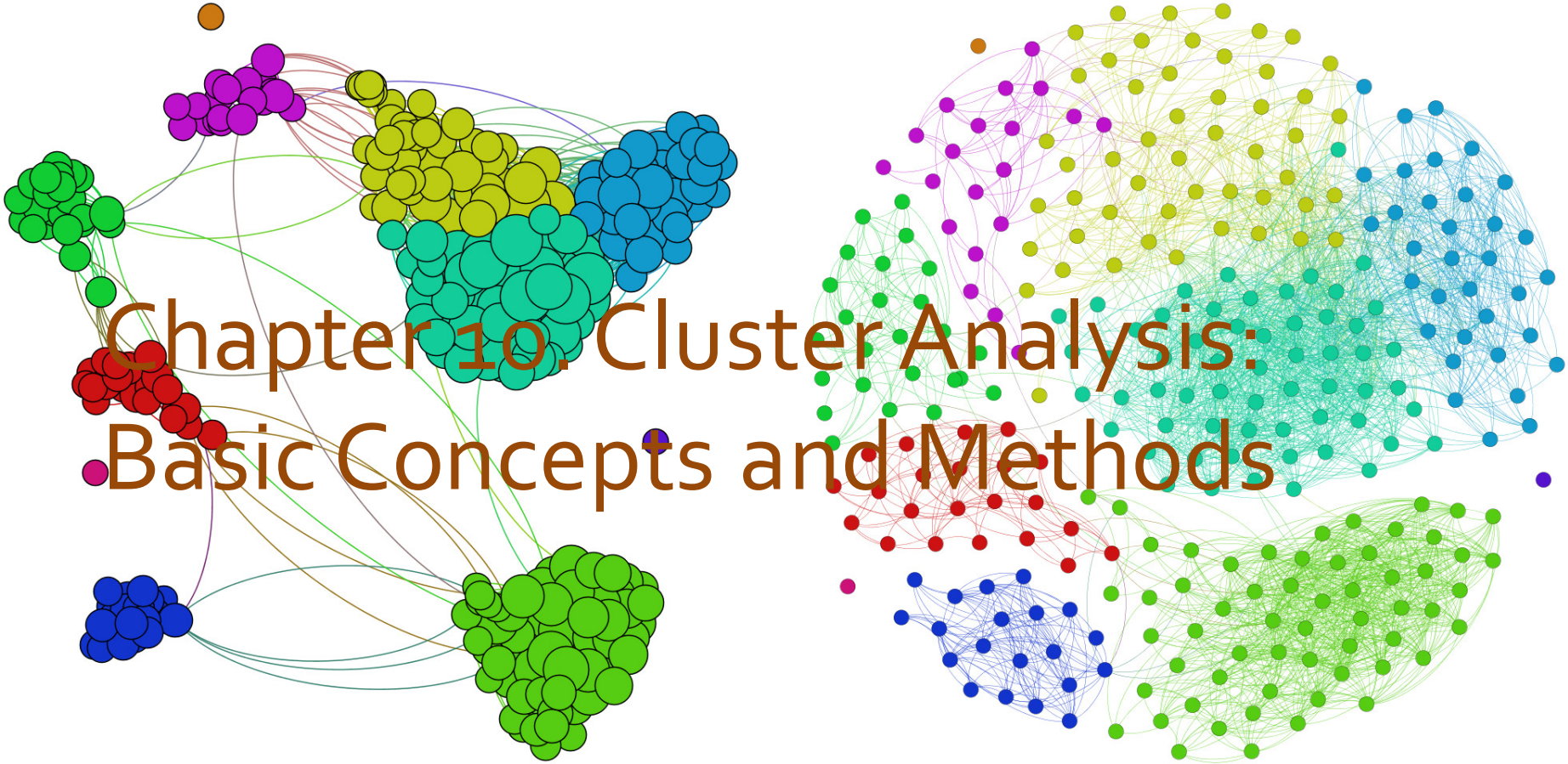


Chapter 10. Cluster Analysis: Basic Concepts and Methods

Meng Jiang

CS412 Summer 2017:

Introduction to Data Mining

Cluster Analysis: Basic Concepts and Methods

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

What Is Cluster Analysis?

- **What is a cluster?**
  - A cluster is a collection of data objects which are
    - Similar (or related) to one another within the same group (i.e., cluster)
    - Dissimilar (or unrelated) to the objects in other groups (i.e., clusters)
- **Cluster analysis** (or ***clustering, data segmentation***, …)
  - Given a set of data points, partition them into a set of groups (i.e., clusters) which are as similar as possible
- Cluster analysis is **unsupervised learning** (i.e., no predefined classes)
  - This contrasts with *classification* (i.e., *supervised learning*)
- Typical ways to use/apply cluster analysis
  - As a stand-alone tool to get insight into data distribution, or
  - As a preprocessing (or intermediate) step for other algorithms

What Is Good Clustering?

- A good clustering method will produce high quality clusters which should have
  - **High intra-class similarity:** Cohesive within clusters
  - **Low inter-class similarity:** Distinctive between clusters
- **Quality function**
  - There is usually a separate "quality" function that measures the "goodness" of a cluster
  - It is hard to define "similar enough" or "good enough"
    - The answer is typically highly subjective
- There exist many similarity measures and/or functions for different applications
- Similarity measure is critical for cluster analysis

Cluster Analysis: Applications

- A key intermediate step for other data mining tasks
  - Generating a compact summary of data for classification, pattern discovery, hypothesis generation and testing, etc.
  - Outlier detection: Outliers—those "far away" from any cluster
- Data summarization, compression, and reduction
  - Ex. Image processing: Vector quantization
- Collaborative filtering, recommendation systems, or customer segmentation
  - Find like-minded users or similar products
- Dynamic trend detection
  - Clustering stream data and detecting trends and patterns
- Multimedia data analysis, biological data analysis and social network analysis
  - Ex. Clustering images or video/audio clips, gene/protein sequences, etc.

Considerations for Cluster Analysis

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

Requirements and Challenges

- **Quality**
  - Ability to deal with different types of attributes: Numerical, categorical, text, multimedia, networks, and mixture of multiple types
  - Discovery of clusters with arbitrary shape
  - Ability to deal with noisy data
- **Scalability**
  - Clustering all the data instead of only on samples
  - High dimensionality
  - Incremental or stream clustering and insensitivity to input order
- **Constraint-based clustering**
  - User-given preferences or constraints; domain knowledge; user queries
- **Interpretability and usability**

Cluster Analysis: A Multi-Dimensional Categorization

- **Technique-Centered**
  - Distance-based methods
  - Density-based and grid-based methods
  - Probabilistic and generative models
  - Leveraging dimensionality reduction methods
  - High-dimensional clustering
  - Scalable techniques for cluster analysis
- **Data Type-Centered**
  - Clustering numerical data, categorical data, text data, multimedia data, time-series data, sequences, stream data, networked data, uncertain data
- **Additional Insight-Centered**
  - Visual insights, semi-supervised, ensemble-based, validation-based

Typical Clustering Methodologies (I)

- Distance-based methods
  - Partitioning algorithms: K-Means, K-Medians, K-Medoids
  - Hierarchical algorithms: Agglomerative vs. divisive methods
- Density-based and grid-based methods
  - Density-based: Data space is explored at a high-level of granularity and then post-processing to put together dense regions into an arbitrary shape
  - Grid-based: Individual regions of the data space are formed into a grid-like structure
- Probabilistic and generative models: Modeling data from a generative process
  - Assume a specific form of the generative model (e.g., mixture of Gaussians)
  - Model parameters are estimated with the Expectation-Maximization (EM) algorithm (using the available dataset, for a maximum likelihood fit)
  - Then estimate the generative probability of the underlying data points

Typical Clustering Methodologies (II)

- High-dimensional clustering
  - Subspace clustering: Find clusters on various subspaces
    - Bottom-up, top-down, correlation-based methods vs. δ-cluster methods
  - Dimensionality reduction: A vertical form (i.e., columns) of clustering
    - Columns are clustered; may cluster rows and columns together (co-clustering)
    - Probabilistic latent semantic indexing (PLSI) then LDA: Topic modeling of text data
      - A cluster (i.e., topic) is associated with a set of words (i.e., dimensions) and a set of documents (i.e., rows) simultaneously
    - Nonnegative matrix factorization (NMF) (as one kind of co-clustering)
      - A nonnegative matrix A (e.g., word frequencies in documents) can be approximately factorized two non-negative low rank matrices U and V
    - Spectral clustering: Use the spectrum of the similarity matrix of the data to perform dimensionality reduction for clustering in fewer dimensions

Clustering Different Types of Data (I)

- **Numerical data**
  - Most earliest clustering algorithms were designed for numerical data
- **Categorical data** (including binary data)
  - Discrete data, no natural order (e.g., sex, race, zip-code, and market-basket)
- **Text data**: Popular in social media, Web, and social networks
  - Features: High-dimensional, sparse, value corresponding to word frequencies
  - Methods: Combination of k-means and agglomerative; topic modeling; co-clustering
- **Multimedia data**: Image, audio, video (e.g., on Flickr, YouTube)
  - Multi-modal (often combined with text data)
  - Contextual: Containing both behavioral and contextual attributes
    - Images: Position of a pixel represents its context, value represents its behavior
    - Video and music data: Temporal ordering of records represents its meaning

Clustering Different Types of Data (II)

- **Time-series data**: Sensor data, stock markets, temporal tracking, forecasting, etc.
  - Data are temporally dependent
  - Time: contextual attribute; data value: behavioral attribute
  - Correlation-based online analysis (e.g., online clustering of stock to find stock tickers)
  - Shape-based offline analysis (e.g., cluster ECG based on overall shapes)
- **Sequence data**: Weblogs, biological sequences, system command sequences
  - Contextual attribute: Placement (rather than time)
  - Similarity functions: Hamming distance, edit distance, longest common subsequence
  - Sequence clustering: Suffix tree; generative model (e.g., Hidden Markov Model)
- **Stream data**:
  - Real-time, evolution and concept drift, single pass algorithm
  - Create efficient intermediate representation, e.g., micro-clustering

Clustering Different Types of Data (III)

- **Graphs and homogeneous networks**
  - Every kind of data can be represented as a graph with similarity values as edges
  - Methods: Generative models; combinatorial algorithms (graph cuts); spectral methods; non-negative matrix factorization methods
- **Heterogeneous networks**
  - A network consists of multiple typed nodes and edges (e.g., bibliographical data)
  - Clustering different typed nodes/links together (e.g., NetClus)
- **Uncertain data**: Noise, approximate values, multiple possible values
  - Incorporation of probabilistic information will improve the quality of clustering
- **Big data**: Model systems may store and process very big data (e.g., weblogs)
  - Ex. Google's MapReduce framework
    - Use *Map* function to distribute the computation across different machines
    - Use R*educe* function to aggregate results obtained from the Map step

User Insights and Interactions in Clustering

- **Visual insights**: One picture is worth a thousand words
  - Human eyes: High-speed processor linking with a rich knowledge-base
  - A human can provide intuitive insights; HD-eye: visualizing HD clusters
- **Semi-supervised insights**: Passing user's insights or intention to system
  - User-seeding: A user provides a number of labeled examples, approximately representing categories of interest
- **Multi-view and ensemble-based insights**
  - Multi-view clustering: Multiple clusterings represent different perspectives
  - Multiple clustering results can be ensembled to provide a more robust solution
- **Validation-based insights**: Evaluation of the quality of clusters generated
  - May use case studies, specific measures, or pre-existing labels

Cluster Analysis: Basic Concepts and Methods

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

Partitioning-Based Clustering Methods

- Basic Concepts of Partitioning Algorithms
- The K-Means Clustering Method
- Initialization of K-Means Clustering
- The K-Medoids Clustering Method
- The K-Medians and K-Modes Clustering Methods
- The Kernel K-Means Clustering Method

Partitioning Algorithms: Basic Concepts

- Partitioning method: Discovering the groupings in the data by optimizing a specific objective function and iteratively improving the quality of partitions
- *K*-partitioning method: Partitioning a dataset ***D*** of ***n*** objects into a set of ***K*** clusters so that an objective function is optimized (e.g., the sum of squared distances is minimized, where c_k is the centroid or medoid of cluster C_k)
  - A typical objective function: **Sum of Squared Errors** (**SSE**)

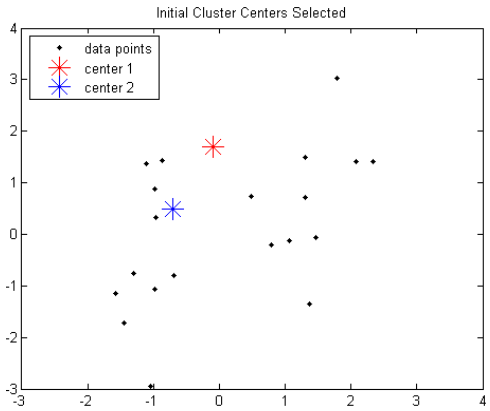
$$SSE(C) = \sum_{k=1}^{K} \sum_{x_i \in C_k} || x_i - c_k ||^2$$

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

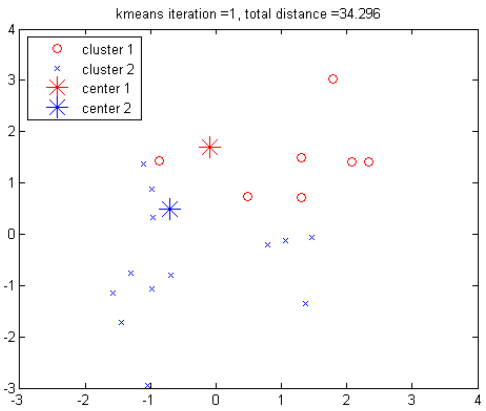
The *K-Means* Clustering Method

- *K-Means* (MacQueen'67, Lloyd'57/'82)
  - Each cluster is represented by the center of the cluster
- Given K, the number of clusters, the *K-Means* clustering algorithm is outlined as follows
    - Select *K* points as initial **centroids**
    - **Repeat**
      - Form *K* clusters by assigning each point to its closest centroid
      - Re-compute the centroids (i.e., *mean point*) of each cluster
    - **Until** convergence criterion is satisfied
- Different kinds of measures can be used
  - Manhattan distance (L_1 norm), Euclidean distance (L_2 norm), Cosine similarity

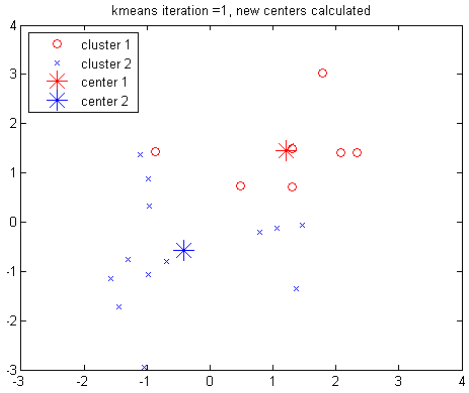
Example: *K-Means* Clustering



Assign points to clusters

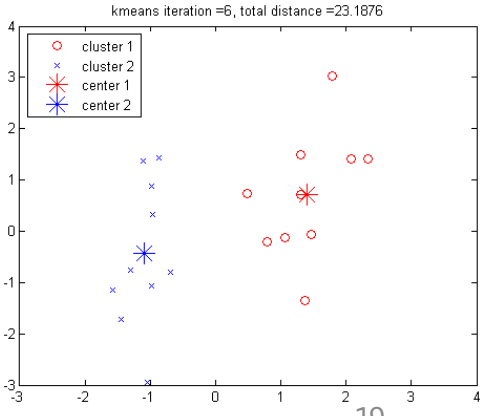


Re-compute cluster centers



The original data points & randomly select $K = 2$ centroids

Redo point assignment



Execution of the K-Means Clustering Algorithm

Select K points as initial centroids

Repeat

- Form K clusters by assigning each point to its closest centroid
- Re-compute the centroids (i.e., *mean point*) of each cluster

Until convergence criterion is satisfied

Discussion on the *K-Means* Method

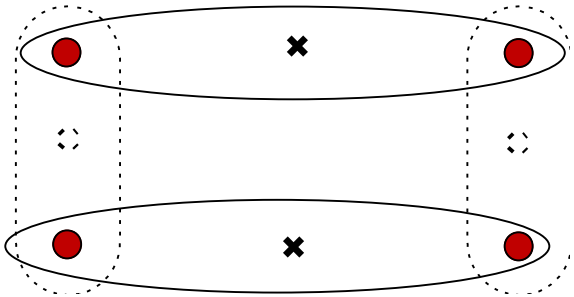
- **Efficiency**: $O(tKn)$ where n: # of objects, K: # of clusters, and t: # of iterations
  - Normally, K, $t << n$; thus, an efficient method
- K-means clustering often ***terminates at* a *local optimal***
  - Initialization can be important to find high-quality clusters
- **Need to specify *K***, the *number* of clusters, in advance
  - There are ways to automatically determine the "*best*" *K*
  - In practice, one often runs a range of values and selected the "*best*" *K* value
- **Sensitive to noisy data and *outliers***
  - Variations: Using K-medians, K-medoids, etc.
- K-means is applicable only to objects in a continuous n-dimensional space
  - Using the K-modes for ***categorical data***
- Not suitable to discover clusters with ***non-convex shapes***
  - Using density-based clustering, kernel *K*-means, etc.

Variations of *K-Means*

- There are many variants of the *K-Means* method, varying in different aspects
  - Choosing better initial centroid estimates
    - *K-means++*, *Intelligent K-Means*, *Genetic K-Means*
  - Choosing different representative prototypes for the clusters
    - *K-Medoids*, *K-Medians*, *K-Modes*
  - Applying feature transformation techniques
    - *Weighted K-Means*, *Kernel K-Means*

Initialization of K-Means

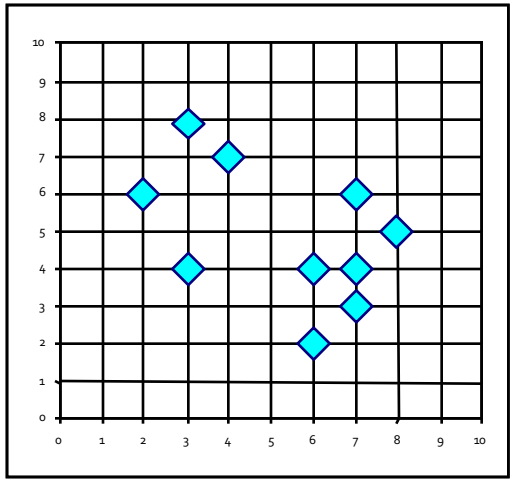
- Different initializations may generate rather different clustering results (some could be far from optimal)
- Original proposal (MacQueen'67): Select *K* seeds randomly
  - Need to run the algorithm multiple times using different seeds
- There are many methods proposed for better initialization of *k* seeds
  - ***K-Means++*** (Arthur & Vassilvitskii'07):
    - The first centroid is selected at random
    - The next centroid selected is the one that is farthest from the currently selected (selection is based on a weighted probability score)
    - The selection continues until *K* centroids are obtained



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

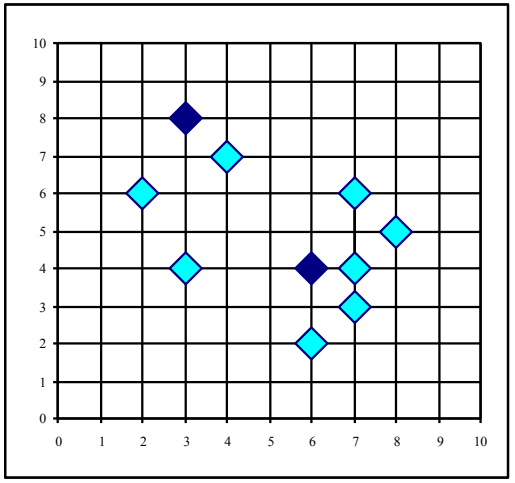
- The *K-Means* algorithm is sensitive to outliers!—since an object with an extremely large value may substantially distort the distribution of the data
- *K-Medoids*: Instead of taking the **mean** value of the object in a cluster as a reference point, **medoids** can be used, which is the **most centrally located** object in a cluster
- The *K-Medoids* clustering algorithm:
    - Select *K* points as the initial representative objects (i.e., as initial *K medoids*)
    - **Repeat**
        - Assigning each point to the cluster with the closest medoid
        - Randomly select a non-representative object o_i
        - Compute the total **cost** *S* of **swapping the medoid** *m* with o_i
        - If $S < 0$, then swap *m* with o_i to form the new set of medoids
    - **Until** convergence criterion is satisfied

PAM: A Typical *K-Medoids* Algorithm

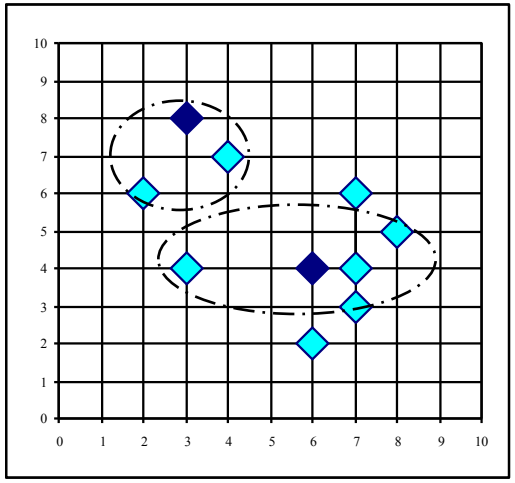


$K = 2$

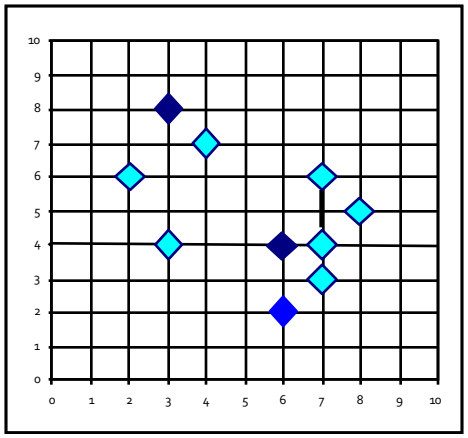
Arbitrary choose K object as initial medoids



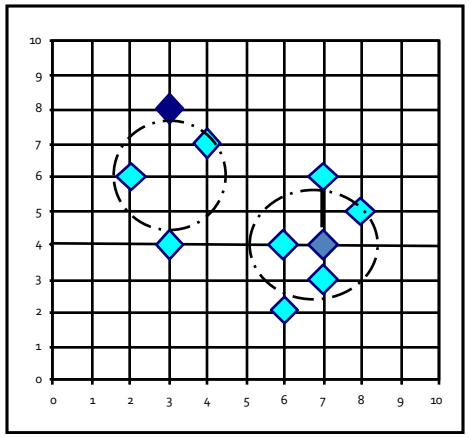
Assign each remaining object to nearest medoids



Randomly select a non-medoid object, O_{ramdom}



Compute total cost of swapping



Swapping O and O_{ramdom}

If quality is improved

Select initial *K medoids* randomly

Repeat

Object re-assignment

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

Until convergence criterion is satisfied

Discussion on *K-Medoids* Clustering

- *K-Medoids* Clustering: Find *representative* objects (medoids) in clusters
- *PAM* (Partitioning Around Medoids: Kaufmann & Rousseeuw 1987)
  - Starts from an initial set of medoids, and
  - Iteratively replaces one of the medoids by one of the non-medoids if it improves the total sum of the squared errors (SSE) of the resulting clustering
  - *PAM* works effectively for small data sets but **does not scale well** for large data sets (due to the computational complexity)
  - Computational complexity: PAM: $O(\mathbf{K(n-K)^2})$ (quite expensive!)
- Efficiency improvements on PAM
  - *CLARA* (Kaufmann & Rousseeuw, 1990):
    - PAM on samples; $O(Ks^2 + K(n-K))$, s is the sample size
  - *CLARANS* (Ng & Han, 1994): Randomized re-sampling, ensuring efficiency + quality

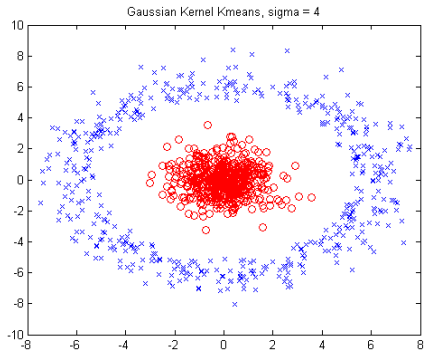
K-Medians: Handling Outliers by Computing Medians

- Medians are less sensitive to outliers than means
  - Think of the median salary vs. mean salary of a large firm when adding a few top executives!
- ***K-Medians***: Instead of taking the **mean** value of the object in a cluster as a reference point, **medians** are used (L_1-norm as the distance measure)
- The criterion function for the *K-Medians* algorithm: $S = \sum_{k=1}^{K} \sum_{x_{i} \in C_k} | x_{ij} - med_{kj} |$
- The *K-Medians* clustering algorithm:
  - Select *K* points as the initial representative objects (i.e., as initial *K medians*)
  - **Repeat**
    - Assign every point to its nearest median
    - Re-compute the median using the median of each individual feature
  - **Until** convergence criterion is satisfied

K-Modes: Clustering Categorical Data

- *K-Means* cannot handle non-numerical (categorical) data
  - Mapping categorical value to 1/0 cannot generate quality clusters for high-dimensional data
- ***K-Modes*:** An extension to *K-Means* by replacing means of clusters with ***modes***
- Dissimilarity measure between object X and the center of a cluster Z
  - $\Phi(x_j, z_j) = 1 - n_j^r/n_l$ when $x_j = z_j$; 1 when $x_j \neq z_j$
    - where z_j is the categorical value of attribute j in Z_l, n_l is the number of objects in cluster *l*, and n_j^r is the number of objects whose attribute value is r
- This dissimilarity measure (distance function) is **frequency-based**
- Algorithm is still based on iterative *object cluster assignment* and *centroid update*
- A ***fuzzy K-Modes*** method is proposed to calculate a ***fuzzy cluster membership value*** for each object to each cluster
- A mixture of categorical and numerical data: Using a ***K-Prototype*** method

Kernel K-Means Clustering



- Kernel K-Means can be used to detect non-convex clusters
  - K-Means can only detect clusters that are linearly separable
- Idea: Project data onto the high-dimensional kernel space, and then perform K-Means clustering
  - Map data points in the input space onto a high-dimensional feature space using the kernel function
  - Perform K-Means on the mapped feature space
- Computational complexity is higher than K-Means
  - Need to compute and store n x n kernel matrix generated from the kernel function on the original data
- The widely studied spectral clustering can be considered as a variant of Kernel K-Means clustering

Kernel Functions and Kernel K-Means Clustering

Kernel matrix:
inner-product(i, j) matrix √
similarity
distance(i, j) matrix ×

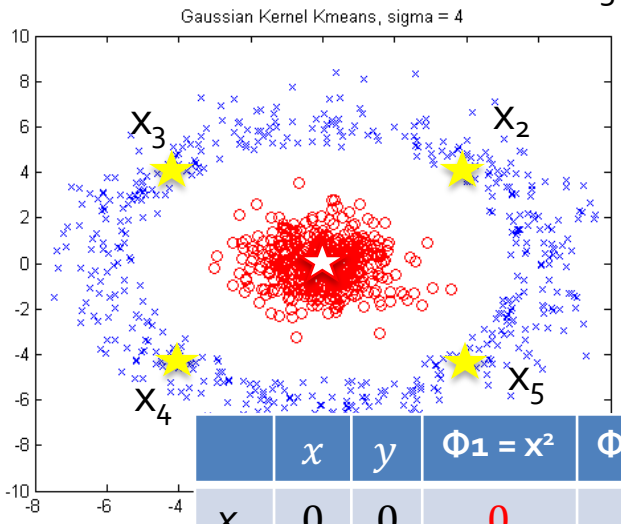
- Typical kernel functions:
  - Polynomial kernel of degree h: $K(\boldsymbol{X}_i, \boldsymbol{X}_j) = (\boldsymbol{X}_i \cdot \boldsymbol{X}_j + c)^h$
  - Gaussian radial basis function (RBF) kernel: $K(\boldsymbol{X}_i, \boldsymbol{X}_j) = e^{-\|X_i - X_j\|^2 / 2\sigma^2}$
  - Sigmoid kernel: $K(\boldsymbol{X}_i, \boldsymbol{X}_j) = \tanh(\kappa \boldsymbol{X}_i \cdot \boldsymbol{X}_j - \delta)$
- The formula for **kernel matrix** K for any two points x_i, $x_j \in C_k$ is $K_{x_i x_j} = \phi(x_i) \bullet \phi(x_j)$
- The SSE criterion of *kernel K-means*: $SSE(C) = \sum_{k=1}^{K} \sum_{x_i \in C_k} \| \phi(x_i) - c_k \|^2$
  - The formula for the cluster centroid: $c_k = \frac{\sum_{x_i \in C_k} \phi(x_i)}{| C_k |}$

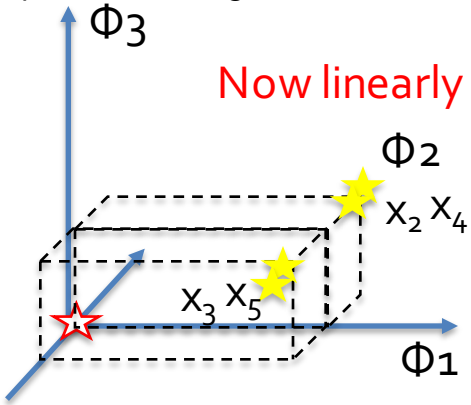
Inderjit S. Dhillon, Yiqiang Guan, Brian Kulis (Univ. of Texas at Austin). "Kernel K-means, Spectral Clustering and Normalized Cuts", KDD 04.

http://www.cs.utexas.edu/users/inderjit/public_papers/kdd_spectral_kernelkmeans.pdf

Example: Kernel Functions and Kernel K-Means Clustering

- Polynomial kernel of degree h=2: $K(\boldsymbol{X}_i, \boldsymbol{X}_j) = \boldsymbol{X}_i \cdot \boldsymbol{X}_j{}^2 \rightarrow \phi(x, y) = (x^2, \sqrt{2}xy, y^2)$
- Suppose there are 5 original 2-dimensional points:
  - $x_1(0, 0)$, $x_2(4, 4)$, $x_3(-4, 4)$, $x_4(-4, -4)$, $x_5(4, -4)$





Now linearly separable!!!

	x	y	Φ1 = x²	Φ2 = xy	ϕ3 = y²	$K(x_i,x_1)$	$K(x_i,x_2)$	$K(x_i,x_3)$	$K(x_i,x_4)$	$K(x_i,x_5)$
x_1	0	0	0	0	0	0	0	0	0	0
x_2	4	4	16	$16\sqrt{2}$	16	0	32^2	0	32^2	0
x_3	-4	4	16	$-16\sqrt{2}$	16	0	0	32^2	0	32^2
x_4	-4	-4	16	$16\sqrt{2}$	16	0	32^2	0	32^2	0
x_5	4	-4	16	$-16\sqrt{2}$	16	0	0	32^2	0	32^2

Example: Kernel Functions and Kernel K-Means Clustering

- Suppose there are 5 original 2-dimensional points:
  - $x_1(0, 0)$, $x_2(4, 4)$, $x_3(-4, 4)$, $x_4(-4, -4)$, $x_5(4, -4)$

Original Space

	x	y	(x_i, x_1)	(x_i, x_2)	(x_i, x_3)	(x_i, x_4)	(x_i, x_5)
x_1	0	0	0	0	0	0	0
x_2	4	4	0	32	0	−32	0
x_3	−4	4	0	0	32	0	−32
x_4	−4	−4	0	−32	0	32	0
x_5	4	−4	0	0	−32	0	32

Example: Kernel Functions and Kernel K-Means Clustering

- Gaussian radial basis function (RBF) kernel: $K(\boldsymbol{X}_i, \boldsymbol{X}_j) = e^{-\|X_i - X_j\|^2 / 2\sigma^2}$
- Suppose there are 5 original 2-dimensional points: $K_{x_i x_j} = \phi(x_i) \bullet \phi(x_j)$
  - $x_1(0, 0)$, $x_2(4, 4)$, $x_3(-4, 4)$, $x_4(-4, -4)$, $x_5(4, -4)$
- If we set σ to 4, we will have the following points in the kernel space
  - E.g., $\left\|x_1 - x_2\right\|^2 = (0-4)^2 + (0-4)^2 = 32$, therefore, $K(x_1, x_2) = e^{-\frac{32}{2 \cdot 4^2}} = e^{-1}$

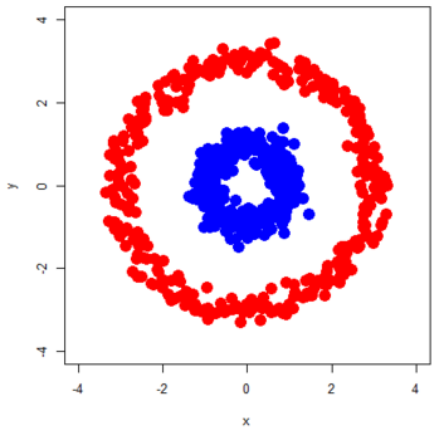
Original Space

RBF Kernel Space ($\sigma = 4$)

	x	y	$K(x_i, x_1)$	$K(x_i, x_2)$	$K(x_i, x_3)$	$K(x_i, x_4)$	$K(x_i, x_5)$
x_1	0	0	1	$e^{-\frac{4^2+4^2}{2 \cdot 4^2}} = e^{-1}$	e^{-1}	e^{-1}	e^{-1}
x_2	4	4	e^{-1}	1	e^{-2}	e^{-4}	e^{-2}
x_3	-4	4	e^{-1}	e^{-2}	1	e^{-2}	e^{-4}
x_4	-4	-4	e^{-1}	e^{-4}	e^{-2}	1	e^{-2}
x_5	4	-4	e^{-1}	e^{-2}	e^{-4}	e^{-2}	1

Example: Kernel Functions and Kernel K-Means Clustering

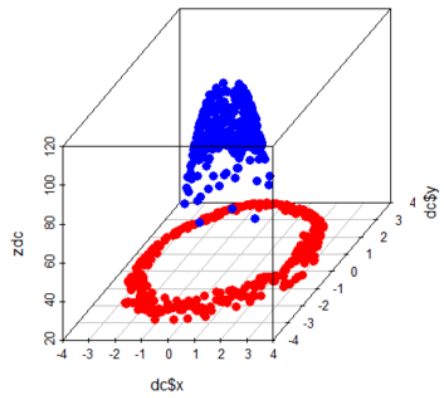
- Gaussian radial basis function (RBF) kernel: $K(\boldsymbol{X}_i, \boldsymbol{X}_j) = e^{-\|X_i - X_j\|^2 / 2\sigma^2}$

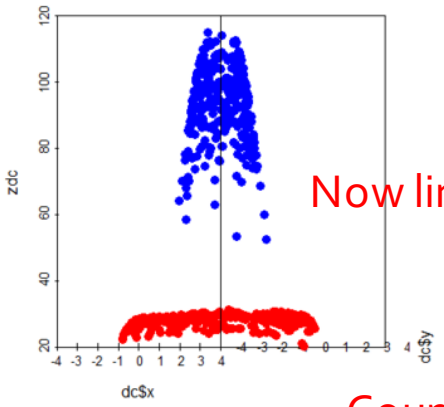


$$\underset{\mathcal{J}_1,\ldots,\mathcal{J}_k}{\operatorname{argmin}} \sum_{i=1}^{k} \sum_{j \in \mathcal{J}_i} \left\| \mathbf{a}_j - \frac{1}{|\mathcal{J}_i|} \sum_{l \in \mathcal{J}_i} \mathbf{a}_l \right\|_2^2.$$

$$\underset{\mathcal{J}_1,\ldots,\mathcal{J}_k}{\operatorname{argmin}} \sum_{i=1}^{k} \sum_{j \in \mathcal{J}_i} \left\| \boldsymbol{\phi}(\mathbf{a}_j) - \frac{1}{|\mathcal{J}_i|} \sum_{l \in \mathcal{J}_i} \boldsymbol{\phi}(\mathbf{a}_l) \right\|_2^2.$$

$$\kappa(\mathbf{a}_i, \mathbf{a}_j) = \langle \boldsymbol{\phi}(\mathbf{a}_i), \boldsymbol{\phi}(\mathbf{a}_j) \rangle.$$



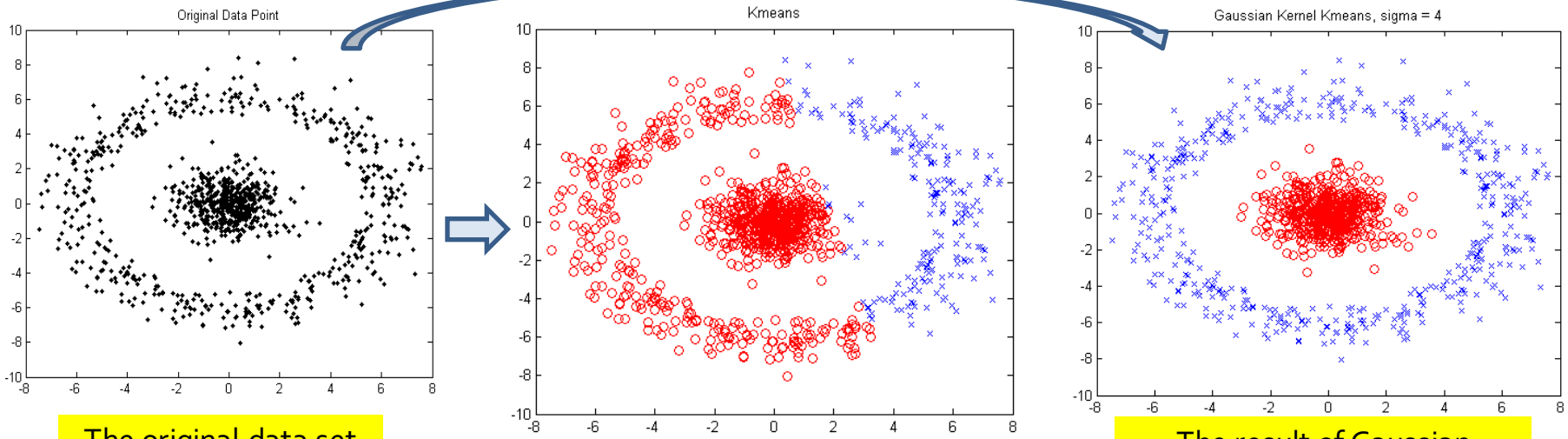


Now linearly separable!!!

$$\exp(z) = \sum_{k=0}^{\infty} \frac{z^k}{k!} = 1 + z + \frac{z^2}{2} + \frac{z^3}{6} + \frac{z^4}{24} + \cdots$$

Countless new features in RBF kernel space…

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: $K_{x_i x_j} = \phi(x_i) \bullet \phi(x_j)$ and Gaussian kernel: $K(\boldsymbol{X}_i, \boldsymbol{X}_j) = e^{-\|X_i - X_j\|^2 / 2\sigma^2}$
- K-Means clustering is conducted on the mapped data, generating quality clusters

Cluster Analysis: Basic Concepts and Methods

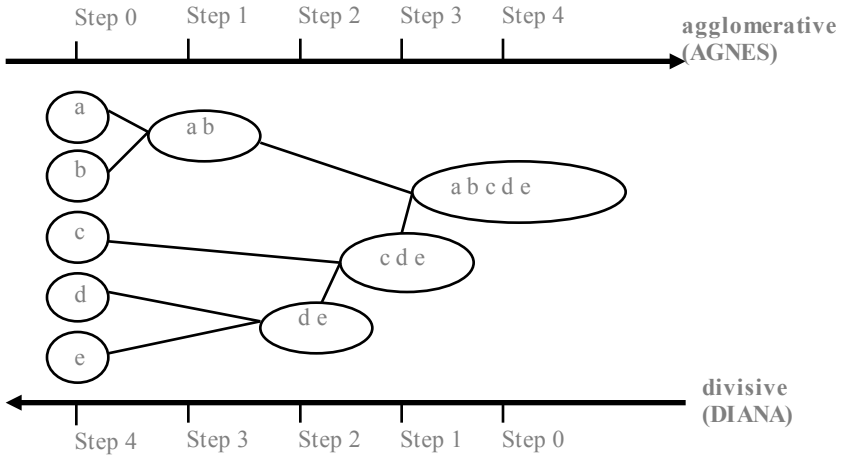
- Cluster Analysis: An Introduction
- Partitioning Methods
- **Hierarchical Methods**
- Density- and Grid-Based Methods
- Evaluation of Clustering

Hierarchical Clustering Methods

- Basic Concepts of Hierarchical Algorithms
- Agglomerative Clustering Algorithms
- Divisive Clustering Algorithms
- Extensions to Hierarchical Clustering
- BIRCH: A Micro-Clustering-Based Approach
- CURE: Exploring Well-Scattered Representative Points
- CHAMELEON: Graph Partitioning on the KNN Graph of the Data
- Probabilistic Hierarchical Clustering

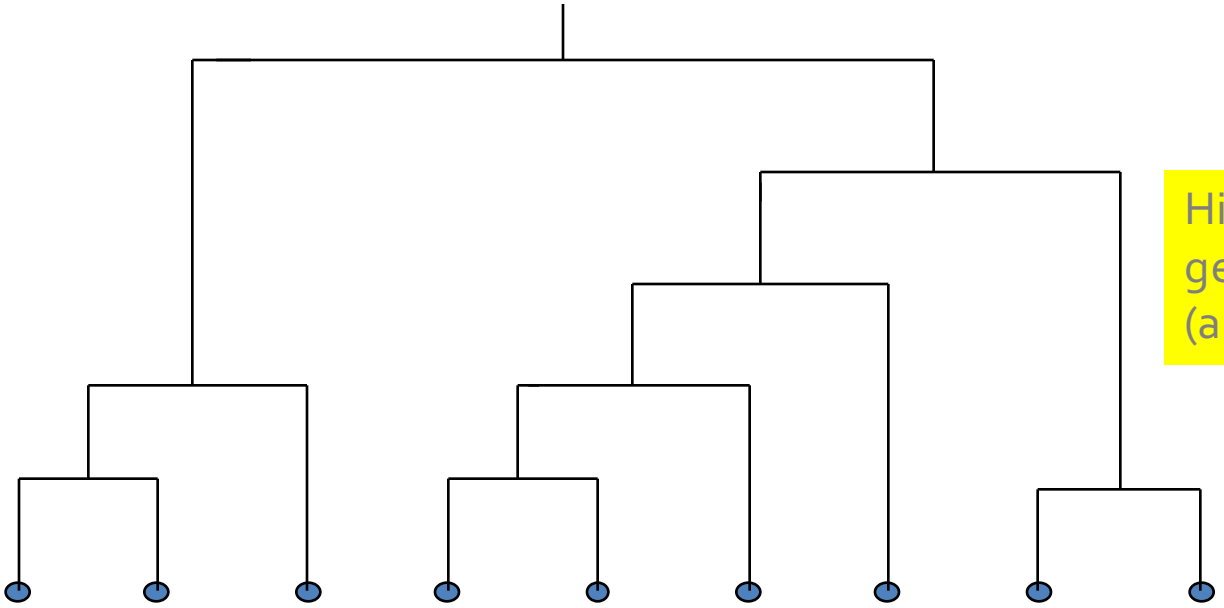
Hierarchical Clustering: Basic Concepts

- Hierarchical clustering
  - Generate a clustering hierarchy (drawn as a dendrogram)
  - Not required to specify K, the number of clusters
  - More deterministic
  - No iterative refinement
- Two categories of algorithms:
  - Agglomerative: Start with singleton clusters, continuously merge two clusters at a time to build a bottom-up hierarchy of clusters
  - Divisive: Start with a huge macro-cluster, split it continuously into two groups, generating a top-down hierarchy of clusters



Dendrogram: Shows How Clusters are Merged

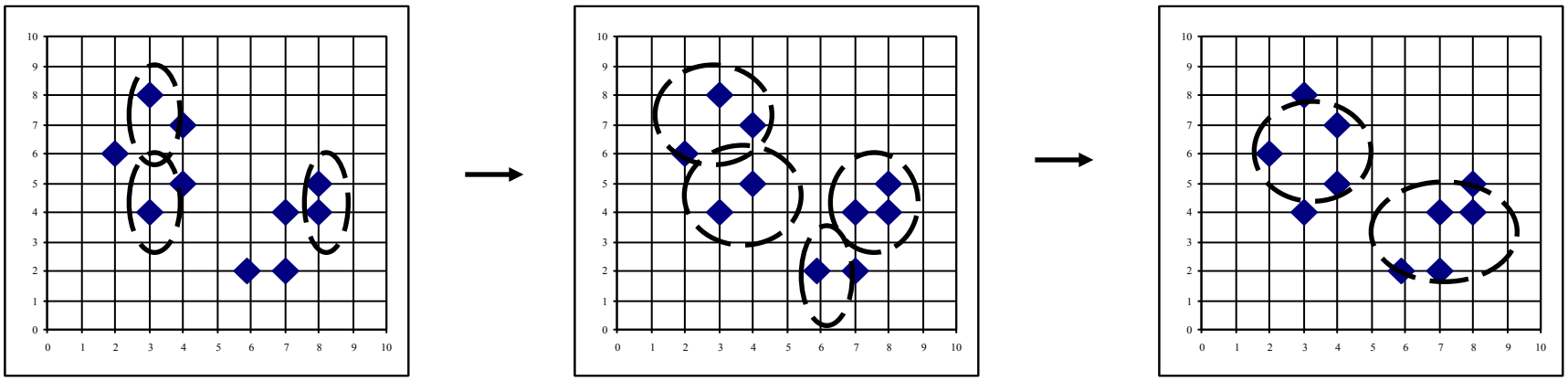
- Dendrogram: Decompose a set of data objects into a tree of clusters by multi-level nested partitioning
- A clustering of the data objects is obtained by cutting the dendrogram at the desired level, then each connected component forms a cluster



Hierarchical clustering generates a dendrogram (a hierarchy of clusters)

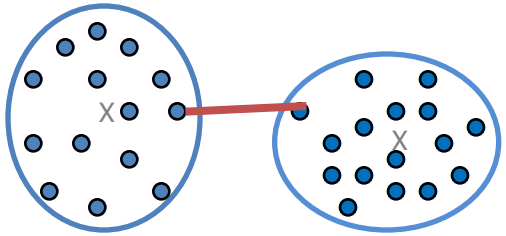
Agglomerative Clustering Algorithm

- AGNES (AGglomerative NESting) (Kaufmann and Rousseeuw, 1990)
  - Use the **single-link** method and the dissimilarity matrix
  - Continuously merge nodes that have the least dissimilarity
  - Eventually all nodes belong to the same cluster
- Agglomerative clustering varies on different similarity measures among clusters
  - Single link (nearest neighbor)
  - Complete link (diameter)
  - Average link (group average)
  - Centroid link (centroid similarity)

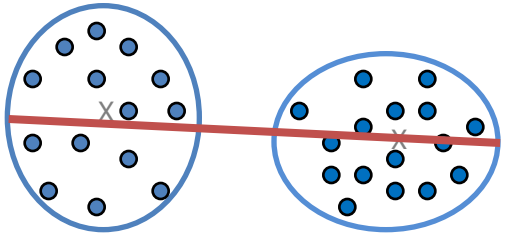


Single Link vs. Complete Link in Hierarchical Clustering

- Single link (nearest neighbor)
  - The similarity between two clusters is the similarity between their most similar (nearest neighbor) members
  - Local similarity-based: Emphasizing more on close regions, ignoring the overall structure of the cluster
  - Capable of clustering non-elliptical shaped group of objects
  - Sensitive to noise and outliers



- Complete link (diameter)
  - The similarity between two clusters is the similarity between their most dissimilar members
  - Merge two clusters to form one with the smallest diameter
  - Nonlocal in behavior, obtaining compact shaped clusters
  - Sensitive to outliers



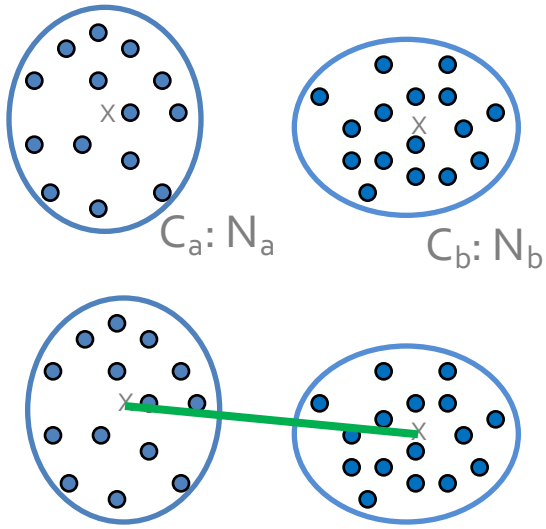
Agglomerative Clustering: Average vs. Centroid Links

- Agglomerative clustering with **average link**
  - **Average link**: The average distance between an element in one cluster and an element in the other (i.e., all pairs in two clusters)
    - Expensive to compute
- Agglomerative clustering with **centroid link**
  - **Centroid link**: The distance between the centroids of two clusters
- **Group Averaged Agglomerative Clustering (GAAC)**
  - Let two clusters C_a and C_b be merged into C_{aUb}. The new centroid is:
    - N_a is the cardinality of cluster C_a, and c_a is the centroid of C_a
  - The similarity measure for GAAC is the average of their distances

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

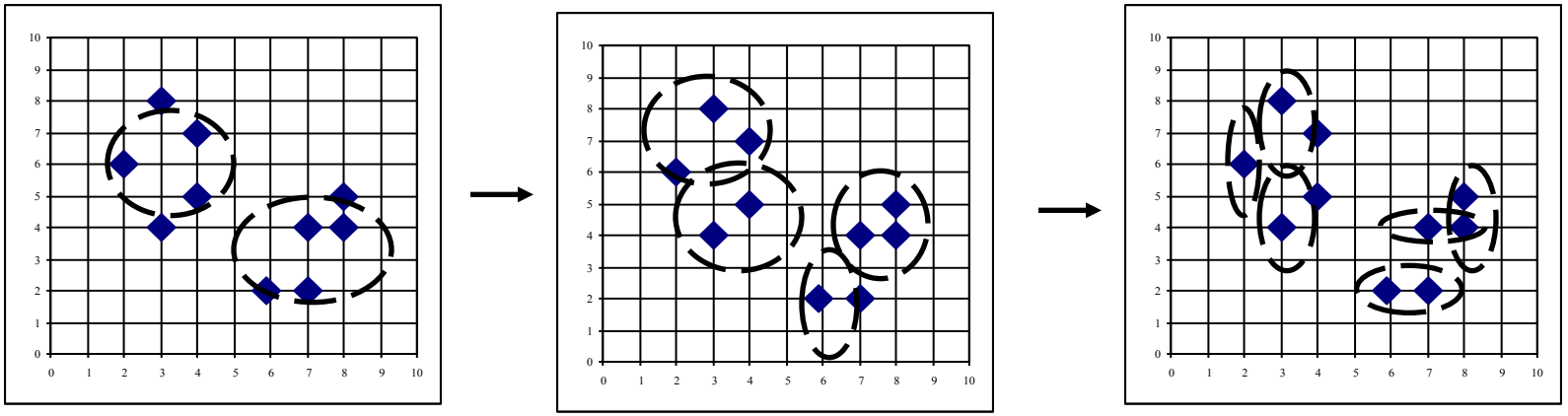
- Agglomerative clustering with **Ward's criterion**
  - **Ward's criterion:** The increase in the value of the SSE criterion for the clustering obtained by merging them into $C_a \cup C_b$:

$$W(C_{a\cup b}, c_{a\cup b}) - W(C,c) = \frac{N_a N_b}{N_a + N_b} d(c_a, c_b)$$



Divisive Clustering

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



- Divisive clustering is a top-down approach
  - The process starts at the root with all the points as one cluster
  - It recursively splits the higher level clusters to build the dendrogram
  - Can be considered as a global approach
  - More efficient when compared with agglomerative clustering

More on Algorithm Design for Divisive Clustering

- Choosing which cluster to split
  - Check the sums of squared errors of the clusters and choose the one with the largest value
- Splitting criterion: Determining how to split
  - One may use Ward's criterion to chase for greater reduction in the difference in the SSE criterion as a result of a split
  - For categorical data, Gini-index can be used
- Handling the noise
  - Use a threshold to determine the termination criterion (do not generate clusters that are too small because they contain mainly noises)

Extensions to Hierarchical Clustering

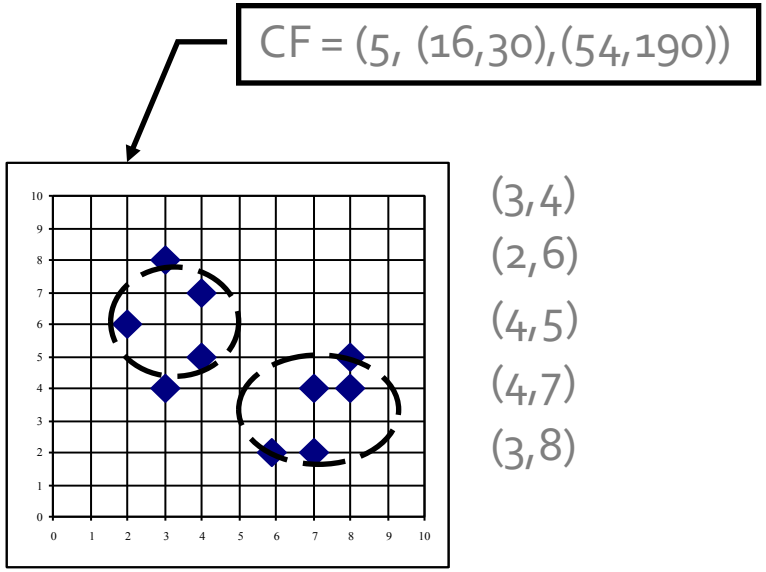
- Major weaknesses of hierarchical clustering methods
  - Can never undo what was done previously
  - Do not scale well
    - Time complexity of at least $O(n^2)$, where n is the number of total objects
- Other hierarchical clustering algorithms
  - BIRCH (1996): Use CF-tree and incrementally adjust the quality of sub-clusters
  - CURE (1998): Represent a cluster using a set of well-scattered representative points
  - CHAMELEON (1999): Use graph partitioning methods on the K-nearest neighbor graph of the data

BIRCH (Balanced Iterative Reducing and Clustering Using Hierarchies)

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

Clustering Feature Vector in BIRCH

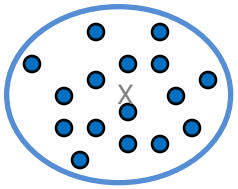
- Clustering Feature (CF): *CF = (N, LS, SS)*
  - *N*: Number of data points
  - *LS: linear sum of N points:* $\sum_{i=1}^{N} X_i$
  - *SS: square sum of N points:* $\sum_{i=1}^{N} X_i^2$



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

Measures of Cluster: Centroid, Radius and Diameter

- Centroid:
  - the "middle" of a cluster
  - n: number of points in a cluster
  - x_i is the *i*-th point in the cluster



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

- Radius: R
  - Average distance from member objects to the centroid
  - The square root of average distance from any point of the cluster to its centroid

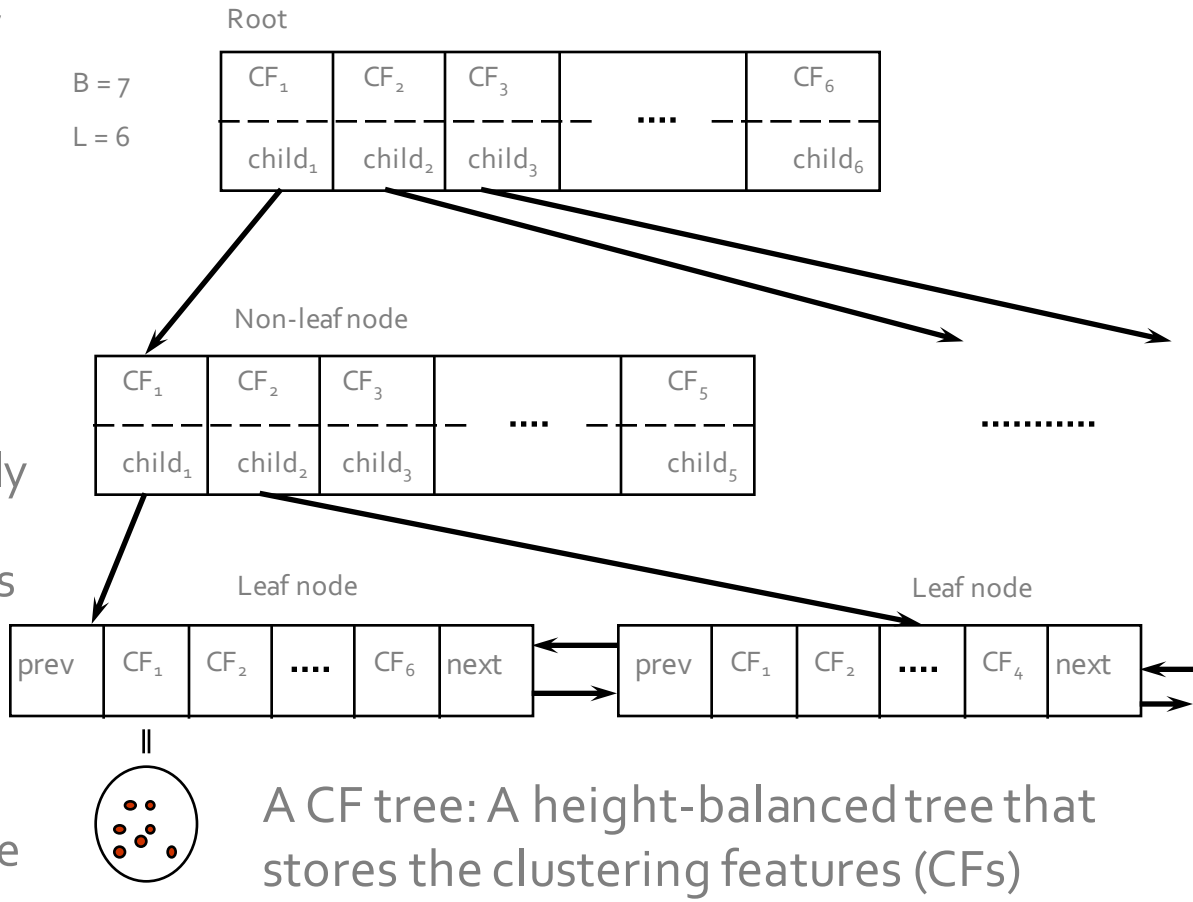
$$R = \sqrt{\frac{\sum_i^n (\vec{x}_i - \vec{x}_0)^2}{n}}$$

- Diameter: D
  - Average pairwise distance within a cluster
  - The square root of average mean squared distance between all pairs of points in the cluster

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

The CF Tree Structure in BIRCH

- Incremental insertion of new points (similar to B+-tree)
- For each point in the input
  - Find closest leaf entry
  - Add point to leaf entry and update CF
  - If entry diameter > max_diameter
    - split leaf, and possibly parents
- A CF tree has two parameters
  - Branching factor: Maximum number of children
  - Maximum diameter of sub-clusters stored at the leaf nodes



A CF tree: A height-balanced tree that stores the clustering features (CFs)

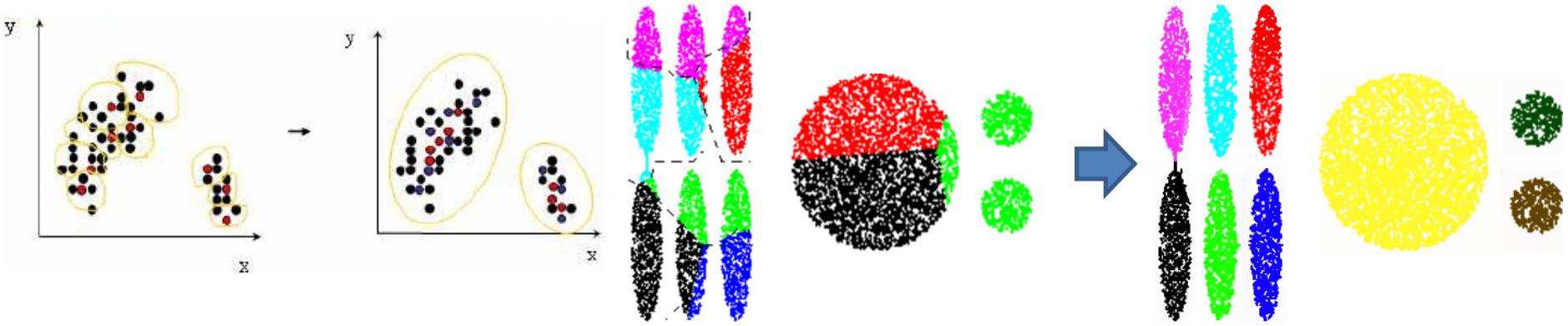
The non-leaf nodes store sums of the CFs of their children

BIRCH: A Scalable and Flexible Clustering Method

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

CURE: Clustering Using Representatives

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

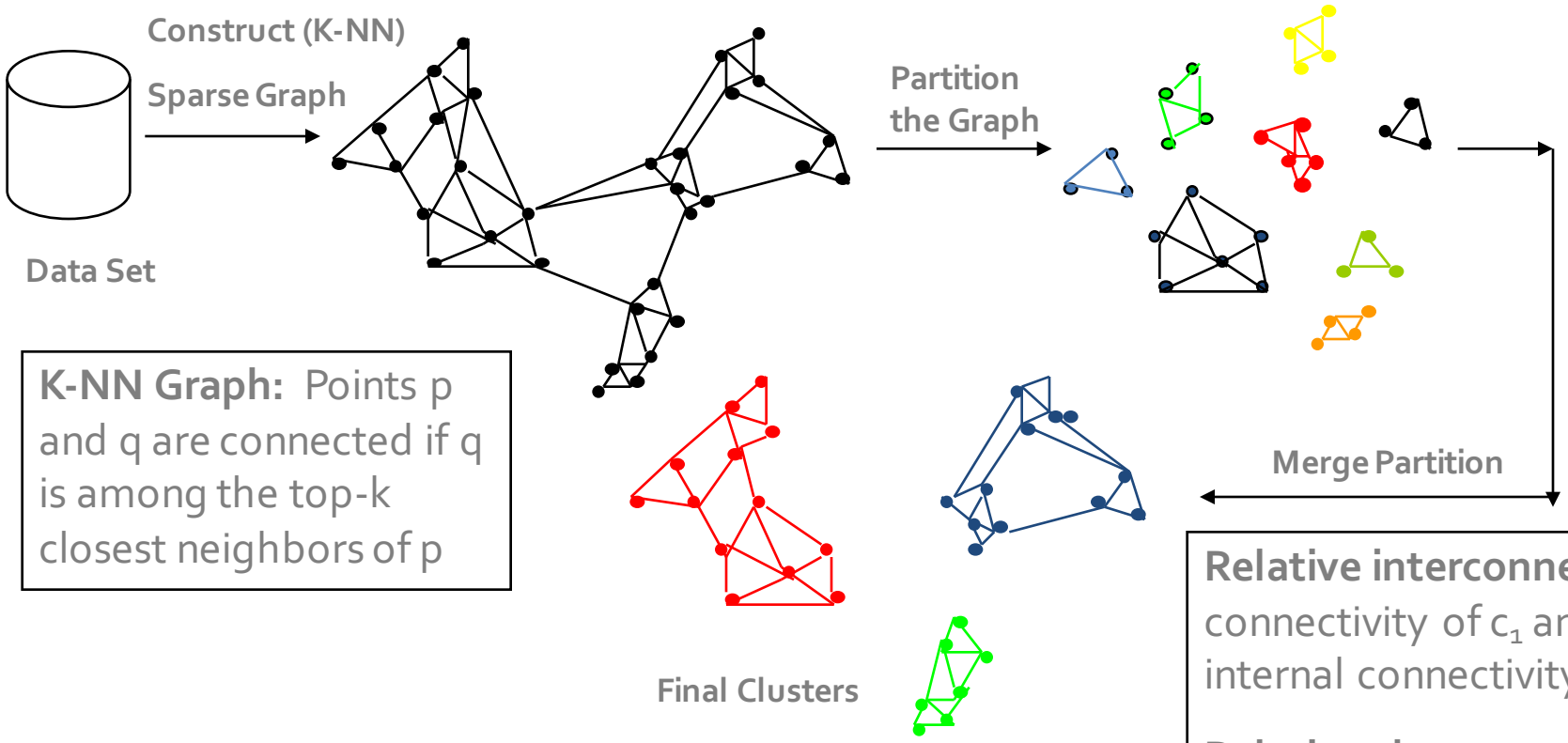


Courtesy: Kyuseok Shim@SNU.KR

CHAMELEON: Hierarchical Clustering Using Dynamic Modeling

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

Overall Framework of CHAMELEON



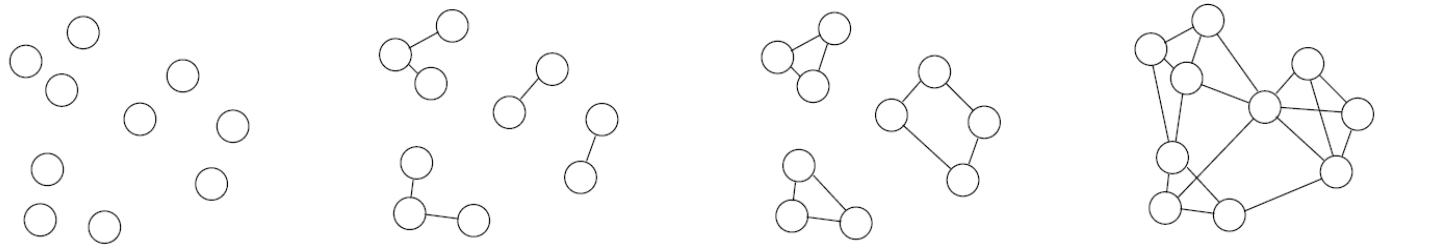
K-NN Graph: Points p and q are connected if q is among the top-k closest neighbors of p

Relative interconnectivity: connectivity of c_1 and c_2 over internal connectivity

Relative closeness: closeness of c_1 and c_2 over internal closeness

KNN Graphs and Interconnectivity

- K-nearest neighbor (KNN) graphs from an original data in 2D:



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

- $EC_{\{Ci,Cj\}}$: The absolute interconnectivity between C_i *and* C_j:
  - *The sum* of the weight of the edges that connect vertices in C_i *to vertices in* C_j
- Internal interconnectivity of a cluster C_i : *The size of its min-cut bisector* EC_{Ci} *(i.e., the weighted sum* of edges that partition the graph into two roughly equal parts)
- Relative Interconnectivity (RI): $$RI(C_i, C_j) = \frac{|EC_{\{C_i,C_j\}}|}{\frac{|EC_{C_i}|+|EC_{C_j}|}{2}}$$

Relative Closeness & Merge of Sub-Clusters

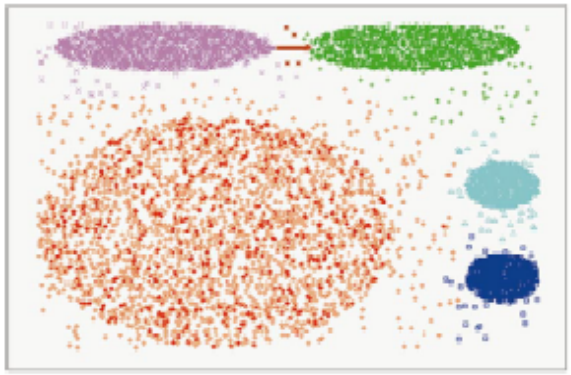
- **Relative closeness** between a pair of clusters C_i *and* C_j *: The absolute closeness* between C_i *and* C_j *normalized w.r.t. the internal closeness of the two clusters* C_i *and* C_j

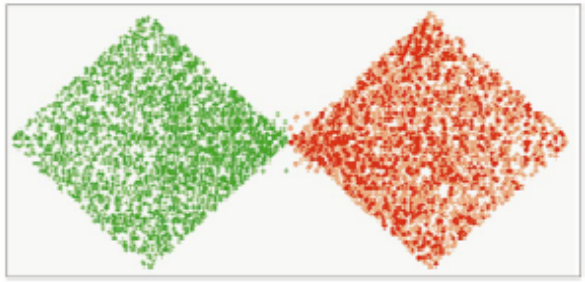
$$RC(C_i, C_j) = \frac{\overline{S}_{EC_{\{C_i, C_j\}}}}{\frac{|C_i|}{|C_i|+|C_j|}\overline{S}_{EC_{C_i}} + \frac{|C_j|}{|C_i|+|C_j|}\overline{S}_{EC_{C_j}}}$$

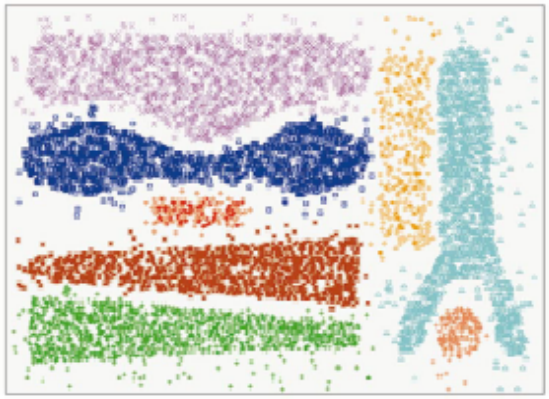
  - where $\overline{S}_{EC_{C_i}}$ and $\overline{S}_{EC_{C_j}}$ are the average weights of the edges that belong to the min-cut bisector of clusters C_i *and* C_j, respectively, and $\overline{S}_{EC_{\{C_i, C_j\}}}$ is the average weight of the edges that connect vertices in C_i *to vertices in* C_j

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

CHAMELEON: Clustering Complex Objects







CHAMELEON is capable to generate quality clusters at clustering complex objects

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 distribution 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:

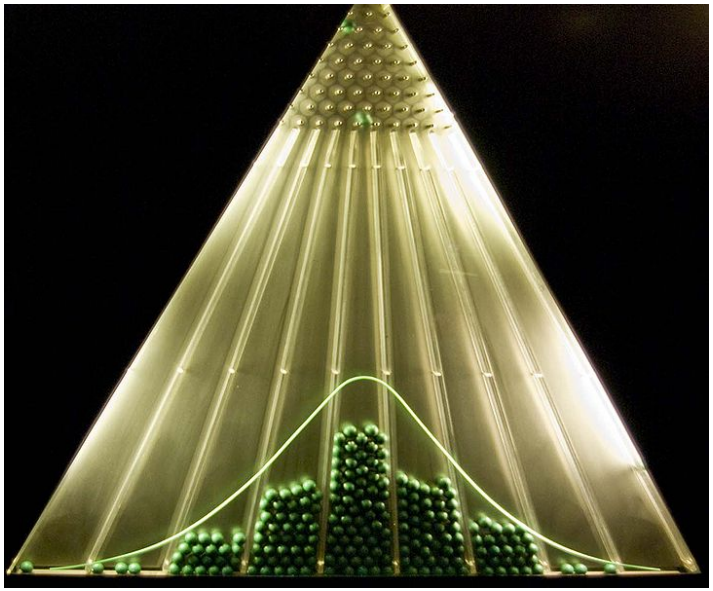
$$L(\mathcal{N}(\mu, \sigma^2) : X) = P(X|\mu, \sigma^2) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x_i-\mu)^2}{2\sigma^2}}$$

- The task of learning the generative model: find the parameters μ and σ² such that

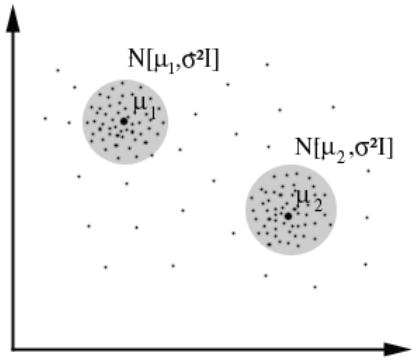
the maximum likelihood

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

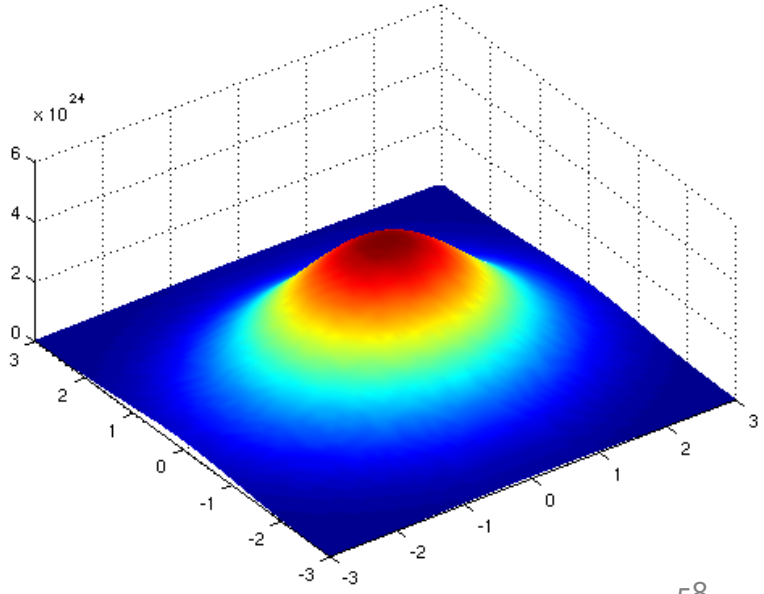
Gaussian Distribution

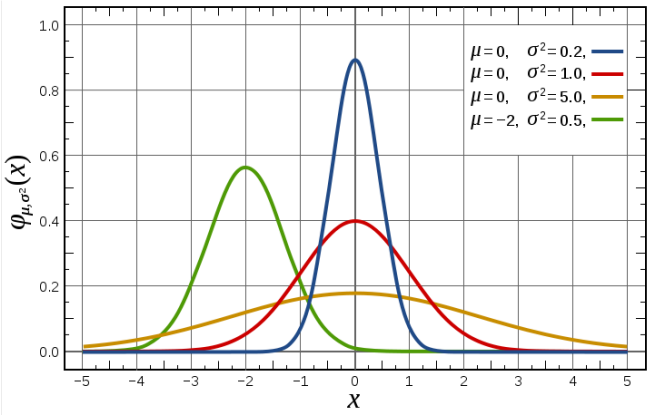


Bean machine: drop ball with pins



2-d Gaussian





1-d Gaussian

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,

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

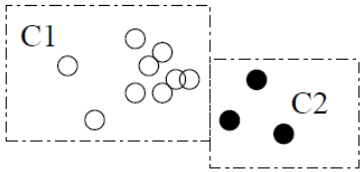
where $P()$ is the maximum likelihood

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

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

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

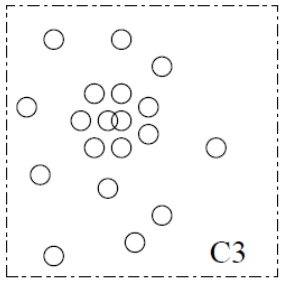
$$= \prod_{i=1}^{m} P(C_i)\left(\frac{P(C_{j_1} \cup C_{j_2})}{P(C_{j_1})P(C_{j_2})} - 1\right)$$

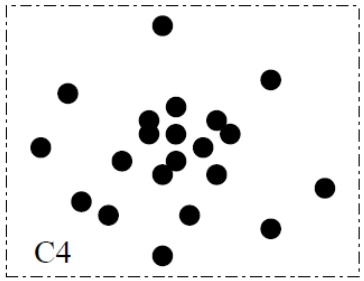


- Distance between clusters C_1 and C_2:

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

- If $dist(C_i, C_j) < 0$, merge C_i and C_j





Cluster Analysis: Basic Concepts and Methods

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

Density-Based and Grid-Based Clustering Methods

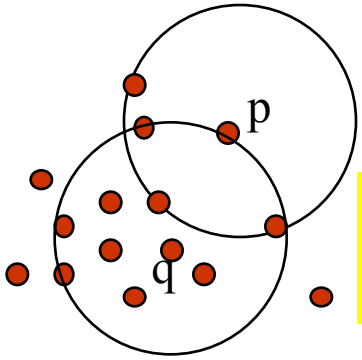
- Density-Based Clustering
  - Basic Concepts
  - **DBSCAN: A Density-Based Clustering Algorithm**
  - OPTICS: Ordering Points To Identify Clustering Structure
- Grid-Based Clustering Methods
  - Basic Concepts
  - STING: A Statistical Information Grid Approach
  - CLIQUE: Grid-Based Subspace Clustering

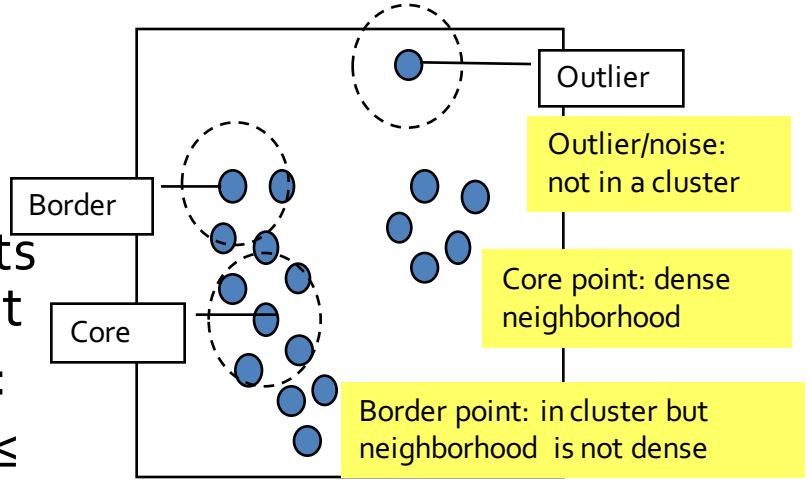
Density-Based Clustering Methods

- Clustering based on density (a local cluster criterion), such as density-connected points
- Major features:
  - Discover clusters of arbitrary shape
  - Handle noise
  - One scan (only examine the local region to justify density)
  - 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) (also, grid-based)

DBSCAN: A Density-Based Spatial Clustering Algorithm

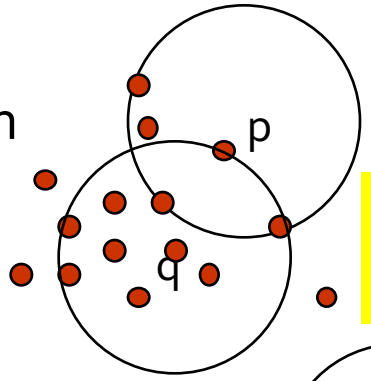
- DBSCAN (M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, KDD'96)
  - Discovers clusters of arbitrary shape: Density-Based Spatial Clustering of Applications with Noise
- A *density-based* notion of cluster
  - A ***cluster*** is defined as a **maximal** set of **density-connected** points
- Two parameters*:*
  - *Eps* (ε): Maximum radius of the neighborhood
  - *MinPts*: Minimum number of points in the Eps-neighborhood of a point
- The Eps(ε)-neighborhood of a point *q*:
  - $N_{Eps}(q)$: {p belongs to D | dist(p, q) ≤ Eps}

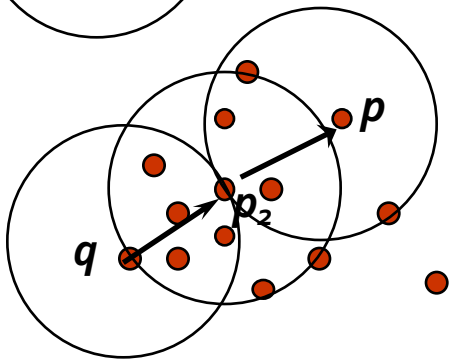


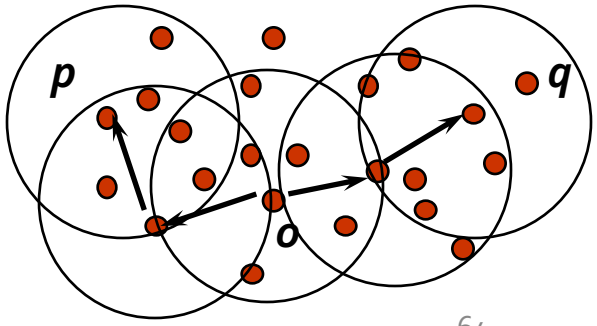


DBSCAN: Density-Reachable and Density-Connected

- **Directly density-reachable**:
  - A point *p* is directly density-reachable from a point *q* w.r.t. *Eps* (ε), *MinPts* if
    - *p* belongs to $N_{Eps}(q)$
    - **core point** condition: $|N_{Eps}(q)| \geq MinPts$
- **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*



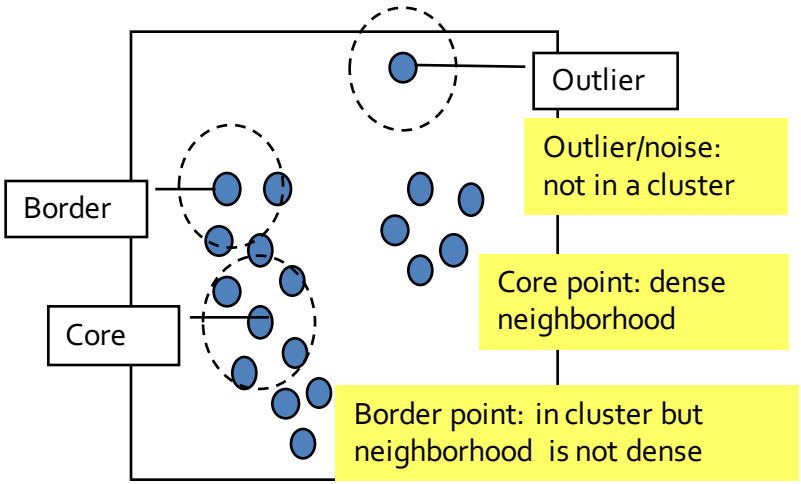




DBSCAN: The Algorithm

- **Algorithm**
  - Arbitrarily select a point *p*
  - Retrieve all points density-reachable

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



DBSCAN: The Algorithm

- **Computational complexity**
  - 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 $O(n^2)$

https://en.wikipedia.org/wiki/DBSCAN

DBSCAN visits each point of the database, possibly multiple times (e.g., as candidates to different clusters). For practical considerations, however, the time complexity is mostly governed by the number of regionQuery invocations. DBSCAN executes exactly one such query for each point, and if an indexing structure is used that executes a neighborhood query in $O(\log n)$, an overall average runtime complexity of $O(n \log n)$ is obtained (if parameter ε is chosen in a meaningful way, i.e. such that on average only $O(\log n)$ points are returned). Without the use of an accelerating index structure, or on degenerated data (e.g. all points within a distance less than ε), the worst case run time complexity remains $O(n^2)$. The distance matrix of size $(n^2\text{-}n)/2$ can be materialized to avoid distance recomputations, but this needs $O(n^2)$ memory, whereas a non-matrix based implementation of DBSCAN only needs $O(n)$ memory.

DBSCAN Is Sensitive to the Setting of Parameters

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

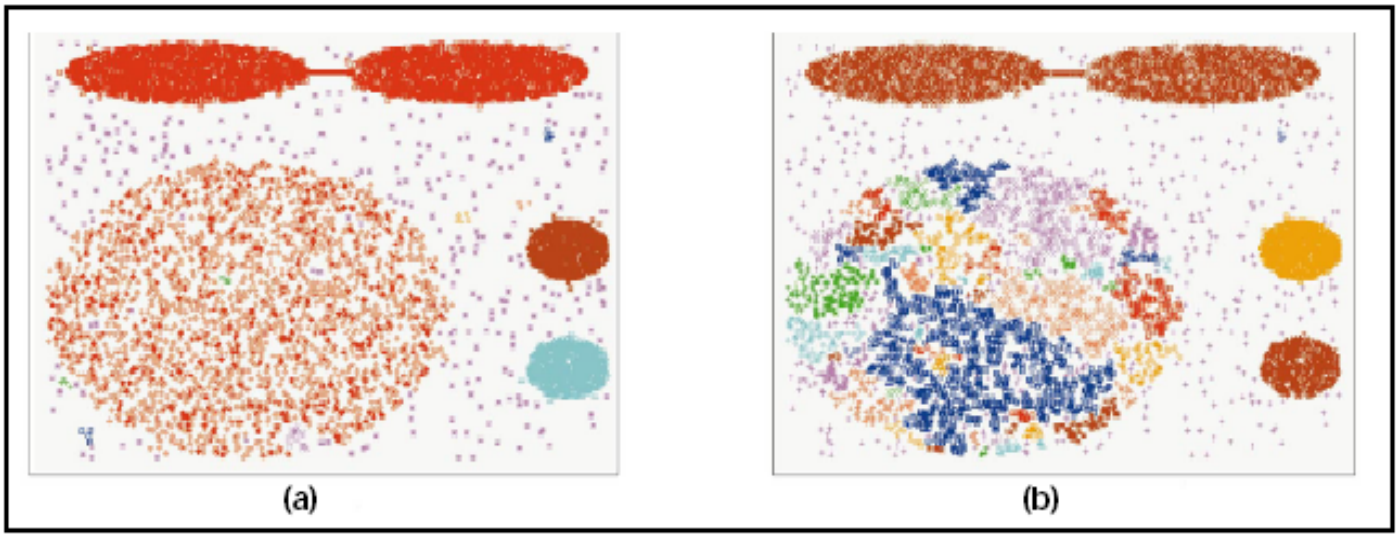
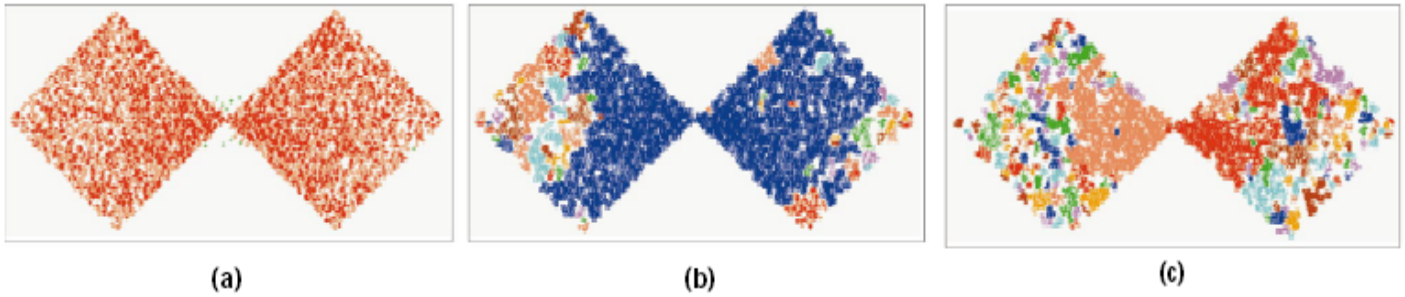


Figure 9. DBScan results for DS2 with MinPts at 4 and Eps at (a) 5.0, (b) 3.5, and (c) 3.0.

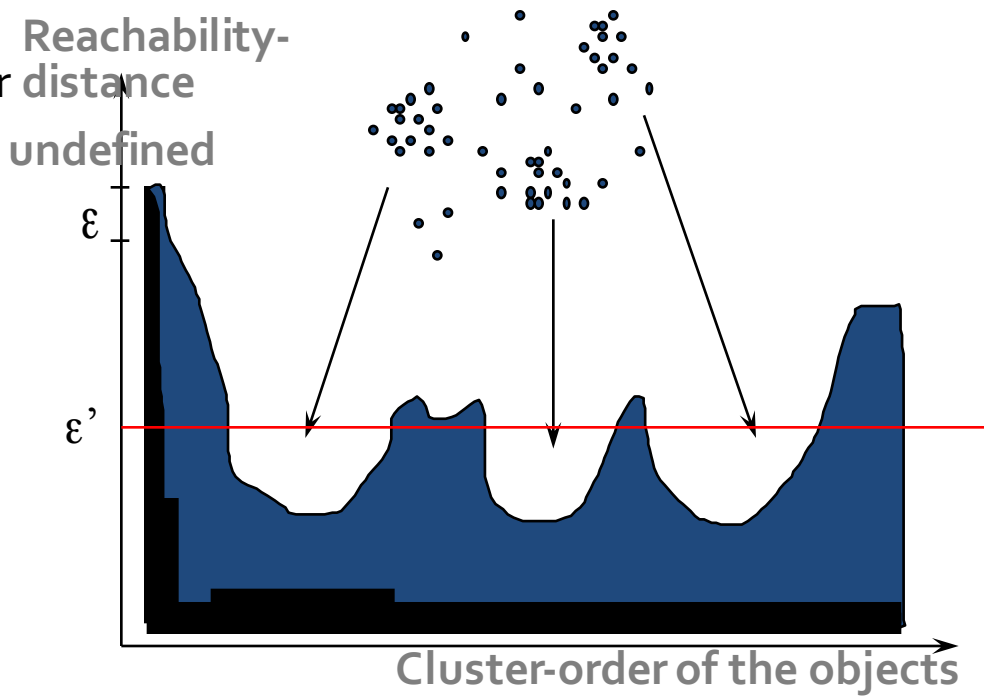


Ack. Figures from G. Karypis, E.-H. Han, and V. Kumar, *COMPUTER*, 32(8), 1999

OPTICS: Ordering Points To Identify Clustering Structure

- OPTICS (Ankerst, Breunig, Kriegel, and Sander, SIGMOD'99)
  - DBSCAN is sensitive to parameter setting
  - An extension: finding clustering structure
- Observation: Given a *MinPts*, density-based clusters w.r.t. a higher density are completely contained in clusters w.r.t. to a lower density
- Idea: Higher density points should be processed first—find high-density clusters first
- OPTICS stores such a clustering order using two pieces of information:
  - *Core distance* and *reachability distance*

Reachability plot for a dataset



Since points belonging to a cluster have a low reachability distance to their nearest neighbor, valleys correspond to clusters

The deeper the valley, the denser the cluster

OPTICS: An 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

  Core-distance$_{\varepsilon, MinPts}(p)$ = Undefined if card($N_\varepsilon(p)$) < *MinPts*

  MinPts-distance(p), otherwise

- **Reachability distance** of object *p* from core object *q* is the min. radius value that makes *p* density-reachable from *q*

  Reachability-distance$_{\varepsilon, MinPts}(p, q)$ =

  Undefined, *if q is not a core object*

  max(core-distance(*q*), distance (*q*, *p*)), *otherwise*

- Complexity: O(*N* log*N*) (if index-based)

  where *N*: # of points

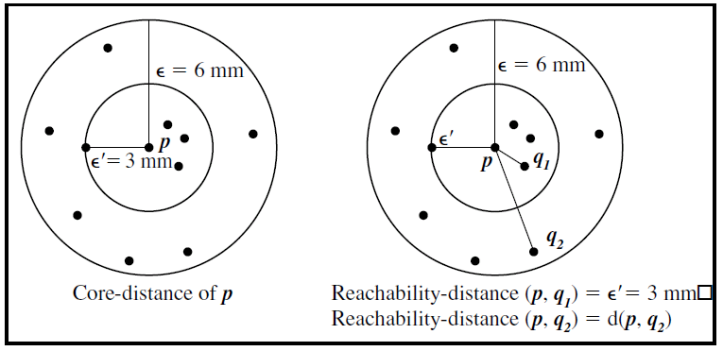
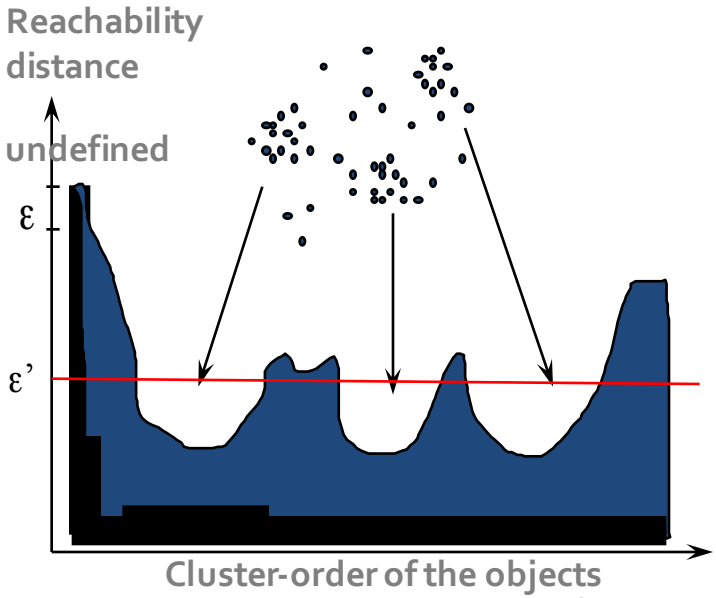
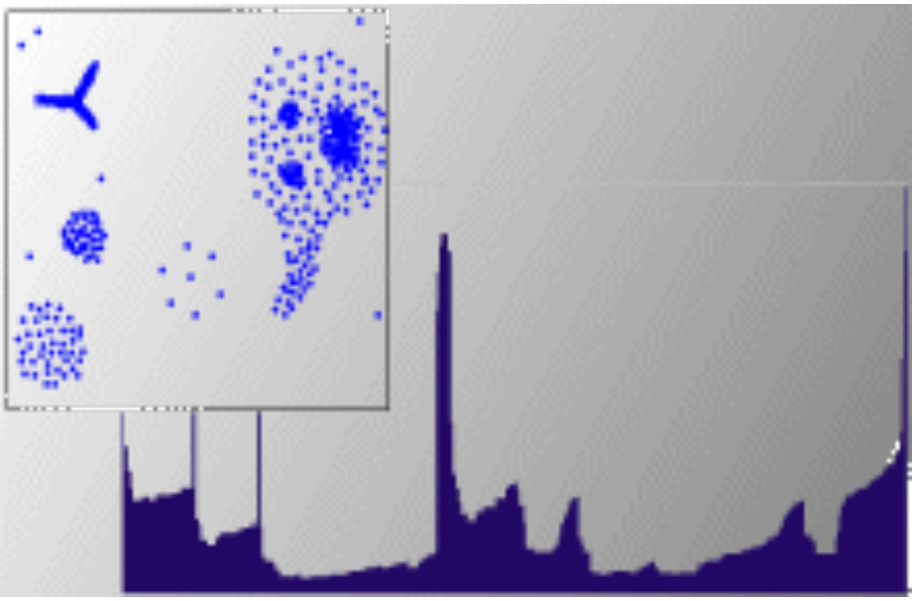


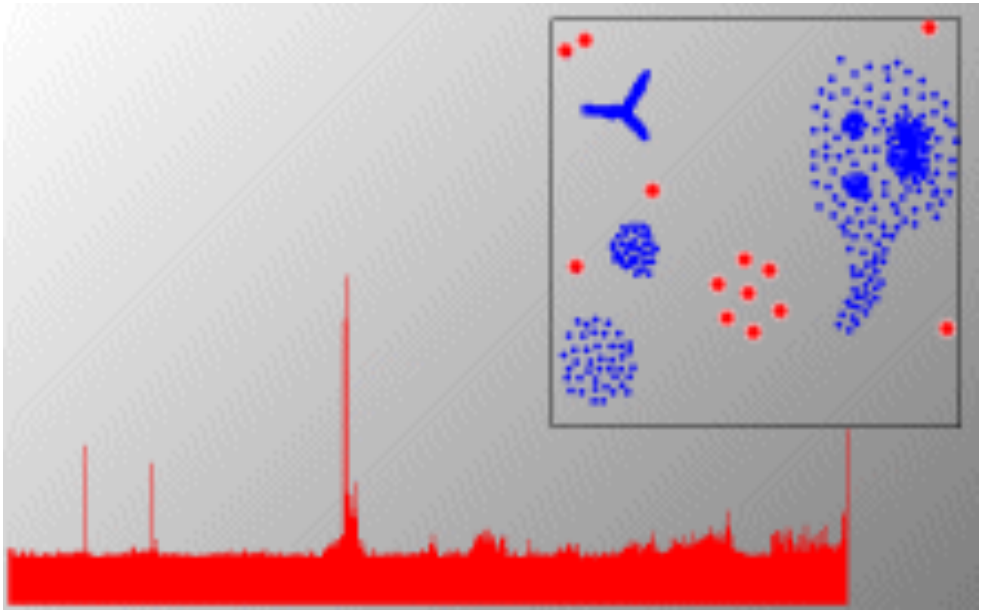
Figure 10.16: OPTICS terminology. Based on [ABKS99].



OPTICS: Finding Hierarchically Nested Clustering Structures

- OPTICS produces a special cluster-ordering of the data points with respect to its density-based clustering structure
  - The 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—finding intrinsic, even hierarchically nested clustering structures





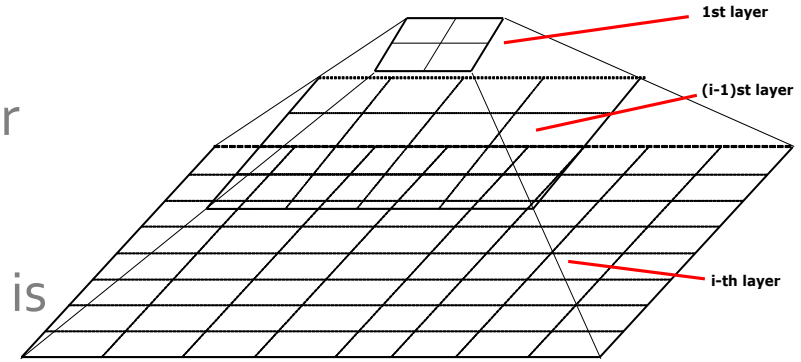
Finding nested clustering structures with different parameter settings

Grid-Based Clustering Methods

- 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 << # 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
- Methods to be introduced
  - **STING** (a STatistical INformation Grid approach) (Wang, Yang and Muntz, VLDB'97)
  - **CLIQUE** (Agrawal, Gehrke, Gunopulos, and Raghavan, SIGMOD'98)
    - Both grid-based and subspace clustering

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
- Statistical information of each cell is calculated and stored beforehand and is used to answer queries
- Parameters of higher level cells can be easily calculated from that of lower level cell, including
  - *count*, *mean*, *s*(standard deviation), *min*, *max*
  - type of distribution—*normal*, *uniform*, etc.



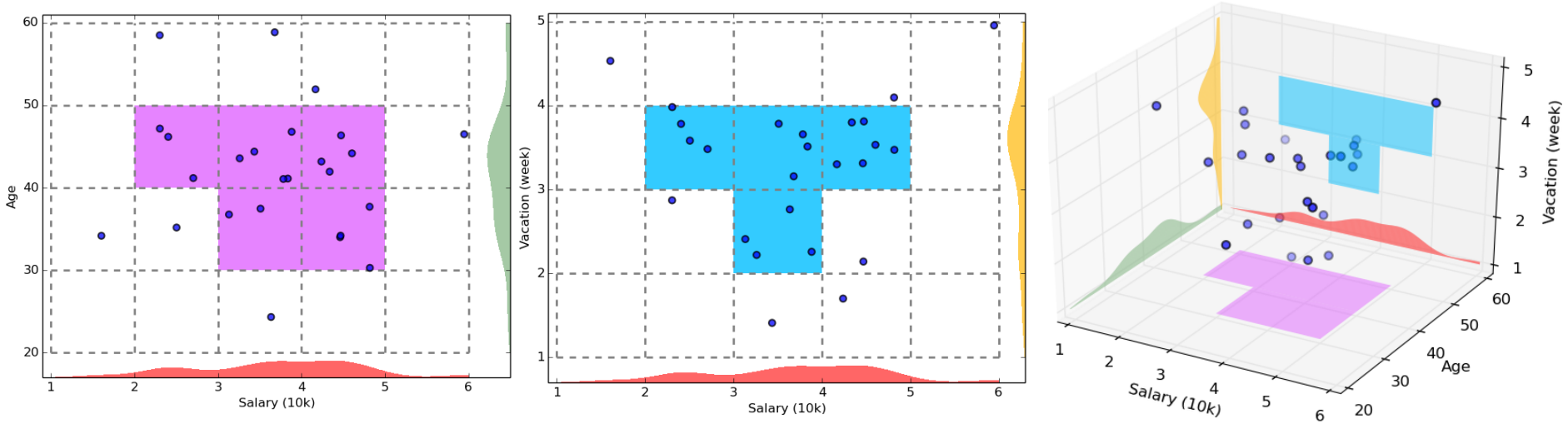
Query Processing in STING and Its Analysis

- To process a region query
  - Start at the root and proceed to the next lower level, using the STING index
  - Calculate the likelihood that a cell is relevant to the query at some confidence level using the statistical information of the cell
  - Only children of likely relevant cells are recursively explored
  - Repeat this process until the bottom layer is reached
- Advantages
  - Query-independent, easy to parallelize, incremental update
  - Efficiency: Complexity is O(K)
    - K: # of grid cells at the lowest level, and K << N (i.e., # of data points)
- Disadvantages
  - 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

Additional 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

Cluster Analysis: Basic Concepts and Methods

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

Clustering Validation

- Clustering Validation: Basic Concepts
- Clustering Evaluation: Measuring Clustering Quality
- External Measures for Clustering Validation
  - I: Matching-Based Measures
  - II: Entropy-Based Measures
  - III: Pairwise Measures
- Internal Measures for Clustering Validation
- Relative Measures
- Cluster Stability
- Clustering Tendency

Clustering Validation and Assessment

- Major issues on clustering validation and assessment
  - Clustering evaluation
    - Evaluating the goodness of the clustering
  - Clustering stability
    - To understand the sensitivity of the clustering result to various algorithm parameters, e.g., # of clusters
  - Clustering tendency
    - Assess the suitability of clustering, i.e., whether the data has any inherent grouping structure

Measuring Clustering Quality

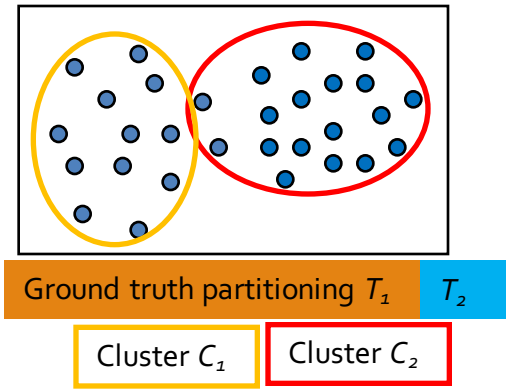
- **Clustering Evaluation**: Evaluating the goodness of clustering results
  - No commonly recognized best suitable measure in practice
- **Three categorization of measures**: External, internal, and relative
  - **External**: Supervised, employ criteria not inherent to the dataset
    - Compare a clustering 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: External Methods

- Given the **ground truth** *T*, *Q*(*C*, *T*) is the **quality measure** for a clustering *C*
- *Q*(*C*, *T*) is good if it satisfies the following **four** essential criteria
  - **Cluster homogeneity**
    - The purer, the better
  - **Cluster completeness**
    - Assign objects belonging to the same category in the ground truth to the same cluster
  - **Rag bag better than alien**
    - 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

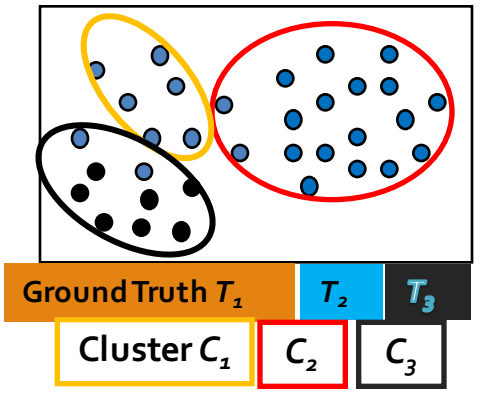
Commonly Used External Measures

- **Matching-based measures**
  - Purity, maximum matching, F-measure
- **Entropy-Based Measures**
  - Conditional entropy
  - Normalized mutual information (NMI)
  - Variation of information
- **Pairwise measures**
  - Four possibilities: True positive (TP), FN, FP, TN
  - Jaccard coefficient, Rand statistic, Fowlkes-Mallow measure
- **Correlation measures**
  - Discretized Huber static, normalized discretized Huber static



Matching-Based Measures (I): Purity vs. Maximum Matching

- **Purity**: Quantifies the extent that cluster C_i contains points only from one (ground truth) partition: $purity_i = \frac{1}{n_i} \max_{j=1}^{k} \{n_{ij}\}$
  - Total purity of clustering C: $purity = \sum_{i=1}^{r} \frac{n_i}{n} purity_i = \frac{1}{n} \sum_{i=1}^{r} \max_{j=1}^{k} \{n_{ij}\}$
  - Perfect clustering if purity = 1 and $r = k$ (the number of clusters obtained is the same as that in the ground truth)
  - Ex. 1 (green or orange): $purity_1$ = 30/50; $purity_2$ = 20/25; $purity_3$ = 25/25; $purity$ = (30 + 20 + 25)/100 = 0.75
  - Two clusters may share the same majority partition
- **Maximum matching**: Only one cluster can match one partition
  - Match: Pairwise matching, weight $w(e_{ij}) = n_{ij}$
  - Maximum weight matching: $w(M) = \sum_{e \in M} w(e)$, $match = \arg\max_{M} \{\frac{w(M)}{n}\}$
  - Ex2. (green) *match* = *purity* = 0.75; (orange) *match* = 0.65 > 0.6

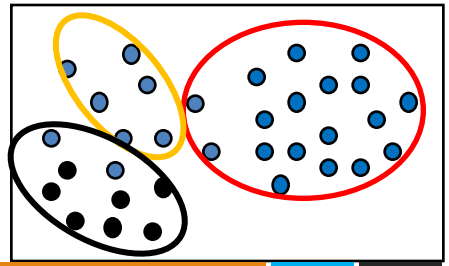


C\T	T_1	T_2	T_3	Sum
C_1	0	20	30	50
C_2	0	20	5	25
C_3	25	0	0	25
m_j	25	40	35	100

C\T	T_1	T_2	T_3	Sum
C_1	0	30	20	50
C_2	0	20	5	25
C_3	25	0	0	25
m_j	25	50	25	100

Matching-Based Measures (II): F-Measure

- **Precision**: The fraction of points in C_i from the majority partition T_{j_i} (i.e., the same as purity), where j_i is the partition that contains the maximum # of points from C_i
  - Ex. For the green table
    - $prec_1 = 30/50$; $prec_2 = 20/25$; $prec_3 = 25/25$
- **Recall**: The fraction of point in partition shared in common with cluster C_i, where $m_{j_i} = |T_{j_i}|$
  - Ex. For the green table
    - $recall_1 = 30/35$; $recall_2 = 20/40$; $recall_3 = 25/25$
- **F-measure** for C_i: The harmonic means of $prec_i$ and $recall_i$: $F_i = \frac{2n_{ij_i}}{n_i + m_{j_i}}$
- F-measure for clustering C: average of all clusters: $F = \frac{1}{r}\sum_{i=1}^{r} F_i$
  - Ex. For the green table
    - $F_1 = 60/85$; $F_2 = 40/65$; $F_3 = 1$; $F = 0.774$



C\T	T_1	T_2	T_3	Sum
C_1	0	20	30	50
C_2	0	20	5	25
C_3	25	0	0	25
m_j	25	40	35	100

$$prec_i = \frac{1}{n_i}\max_{j=1}^{k}\{n_{ij}\} = \frac{n_{ij_i}}{n_i}$$

$$recall_i = \frac{n_{ij_i}}{|T_{j_i}|} = \frac{n_{ij_i}}{m_{j_i}}$$

Entropy-Based Measures (I): Conditional Entropy

- **Entropy of clustering C:** $H(\mathcal{C}) = -\sum_{i=1}^{r} p_{C_i} \log p_{C_i}$ $p_{C_i} = \frac{n_i}{n}$ (i.e., the probability of cluster C_i)
- **Entropy of partitioning T:** $H(\mathcal{T}) = -\sum_{j=1}^{k} p_{T_i} \log p_{T_j}$
- **Entropy of T with respect to cluster C_i:** $H(\mathcal{T}|C_i) = -\sum_{j=1}^{k} (\frac{n_{ij}}{n_i}) \log(\frac{n_{ij}}{n_i})$
- **Conditional entropy of T with respect to clustering C:**

$$H(\mathcal{T}|\mathcal{C}) = -\sum_{i=1}^{r} (\frac{n_i}{n}) H(\mathcal{T}|C_i) = -\sum_{i=1}^{r}\sum_{j=1}^{k} p_{ij} \log(\frac{p_{ij}}{p_{C_i}})$$

  - The more a cluster's members are split into different partitions, the higher the conditional entropy
  - For a perfect clustering, the conditional entropy value is o, where the worst possible conditional entropy value is *log k*

$$H(\mathcal{T}|\mathcal{C}) = -\sum_{i=1}^{r}\sum_{j=1}^{k} p_{ij}(\log p_{ij} - \log p_{C_i}) = -\sum_{i=1}^{r}\sum_{j=1}^{k} p_{ij} \log p_{ij} + \sum_{i=1}^{r}(\log p_{C_i} \sum_{j=1}^{k} p_{ij})$$

$$= -\sum_{i=1}^{r}\sum_{j=1}^{k} p_{ij} \log p_{ij} + \sum_{i=1}^{r}(p_{C_i} \log p_{C_i}) = H(\mathcal{C}, \mathcal{T}) - H(\mathcal{C})$$

Entropy-Based Measures (II): Normalized Mutual Information (NMI)

- **Mutual information**:
  - Quantifies the amount of shared info between the clustering C and partitioning T $I(C,T) = \sum_{i=1}^{r}\sum_{j=1}^{k} p_{ij} \log(\frac{p_{ij}}{p_{C_i} \cdot p_{T_j}})$
  - Measures the dependency between the observed joint probability p_{ij} of C and T, and the expected joint probability $p_{Ci} \cdot p_{Tj}$ under the independence assumption
  - When C and T are independent, $p_{ij} = p_{Ci} \cdot p_{Tj}$, $I(C, T) = 0$. However, there is no upper bound on the mutual information
- **Normalized mutual information** (NMI)

$$NMI(\mathcal{C},\mathcal{T}) = \sqrt{\frac{I(\mathcal{C},\mathcal{T})}{H(\mathcal{C})} \cdot \frac{I(\mathcal{C},\mathcal{T})}{H(\mathcal{T})}} = \frac{I(\mathcal{C},\mathcal{T})}{\sqrt{H(\mathcal{C}) \cdot H(\mathcal{T})}}$$

  - Value range of NMI: [0,1]. Value close to 1 indicates a good clustering

Pairwise Measures: Four Possibilities for Truth Assignment

- **Four possibilities** based on the agreement between cluster label and partition label
  - *TP*: true positive—Two points $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ belong to the same partition T, and they also in the same cluster C

$$TP = |\{(\mathbf{x}_i, \mathbf{x}_j) : y_i = y_j \text{ and } \hat{y}_i = \hat{y}_j\}|$$

where y_i: the true partition label , and $\hat{y}_i$: the cluster label for point $\boldsymbol{x}_i$

  - *FN*: false negative: $FN = |\{(\mathbf{x}_i, \mathbf{x}_j) : y_i = y_j \text{ and } \hat{y}_i \neq \hat{y}_j\}|$
  - *FP: false positive* $FP = |\{(\mathbf{x}_i, \mathbf{x}_j) : y_i \neq y_j \text{ and } \hat{y}_i = \hat{y}_j\}|$
  - *TN*: true negative $TN = |\{(\mathbf{x}_i, \mathbf{x}_j) : y_i \neq y_j \text{ and } \hat{y}_i \neq \hat{y}_j\}|$
- Calculate the four measures:

$$TP = \sum_{i=1}^{r}\sum_{j=1}^{k}\binom{n_{ij}}{2} = \frac{1}{2}\left(\left(\sum_{i=1}^{r}\sum_{j=1}^{k} n_{ij}^{\,2}\right) - n\right) \qquad FN = \sum_{j=1}^{k}\binom{m_j}{2} - TP \qquad N = \binom{n}{2}$$

Total # of pairs of points

$$FP = \sum_{i=1}^{r}\binom{n_i}{2} - TP \qquad TN = N - (TP + FN + FP) = \frac{1}{2}\left(n^2 - \sum_{i=1}^{r} n_i^{\,2} - \sum_{j=1}^{k} m_j^{\,2} + \sum_{i=1}^{r}\sum_{j=1}^{k} n_{ij}^{\,2}\right)$$

Pairwise Measures: Jaccard Coefficient and Rand Statistic

- Jaccard coefficient: Fraction of true positive point pairs, but after ignoring the true negatives (thus asymmetric)
  - Jaccard = TP/(TP + FN + FP) [i.e., denominator ignores TN]
  - Perfect clustering: Jaccard = 1
- Rand Statistic:
  - Rand = (TP + TN)/N
  - Symmetric; perfect clustering: Rand = 1
- Fowlkes-Mallow Measure:
  - Geometric mean of precision and recall

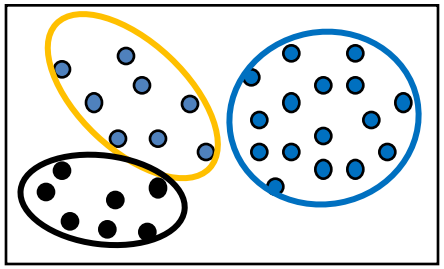
$$FM = \sqrt{prec \times recall} = \frac{TP}{\sqrt{(TP + FN)(TP + FP)}}$$

$C \backslash T$	T_1	T_2	T_3	Sum
C_1	0	20	30	50
C_2	0	20	5	25
C_3	25	0	0	25
m_j	25	40	35	100

- Using the above formulas, one can calculate all the measures for the green table (leave as an exercise)

Internal Measures (I): BetaCV Measure

- A trade-off in maximizing intra-cluster compactness and inter-cluster separation
- Given a clustering $C = \{C_1, \ldots, C_k\}$ with k clusters, cluster C_i containing $n_i = |C_i|$ points
  - Let $W(S, R)$ be sum of weights on all edges with one vertex in S and the other in R
  - The sum of all the intra-cluster weights over all clusters: $W_{in} = \frac{1}{2}\sum_{i=1}^{k} W(C_i, C_i)$
  - The sum of all the inter-cluster weights: $W_{out} = \frac{1}{2}\sum_{i=1}^{k} W(C_i, \overline{C_i}) = \sum_{i=1}^{k-1}\sum_{j>i} W(C_i, C_j)$
  - The number of distinct intra-cluster edges: $N_{in} = \sum_{i=1}^{k} \binom{n_i}{2}$
  - The number of distinct inter-cluster edges: $N_{out} = \sum_{i=1}^{k-1}\sum_{j=i+1}^{k} n_i n_j$



- **Beta-CV measure**:
  - The ratio of the mean intra-cluster distance to the mean inter-cluster distance
  - The smaller, the better the clustering

$$BetaCV = \frac{W_{in} / N_{in}}{W_{out} / N_{out}}$$

Internal Measures (II): Normalized Cut and Modularity

- **Normalized cut**: $$NC = \sum_{i=1}^{k} \frac{W(C_i, \overline{C_i})}{vol(C_i)} = \sum_{i=1}^{k} \frac{W(C_i, \overline{C_i})}{W(C_i, V)} = \sum_{i=1}^{k} \frac{W(C_i, \overline{C_i})}{W(C_i, C_i) + W(C_i, \overline{C_i})} = \sum_{i=1}^{k} \frac{1}{\frac{W(C_i, C_i)}{W(C_i, \overline{C_i})} + 1}$$

  where $vol(C_i) = W(C_i, V)$ is the volume of cluster C_i

  - The higher normalized cut value, the better the clustering
- **Modularity** (for graph clustering)
  - Modularity Q is defined as $$Q = \sum_{i=1}^{k} \left(\frac{W(C_i, C_i)}{W(V, V)} - \left(\frac{W(C_i, V)}{W(V, V)} \right)^2 \right)$$

    where $$W(V, V) = \sum_{i=1}^{k} W(C_i, V) = \sum_{i=1}^{k} W(C_i, C_i) + \sum_{i=1}^{k} W(C_i, \overline{C_i}) = 2(W_{in} + W_{out})$$
  - Modularity measures the difference between the observed and expected fraction of weights on edges within the clusters.
  - The smaller the value, the better the clustering—the intra-cluster distances are lower than expected

Relative Measure

- Relative measure: Directly compare different clusterings, usually those obtained via different parameter settings for the same algorithm
- **Silhouette coefficient** as an **internal measure**: Check cluster cohesion and separation
  - For each point $\boldsymbol{x}_i$, its silhouette coefficient s_i is: $s_i = \frac{\mu_{out}^{\min}(\mathbf{x}_i) - \mu_{in}(\mathbf{x}_i)}{\max\{\mu_{out}^{\min}(\mathbf{x}_i), \mu_{in}(\mathbf{x}_i)\}}$
    where $\mu_{in}(\mathbf{x}_i)$ is the mean distance from $\boldsymbol{x}_i$ to points in its own cluster
    $\mu_{out}^{\min}(\mathbf{x}_i)$ is the mean distance from $\boldsymbol{x}_i$ to points in its closest cluster
  - Silhouette coefficient (*SC*) is the mean values of s_i across all the points: $SC = \frac{1}{n}\sum_{i=1}^{n} s_i$
  - *SC* close to +1 implies good clustering
    - Points are close to their own clusters but far from other clusters
- **Silhouette coefficient** as a **relative measure**: Estimate the # of clusters in the data

$$SC_i = \frac{1}{n_i}\sum_{x_j \in C_i} s_j$$

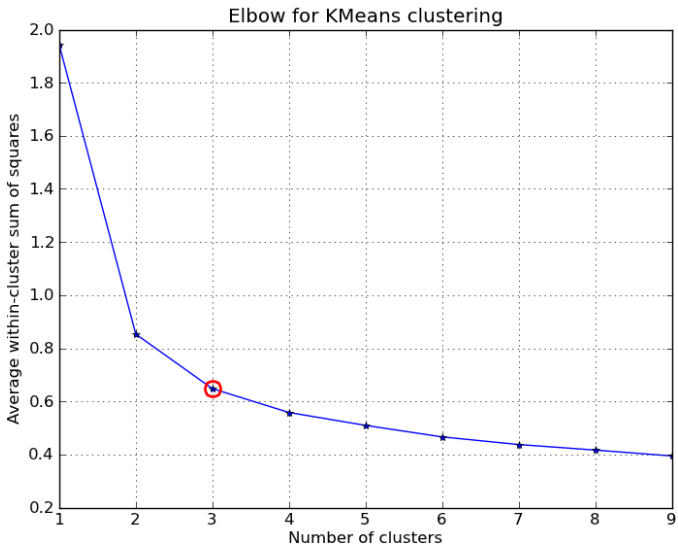
Pick the *k* value that yields the best clustering, i.e., yielding high values for *SC* and SC_i ($1 \le i \le k$)

Cluster Stability

- Clusterings obtained from several datasets sampled from the same underlying distribution as ***D*** should be similar or "stable"
- Typical approach:
  - Find good parameter values for a given clustering algorithm
- Example: Find a good value of *k*, the correct number of clusters
- A **bootstrapping approach** to find the best value of *k* (judged on stability)
  - Generate *t* samples of size *n* by sampling from ***D*** with replacement
  - For each sample $\boldsymbol{D}_i$, run the same clustering algorithm with *k* values from 2 to k_{max}
  - Compare the distance between all pairs of clusterings $C_k(\boldsymbol{D}_i)$ and $C_k(\boldsymbol{D}_j)$ via some distance function
    - Compute the expected pairwise distance for each value of *k*
  - The value *k** that exhibits the least deviation between the clusterings obtained from the resampled datasets is the best choice for *k* since it exhibits the most stability

Other Methods for Finding K, the Number of Clusters

- **Empirical method**
  - # of clusters: $k \approx \sqrt{n/2}$ for a dataset of n points (e.g., $n = 200$, $k = 10$)
- **Elbow method**: Use the turning point in the curve of the 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
    - For example, 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

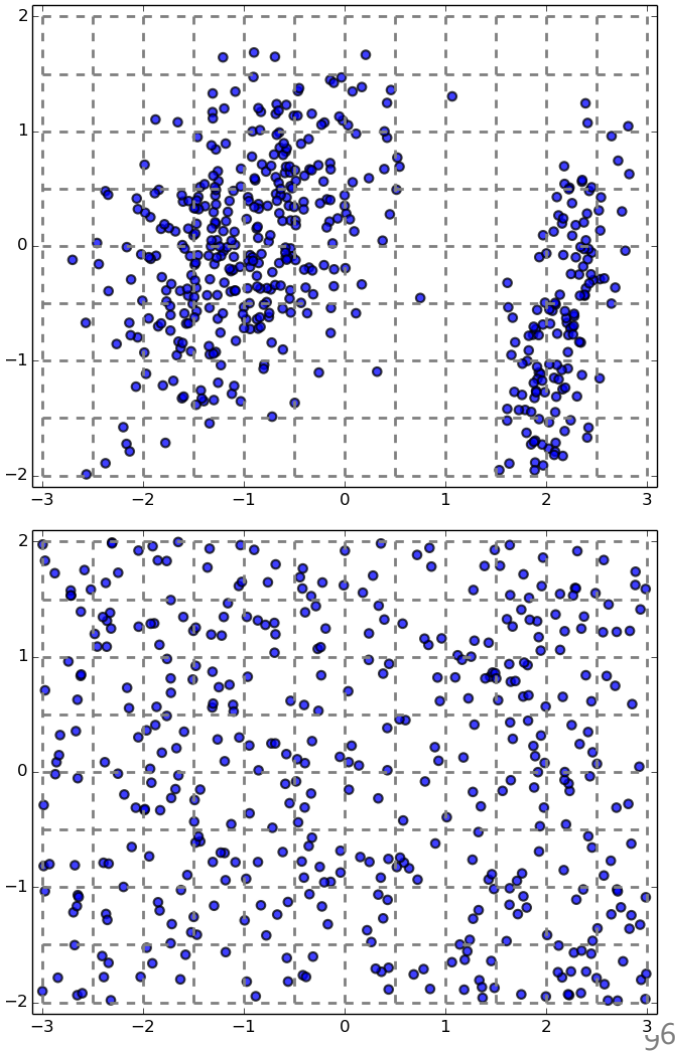


Clustering Tendency: Whether the Data Contains Inherent Grouping Structure

- Assessing the suitability of clustering
  - (i.e., whether the data has any inherent grouping structure)
- Determining clustering tendency or clusterability
  - A hard task because there are so many different definitions of clusters
    - E.g., partitioning, hierarchical, density-based, graph-based, etc.
  - Even fixing cluster type, still hard to define an appropriate null model for a data set
- Still, there are some clusterability assessment methods, such as
  - Spatial histogram: Contrast the histogram of the data with that generated from random samples
  - Distance distribution: Compare the pairwise point distance from the data with those from the randomly generated samples
  - Hopkins Statistic: A sparse sampling test for spatial randomness

Testing Clustering Tendency: A Spatial Histogram Approach

- **Spatial Histogram Approach:** Contrast the *d*-dimensional histogram of the input dataset ***D*** with the histogram generated from random samples
  - Dataset D is clusterable if the distributions of two histograms are rather different
- Method outline
  - Divide each dimension into equi-width bins, count how many points lie in each cells, and obtain the empirical joint probability mass function (EPMF)
- Do the same for the randomly sampled data
- Compute how much they differ using the *Kullback-Leibler* (*KL*) *divergence* value



Summary

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

References: (I) Cluster Analysis: An Introduction

- 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

References: (II) Partitioning Methods

- 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
- 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

References: (III) Hierarchical Methods

- 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.
- 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

References: (IV) Density- and Grid-Based Methods

- 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: (IV) Evaluation of Clustering

- 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