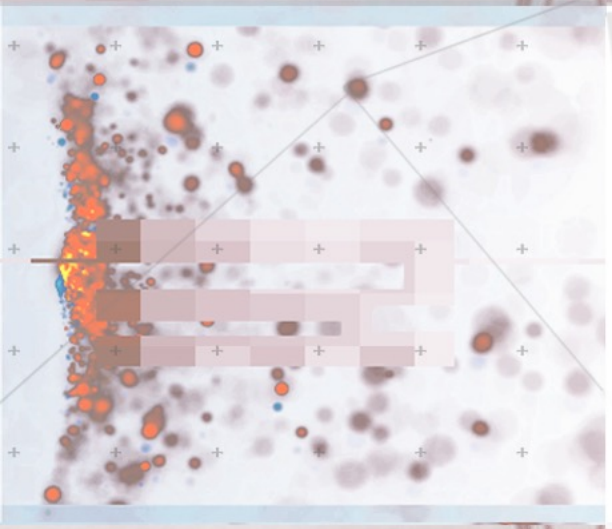
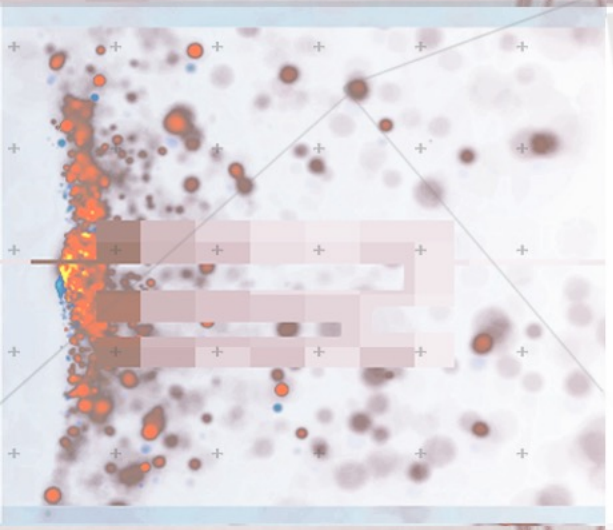


Partitioning-Based Clustering Methods

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



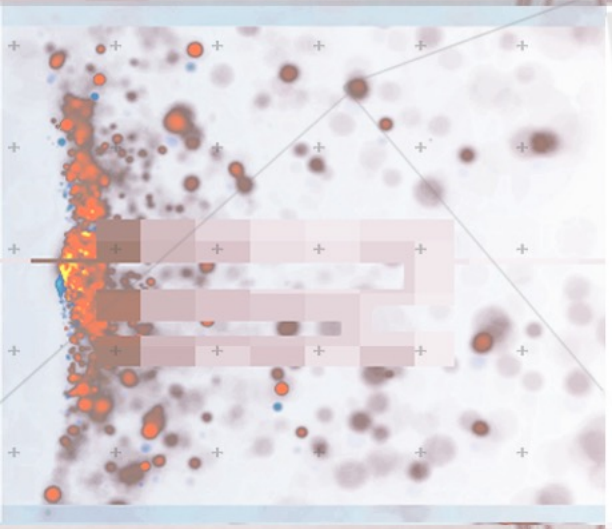
Session 1: Basic Concepts of Partitioning Algorithms

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.

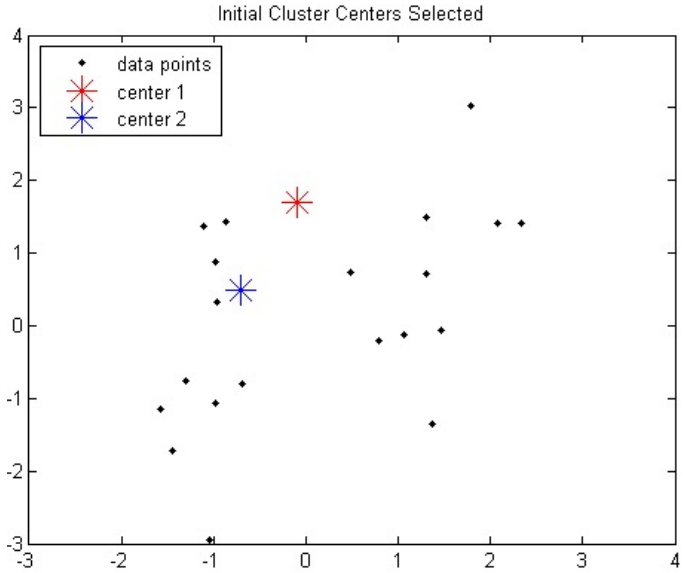


Session 2: The *K-Means* Clustering Method

The *K-Means* Clustering Method

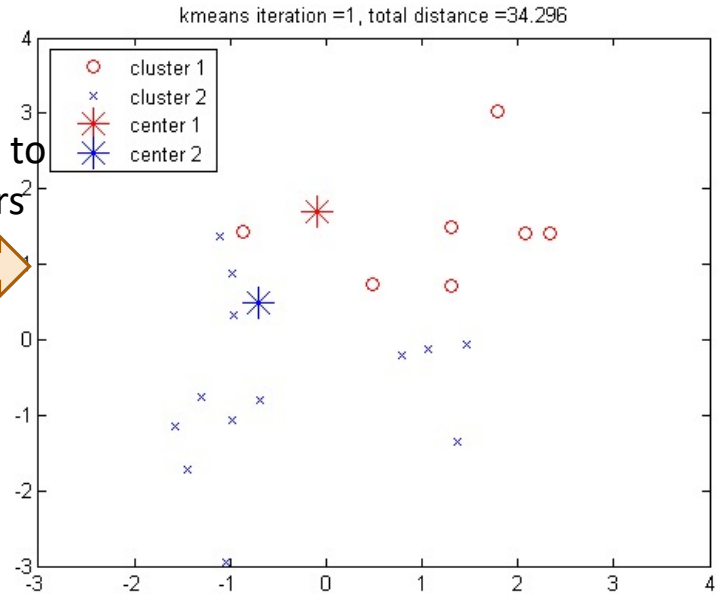
- *K-Means* (MacQueen'67, Lloyd'57/'82)
  - Each cluster is represented by the center of the cluster
- Given K, the number of clusters, the *K-Means* clustering algorithm is outlined as follows
  - Select *K* points as initial centroids
  - **Repeat**
    - Form *K* clusters by assigning each point to 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

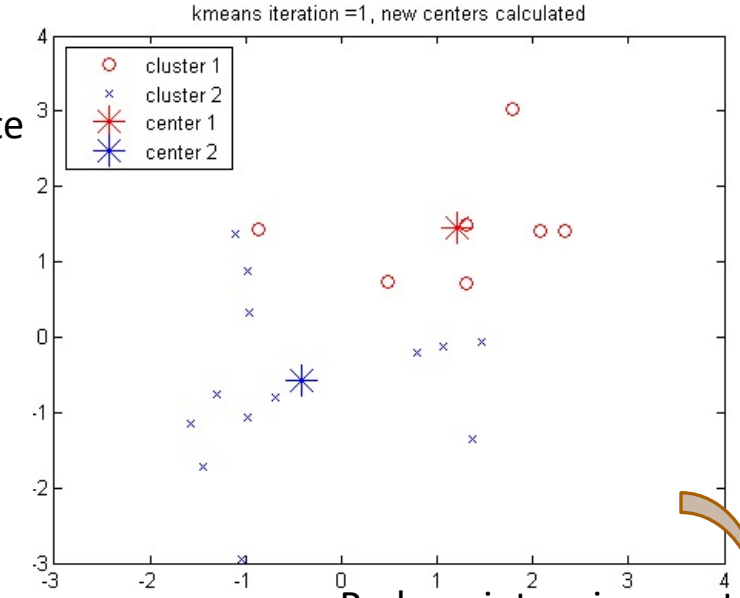


The original data points & randomly select $K = 2$ centroids

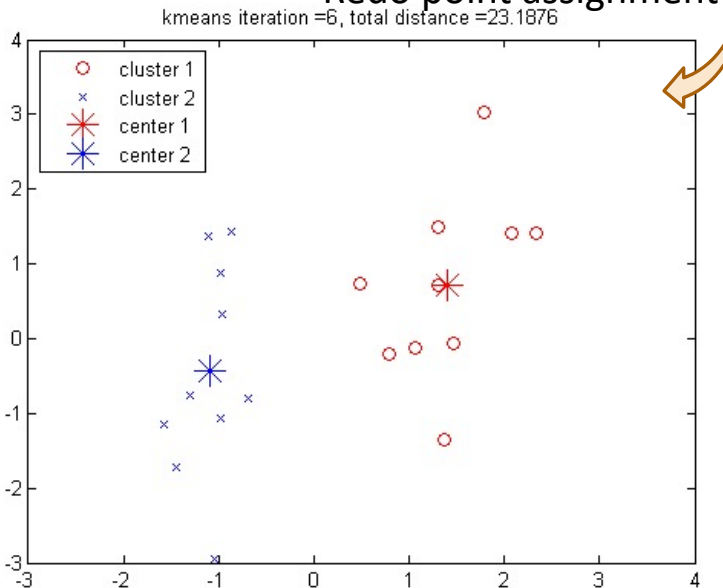
Assign points to clusters



Recompute cluster centers



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*

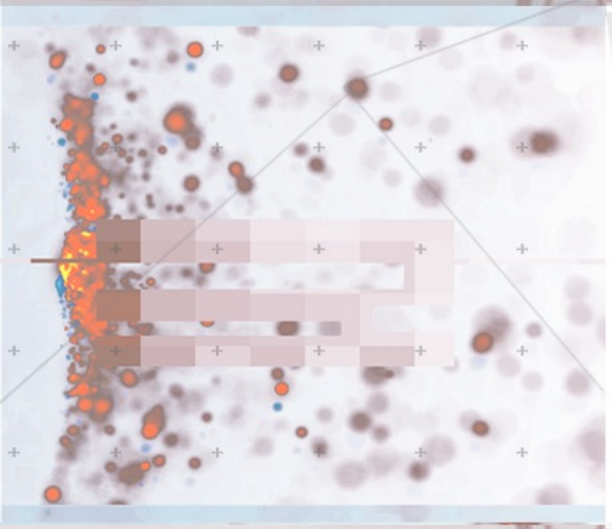
To be discussed in this lecture

  - Choosing different representative prototypes for the clusters
    - *K-Medoids*, *K-Medians*, *K-Modes*

To be discussed in this lecture

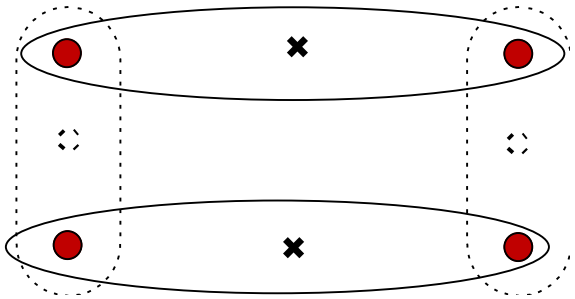
  - Applying feature transformation techniques
    - *Weighted K-Means*, *Kernel K-Means*

To be discussed in this lecture



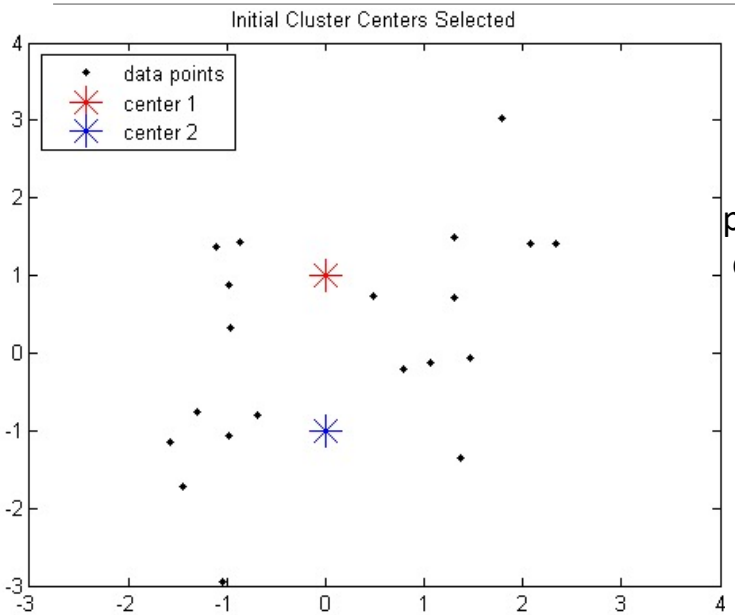
Session 3: Initialization of K-Means Clustering

Initialization of K-Means



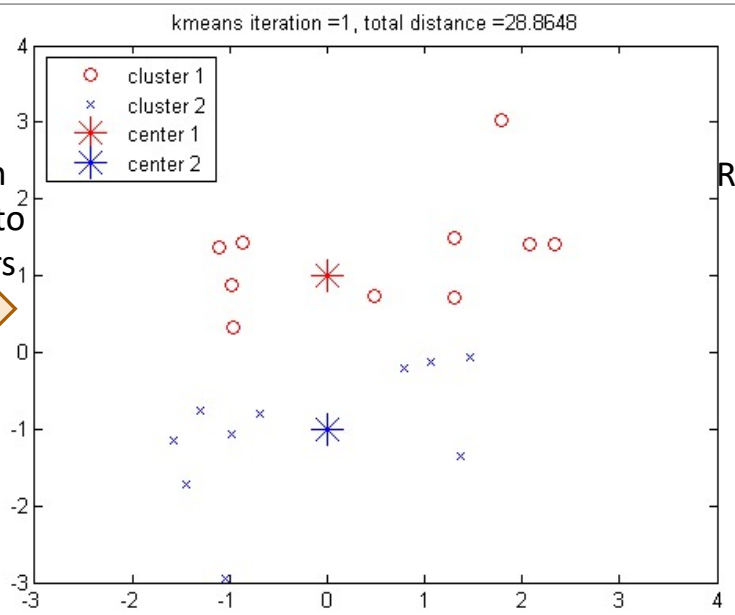
- Different initializations may generate rather different clustering results (some could be far from optimal)
- Original proposal (MacQueen'67): Select *K* seeds randomly
  - Need to run the algorithm multiple times using different seeds
- There are many methods proposed for better initialization of *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

Example: Poor Initialization May Lead to Poor Clustering

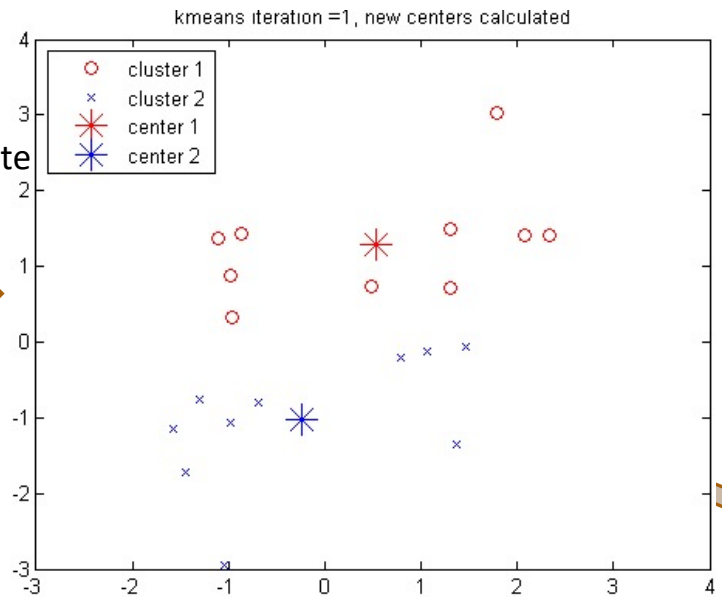


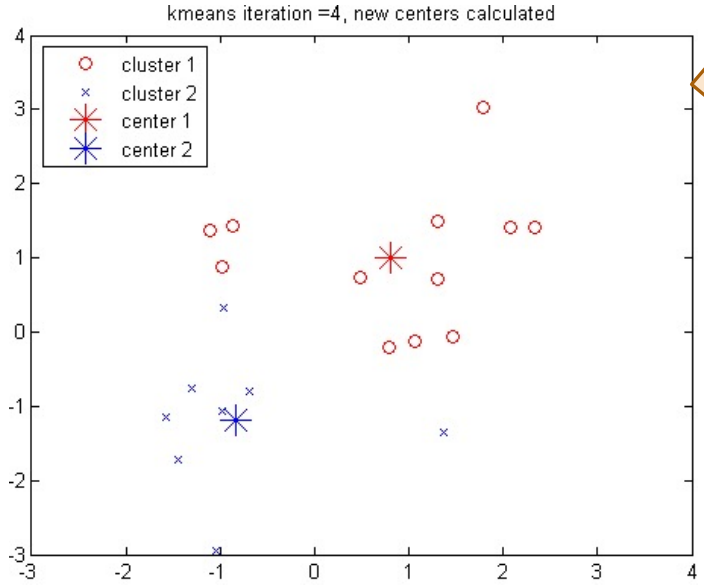
Another random selection of k centroids for the same data points

Assign points to clusters

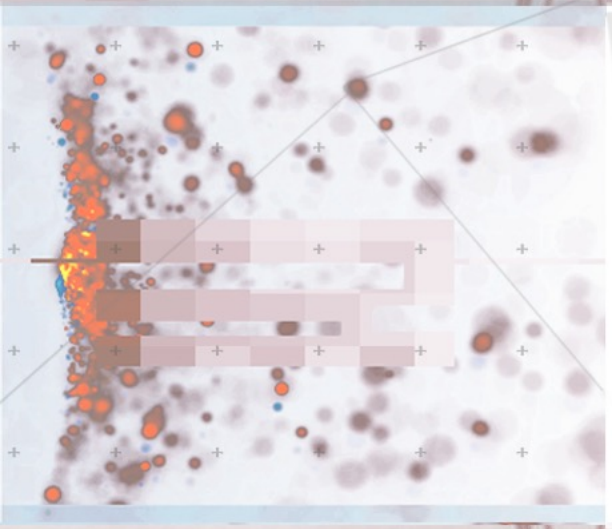


Recompute cluster centers





- Rerun of the *K-Means* using another random *K* seeds
- This run of *K*-Means generates a poor quality clustering

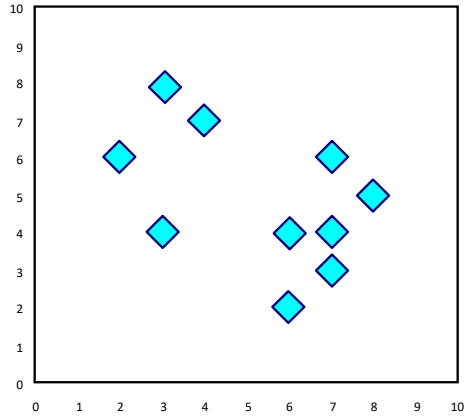


Session 4: The *K-Medoids* Clustering Method

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

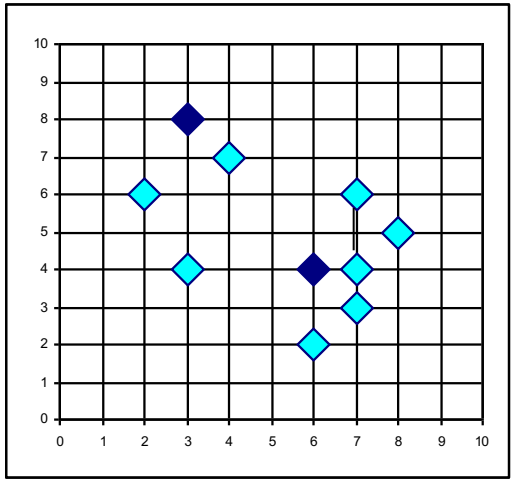
- The *K-Means* algorithm is sensitive to outliers!—since an object with an extremely large value may substantially distort the distribution of the data
- *K-Medoids*: Instead of taking the **mean** value of the object in a cluster as a reference point, **medoids** can be used, which is the **most centrally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

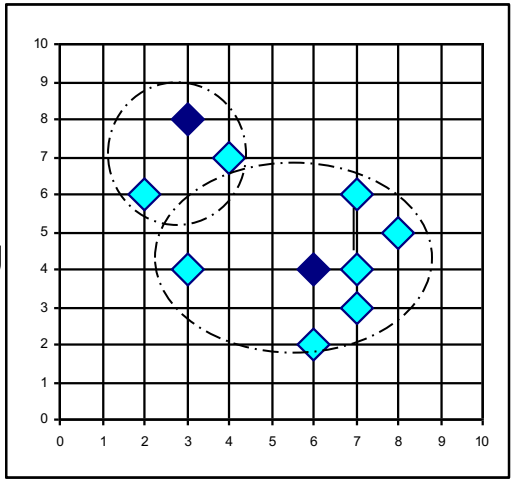


$K = 2$

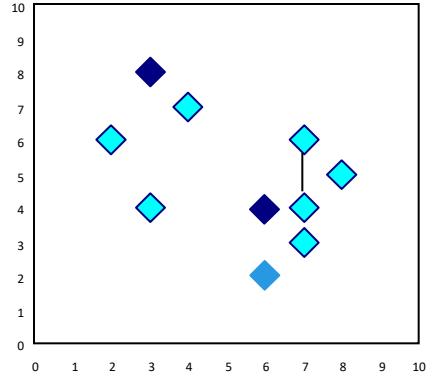
Arbitrary choose *K* object as initial medoids



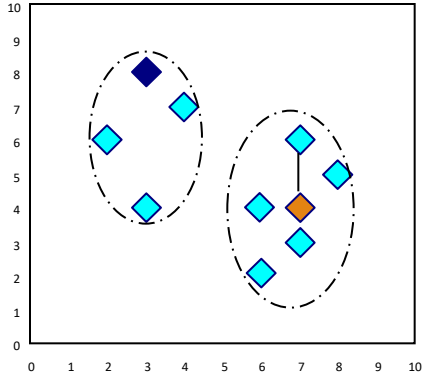
Assign each remaining object to nearest medoids



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

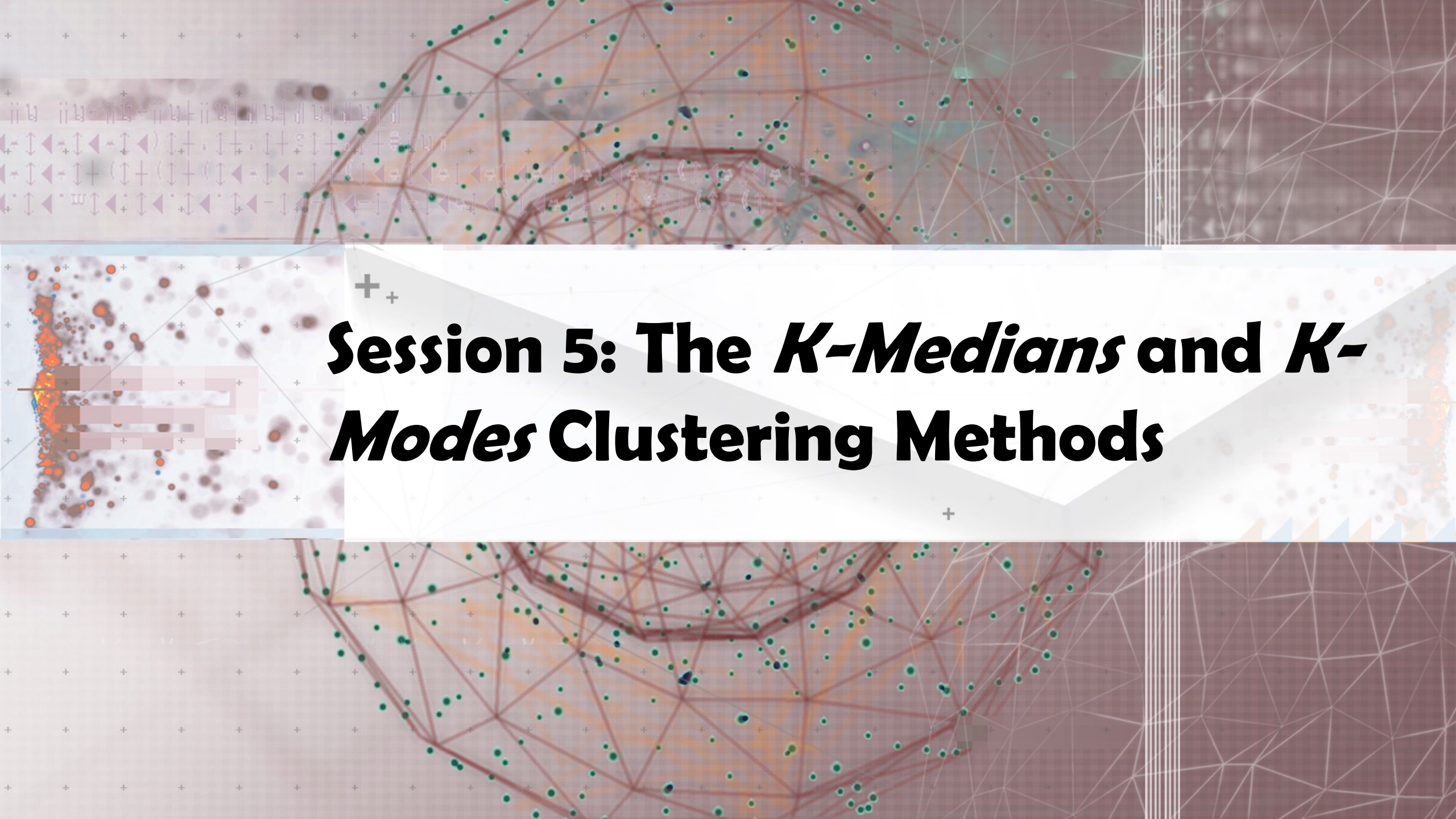
Object re-assignment

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: $O(K(n - K)^2)$ (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



Session 5: The *K-Medians* and *K-Modes* Clustering Methods

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:

$$S = \sum_{k=1}^{K} \sum_{x_{i \in C_k}} | x_{ij} - med_{kj} |$$

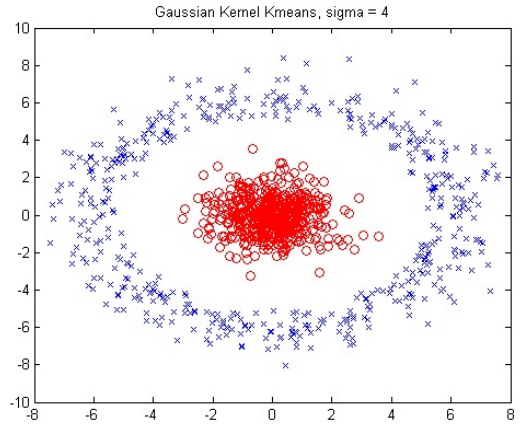
- The *K-Medians* clustering algorithm:
  - Select *K* points as the initial representative objects (i.e., as initial *K medians*)
  - **Repeat**
    - Assign every point to its nearest median
    - Re-compute the median using the median of each individual feature
  - **Until** convergence criterion 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

Session 6: Kernel K-Means Clustering

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

- Typical kernel functions:
  - Polynomial kernel of degree h: $K(\boldsymbol{X}_i, \boldsymbol{X}_j) = (\boldsymbol{X}_i \cdot \boldsymbol{X}_j + 1)^h$
  - 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: $c_k = \frac{\sum_{x_{i \in C_k}} \phi(x_i)}{| C_k |}$
- Clustering can be performed without the actual individual projections $\phi(x_i)$ and $\phi(x_j)$ for the data points $x_i, x_j \in C_k$

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:
  - $x_1(0, 0)$, $x_2(4, 4)$, $x_3(-4, 4)$, $x_4(-4, -4)$, $x_5(4, -4)$
- If we set σ to 4, we will have the following points in the kernel space
  - E.g., $\left|\left|x_1 - x_2\right|\right|^2 = (0-4)^2 + (0-4)^2 = 32$, therefore, $K(x_1, x_2) = e^{-\frac{32}{2 \cdot 4^2}} = e^{-1}$

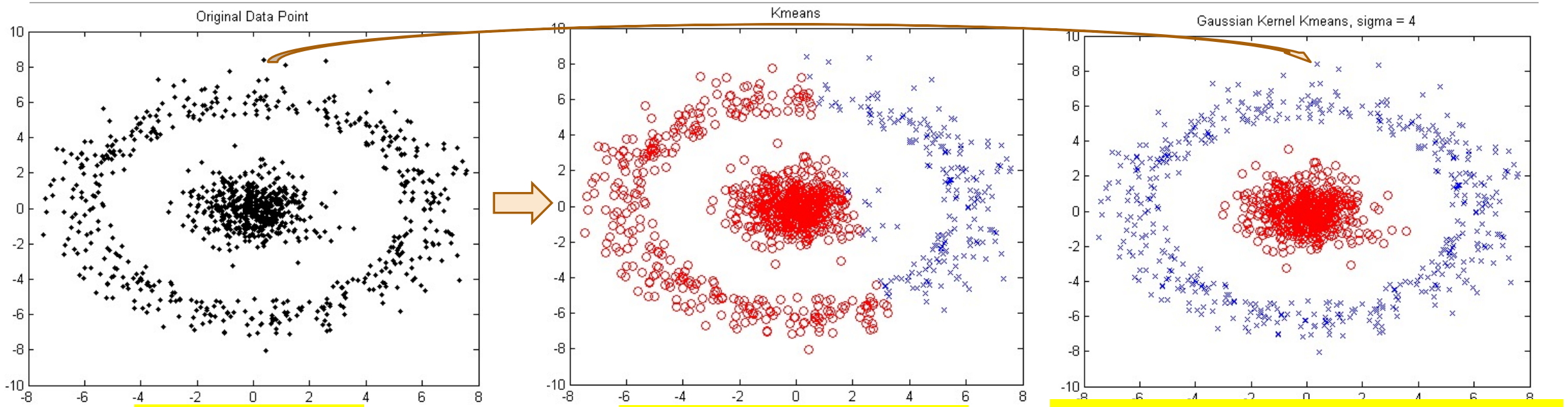
Original Space

	x	y
x_1	0	0
x_2	4	4
x_3	-4	4
x_4	-4	-4
x_5	4	-4

RBF Kernel Space ($\sigma = 4$)

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

Example: Kernel K-Means Clustering

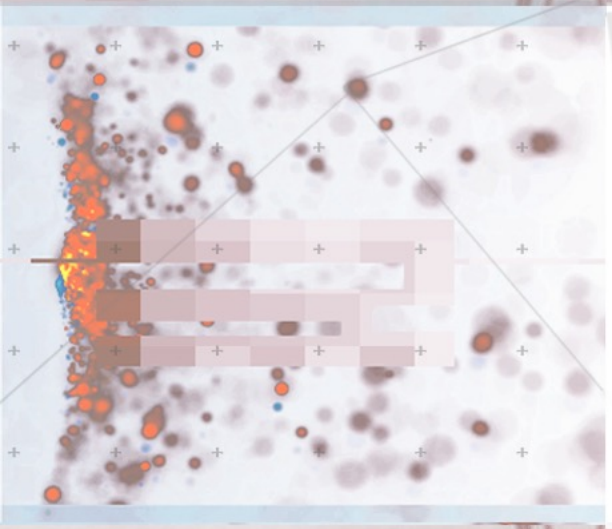


The original data set

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-covex 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



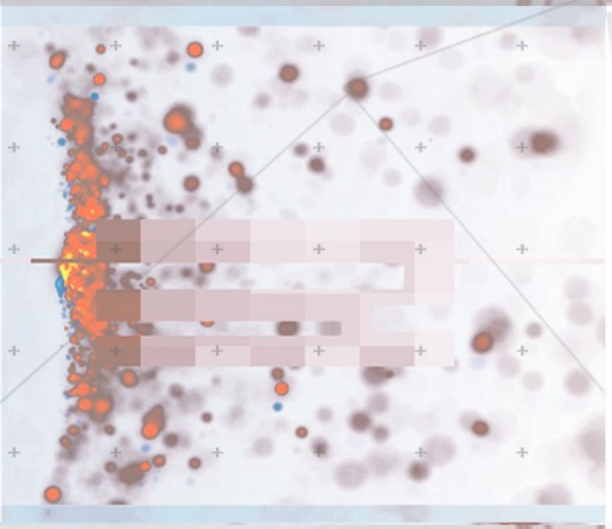
Summary

Summary: 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

Recommended Readings

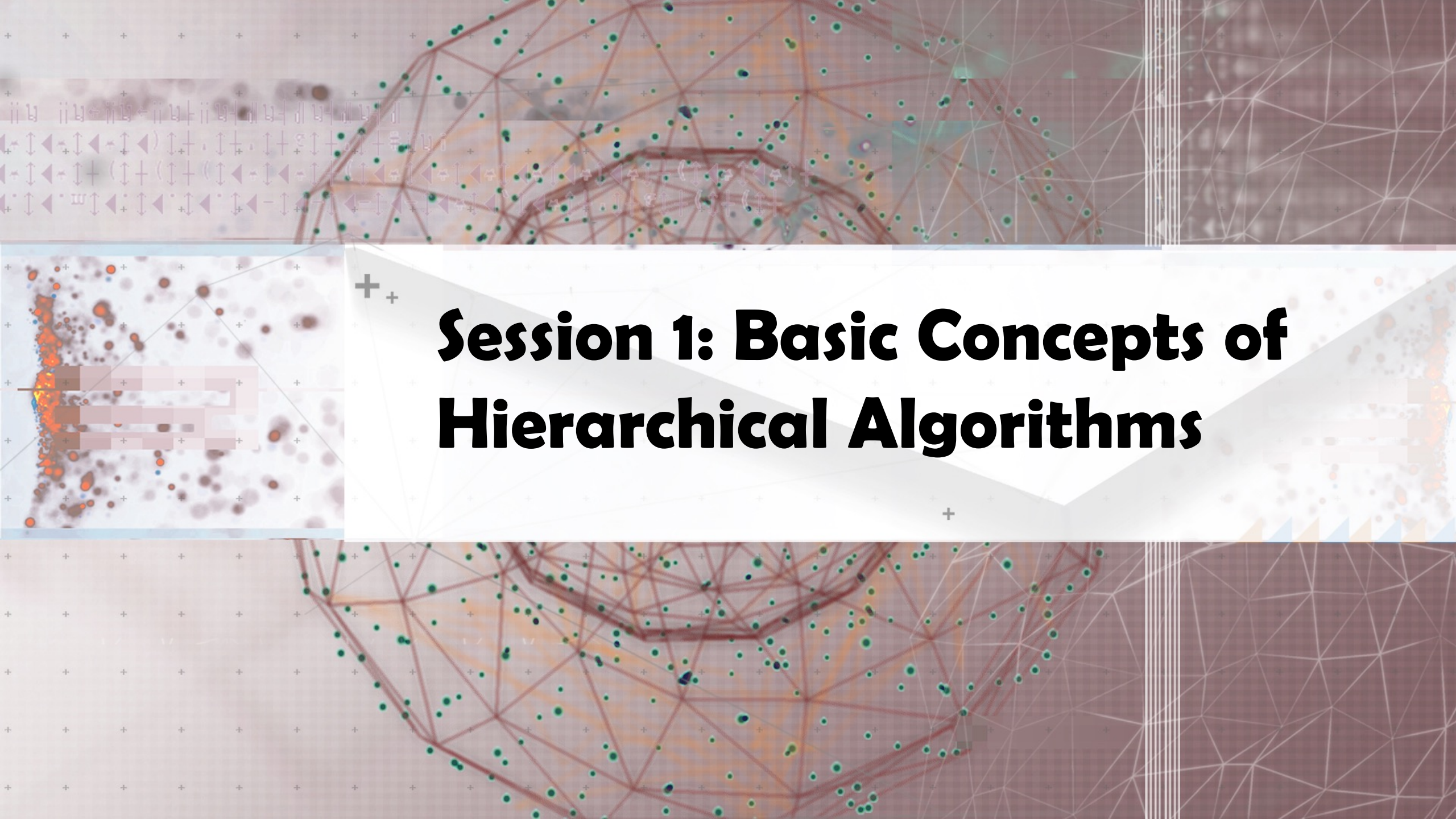
- J. MacQueen. Some Methods for Classification and Analysis of Multivariate Observations. In *Proc. of the 5th Berkeley Symp. on Mathematical Statistics and Probability*, 1967
- S. Lloyd. Least Squares Quantization in PCM. *IEEE Trans. on Information Theory*, 28(2), 1982
- A. K. Jain and R. C. Dubes. Algorithms for Clustering Data. Prentice Hall, 1988
- L. Kaufman and P. J. Rousseeuw. Finding Groups in Data: An Introduction to Cluster Analysis. John Wiley & Sons, 1990
- 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
- M. J. Zaki and W. Meira, Jr.. Data Mining and Analysis: Fundamental Concepts and Algorithms. Cambridge Univ. Press, 2014



Hierarchical Clustering Methods

Hierarchical Clustering Methods

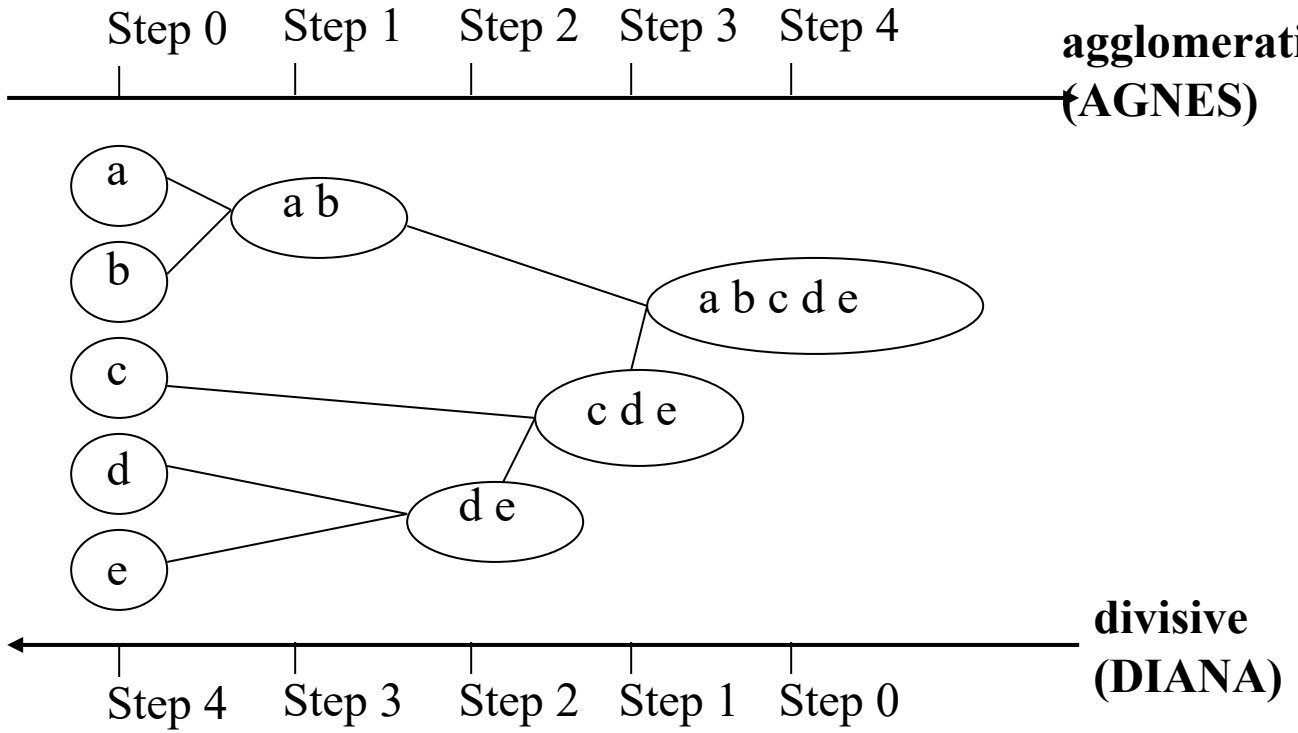
- Basic Concepts of Hierarchical Algorithms
- Agglomerative Clustering Algorithms
- Divisive Clustering Algorithms
- Extensions to Hierarchical Clustering
- BIRCH: A Micro-Clustering-Based Approach
- Probabilistic Hierarchical Clustering



Session 1: Basic Concepts of Hierarchical Algorithms

Hierarchical Clustering: Basic Concepts

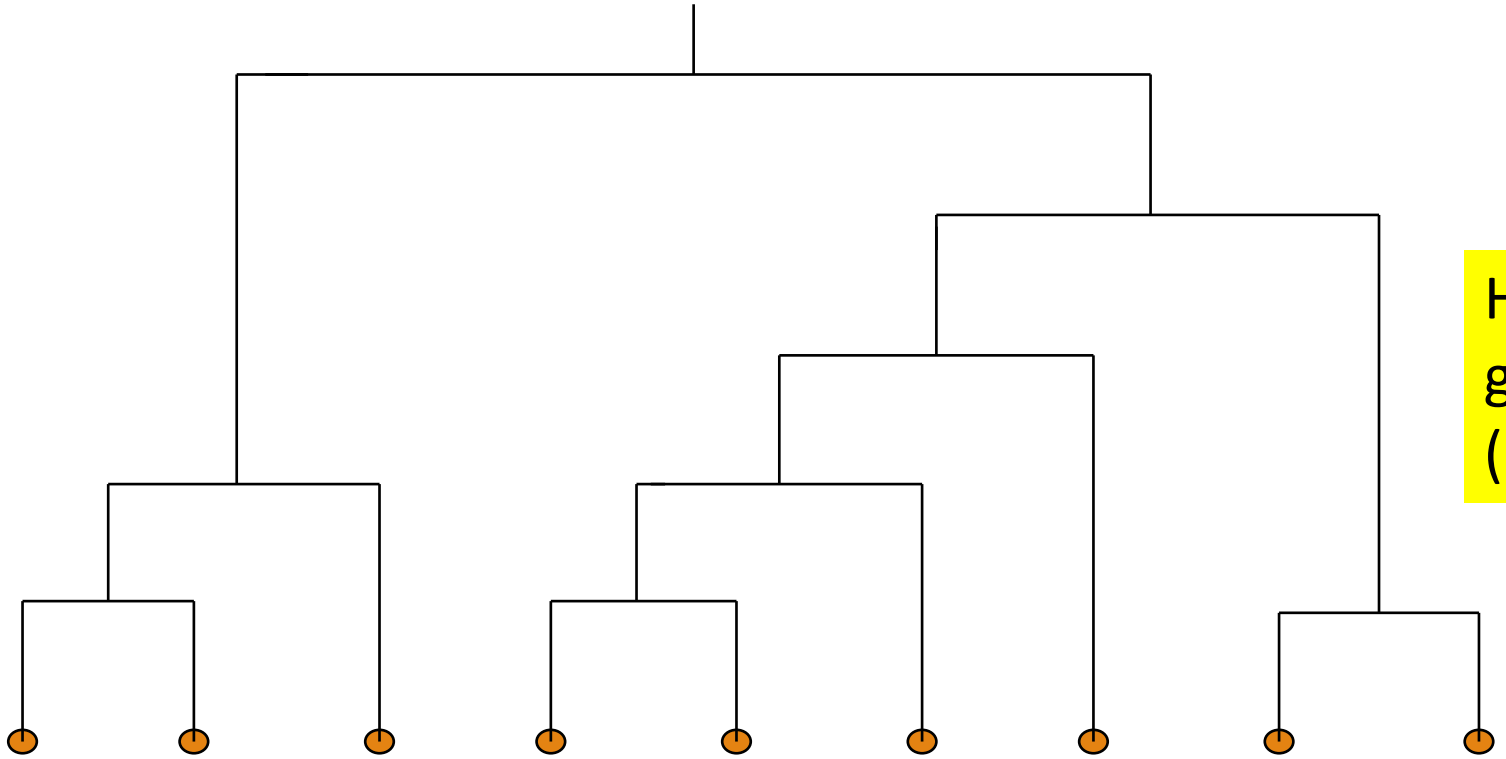
- Hierarchical clustering
  - Generate a clustering hierarchy (drawn as a **dendrogram**)
  - Not required to specify ***K,*** the number of clusters
  - More deterministic
  - No iterative refinement
- Two categories of algorithms:



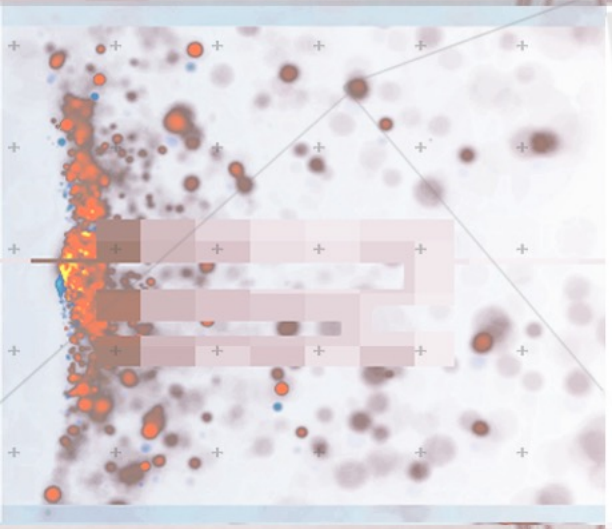
  - **Agglomerative**: Start with singleton clusters, continuously merge two clusters at a time to build a **bottom-up** hierarchy of clusters
  - **Divisive:** Start with a huge macro-cluster, split it continuously into two groups, generating a **top-down** hierarchy of clusters

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



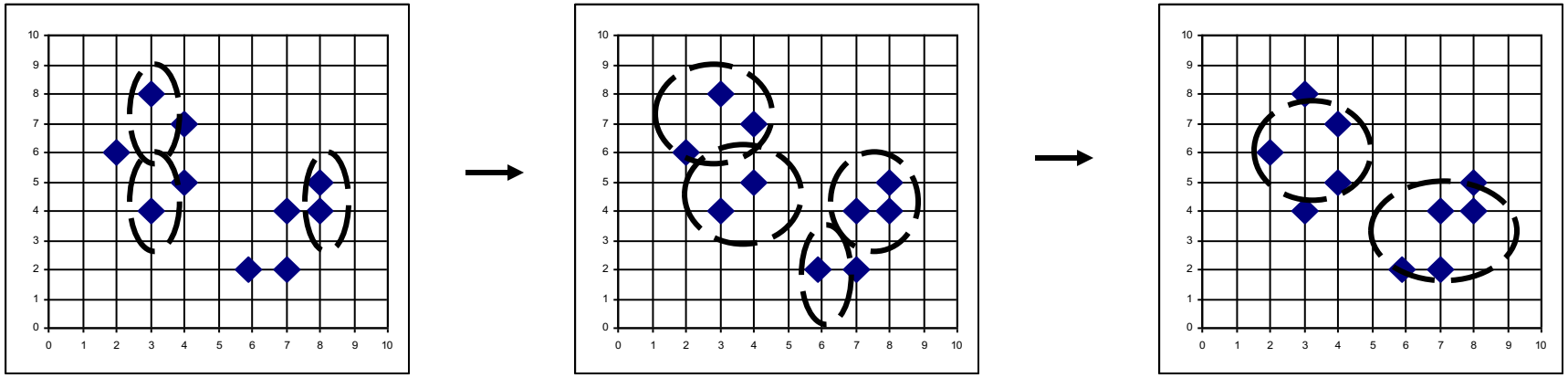
Hierarchical clustering generates a dendrogram (a hierarchy of clusters)



Session 2: Agglomerative Clustering Algorithms

Agglomerative Clustering Algorithm

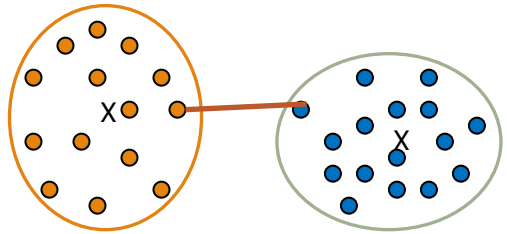
- AGNES (AGglomerative NESting) (Kaufmann and Rousseeuw, 1990)
  - Use the **single-link** method and the dissimilarity matrix
  - Continuously merge nodes that have the least dissimilarity
  - Eventually all nodes belong to the same cluster



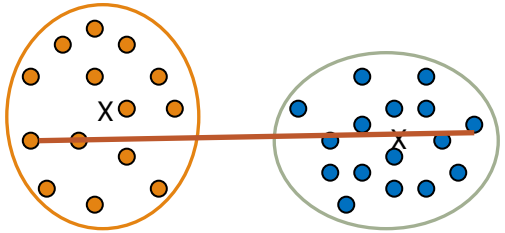
- Agglomerative clustering varies on different similarity measures among clusters
  - Single link (nearest neighbor)
  - Complete link (diameter)
  - Average link (group average)
  - Centroid link (centroid similarity)

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

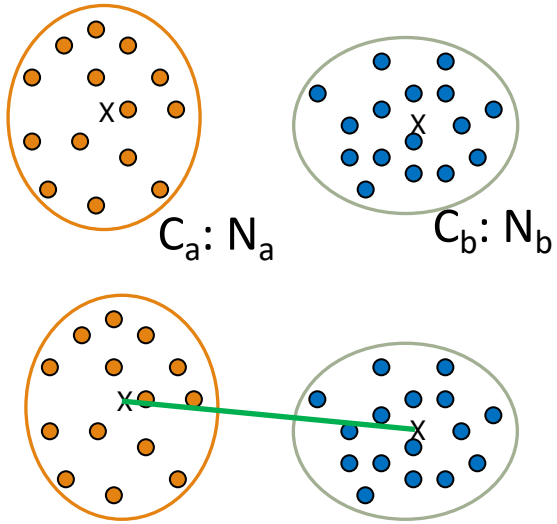


- Complete link (diameter)
  - The similarity between two clusters is the similarity between their most dissimilar members
  - Merge two clusters to form one with the smallest diameter
  - Nonlocal in behavior, obtaining compact shaped clusters
  - Sensitive to outliers



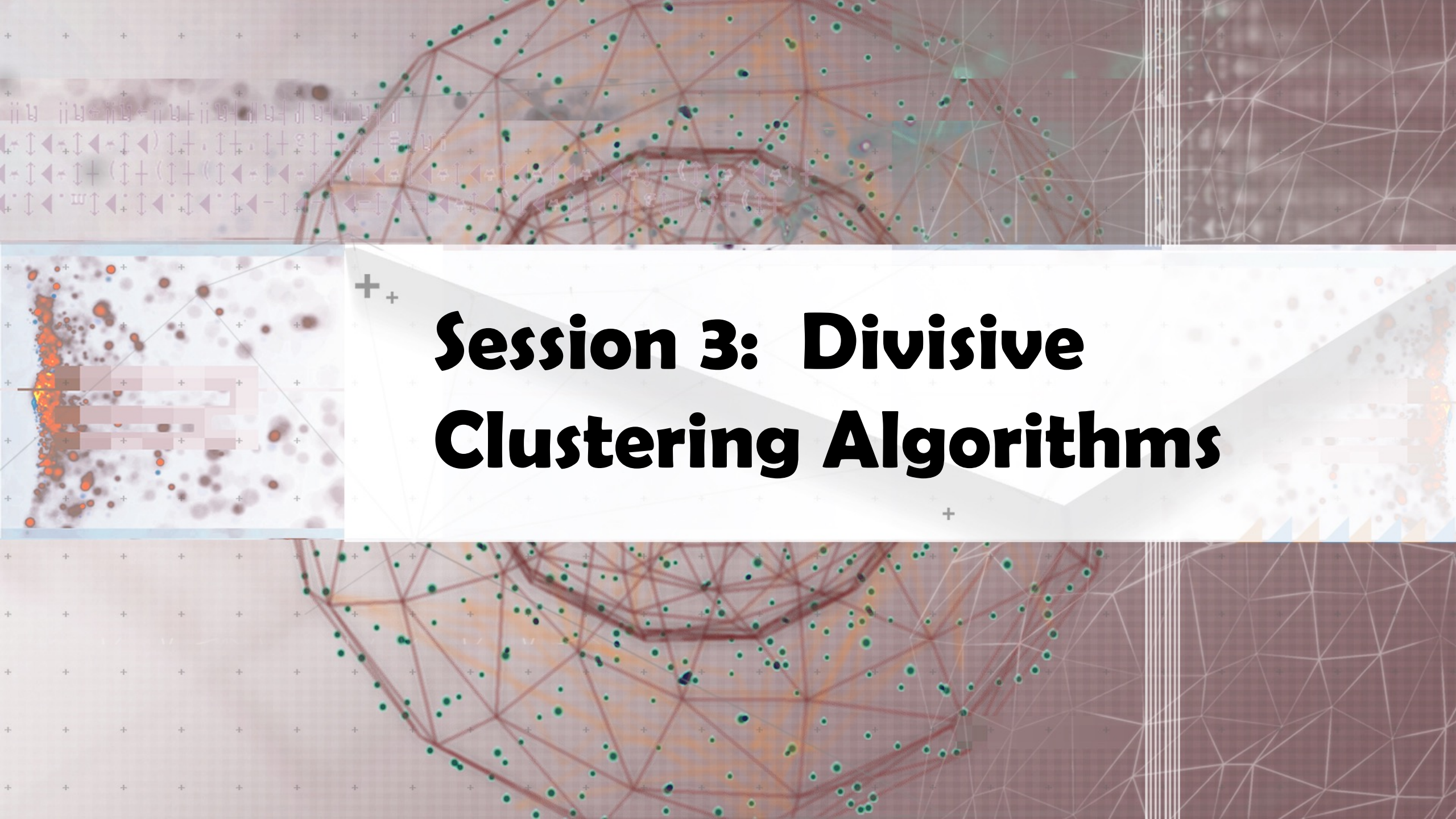
Agglomerative Clustering: Average vs. Centroid Links

- 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
- 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 U C_b:



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

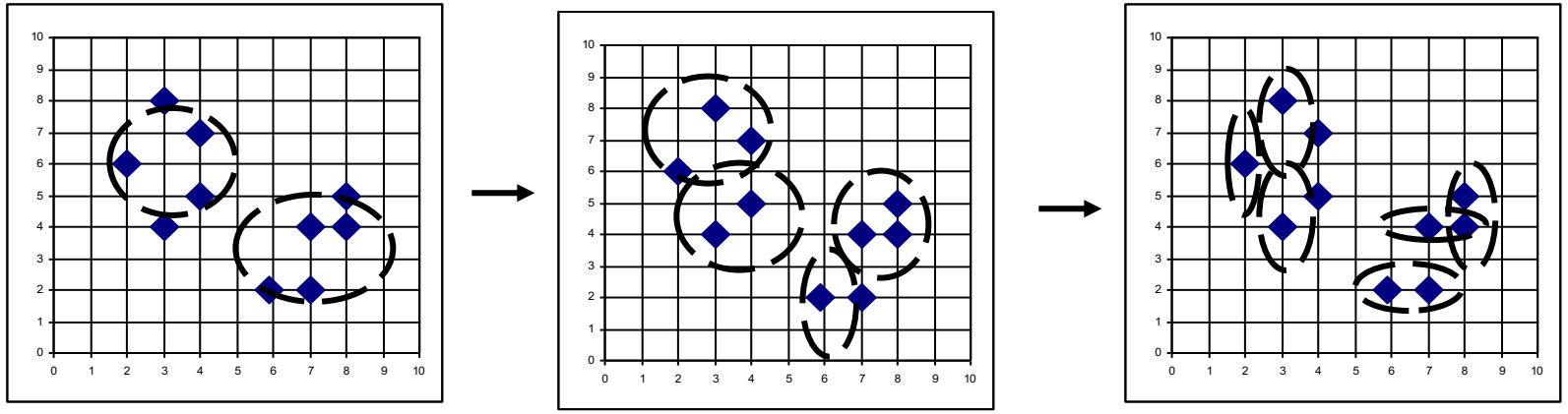
$$W(C_{a\cup b}, c_{a\cup b}) - W(C, c) = \frac{N_a N_b}{N_a + N_b} d(c_a, c_b)$$



Session 3: Divisive Clustering Algorithms

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)

Session 4: Extensions to Hierarchical Clustering & BIRCH

Extensions to Hierarchical Clustering

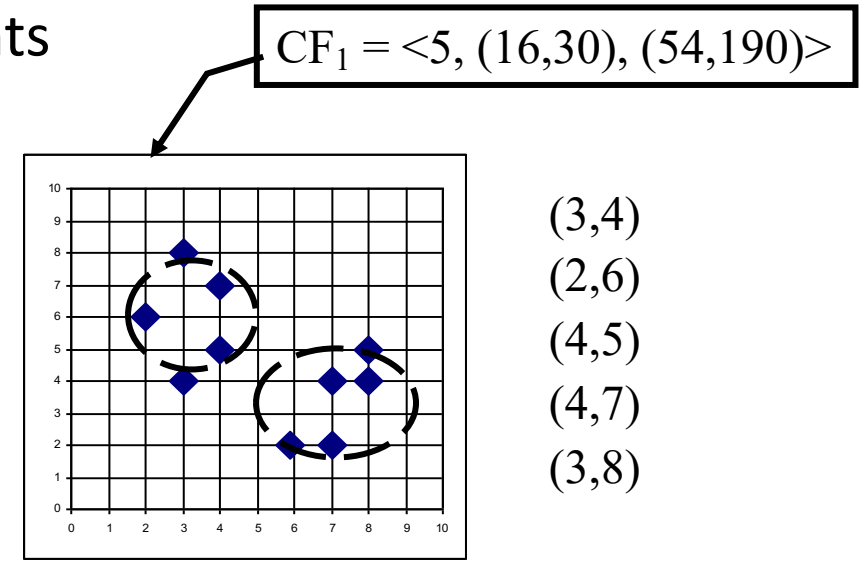
- Weakness of the agglomerative & divisive hierarchical clustering methods
  - No revisit: cannot undo any merge/split decisions made before
  - Scalability bottleneck: Each merge/split needs to examine many possible options
    - Time complexity: at least $O(n^2)$, where n is the number of total objects
- Several other hierarchical clustering algorithms
  - BIRCH (1996): Use CF-tree and incrementally adjust the quality of sub-clusters (To be covered)
  - 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: A Multi-Phase Hierarchical Clustering Method

- BIRCH (Balanced Iterative Reducing and Clustering Using Hierarchies)
  - Developed by Zhang, Ramakrishnan & Livny (SIGMOD'96)
  - Impact many new clustering methods and applications (received 2006 SIGMOD Test of Time award)
- Major innovation
  - Integrating hierarchical clustering (initial micro-clustering phase) and other clustering methods (at the later macro-clustering phase)
- Multi-phase hierarchical clustering
  - Phase1 (initial micro-clustering): Scan DB to build an initial CF tree, a multi-level compression of the data to preserve the inherent clustering structure of the data
  - Phase 2 (later macro-clustering): Use an arbitrary clustering algorithm (e.g., iterative partitioning) to cluster flexibly the leaf nodes of the CF-tree

Clustering Feature Vector

- Consider a cluster of multi-dimensional data objects/points
- The clustering feature (CF) of the cluster is a 3-D vector summarizing info about clusters of objects
  - Register the 0-th, 1st, and 2nd moments of a cluster
- Clustering Feature (CF): *CF = <N, LS, SS>*
  - *N*: Number of data points
  - *LS: linear sum of N points:* $LS = \sum_{i=1}^{n} x_i$
  - *SS: square sum of N points:* $SS = \sum_{i=1}^{n} x_i^2$



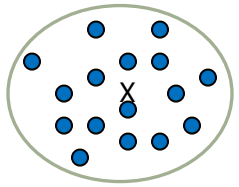
N = 5; LS = ((3+2+4+4+3), (4+6+5+7+8))=(16,30);
SS=((3²+2²+4²+4²+3²), (4²+6²+5²+7²+8²))=(54,190)

- Clustering feature: a summary of the statistics for the given cluster
  - Registers crucial measurements for computing cluster and utilizes storage efficiently
  - Clustering features are additive: Merging clusters C_1 and C_2—linear summation of CFs

$$CF_1 + CF_2 = \langle n_1 + n_2, LS_1 + LS_2, SS_1 + SS_2 \rangle$$

Essential Measures of Cluster: Centroid, Radius and Diameter

- Centroid: x_0
  - The "middle" of a cluster
  - n: number of points in a cluster
  - x_i is the *i*-th point in the cluster



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

- 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

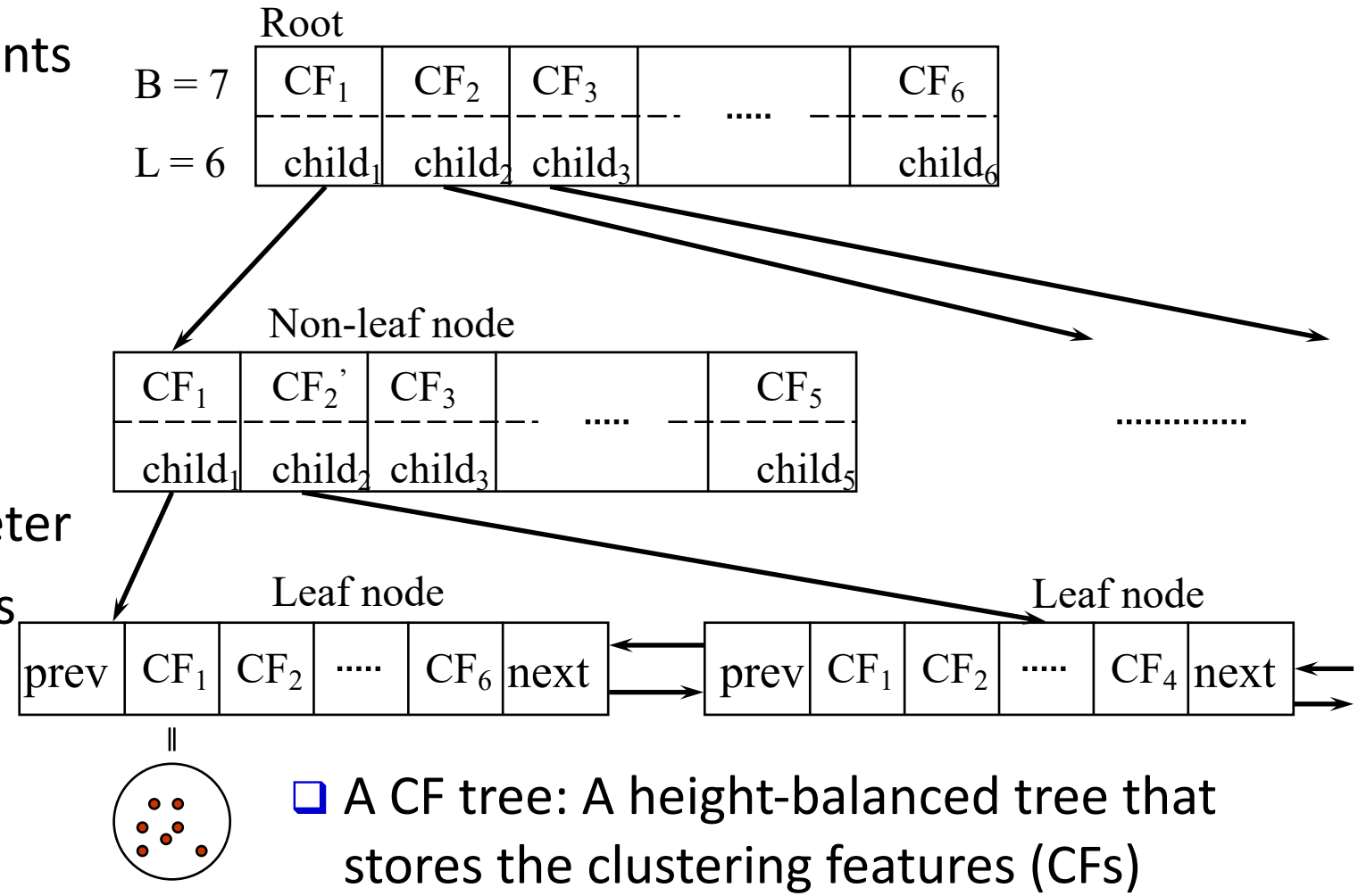
$$R = \sqrt{\frac{\sum_{i=1}^{n} (x_i - x_0)^2}{n}} = \sqrt{\frac{SS}{n} - (\frac{LS}{n})^2}$$

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

$$D = \sqrt{\frac{\sum_{i=1}^{n}\sum_{j=1}^{n} (x_i - x_j)^2}{n(n-1)}} = \sqrt{\frac{2nSS - 2LS^2}{n(n-1)}}$$

CF Tree: A Height-Balanced Tree Storing Clustering Features for Hierarchical Clustering

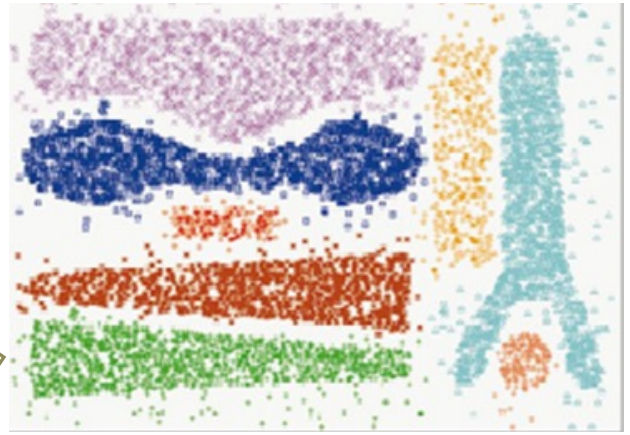
- Incremental insertion of new points (similar to B+-tree)
- For each point in the input
  - Find its closest leaf entry
  - Add point to leaf entry and update CF
  - If entry diameter > max_diameter
    - split leaf, and possibly parents
- A CF tree has two parameters
  - Branching factor: Maximum number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



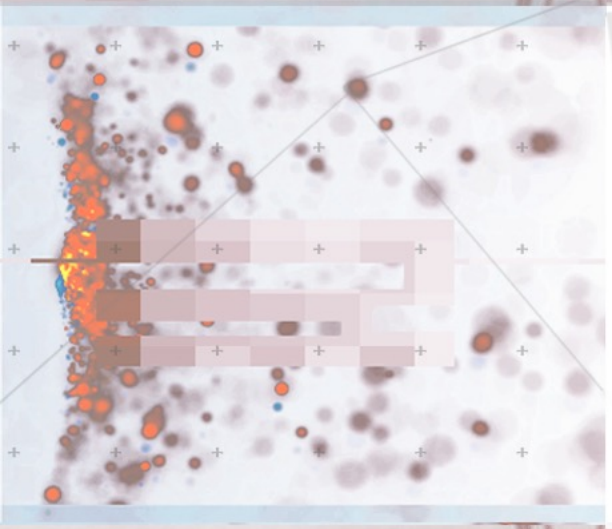
- A CF tree: A height-balanced tree that stores the clustering features (CFs)
- The non-leaf nodes store sums of the CFs of their children

BIRCH: A Scalable and Flexible Clustering Method

- An integration of agglomerative clustering with other (flexible) clustering methods
  - Low-level micro-clustering
    - Exploring CF-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
- Strength: Good quality of clustering; linear scalability in large/stream databases; effective for incremental and dynamic clustering of incoming objects
- Concerns
  - Due to the fixed size of leaf nodes, clusters so formed may not be very natural
  - Clusters tend to be spherical given the radius and diameter measures



Images like this may give BIRCH a hard time



Session 5: Probabilistic Hierarchical Clustering

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:

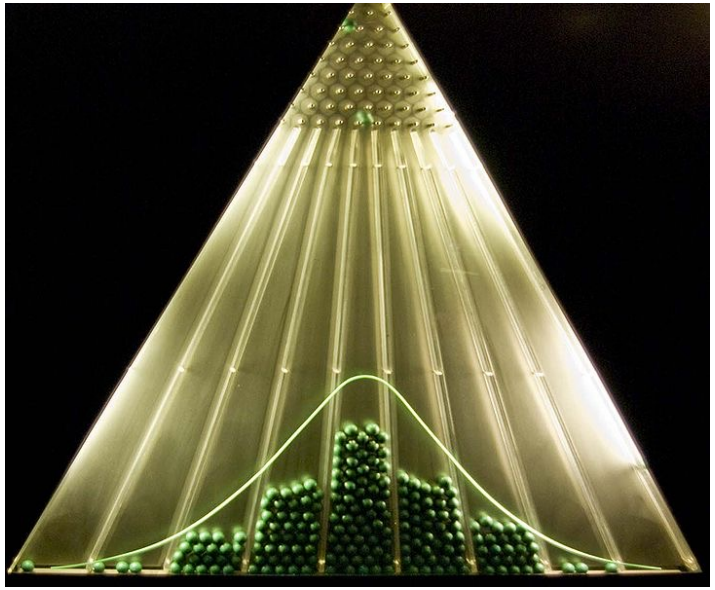
$$L(\mathcal{N}(\mu, \sigma^2) : X) = P(X|\mu, \sigma^2) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x_i-\mu)^2}{2\sigma^2}}$$

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

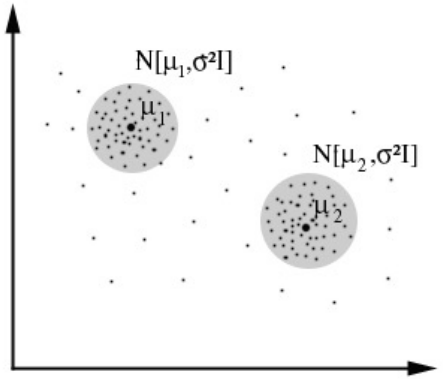
the maximum likelihood

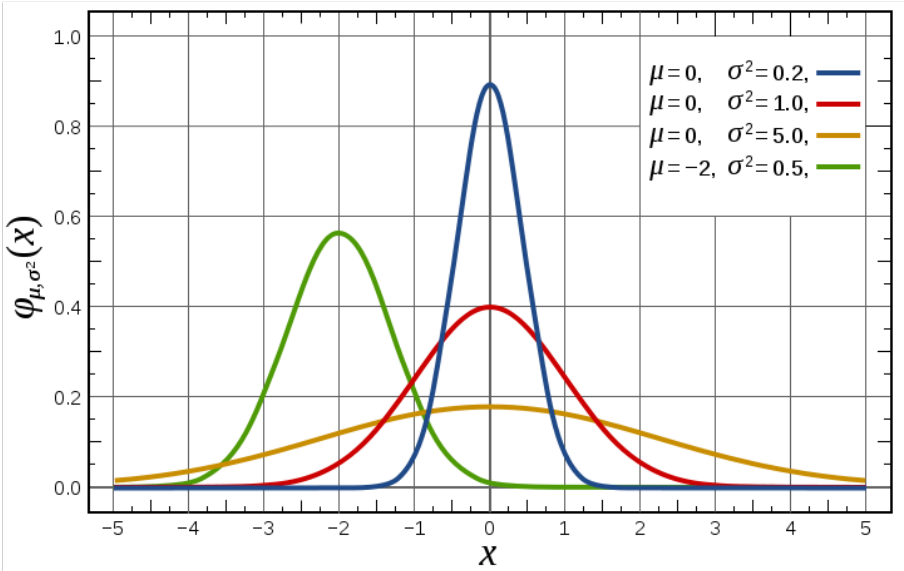
$$\mathcal{N}(\mu_0, \sigma_0^2) = \arg\max\{L(\mathcal{N}(\mu, \sigma^2) : X)\}$$

Gaussian Distribution

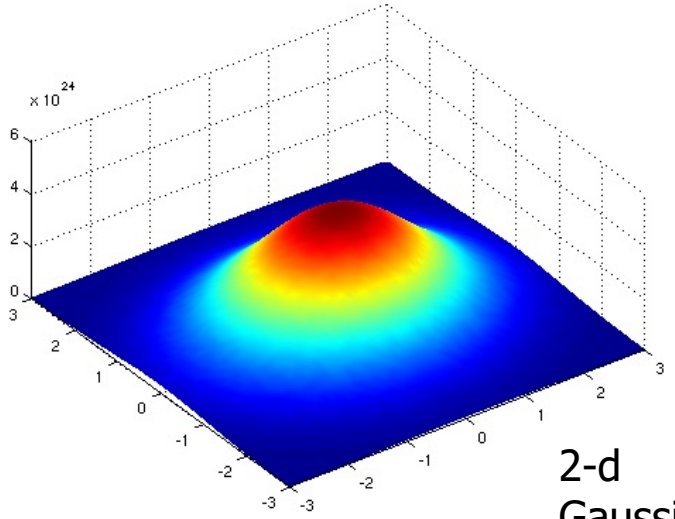


Bean machine: drop ball with pins





1-d Gaussian



2-d Gaussian

From wikipedia and http://home.dei.polimi.it

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

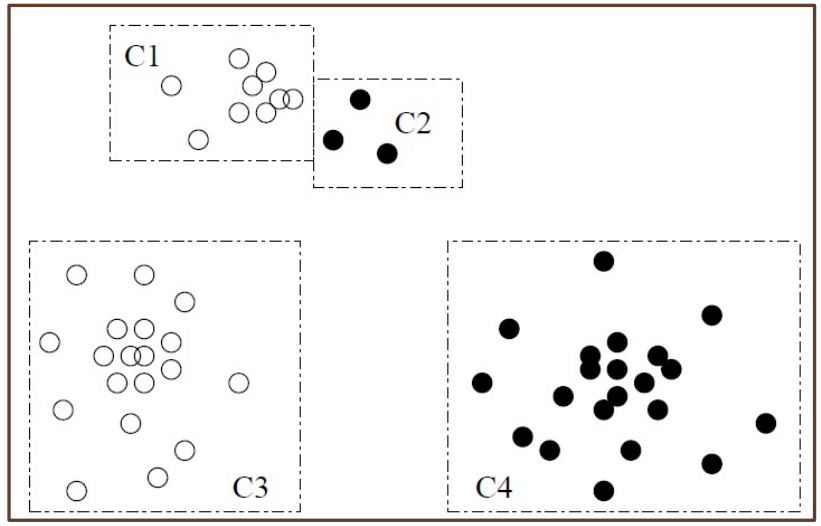
- If we merge two clusters C_{j1} and C_{j2} into a cluster $C_{j1} \cup C_{j2}$, the change in quality of the overall clustering is

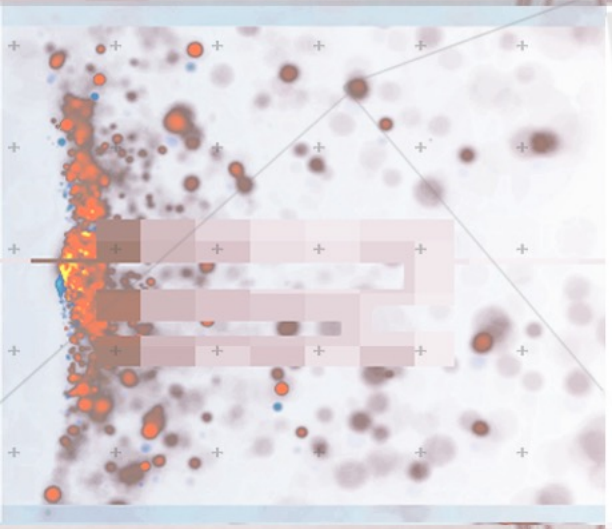
$$\begin{aligned} & Q((\{C_1, \ldots, C_m\} - \{C_{j_1}, C_{j_2}\}) \cup \{C_{j_1} \cup C_{j_2}\}) - Q(\{C_1, \ldots, C_m\}) \\ = & \frac{\prod_{i=1}^{m} P(C_i) \cdot P(C_{j_1} \cup C_{j_2})}{P(C_{j_1}) P(C_{j_2})} - \prod_{i=1}^{m} P(C_i) \\ = & \prod_{i=1}^{m} P(C_i) \left(\frac{P(C_{j_1} \cup C_{j_2})}{P(C_{j_1})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

- 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





Summary

Summary: Hierarchical Clustering Methods

- Basic Concepts of Hierarchical Algorithms
- Agglomerative Clustering Algorithms
- Divisive Clustering Algorithms
- Extensions to Hierarchical Clustering
- BIRCH: A Micro-Clustering-Based Approach
- Probabilistic Hierarchical Clustering

Recommended Readings

- 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
- Jiawei Han, Micheline Kamber, and Jian Pei. Data Mining: Concepts and Techniques. Morgan Kaufmann, 3rd ed. , 2011 (Chap. 10)
- C. K. Reddy and B. Vinzamuri. A Survey of Partitional and Hierarchical Clustering Algorithms, in (Chap. 4) Aggarwal and Reddy (eds.), Data Clustering: Algorithms and Applications. CRC Press, 2014
- M. J. Zaki and W. Meira, Jr.. Data Mining and Analysis: Fundamental Concepts and Algorithms. Cambridge Univ. Press, 2014