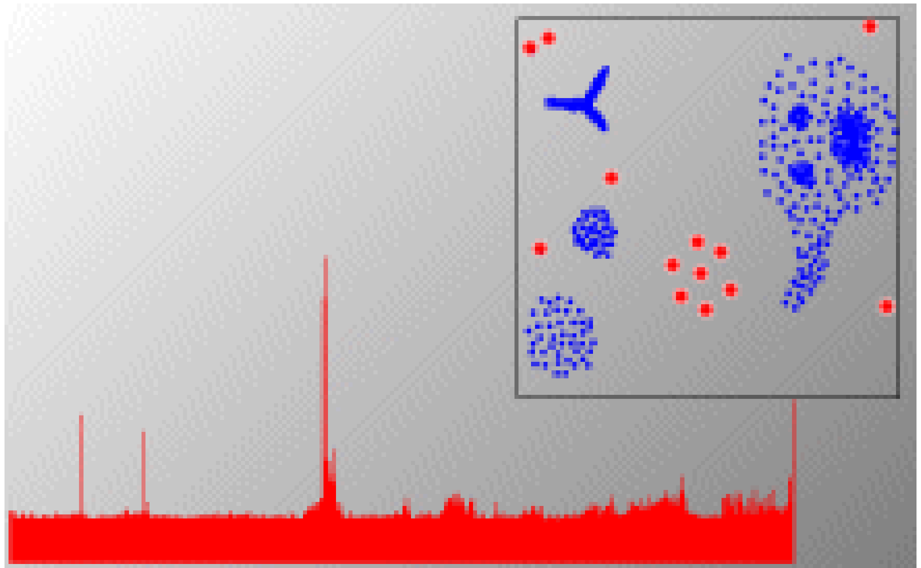
```
ExtractDBSCAN-Clustering (ClusterOrderedObjs,$\varepsilon'$, MinPts)
// Precondition: $\varepsilon' \leq$ generating dist $\varepsilon$ for ClusterOrderedObjs
   ClusterId := NOISE;
   FOR i FROM 1 TO ClusterOrderedObjs.size DO
      Object := ClusterOrderedObjs.get(i);
      IF Object.reachability_distance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

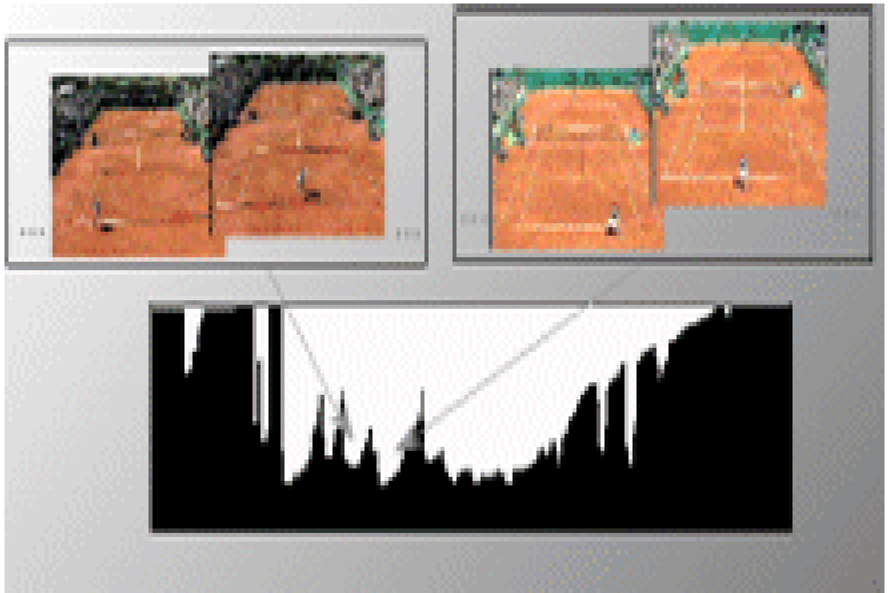
Density-Based Clustering: OPTICS & Applications

demo: http://www.dbs.informatik.uni-muenchen.de/Forschung/KDD/Clustering/OPTICS/Demo









*DENCLUE: Using Statistical Density Functions

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

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

influence of y on x

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

total influence on x

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

gradient of x in the direction of x_i

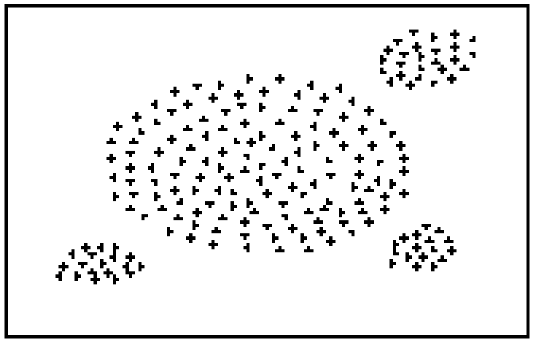
- Major features
  - Solid mathematical foundation
  - Good for data sets with large amounts of noise
  - Allows a compact mathematical description of arbitrarily shaped clusters in high-dimensional data sets
  - Significant faster than existing algorithm (e.g., DBSCAN)
  - But needs a large number of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

Denclue: Technical Essence

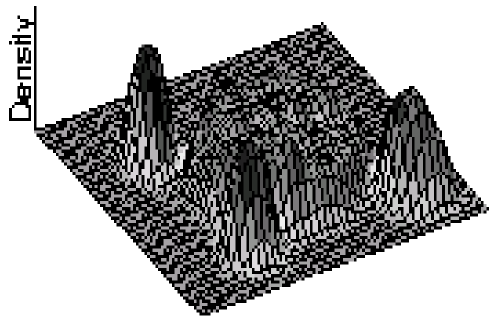
- Overall density of the data space can be calculated as the sum of the influence function of all data points
  - Influence function: describes the impact of a data point within its neighborhood
- Clusters can be determined mathematically by identifying density attractors
  - Density attractors are local 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

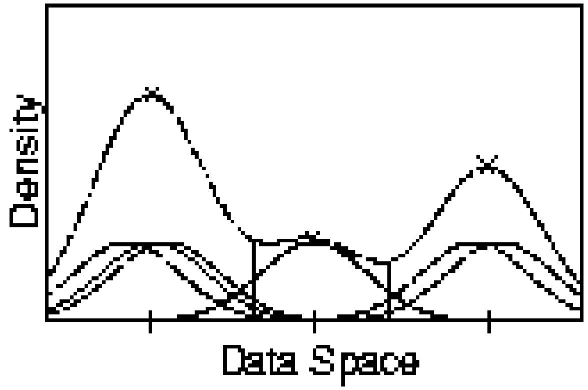
Can be detected by hill-climbing procedure of finding local maximums



(a) Data Set



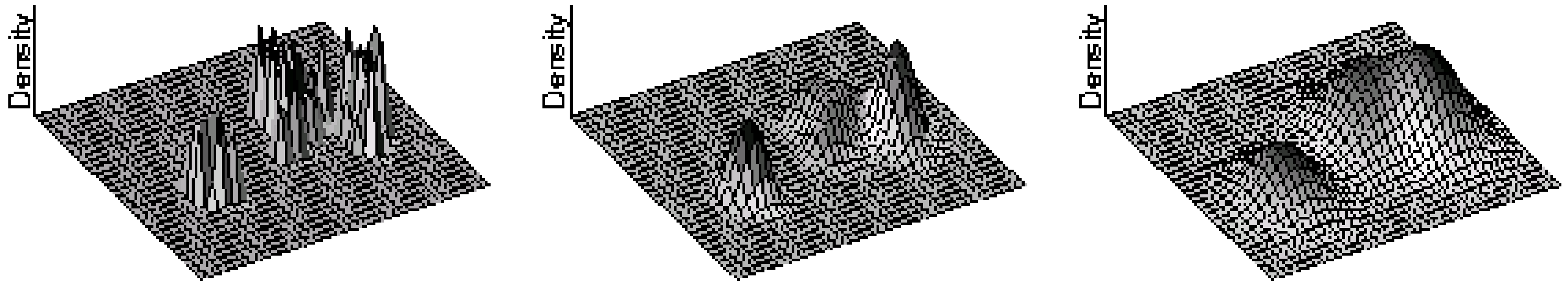
(c) Gaussian



Noise Threshold

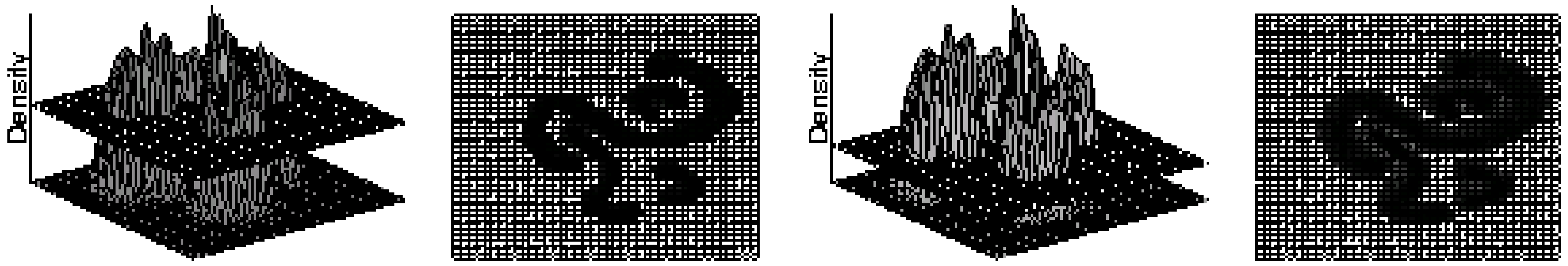
- Noise Threshold ξ
  - Avoid trivial local maximum points
  - A point can be a density attractor only if $\hat{f}(x) \geq \xi$

Center-Defined and Arbitrary



(a) $\sigma = 0.2$ (b) $\sigma = 0.6$ (d) $\sigma = 1.5$

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



(a) $\xi = 2$ (b) $\xi = 2$ (c) $\xi = 1$ (d) $\xi = 1$

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

Matrix Data: Clustering: Part 1

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

Evaluation of Clustering

- Assessing Clustering Tendency
- Determining the number of clusters
- Measuring clustering quality

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 Statistic**

Determine the Number of Clusters

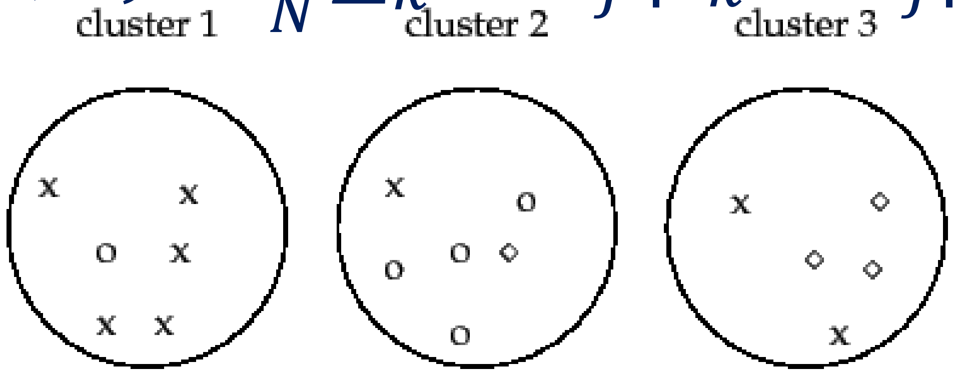
- Empirical method
  - # of clusters ≈$\sqrt{n/2}$ for a dataset of n points
- Elbow method
  - Use the turning point in the curve of sum of within cluster variance w.r.t the # of clusters
- Cross validation method
  - Divide a given data set into *m* parts
  - Use *m* – 1 parts to obtain a clustering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

Measuring Clustering Quality

- Two methods: extrinsic vs. intrinsic
- Extrinsic: supervised, i.e., the ground truth is available
  - Compare a clustering against the ground truth using certain clustering quality measure
  - Ex. Purity, BCubed precision and recall metrics, normalized mutual information
- 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

Purity

- Let $\boldsymbol{C} = \{c_1, \dots, c_k\}$ be the output clustering result, $\boldsymbol{\Omega} = \{\omega_1, \dots, \omega_k\}$ be the ground truth clustering result (ground truth class)
  - $purity(C, \Omega) = \frac{1}{N} \sum_k \max_j |c_k \cap \omega_j|$



► **Figure 16.1** Purity as an external evaluation criterion for cluster quality. Majority class and number of members of the majority class for the three clusters are: x, 5 (cluster 1); o, 4 (cluster 2); and ◇, 3 (cluster 3). Purity is $(1/17) \times (5+4+3) \approx 0.71$.

Normalized Mutual Information

- $NMI(\Omega, C) = \frac{I(\Omega,C)}{\sqrt{H(\Omega)H(C)}}$
  - $I(\Omega, C) = \underline{\sum_k \sum_j P(\omega_k \cap c_j) \log \frac{P(\omega_k \cap c_j)}{P(\omega_k)P(c_j)}}$

    $= \underline{\sum_k \sum_j \frac{|\omega_k \cap c_j|}{N} \log \frac{N|\omega_k \cap c_j|}{|\omega_k||c_j|}}$

  - $H(\Omega) = \underline{-\sum_k \frac{|\omega_k|}{N} \log \frac{|\omega_k|}{N}}$

Precision and Recall

- P = TP/(TP+FP)
- R = TP/(TP+FN)
- F-measure: 2P*R/(P+R)

	Same cluster	Different clusters
Same class	TP	FN
Different classes	FP	TN

Matrix Data: Clustering: Part 1

- Cluster Analysis: Basic Concepts
- Partitioning Methods
- Hierarchical Methods
- Density-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
- 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
- Quality of clustering results can be evaluated in various ways
- Clustering evaluation

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.l 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.
- Xiaoxin Yin, Jiawei Han, and Philip Yu, "LinkClus: Efficient Clustering via Heterogeneous Semantic Links", in Proc. 2006 Int. Conf. on Very Large Data Bases (VLDB'06), Seoul, Korea, Sept. 2006.

*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

Clustering Feature Vector in BIRCH

Clustering Feature (CF): ***CF = (N, LS, SS)***

N: **Number of data points**

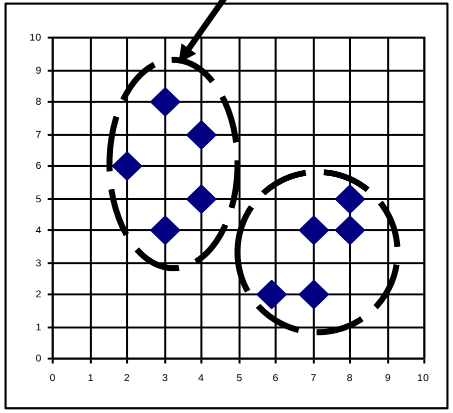
LS: linear sum of N points:

$$\sum_{i=1}^{N} X_i$$

SS: square sum of N points

$$\sum_{i=1}^{N} X_i^{\ 2}$$

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

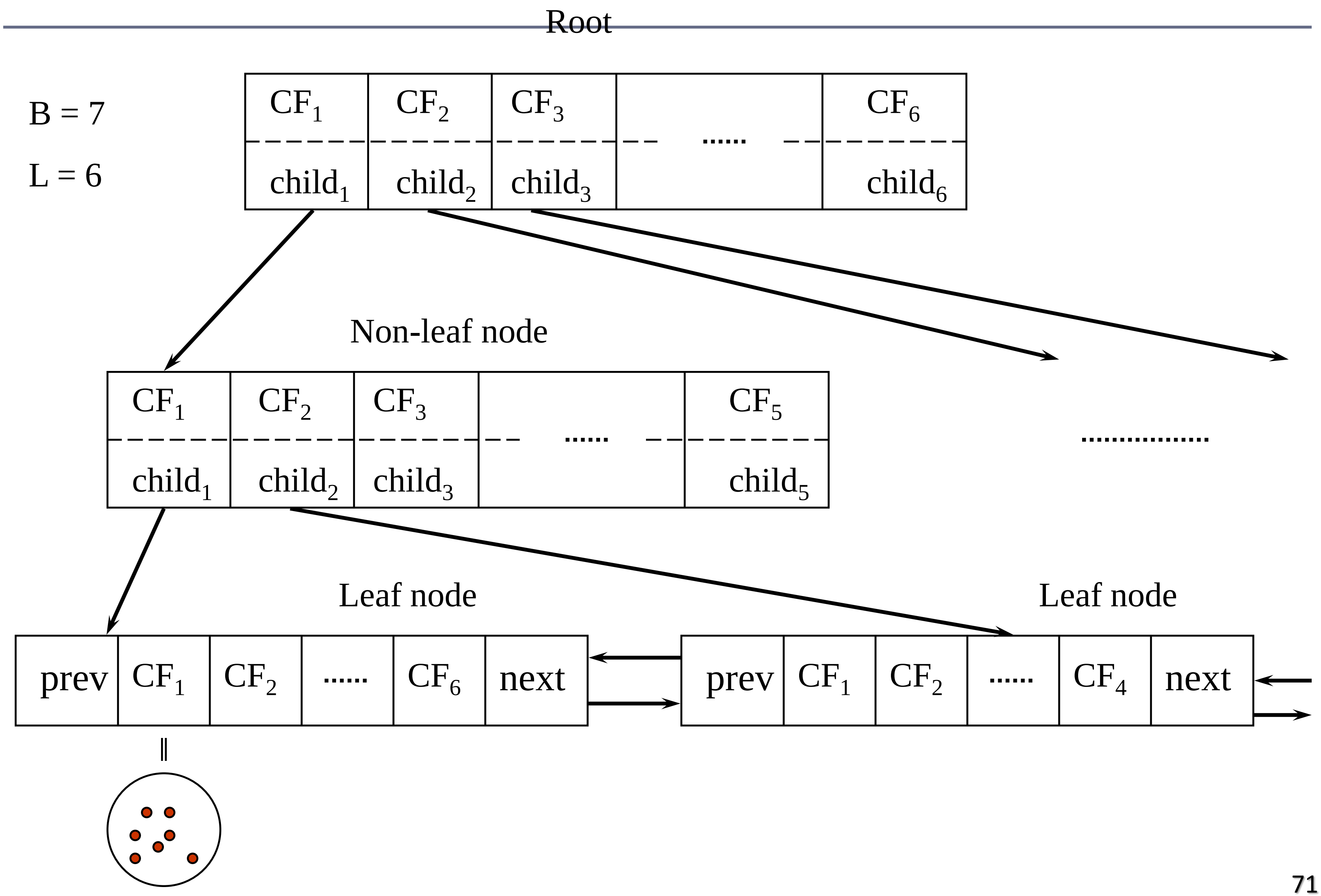


(3,4)
(2,6)
(4,5)
(4,7)
(3,8)

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 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


Root
B = 7
L = 6
CF_1 CF_2 CF_3 ...... CF_6
$child_1$ $child_2$ $child_3$ $child_6$
Non-leaf node
CF_1 CF_2 CF_3 ...... CF_5
$child_1$ $child_2$ $child_3$ $child_5$
....
Leaf node
prev CF_1 CF_2 ...... CF_6 next
Leaf node
prev CF_1 CF_2 ...... CF_4 next

The Birch Algorithm

- Cluster Diameter

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

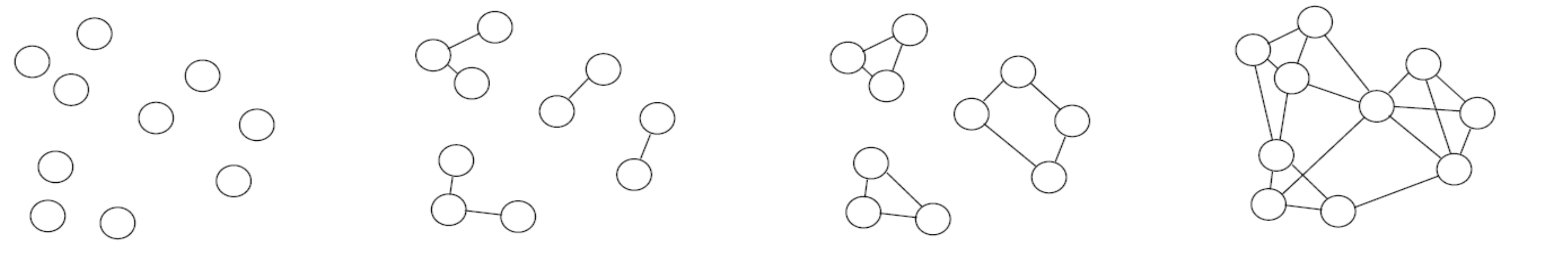
- For each point in the input
  - Find closest leaf entry
  - Add point to leaf entry and update CF
  - If entry diameter > max_diameter, then split leaf, and possibly parents
- Algorithm is O(n)
- Concerns
  - Sensitive to insertion order of data points
  - Since we fix the size of leaf nodes, so clusters may not be so natural
  - Clusters tend to be spherical given the radius and diameter measures

*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 high *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

KNN Graphs & Interconnectivity

- k-nearest graphs from an original data in 2D:

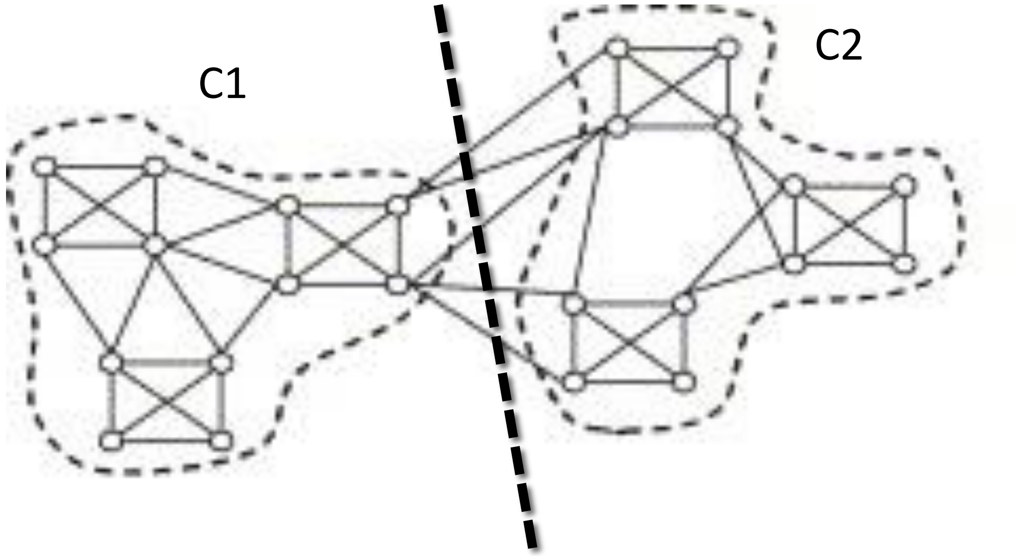


(a) Original Data in 2D (b) 1-nearest neighbor graph (c) 2-nearest neighbor graph (d) 3-nearest neighbor graph

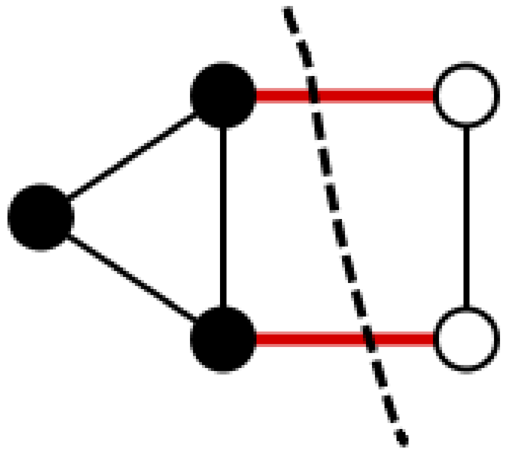
- $EC_{\{C_i, C_j\}}$: The absolute inter-connectivity between C_i *and* C_j*: the sum* of the weight of the edges that connect vertices in C_i *to vertices in* C_j
- Internal inter-connectivity of a cluster C_i : *the size of its min-cut bisector* EC_{C_i} *(i.e., the weighted sum* of edges that partition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

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

Edge Cut for Graphs



The edge cut between two clusters C1 and C2: Size =5



Min-cut that partitions the graph into roughly equal parts: size = 2

Relative Closeness & Merge of Sub-Clusters

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

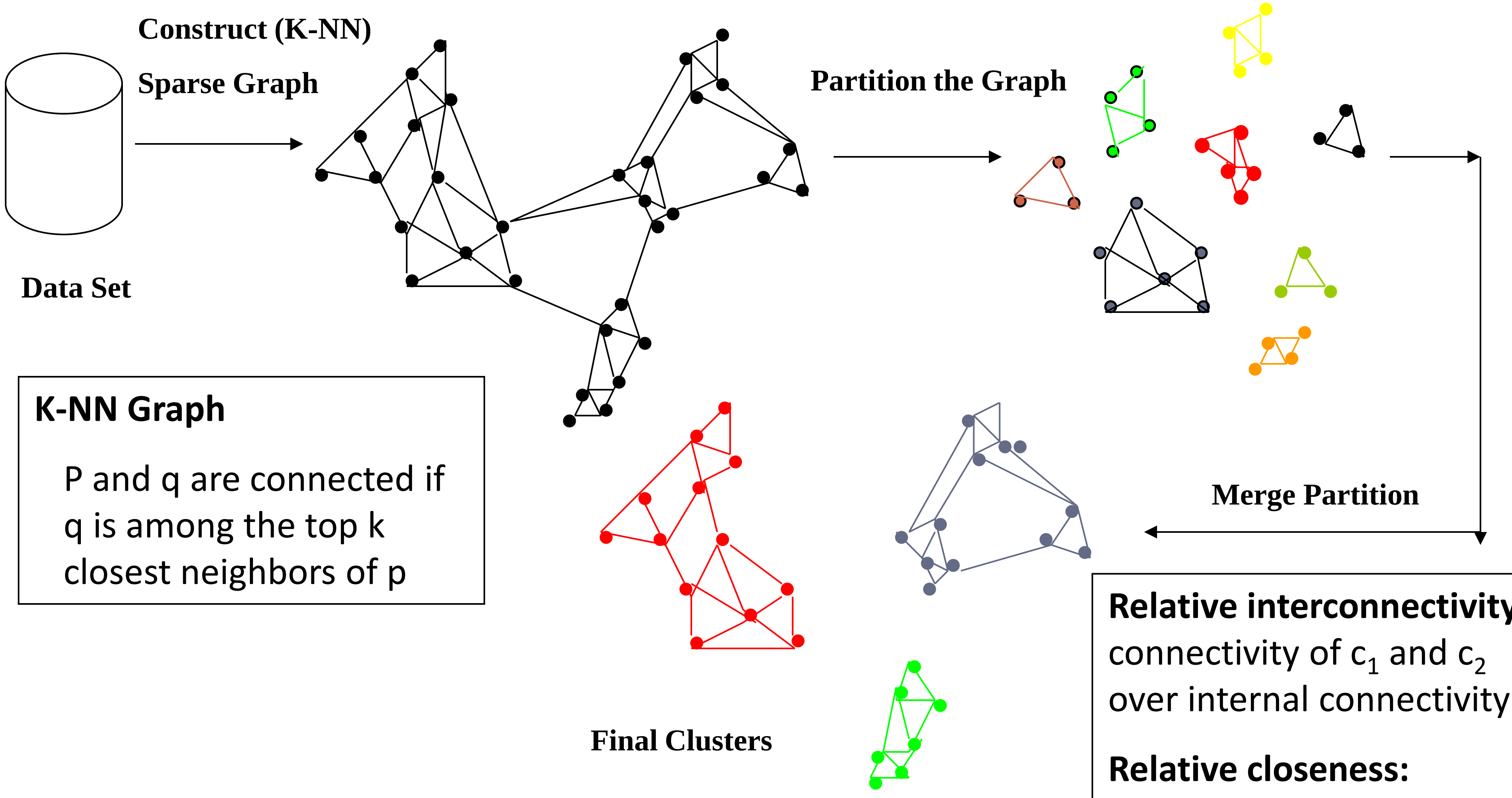
$$RC(C_i, C_j) = \frac{\overline{S}_{EC_{\{C_i, C_j\}}}}{\frac{|C_i|}{|C_i|+|C_j|}\overline{S}_{EC_{C_i}} + \frac{|C_j|}{|C_i|+|C_j|}\overline{S}_{EC_{C_j}}}$$

  - $\overline{S}_{EC_{C_i}}$ and $\overline{S}_{EC_{C_j}}$ are the average weights of the edges that belong in the min-cut bisector of clusters C_i *and* C_j, respectively, and $\overline{S}_{EC_{\{C_i, C_j\}}}$ is the average weight of the edges that connect vertices in C_i *to vertices in* C_j

Weight of edge is determined by KNN calculation

- **Merge Sub-Clusters:**
  - Merges only those pairs of clusters whose RI and RC are both above some user-specified thresholds
  - Merge those maximizing the function that combines RI and RC

Overall Framework of CHAMELEON



K-NN Graph

P and q are connected if q is among the top k closest neighbors of p

Relative interconnectivity: connectivity of c_1 and c_2 over internal connectivity

Relative closeness: closeness of c_1 and c_2 over internal closeness

CHAMELEON (Clustering Complex Objects)