Cluster Analysis

COSC 757 Data Mining, Spring 2016

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 gain insight into data distribution
  - As a preprocessing step for other algorithms

Clustering 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 atmospheric and oceanic paterns
- Economic Science: market resarch

Clustering as a Preprocessing Tool

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

Quality: What is Good Clustering?

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

Cluster Quality

- Dissimilarity/Similarity metric
  - Expressed in terms of a distance function, typically metric: $d(i, j)$
  - The definitions of distance functions are usually rather different for interval-scaled, boolean, categorical, ordinal ratio, and vector variables
  - Weights should be associated with different variables based on applications and data semantics
- Quality of clustering:
  - There is usually a separate "quality" function that measures the "goodness" of a cluster.
  - It is hard to define "similar enough" or "good enough"
    - The answer is typically highly subjective

Considerations for Cluster Analysis

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

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

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, CAMELEON
- Density-based approach:
  - Based on connectivity and density functions
  - Typical methods: DBSCAN, OPTICS, DenClue
- Grid-based approach:
  - based on a multiple-level granularity structure
  - Typical methods: STING, WaveCluster, CLIQUE

Major Clustering Approaches (2)

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

Partitioning Algorithms: Basic Concept

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

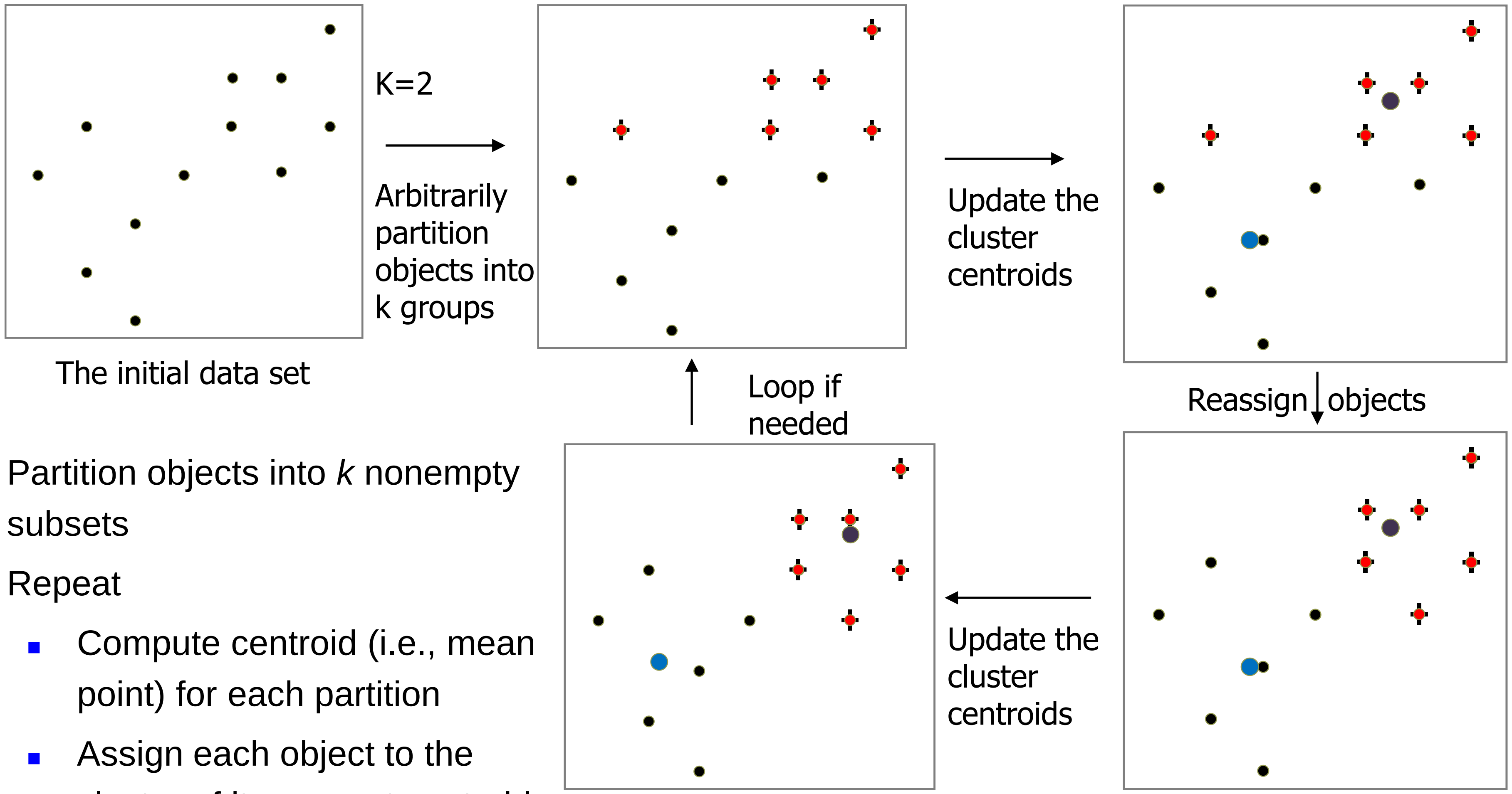
$$E = \Sigma_{i=1}^{k} \Sigma_{p \in C_i} (p - c_i)^2$$

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

K-means Example


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, $t << n$.
    - Comparing: PAM: $O(k(n-k)^2)$, CLARA: $O(ks^2 + k(n-k))$
- Comment: Often terminates at a *local optimal*. (Greedy)
- 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*

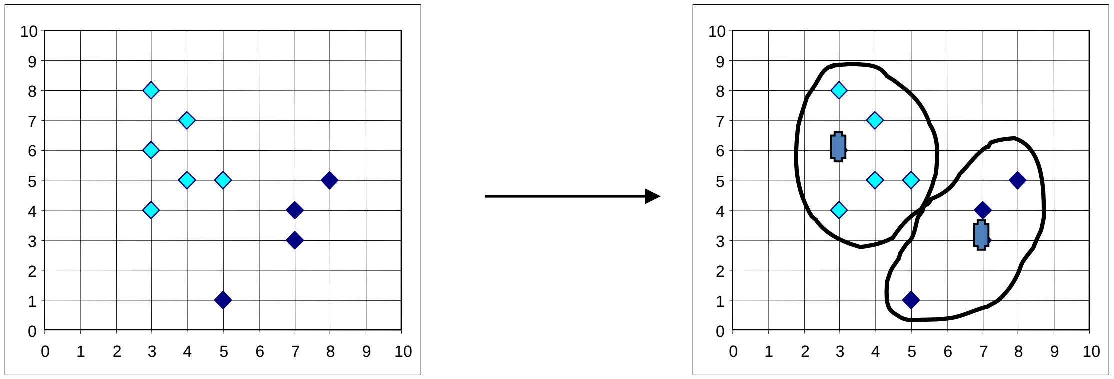
R Demo

Variations of the *K*-means Method

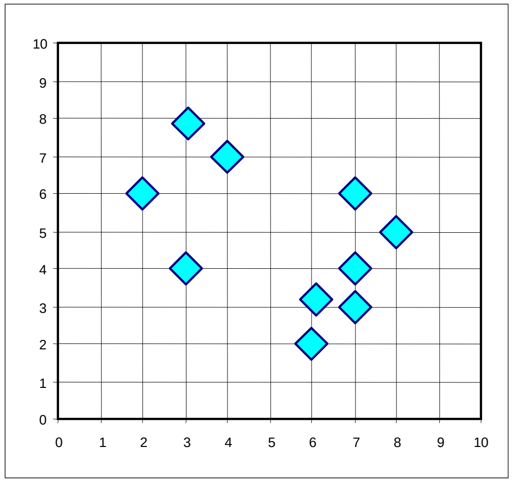
- Most of the variants of the *k-means* which differ in
  - Selection of the initial *k* means
  - Dissimilarity calculations
  - Strategies to calculate cluster means
- Handling categorical data: *k-modes*
  - Replacing means of clusters with <u>modes</u>
  - Using new dissimilarity measures to deal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

K-Medoids

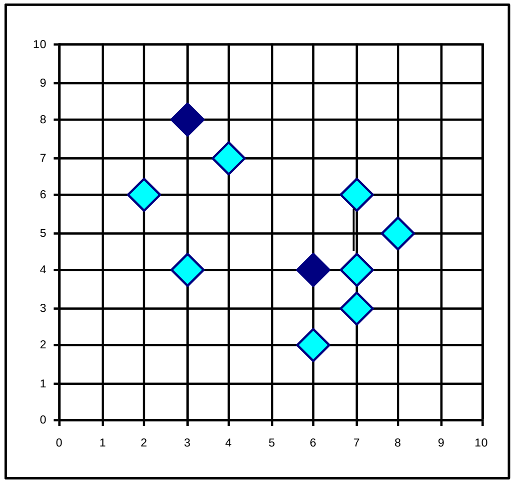
- The k-means algorithm is sensitive to outliers
  - Since an object with an extremely large value may substantially distort the distribution of the data
- K-Medoids: Instead of taking the **mean** value of the object in a cluster as a reference point, **medoids** can be used, which is the **most centrally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



PAM: A Typical K-Medoids Algorithm

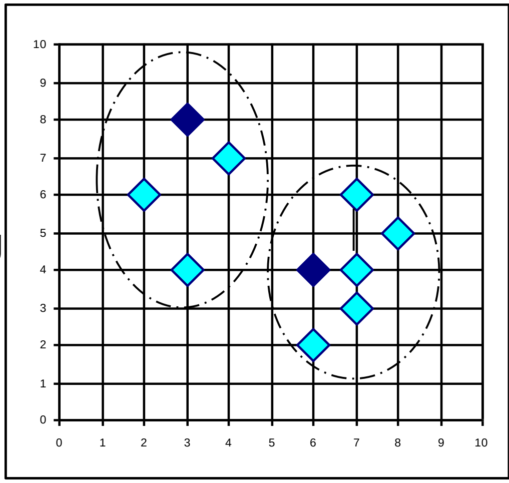


Arbitrarily choose k objects as initial medoids



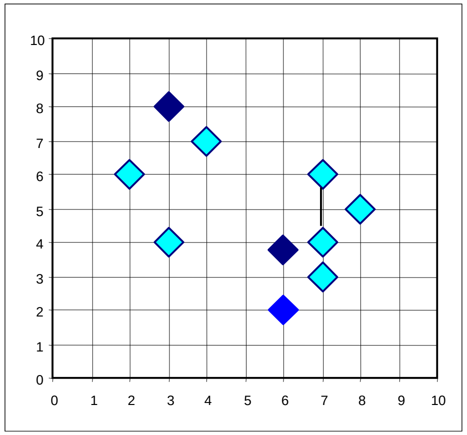
Assign each remaining object to nearest medoids

Total Cost = 20



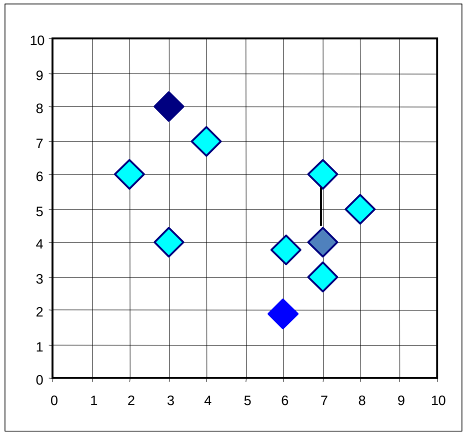
K=2

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

$$\mathrm{cost}(x,c) = \sum_{i=1}^{d} |x_i - c_i|$$

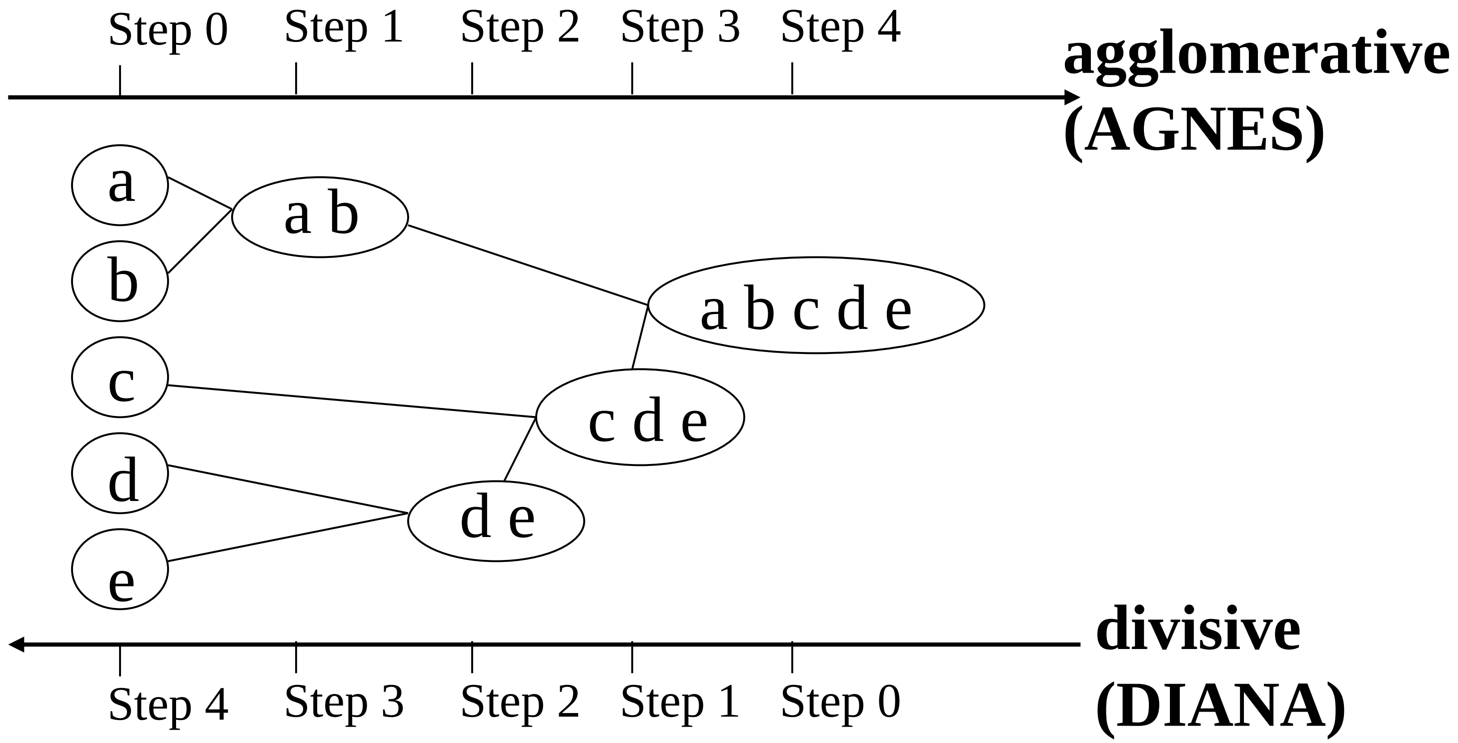
The *K-medoid Clustering Method*

- *K-Medoids* Clustering: Find *representative* objects (<u>medoids</u>) 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

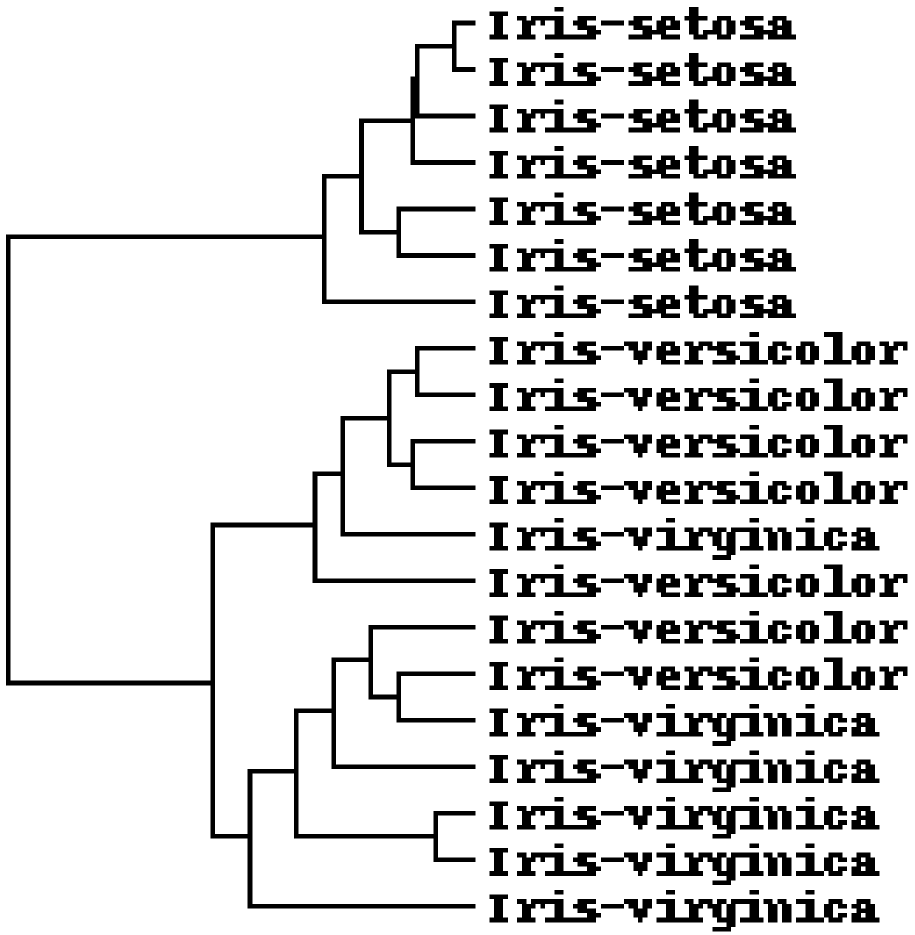
R Demo

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

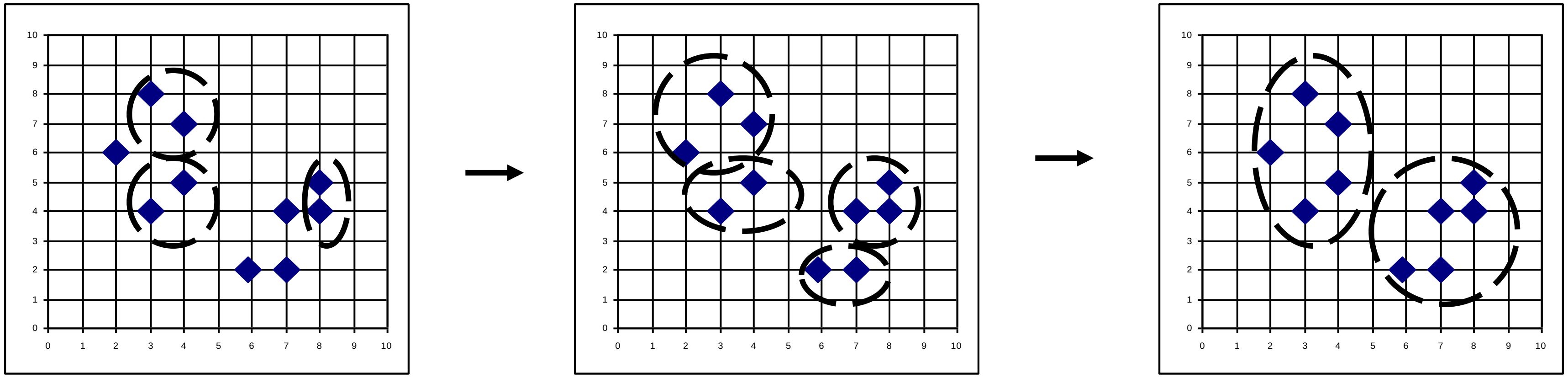


Dendrogram


Iris-setosa
Iris-setosa
Iris-setosa
Iris-setosa
Iris-setosa
Iris-setosa
Iris-setosa
Iris-versicolor
Iris-versicolor
Iris-versicolor
Iris-versicolor
Iris-virginica
Iris-versicolor
Iris-versicolor
Iris-versicolor
Iris-virginica
Iris-virginica
Iris-virginica
Iris-virginica
Iris-virginica

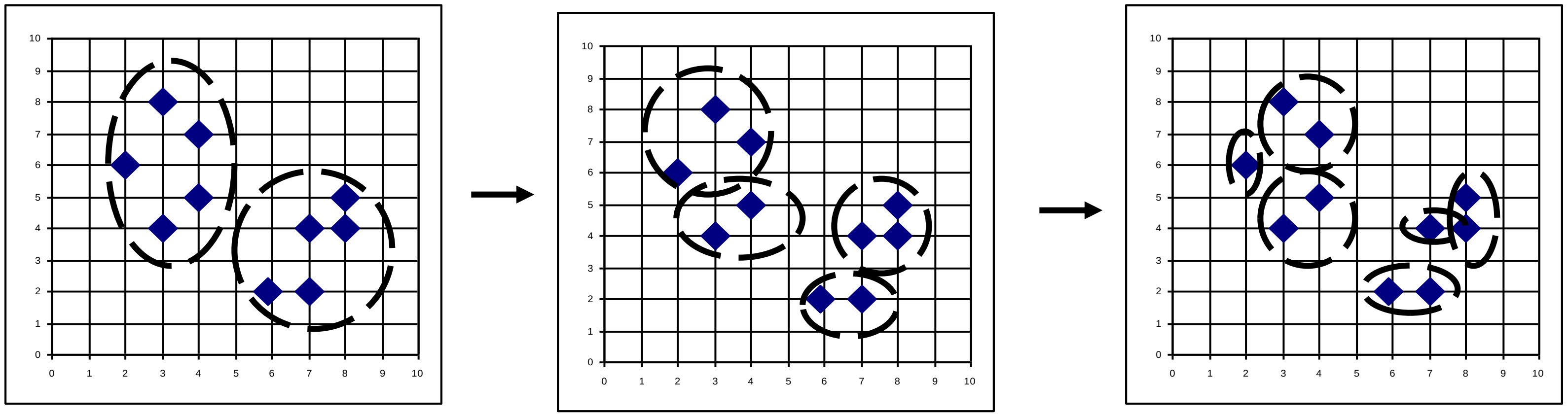
AGNES (Agglomerative Nesting)

- Introduced in Kaufmann and Rousseeuw (1990)
- Use the **single-linkage** method and the dissimilarity matrix
- Merge nodes that have the lowest dissimilarity
- Progress in a non-descending fashion
- Eventually all nodes belong to the same cluster



DIANA (Divisive Analysis)

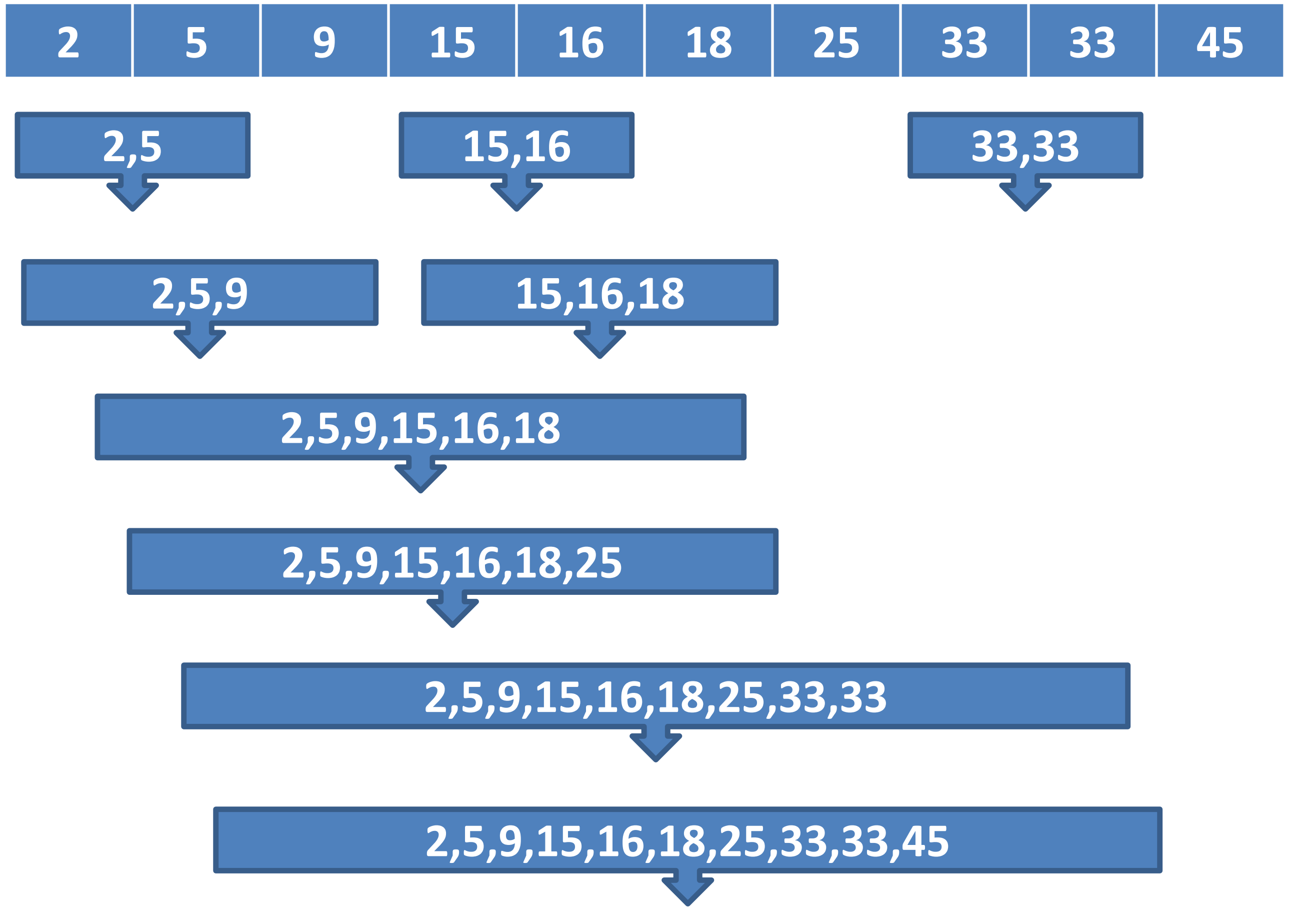
- Introduced in Kaufmann and Rousseeuw (1990)
- Implemented in statistical analysis packages
- Inverse order of AGNES
- Eventually each node forms a cluster on its own



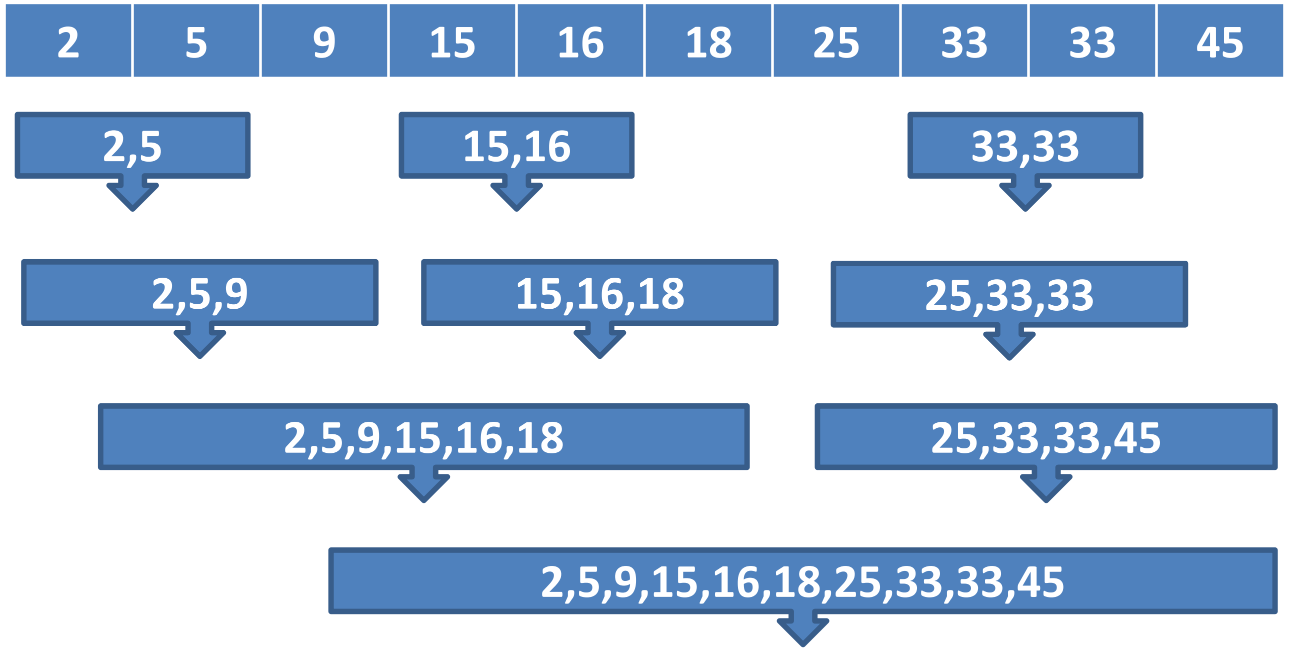
Distance Between Hierarchical Clusters

- A core need is to be able to measure the distance between two hierarchical clusters.
- **Single linkage:** 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 linkage:** 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 linkage:** 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

Single Linkage Example (Agglomerative)



Complete Linkage Example (Agglomerative)



R Demo

Extensions to Hierarchical Clustering

- Major weakness of agglomerative clustering methods
  - Can never undo what was done previously
  - Do not scale well: time complexity of at least $O(n^2)$, where n is the number of total objects
- Integration of hierarchical & distance-based clustering
  - BIRCH (1996): uses CF-tree and incrementally adjusts the quality of sub-clusters
  - CHAMELEON (1999): hierarchical clustering using dynamic modeling

BIRCH (Balanced iterative Reducing and Clustering Using Hierarchies

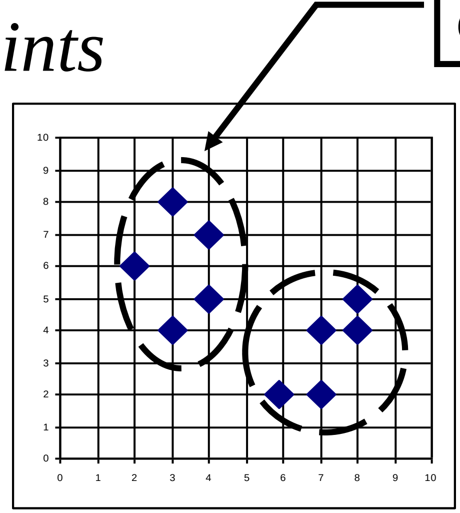
- Zhang, Ramakrishnan & Livny, SIGMOD'96
- Integrates hierarchical clustering and iterative partitioning
- Incrementally construct a CF (Clustering Feature) tree, a hierarchical data structure for multiphase clustering
  - Phase 1: scan DB to build an initial in-memory CF tree (a multi-level compression 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))



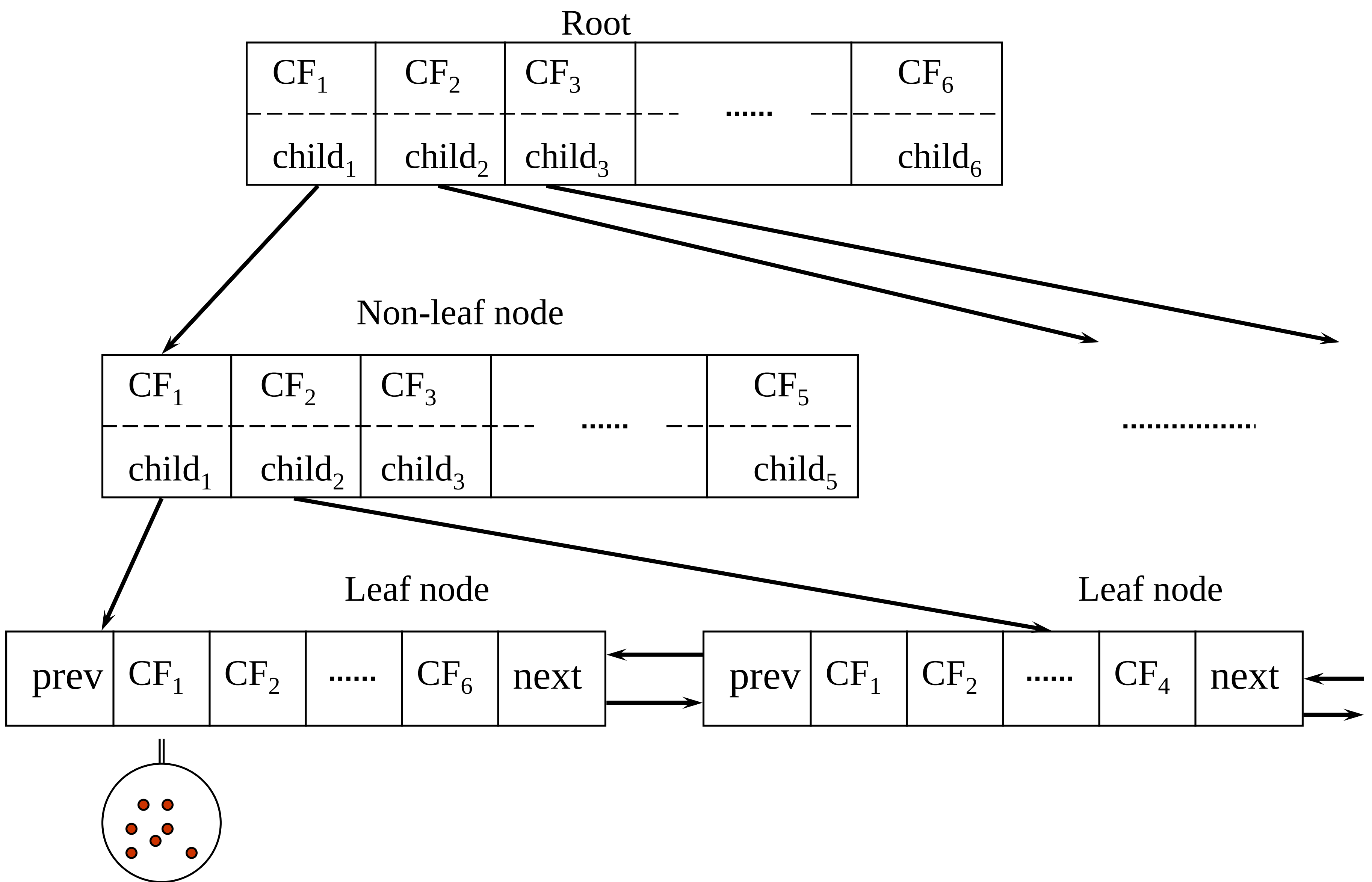
(3,4)
(2,6)
(4,5)
(4,7)
(3,8)

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

$$\sqrt{\frac{2nSS-2LS^2}{n(n-1)}}$$

The CF Tree Structure


Root
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

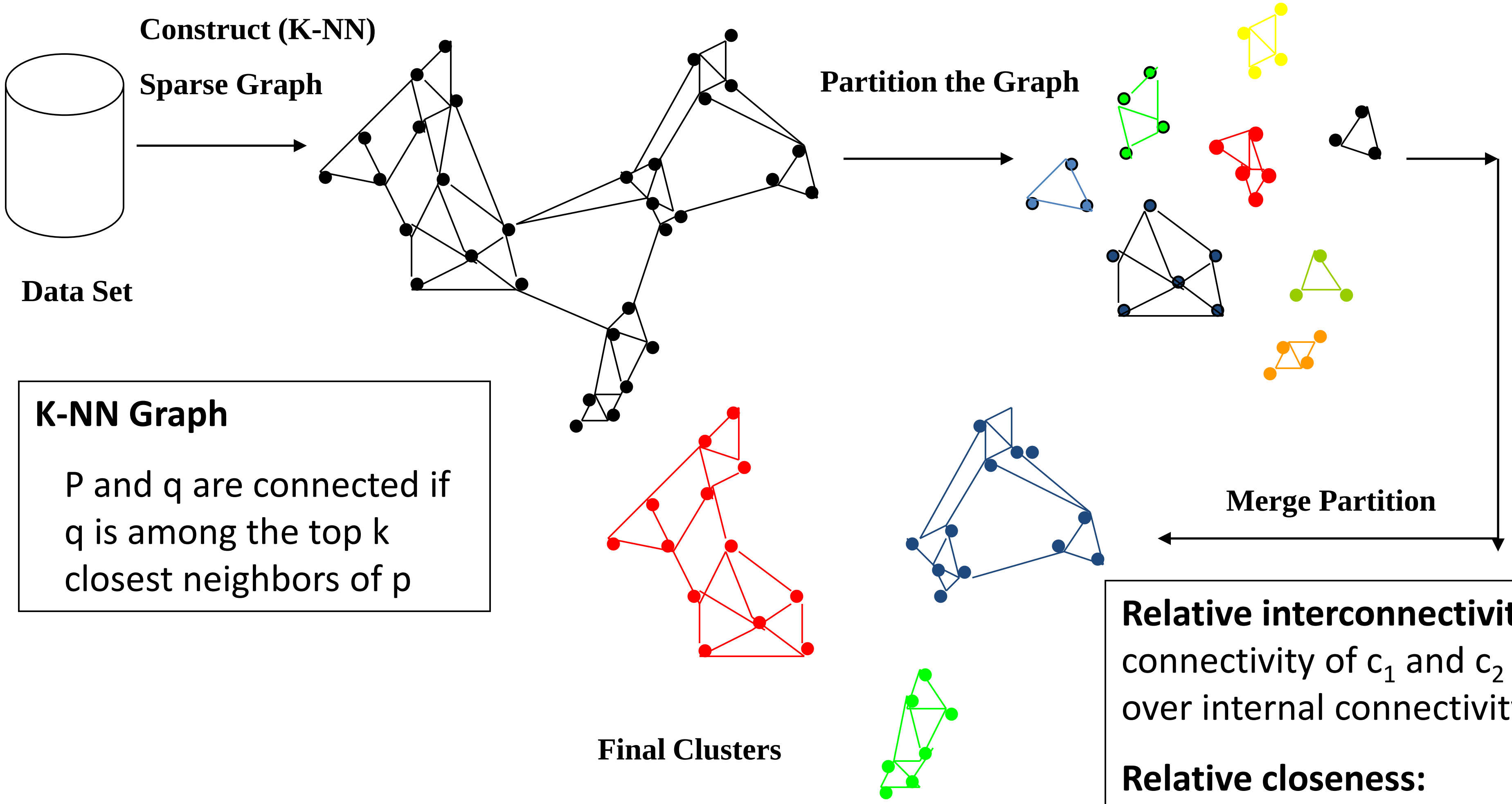
The BIRCH Algorithm

- 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 (not order invariant)
  - 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

Overall Framework of CHAMELEON



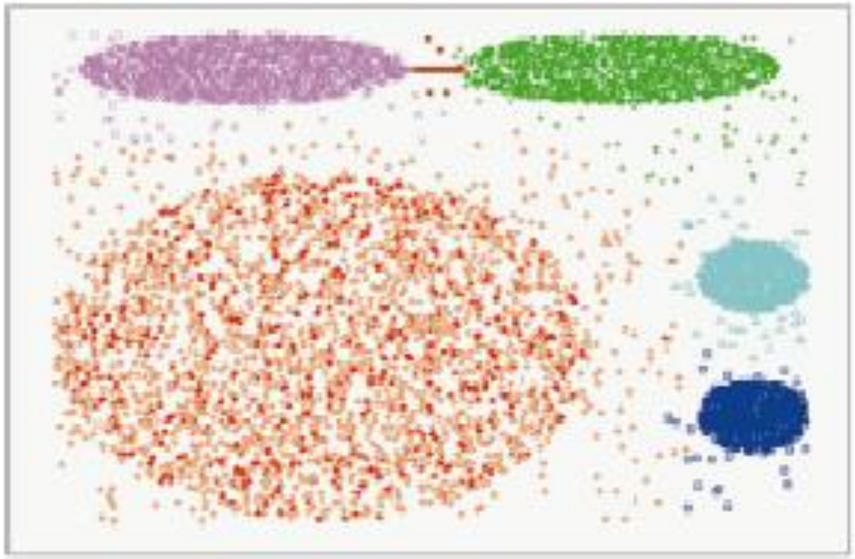
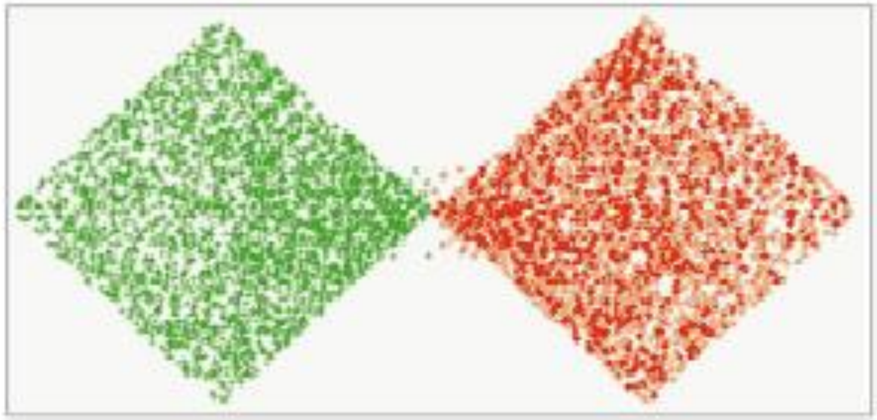
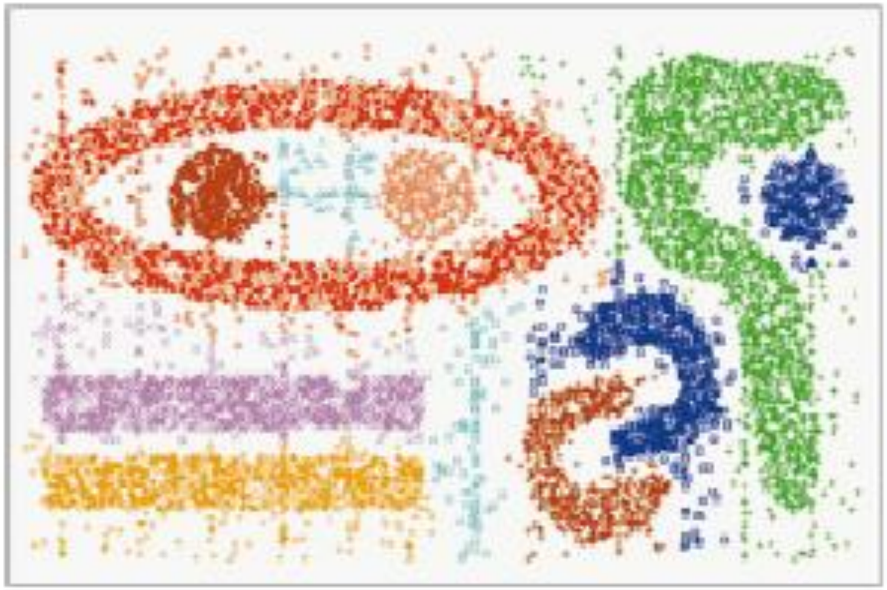
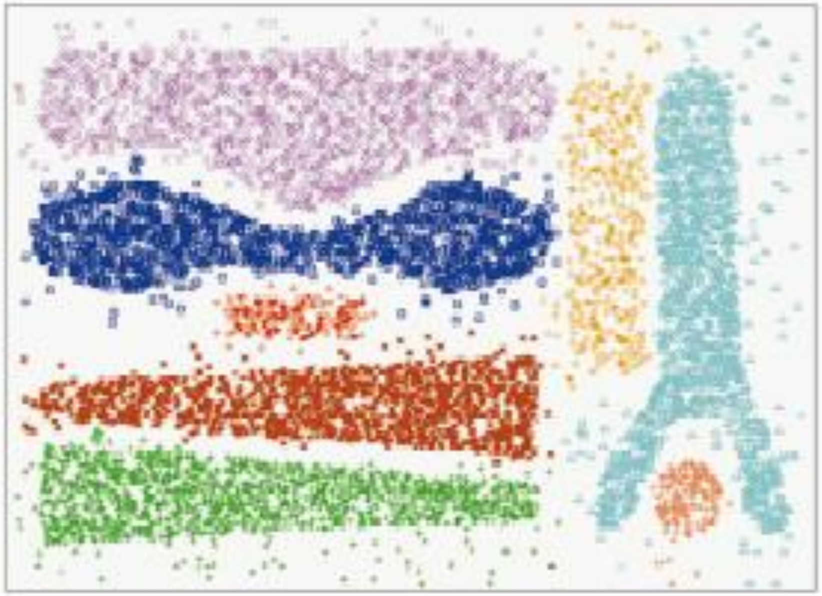
K-NN Graph

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 (Complex Objects)

Probabilistic Hierarchical Clustering

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

Generative Model

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

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

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

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

- The likelihood that X is generated by the model:

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

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

the maximum likelihood

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

A Probabilistic Hierarchical Clustering Algorithm

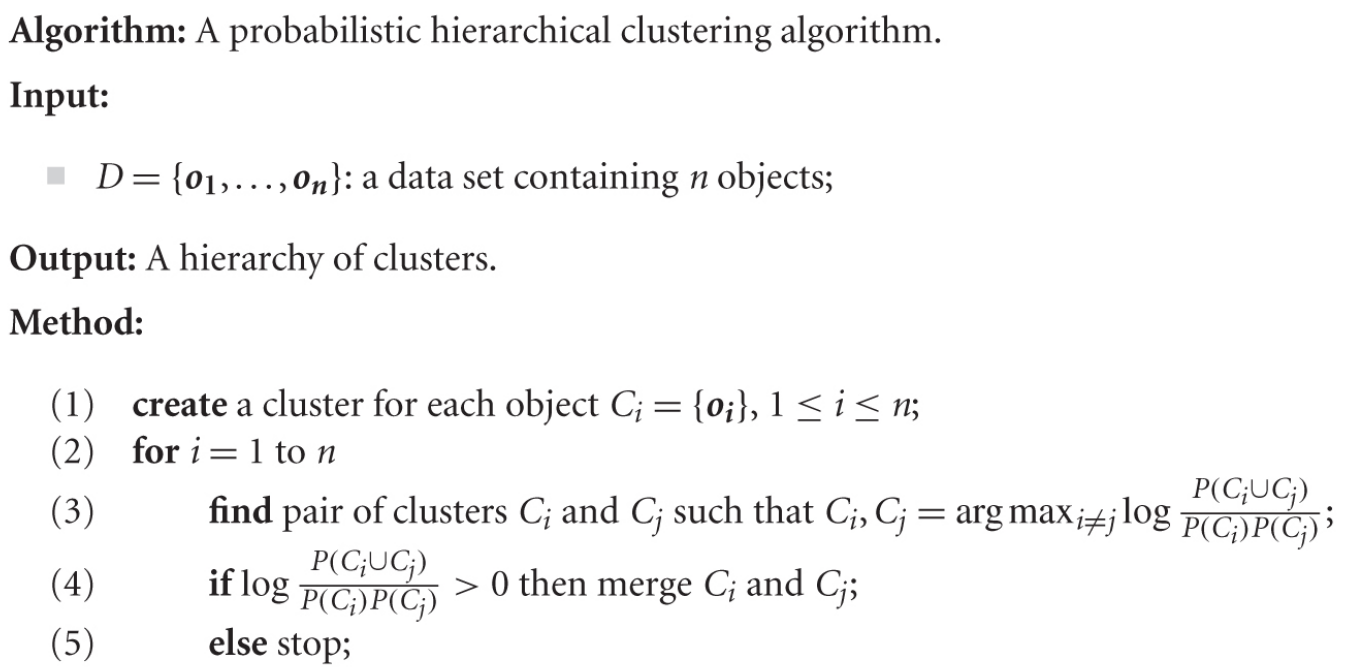
- For a set of objects partitioned into *m* clusters $C_1, \ldots, C_m$, the quality can be measured by,

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

  where *P*() is the maximum likelihood

- Distance between clusters C_1 and C_2:

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



Algorithm: A probabilistic hierarchical clustering algorithm.

Input:

- $D = \{\boldsymbol{o_1}, \ldots, \boldsymbol{o_n}\}$: a data set containing *n* objects;

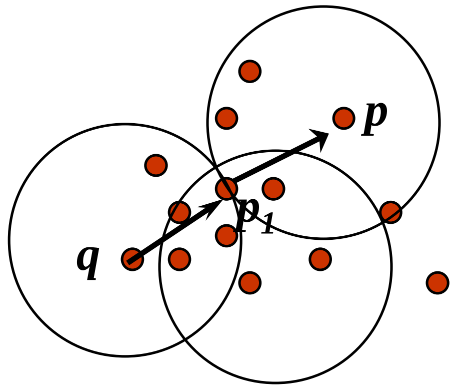
Output: A hierarchy of clusters.

Method:

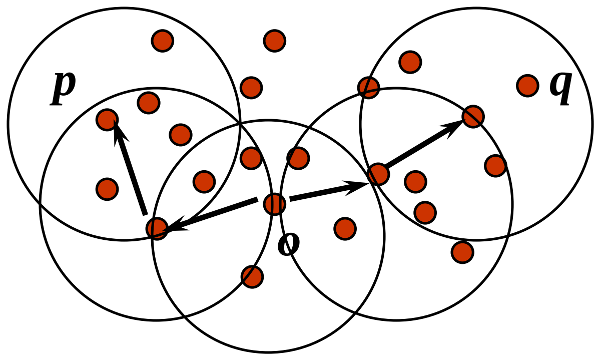
```
(1)  create a cluster for each object $C_i = \{\boldsymbol{o_i}\}$, $1 \le i \le n$;
(2)  for $i = 1$ to $n$
(3)      find pair of clusters $C_i$ and $C_j$ such that $C_i, C_j = \arg\max_{i \neq j} \log \frac{P(C_i \cup C_j)}{P(C_i)P(C_j)}$;
(4)      if $\log \frac{P(C_i \cup C_j)}{P(C_i)P(C_j)} > 0$ then merge $C_i$ and $C_j$;
(5)      else stop;
```

Density-Based Clustering

- 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 such as termination criteria



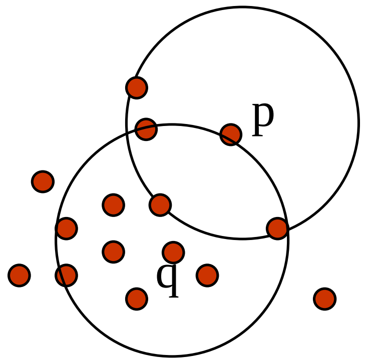
- Several interesting studies:
  - DBSCAN: Ester, et al. (KDD'96)
  - OPTICS: Ankerst, et al (SIGMOD'99).
  - DENCLUE: Hinneburg & D. Keim (KDD'98)
  - CLIQUE: Agrawal, et al. (SIGMOD'98) (more grid-based)



Density-Based Clustering: Basic Concepts

- Two parameters*:*
  - *Eps*: Maximum radius of the neighborhood
  - *MinPts*: Minimum number of points in an Eps-neighbourhood (N_{Eps}) of that point
- 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:

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

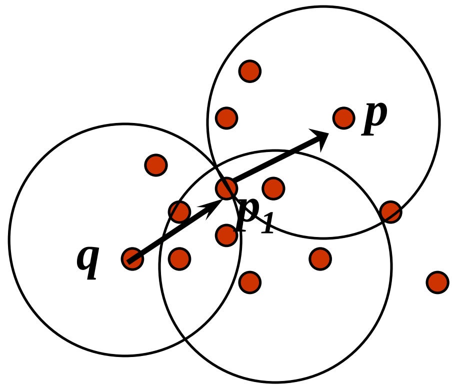


MinPts = 5

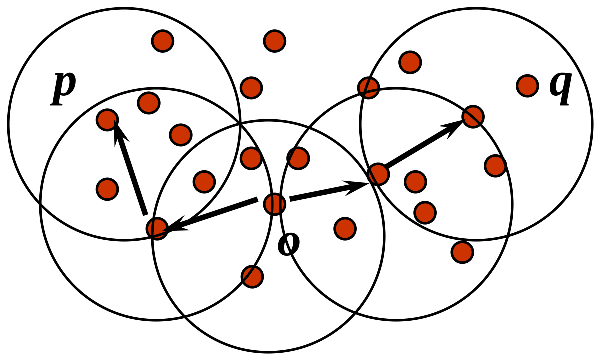
Eps = 1 cm

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

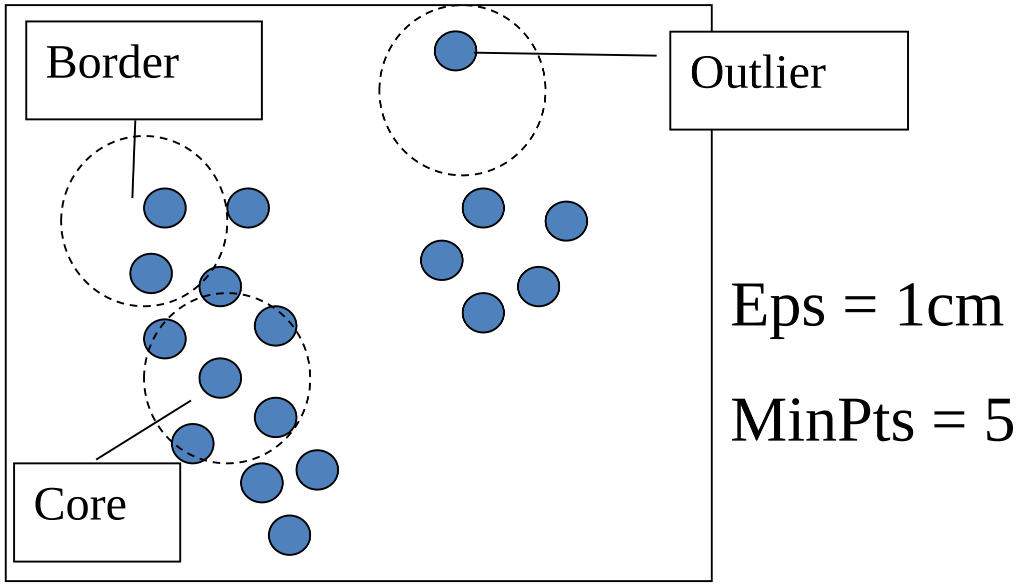


- Density-connected
  - A point *p* is density-connected to a point *q* w.r.t. *Eps*, *MinPts* if there is a point *o* such that both, *p* and *q* are density-reachable from *o* w.r.t. *Eps* and *MinPts*



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
- Discovers clusters of arbitrary shape in spatial databases with noise



DBSCAN: The Algorithm

- Arbitrary select a point *p*
- Retrieve all points density-reachable from *p* w.r.t. *Eps* and *MinPts*
- If *p* is a core point, a cluster is formed
- If *p* is a border point, no points are density-reachable from *p* and DBSCAN visits the next point of the database
- Continue the process until all of the points have been processed

DBSCAN: Sensitive to Parameters

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

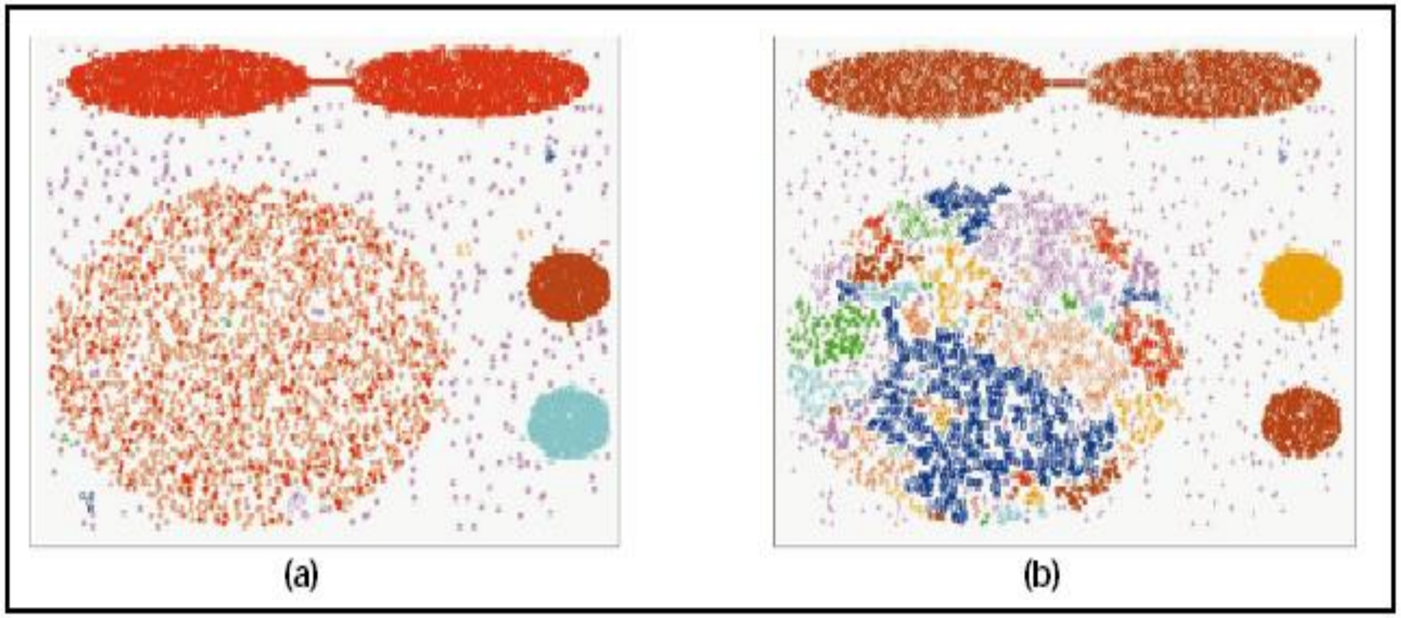
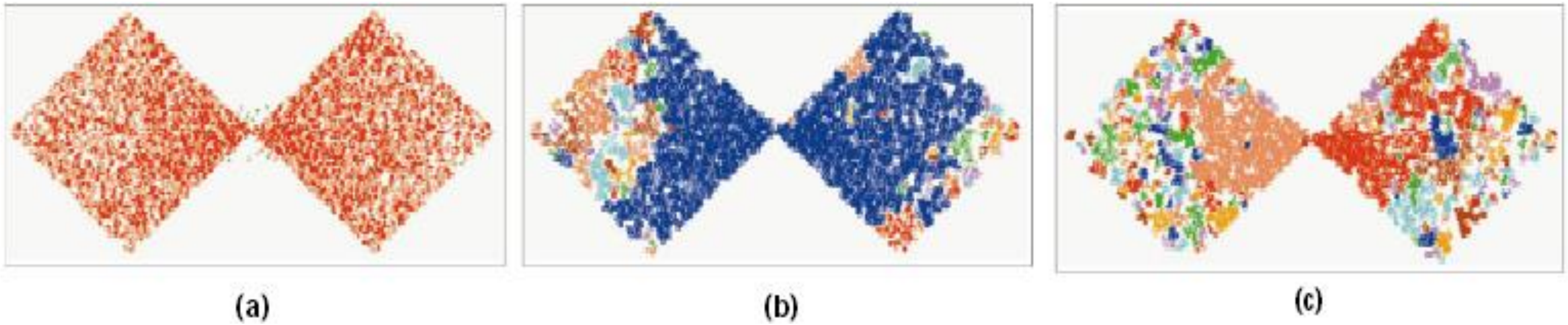


Figure 9. DBScan results for DS2 with MinPts at 4 and Eps at (a) 5.0, (b) 3.5, and (c) 3.0.

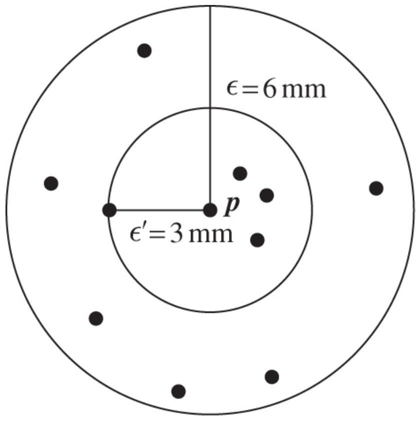


OPTICS: A Cluster-Ordering Method (1999)

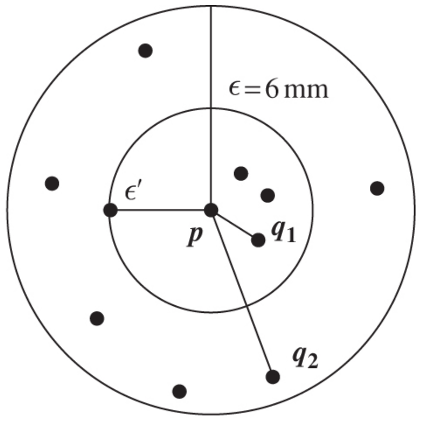
- OPTICS: Ordering Points To Identify the Clustering Structure
  - Ankerst, Breunig, Kriegel, and Sander (SIGMOD'99)
  - Produces a special order of the database with respect to its density-based clustering structure
  - This cluster-ordering contains information equivalent to the density-based clusterings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

OPTICS: Terminology

- Core-Distance
  - The smallest value of $\in$ such that an object *p* is a core object (i.e. its neighborhood is MinPts)
- Reachability-Distance
  - The minimum radius value between two objects *p* and *q* to make *q* density-reachable from *q* (i.e. *q* has to be a core object and *p* has to be in the neighborhood of *q*
- OPTICS orders points by their reachability distance from their respective closest core objects which is essentially by the smallest reachability-distance of each object

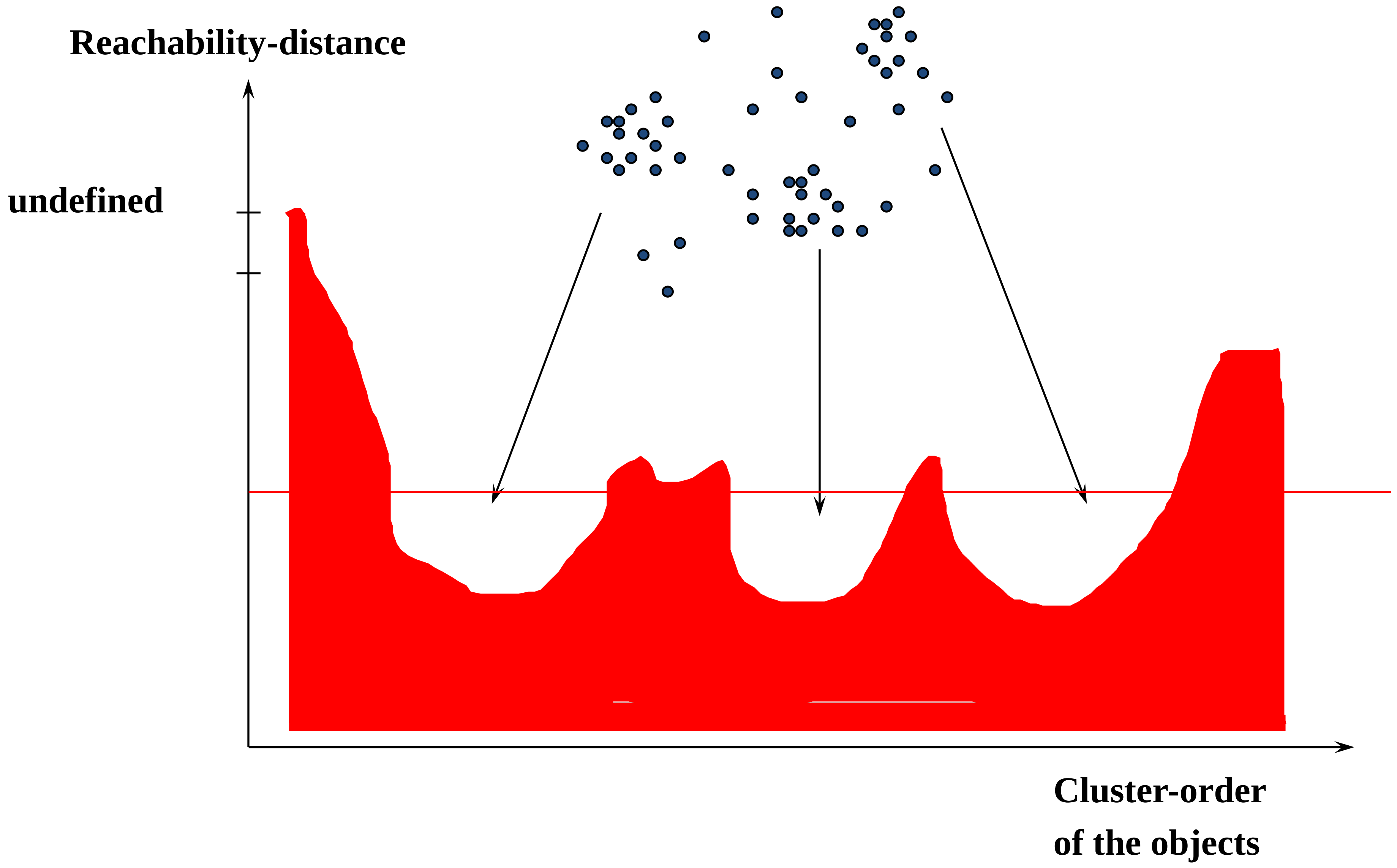


Core-distance of $\boldsymbol{p}$



Reachability-distance $(\boldsymbol{p}, \boldsymbol{q_1}) = \epsilon' = 3$ mm
Reachability-distance $(\boldsymbol{p}, \boldsymbol{q_2}) = dist\ (\boldsymbol{p}, \boldsymbol{q_2})$

Reachability Distance



DENCLUE: Using Statistical Density Functions

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

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

influence of y on x

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

total influence on x

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

gradient of x in the direction of x_i

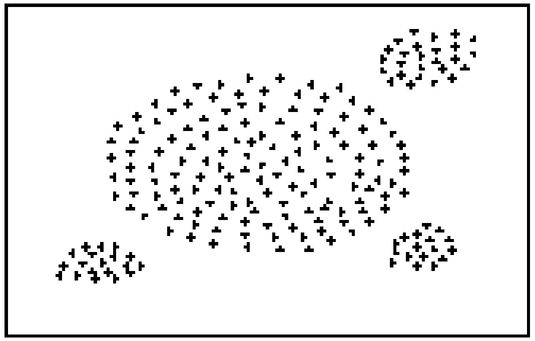
- Major features
  - Solid mathematical foundation
  - Good for data sets with large amounts of noise
  - Allows a compact mathematical description of arbitrarily shaped clusters in high-dimensional data sets
  - Significant faster than existing algorithm (e.g., DBSCAN)
  - But needs a large number of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

Denclue: Technical Essence

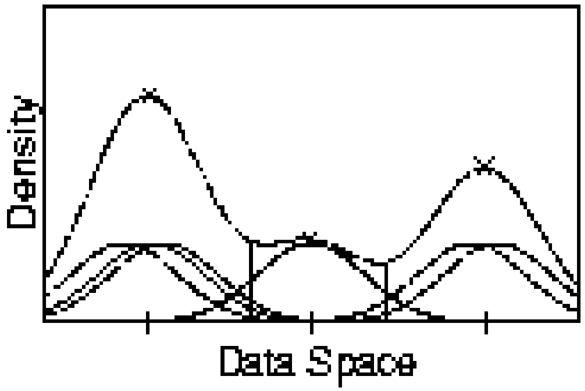
- Influence function: describes the impact of a data point within its neighborhood
- Overall density of the data space can be calculated as the sum of the influence function of all data points
- Clusters can be determined mathematically by identifying density attractors
- Density attractors are local maximal of the overall density function
- Center defined clusters: assign to each density attractor the points density attracted to it
- Arbitrary shaped cluster: merge density attractors that are connected through paths of high density (> threshold)

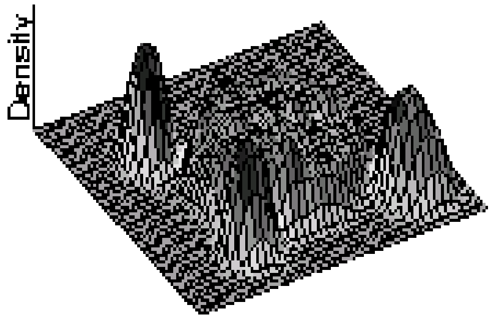
Density Attractor

- A density attractor is a point that is a local maximum of the estimated kernel density function
- Represent the centers of the clusters
- A noise threshold is used to avoid trivial local maximum points



(a) Data Set





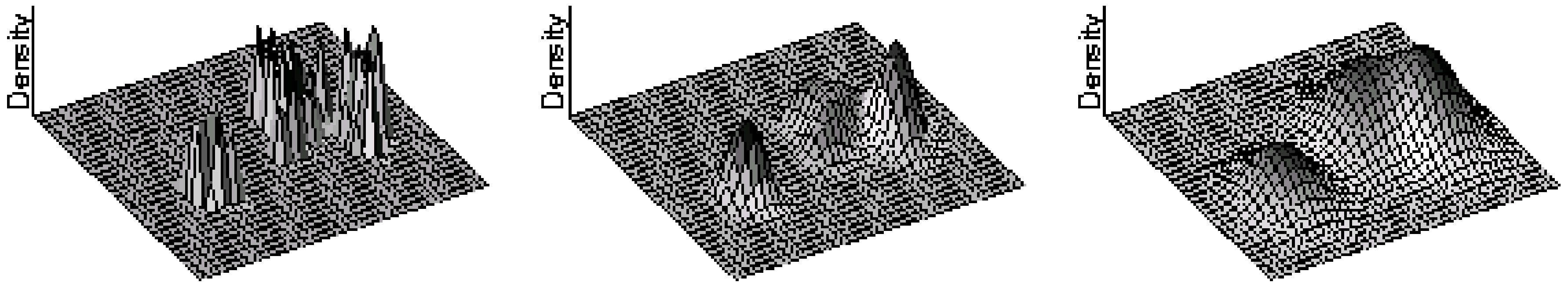
(c) Gaussian

Cluster Assignment

- Uses a hill climbing procedure to assign a data point to a density attractor
- A cluster is a set of density attractors X and a set of input objects C
- Each input object in C is assigned to a density attractor in X and there is a path between every pair of density attractors where the density is above a threshold.

Center-Defined and Arbitrary

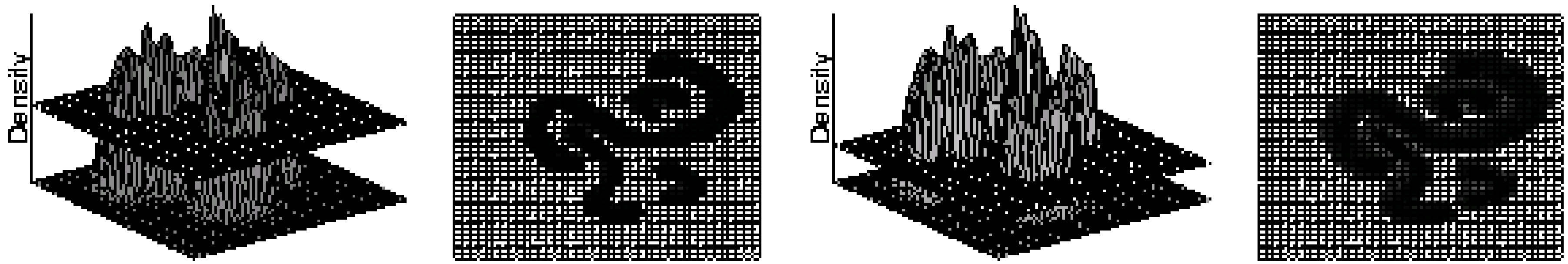
Smoothing parameter (bandwidth)



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

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

Noise Threshold



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

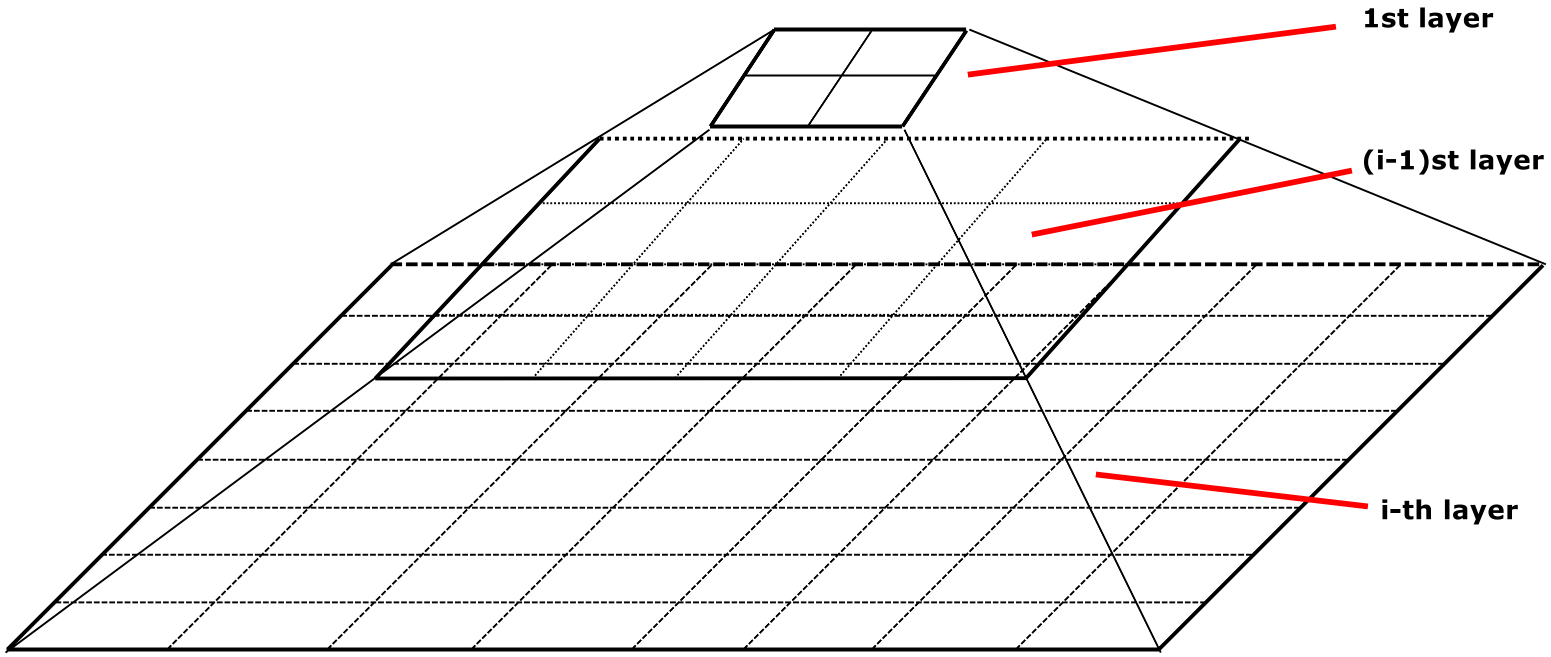
Figure 4: Example of Arbitray-Shape Clusters for different ξ

Grid-Based Clustering Method

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

STING: A Statistical Information Grid Approach

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



The STING Clustering Method

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

STING Algorithm and its Analysis

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

CLIQUE (Clustering in QUEst)

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

CLIQUE: Steps

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

Strength and Weakness of CLIQUE

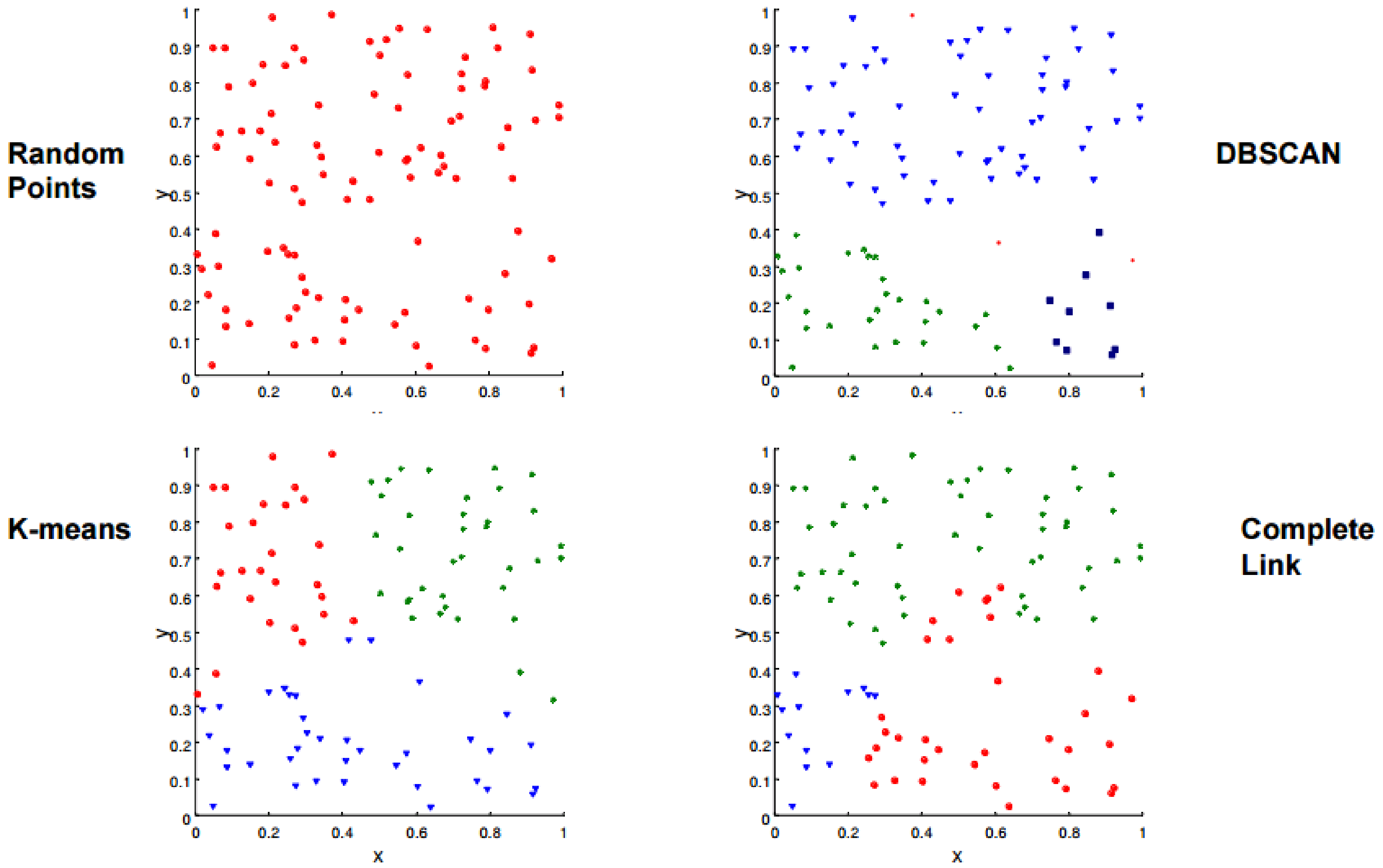
- Strength
  - *automatically* finds subspaces of the highest dimensionality such that high density clusters exist in those subspaces
  - *insensitive* to the order of records in input and does not presume some canonical data distribution
  - scales *linearly* with the size of input and has good scalability as the number of dimensions in the data increases
- Weakness
  - The accuracy of the clustering result may be degraded at the expense of simplicity of the method

Evaluation of Clustering

- Assessing Cluster Tendency
  - Assess whether a non-random structure exists in the data
- Measuring Cluster Validity
  - How good are the resulting clusters?
- Determining the number of clusters
  - Estimate this number before applying clustering

Assessing Clustering Tendency

- What happens when we cluster a completely random



Hopkins Statistic

- Given a dataset D regarded as a sample of a random variable o, determine how far away o is from being uniformly distributed in the data space
- Sample *n* points, $p_1, \ldots, p_n$, uniformly from D. For each p_i, find its nearest neighbor in D: $x_i = \min\{dist(p_i, v)\}$ where *v* in D
- Sample *n* points, $q_1, \ldots, q_n$, uniformly from D. For each q_i, find its nearest neighbor in D – $\{q_i\}$: $y_i = \min\{dist(q_i, v)\}$ where *v* in D and $v \neq q_i$
- Calculate the Hopkins Statistic: $H = \frac{\sum_{i=1}^{n} y_i}{\sum_{i=1}^{n} x_i + \sum_{i=1}^{n} y_i}$
- If D is uniformly distributed, $\sum x_i$ and $\sum y_i$ will be close to each other and H is close to 0.5.
- H > 0.75 indicates a clustering tendency at the 90% confidence level.

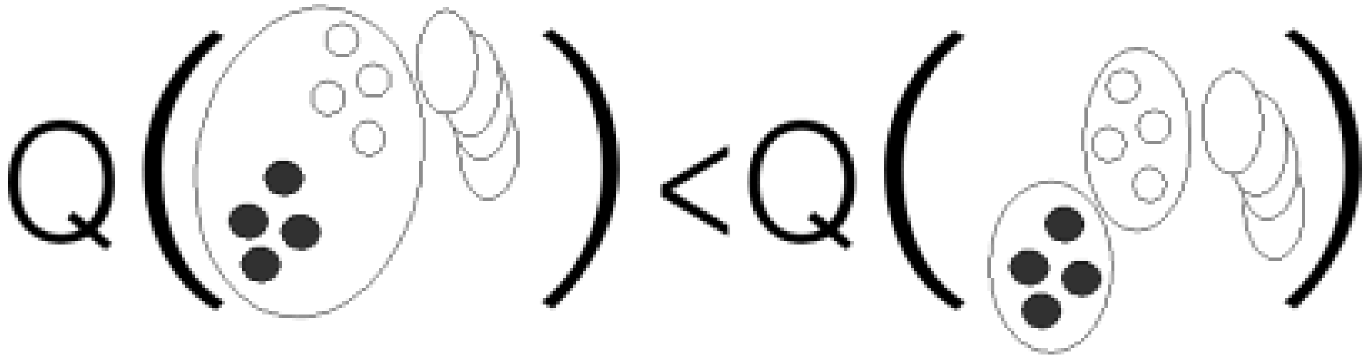
Measuring Cluster Quality

- Two methods: extrinsic vs. intrinsic
- Extrinsic (external): supervised (i.e. ground truth data is available)
  - Compare a clustering against the ground truth using certain clustering quality measure
- Intrinsic (internal): unsupervised (i.e. ground truth data is unavailable)
  - Evaluate the goodness of a clustering by considering how well the clusters are separated, and how compact the clusters are

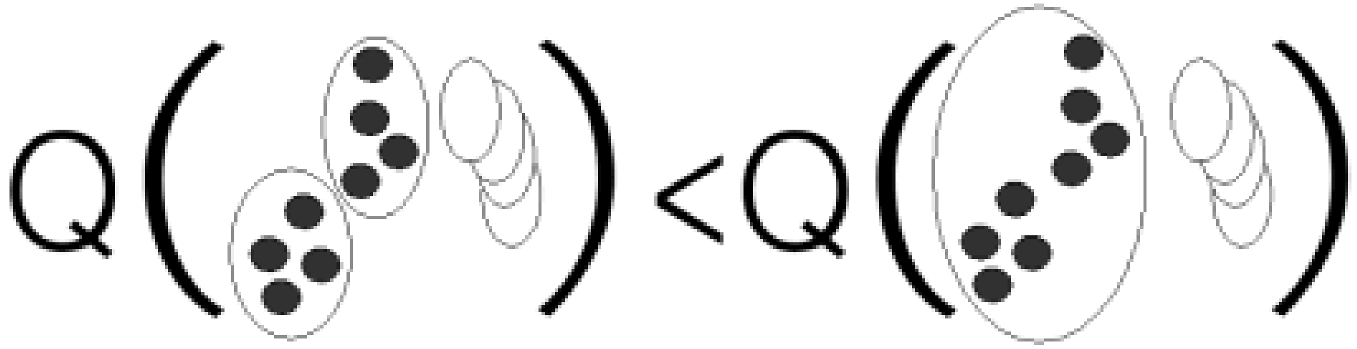
Extrinsic Methods: Criteria

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

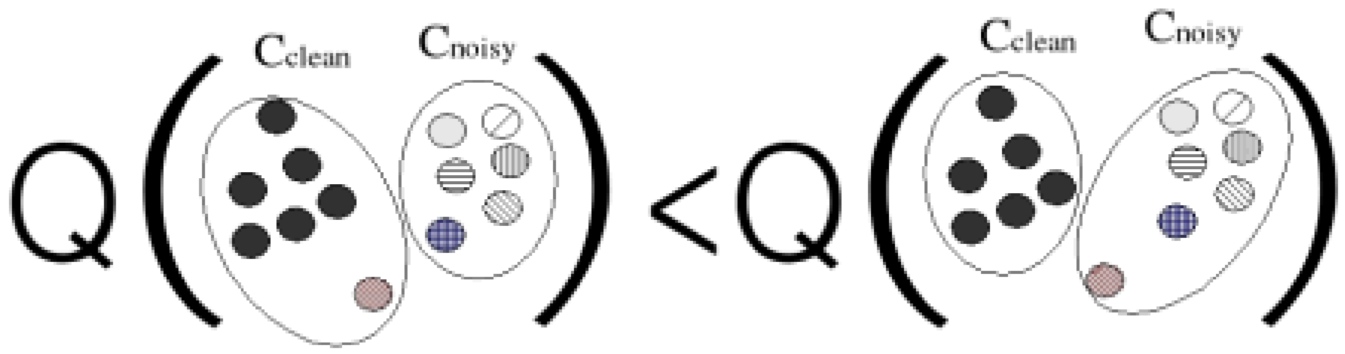
Homogeneity



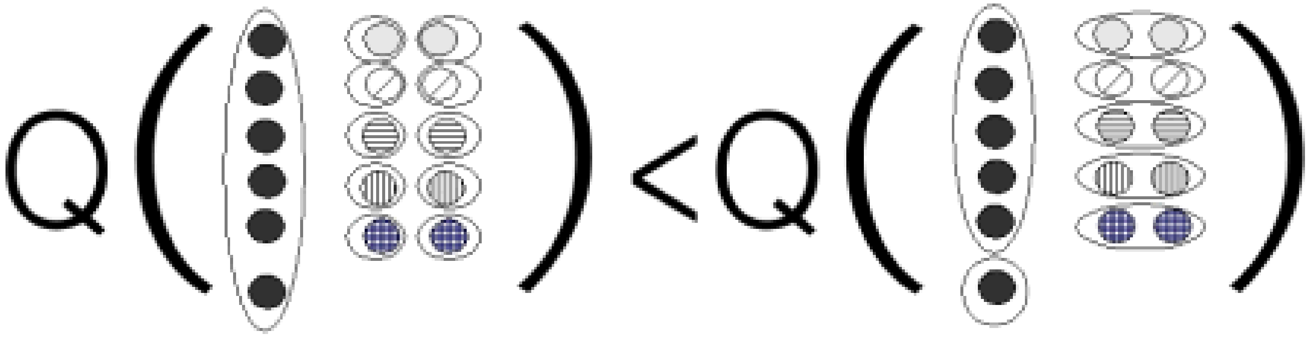
Completeness



Rag Bag



Small Cluster Preservation



BCubed Metric

- Based on precision and recall statistics
- Has been shown to satisfy 4 constraints
  - Homogeneity
    - Splitting a cluster that mixes two categories into two “pure” clusters increases the BCubed precision and does not affect BCubed recall
  - Completeness
    - Unifying two clusters which contain only items from the same category increases the Bcubed recall measure and the precision remains maximal
  - Rag Bag
    - Introducing a unique item in a clean cluster decreases the precision and the recall is unaffected
  - Cluster Size
    - Splitting small clusters decreases recall

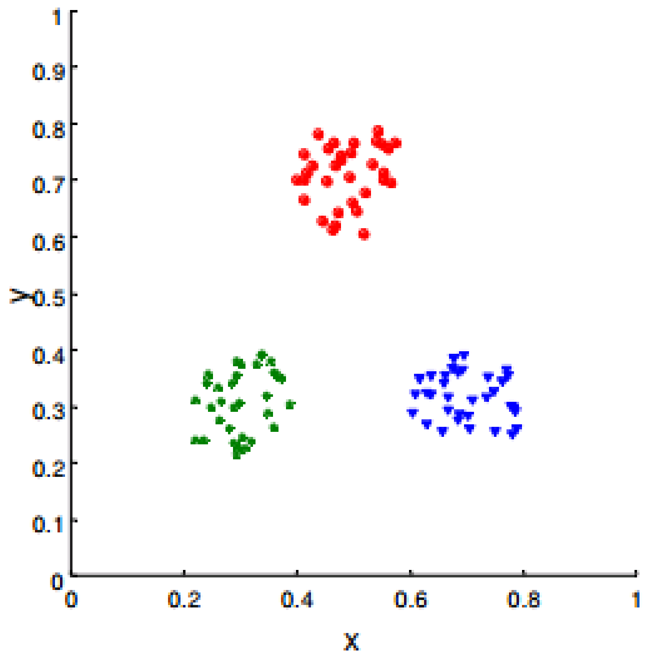
Intrinsic Methods

- Used when the ground truth of a dataset is not available
- Evaluate clusters based on how well they are **separated** and how **compact** they are
- Intrinsic methods typically have the advantage of a similarity metric

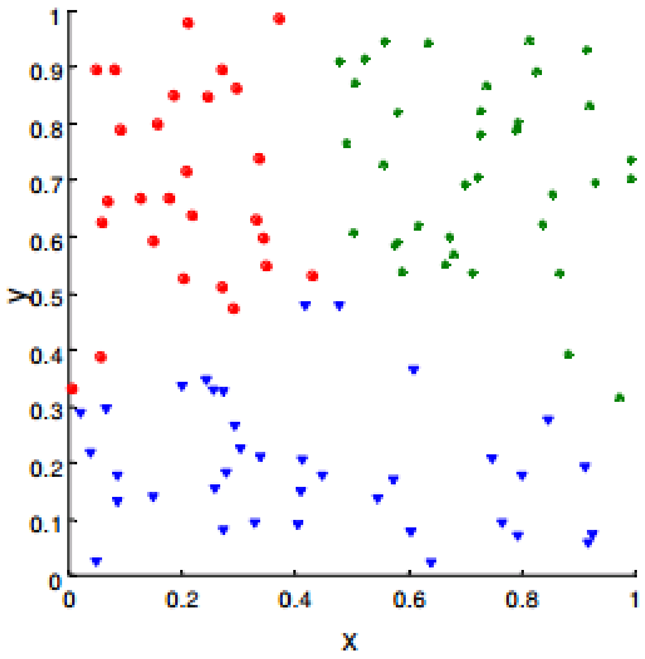
Using Correlation

- Two matrices:
  - Proximity Matrix
  - Incidence Matrix
    - One row and one column for each data point
    - An entry is 1 if the associated pair of points belong to the same cluster
    - An entry is 0 if the associated pair of points belongs to different clusters
- Compute the correlation between the two matrices

Measuring Cluster Validity Via Correlation



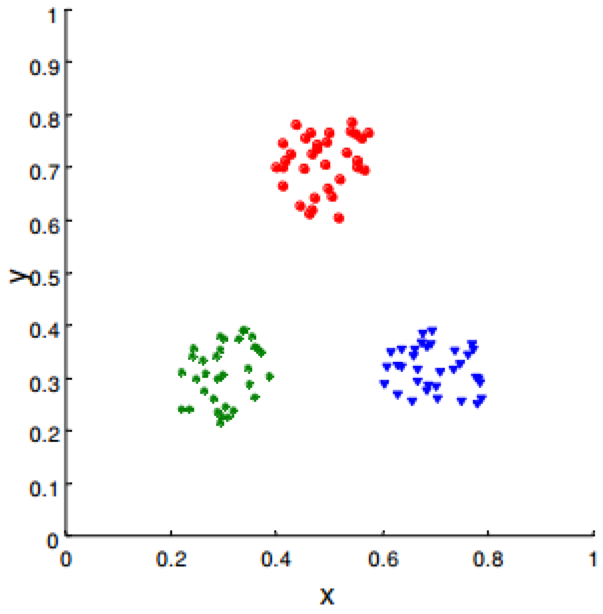
Corr = -0.9235

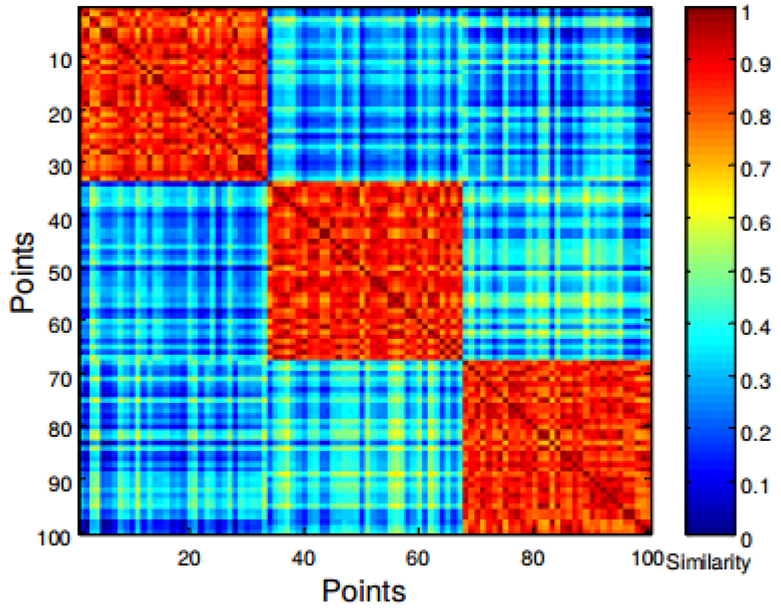


Corr = -0.5810

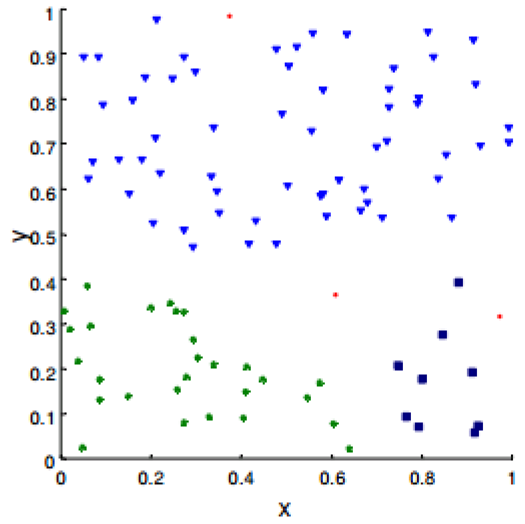
Use Similarity Matrix

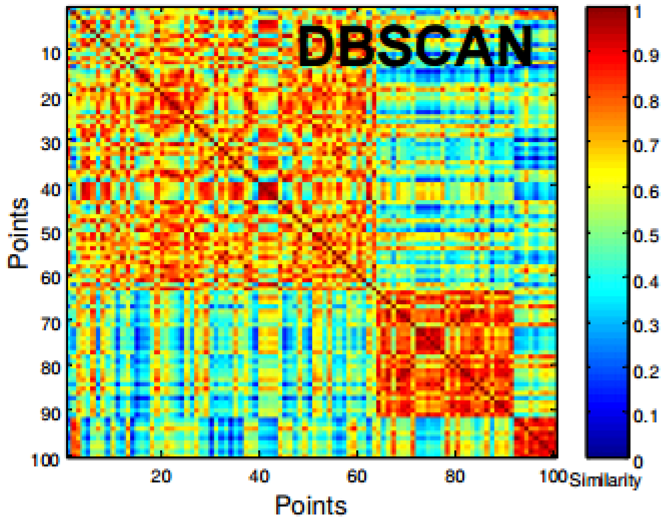
- Order the similarity matrix with respect to cluster labels (visual inspection)

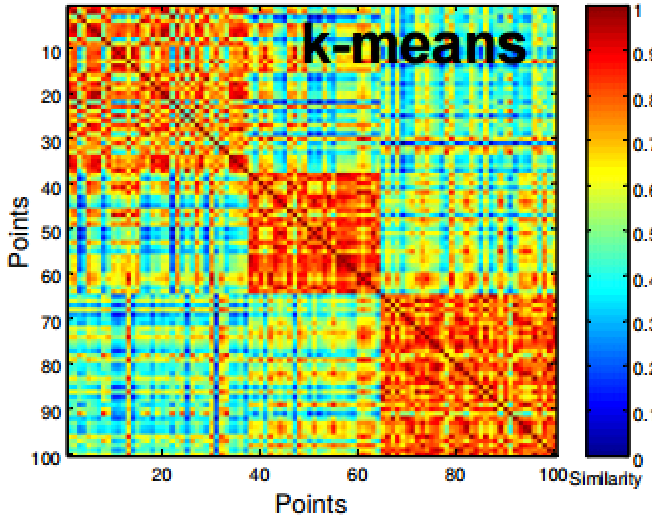


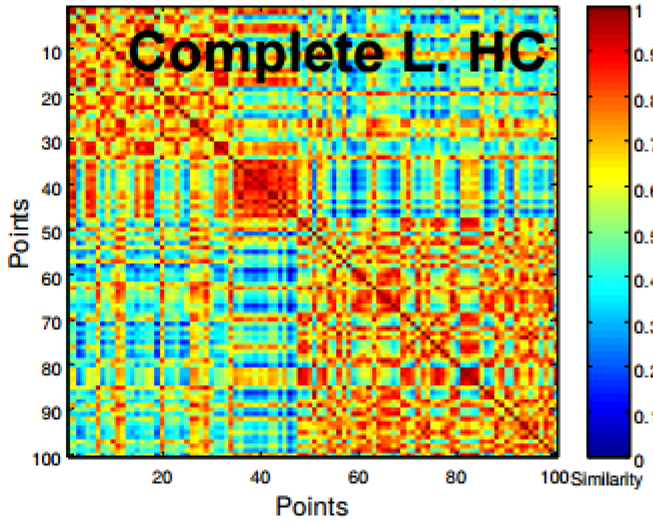


Similarity Matrix for Random Data









Sum of Squares (SSE)

- SSE is good for comparing two clusters (average SSE)

$$SSE = \sum_{i=1}^{K} \sum_{\boldsymbol{x} \in C_i} \|\boldsymbol{x} - \boldsymbol{m_i}\|^2$$

- Can also be used to estimate the number of clusters

Cohesion and Separation

- Cluster Cohesion: Measures how closely related objects are in a cluster:
  - Within cluster sum of squares:

$$WSS = \sum_i \sum_{x \in C_i} \|x - m_i\|^2$$

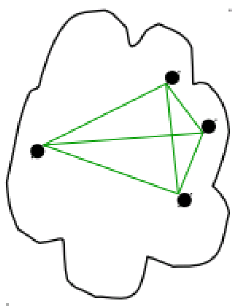
- Cluster Separation: Measure how distinct or well separated a cluster is from other clusters
  - Between cluster sum of squares

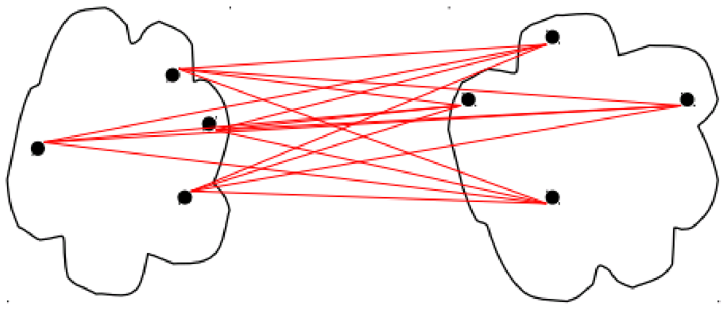
$$BSS = \sum_i |C_i| \|m - m_i\|^2$$

- Total sum of squares *TSS=WSS+BSS*

Proximity Graph for Cohesion and Separation

- A proximity graph-based approach can also be used for cohesion and separation.
  - Cohesion is the sum of the weights of all links within a cluster
  - Separation is the sum of the weights between nodes in the cluster and nodes outside the cluster

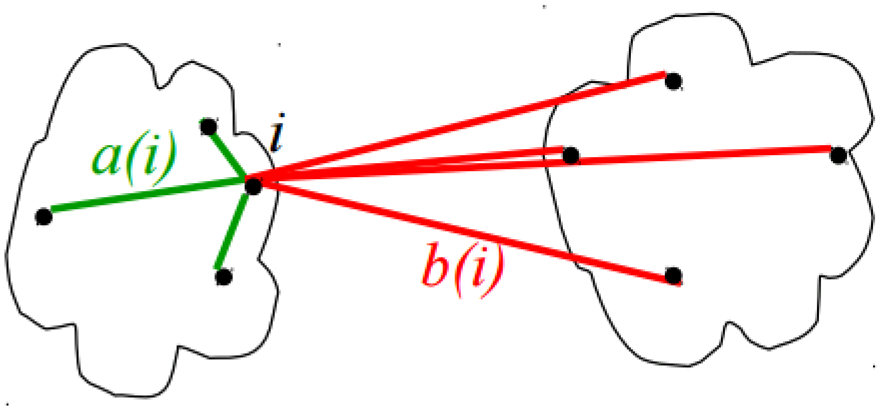

cohesion


separation

Silhouette Coefficient

- Silhouette Coefficient combine ideas of both cohesion and separation, but for individual points. For an individual point *i*:
  - Calculate *a(i)* = average dissimilarity of *i* to all other points in its cluster
  - Calculate *b(i)* = lowest average dissimilarity of *i* to any other)

$$s(i) = \begin{cases} 1 - a(i)/b(i), & \text{if } a(i) < b(i) \\ 0, & \text{if } a(i) = b(i) \\ b(i)/a(i) - 1, & \text{if } a(i) > b(i) \end{cases}$$



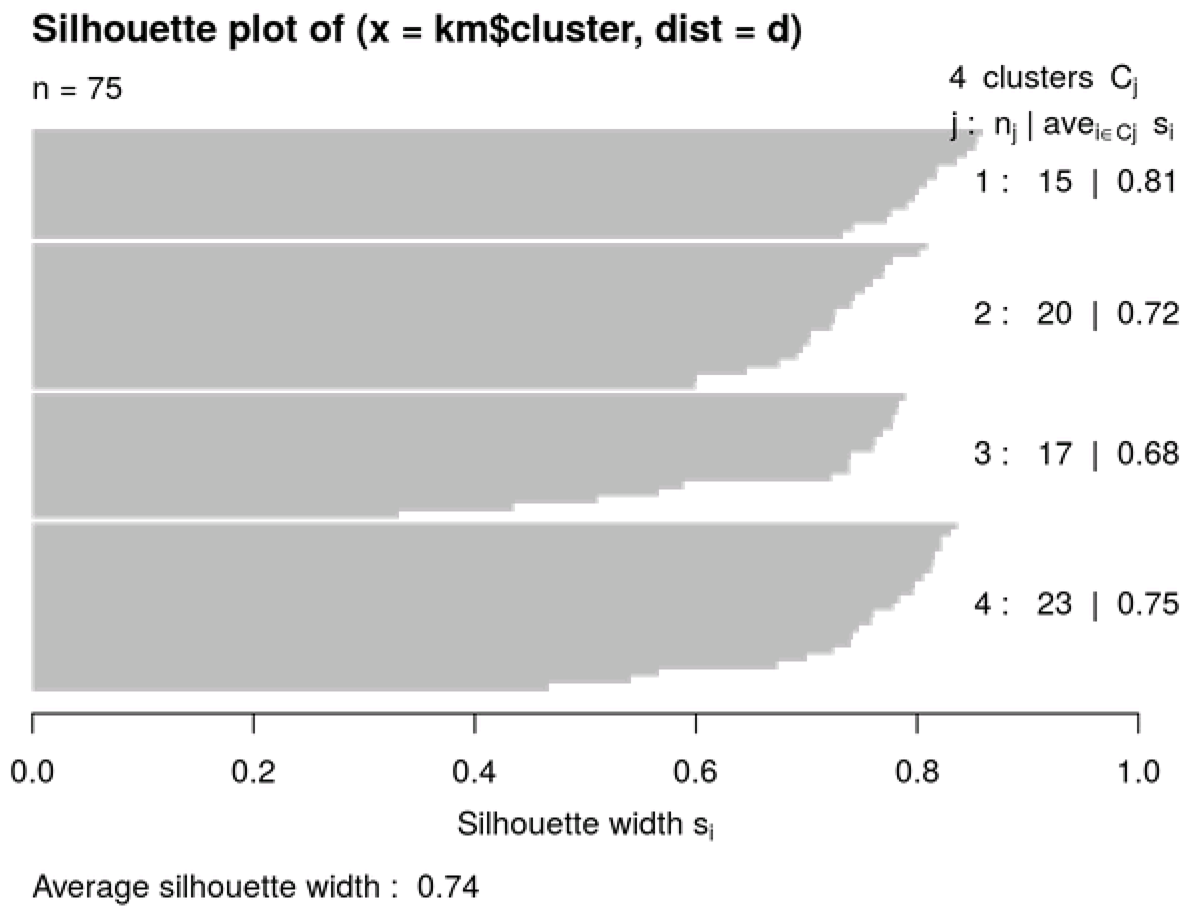
Cohersion Separation

$$-1 \le s(i) \le 1$$

  - The closer to 1 the better.

- Can calculate the Average Silhouette width for a cluster or a clustering

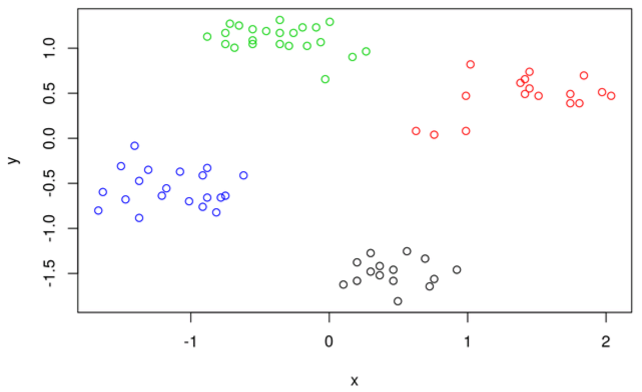
Silhouette Plot

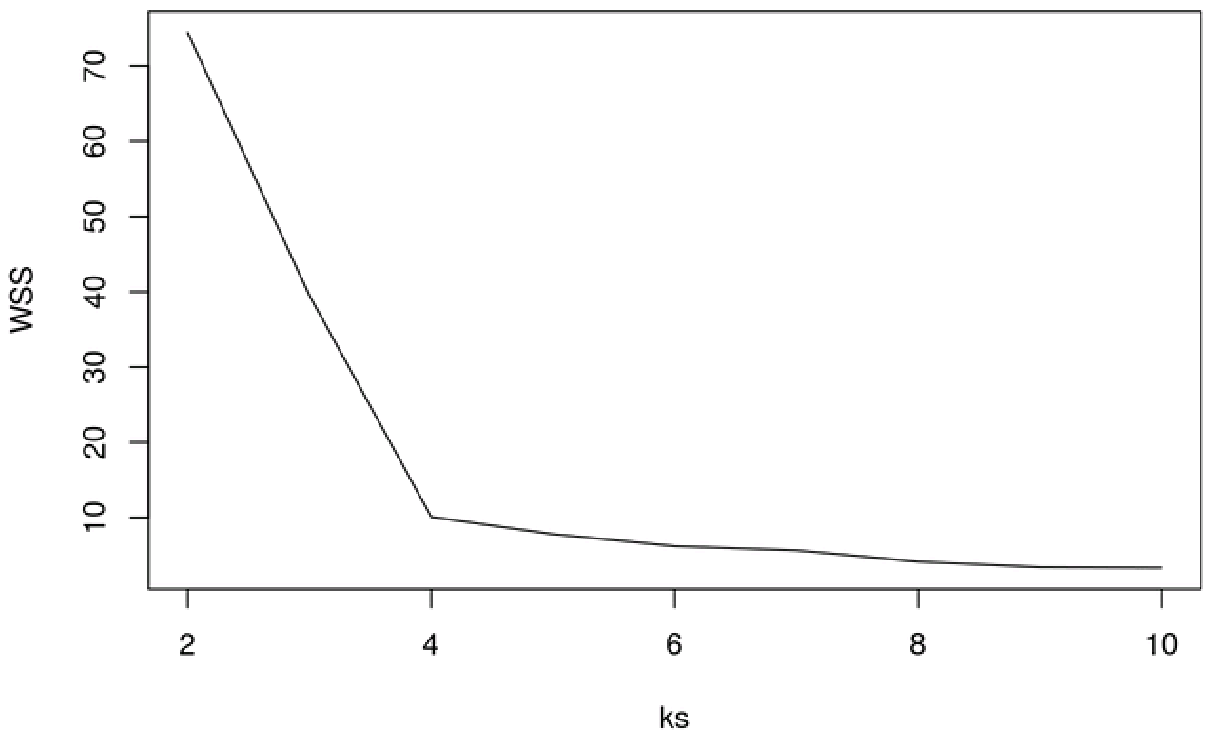


Determine the Number of Clusters

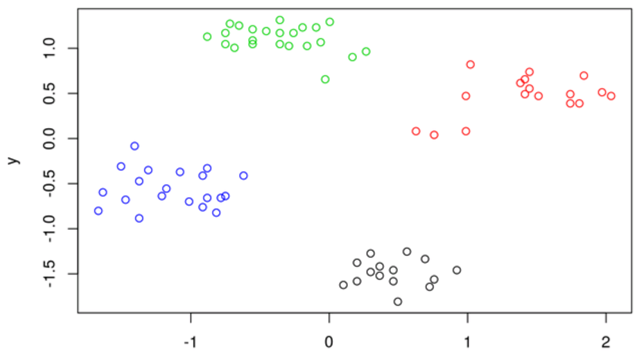
- Empirical method
  - # of clusters ≈$\sqrt{n/2}$ for a dataset of n points
- Elbow method
  - Use the turning point in the curve of sum of within cluster variance with respect to the # of clusters
- Cross validation method
  - Divide a given data set into *m* parts
  - Use *m* – 1 parts to obtain a clustering model
  - Use the remaining part to test the quality of the clustering
    - 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 with respect to different *k's*, and find # of clusters that fits the data the best

Elbow Method Using SSE

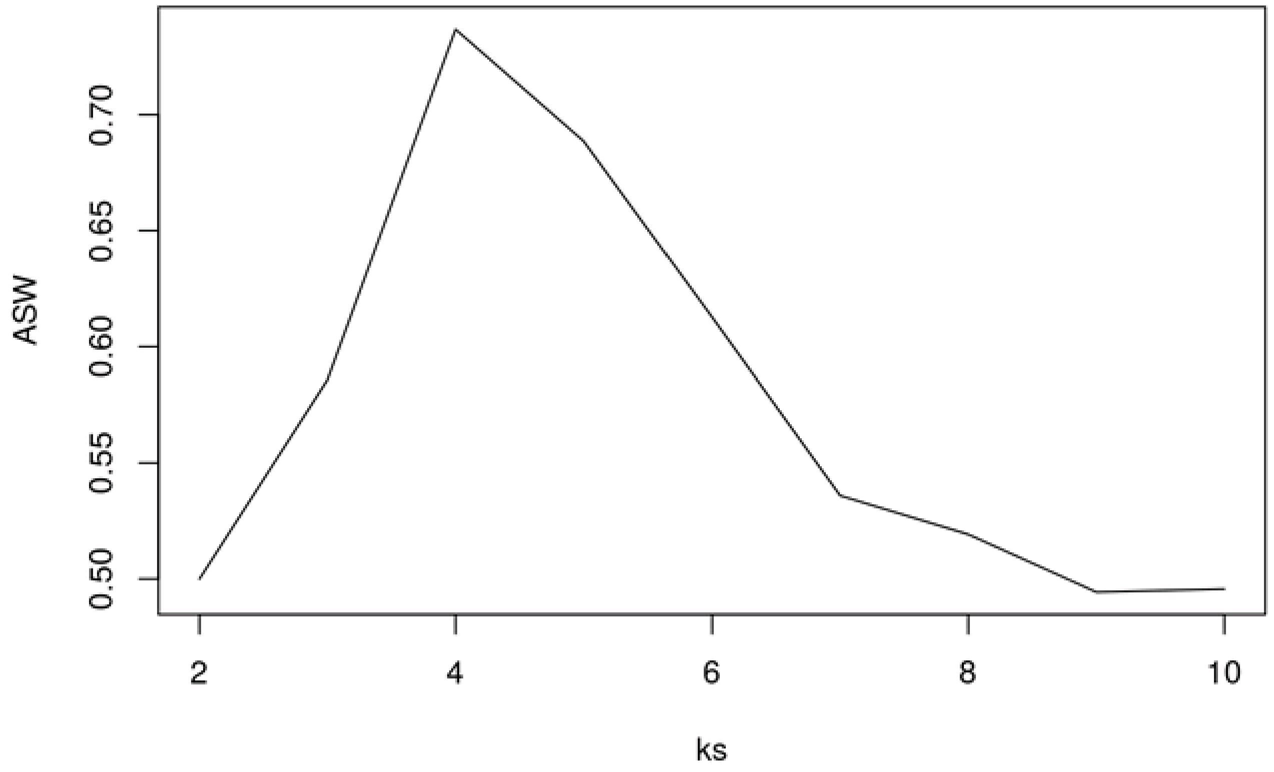




Elbow Method using Average Sillouette



y
1.0
0.5
0.0
-0.5
-1.0
-1.5
-1
0
1
2


ASW
ks