# Clustering for Mining in Spatial Databases

Presented by: Chang-tien Lu

## Overview

- Introduction
- Partitioning Clustering Algorithm
- Hierarchical Clustering Algorithm
- Density-Based Clustering Algorithm
- Grid-Based Clustering Algorithm
- Summary

# Introduction

- Huge amounts of Spatial data
  - *NASA EOS: generate 50GB of date per hour*
- Data mining
  - *Discover interesting, implicit, and previous unknown knowledge from large databases*
- Spatial data mining
  - *Discover interesting spatial patterns and features*
  - *Capture intrinsic relationships between spatial and non-spatial data*
  - *Present data regularity concisely and at higher conceptual levels*
- Applications of spatial data mining
  - *GIS systems*
  - *Image Database Exploration*
  - *Medical Imaging*

## Introduction

- Cluster Analysis
  - *Unsupervised learning*
  - *Find structure of clusters directly from the given data*
  - *Not rely on predefined class and class-labeled training examples*
  - *A data mining function*
  - *A tool:*
    - Gain insight into the distribution of data
    - Observe the characteristics of each cluster
    - Focus on particular set of clusters for further analysis
    - As a preprocessing step for classification and characterization

- Problem of Clustering
  - *Given n data points in a d-dimensional metric space*
  - *Partition the data points into k clusters*
  - *The data points within a cluster are more similar to each other than data points in different clusters*

# Partitioning Clustering Algorithm

- Method
  - *Objective function: Minmize* $E = \sum_{i=1}^{n} d(s_i, rep[s_i, C])$
  - *Arbitrarily choose k centres as the initial solution*
  - *repeat*
    - Compute membership of each object according to present solution
    - Update cluster centres according to new memberships of the objects
  - *until (No change to E)*
- Example: K-means, K mediod, EM(Expectation Maximization)
- Find the global optimal k centres: NP-hard
  - *Local optimal: an iterative relocation technique*
- General weakness
  - *Requirement to specify the parameter k*
  - *Inability fo find arbitrarily shaped clusters*

## Partitioning Clustering Algorithm: K-means

- Construct a partition of a database D of n objects in a d-dimensional space into a set of k clusters
  - *Cluster centre: the centroid(mean) of the objects in the cluster*
- Objective criterion(Square error): $\sum_{i=1}^{k} \sum_{x \in C_i} |x - m_i|^2$
  - *x: point in space represting the give object*
  - $m_i$*: mean of cluster* $C_i$
- Time compelexity: O(nkt)
  - *n: # of objects, k: # of clusters, t: # of iterations*
  - *Normally:* $k \ll n$ *and* $t \ll n$
  - *Relatively scalable and efficient in large data sets*
  - *Terminate at local optimum*
- Sensitive to noise and outlier
  - *a small number of such data substantially influence the mean value*

## Partitioning Clustering Algorithm: K mediod(CLARANS)

- Construct a partition of a database D of n objects in a d-dimensional space into a set of k clusters
  - *Cluster centre: the most centrally located object*
  - *Randomized and bounded search for improving the initial partition*
    - Randomly pick one of the k centries
    - Replace it with another randomly chosen object from the other (n-k) objects
  - *Each cluster is represented by the gravity center*
  - *Each object is assigned to the cluster with the closest representative*

- Drawbacks
  - *Time complexity* $O(KN^2)$, *too high*
  - *With large N, the quality of the results can not be guaranteed by randomized search*
  - *Assume all objects are stored in main memory*
  - *When there are large differences in the sizes or geometries, the method could split large clusters to minimize the square error*

- Less sensitive to noise and outlier

# Partitioning Clustering Algorithm: EM(Expectation Maximization

- Construct a partition of a database D of n objects in a d-dimensional space into a set of k clusters
  - *Assign each object to a cluster accroding to a probability of membership(fuzzy partitioning technique)*
  - *A d-dimensional Gaussian distribution representing a cluster $C_i$: Mean of the cluster $\mu_i$, d*d covariance matrix $M_i$*
  - *Probability of an object occurring at location x :* $P(x|i) = \frac{1}{\sqrt{(2\pi)^d |M_i|}} e^{\frac{1}{2}(x-\mu_i)^T (M_i)^{-1} (x-\mu_i)}$
  - *Combine the effect of the different cluster distributions at x:* $P(x) = \sum_{i=1}^{k} W_i P(x|i)$
    - $W_i$ is the fraction of the databasee represented by $C_i$
  - *Probability(membership) of an object at x belonging to cluster $C_i$:* $P(i|x) = W_i \frac{P(x|i)}{P(x)}$
  - *New value:*
    $W_i = \frac{1}{n} \sum_{x \in D} P(i|x)$ $\quad \mu_i = \frac{\sum_{x \in D} x P(i|x)}{\sum_{x \in D} P(i|x)}$ $\quad M_i = \frac{\sum_{x \in D} P(i|x)(x-\mu_i)(x-\mu_i)^T}{\sum_{x \in D} P(i|x)}$
  - *Criterion function:*
    $E = \sum_{x \in D} log(P(x))$
    - Converge to a local optimal

# Hierarchical Clustering Algorithm

- Hierarchical Clustering Algorithm
  - *Decompose a database D into several levels of partitions(clusters)*
  - *Dendrogram: A tree which splits the database recursively into smaller subsets*
  - *Can either be created from the leaves up to the root, or from the root down to the leaves*
  - *Bottom-up approach(agglomerative):*
    - Start with each object forming a separate group
    - Merge groups accroding to some principle: e.g., distance between centers
    - Until a termination condition holds
  - *Top-down approach(divisive):*
    - Start with all objects in the same cluster
    - A cluster is split into smaller clusters
    - Until a termination condition holds
  - *Example: BIRCH, CHAMELEON*

## Hierarchical Clustering Algorithm:BIRCH

- Balanced Iterative Reducing and Clustering using Hierarchies
  - *Clustering Feature: summarizeing information about a cluster*
  - *CF =* $(n, \sum x_i, \sum x_i^2)$*, no of points, linear sum, square sum of all points in the cluster*
  - *CF tree: A balanced tree used to store the clustering features*
    - Branching factor(B): max number of children per non-leaf node
    - Threshold(T): max diameter of subclusters stored at the leaf node of the tree
- Steps
  - *Load the data into memory be creating a CF tree that summarizes the data*
  - *Build a smaller CF tree if necessary: T is increased*
  - *Perform global clustering: Centroid-based alg is used to cluster the set of summaries*
  - *Redistribute the data point using the centroids discovered in step 3, and thus discover a new set of clusters*

## Hierarchical Clustering Algorithm:BIRCH

- Advantage
  - *By combining closed clusters together and rebuilding CF tree*
    - Make full use of the available memory
    - Require a single scan of the data set
  - *Time complexity: Linearly proportional to the no of objects.*
- Drawbacks
  - *Use radius or diameter to control the boundary of a cluster*
  - *May not work well when clusters are not spherical*
  - *With non-uniform sizes, problem of labeling*

# Hierarchical Clustering Algorithm:CHAMELEON

- Steps
  - *A graph partitioning alg. to cluster the objects into small subclusters*
  - *An agglomerative hierarchical clustering alg. to find the clusters by repeatedly combining these clusters*
  - *Two clusters are merged*
    - The inter-connectivity and closeness between two clusters are highly related to the internal inter-connectivity and closeness of objects within the clusters
- Represents its objects:
  - *K-nearest neighbor graph*
  - *Each vertex represents a data object*
  - *An edge between two vertices: if one object is among the k-most similar objects of the other*

# Hierarchical Clustering Algorithm:CHAMELEON

- Similarity between each pair of clusters $C_i$ and $C_j$
  - *Relative inter-connectivity:* $RI(C_i, C_j) = \frac{|EC_{\{C_i,C_j\}}|}{\frac{1}{2}(|EC_{C_i}|+|EC_{C_j}|)}$
    - $EC_{\{C_i,C_j\}}$ : Edge cute of the clusters, broked into $C_i$ and $C_j$
    - $EC_{C_i}$: size of its min-cut bisector
  - *Relative closeness:* $RC(C_i, C_j) = \frac{\overline{S}_{EC_{\{C_i,C_j\}}}}{\frac{|C_i|}{|C_i|+|C_j|}\overline{S}_{EC_{C_i}}+\frac{|C_i|}{|C_i|+|C_j|}\overline{S}_{EC_{C_j}}}$
    - $\overline{S}_{EC_{\{C_i,C_j\}}}$ : the average weight of the edges that connect vertices in $C_i$ to vertices in $C_j$
    - $\overline{S}_{EC_{C_j}}$: the average weithg of the edge that belong to the min-cut bisector of cluster $C_j$

- More power at discovering arbitrarily-shaped clusters

- Processing cost: $O(n^2)$ time for high-dimensional data for n objects in the worst case

## Single Scan Clustering Algorithm: DBSCAN

- Density Based Spatial Clustering of Applications with Noise
  - Key idea
    - *Eps-neighborhood: within a radius Eps of a given object*
    - *Core object: Eps-neighborhood of an object contains at least a minimum number, MinPts*
    - *Density-reachable: p is within the Eps-neighborhood of q, and q is a core object*
  - Steps
    - *Find the core objects as new clusters*
    - *Iteratively collects density-reachable objects from core objects*
      - Merge of a few density-reachable clusters
  - *Make use of an R* Tree to achieve good performance*
    - Process neighborhood queries efficiently
- Advantage
  - *Can be used to discover clusters of arbitray shape*
  - *Can separate noise from clusters of points*

## Density Based Clustering Algorithm: DBSCAN

- Drawbacks
  - *Time complexity O(Nlog(N))*
  - *Requires a human participant to determine the global parameters Eps.*
  - *A dense string of points connecting two clusters, DBSCAN could end up merging the two clusters*
  - *Execute directly on the entire DB, for large DB, could incur substntial I/O costs*
  - *With density-based alg, using random sampling to reduce the input size may not be feasible*
  - *Not very efficient for high dimensional data: efficiency neighborhood query drops increasingly with the number of dimensions*

# Density Based Clustering Algorithm: DENCLUE

- Clustering based on density distribution functions
  - *Influence function: impact of a data point within its neighborhood*
  - *Overall density: the sum of the influene function of all data points*
  - *Determine clusters by identify density attractors, which are local maxima of the overall density function*
- Influence function
  - *Influence function of y on x :* $f_B^y(x) = f_B(x, y)$
  - *Square wave influence function:* $f_{Square}(x, y) = \{0 : if \;\; d(x, y) > \sigma, 1 : otherwise\}$
  - *Gaussian influence function:* $f_{Gauss}(x, y) = e^{-\frac{d(x,y)^2}{2\sigma^2}}$
- Density function
  - $f_B^D(x) = \sum_{i=1}^n f_B^{x_i}(x)$
  - $f_{Gaussian}^D(x) = \sum_{i=1}^n e^{-\frac{d(x,y)^2}{2\sigma^2}}$
- Advantage
  - *Handle high dimensional data efficiently*
    - A grid structure is utilized
    - Outperform DBSCAN by as much as 45 times

# Density Based Clustering Algorithm: DENCLUE (Example)

## Grid-base cluster methods

- Grid data structure
  - *Enhance the efficiency of clustering*
  - *Quantize the space into a finit number of cells*
  - *Perform clustering on the grid structure*
  - *Fast processing time*
    - Independent of the number of data objects
    - Dependent on only the number of cells in each dimension
- Approaches
  - *WaveCluster: clusters objects using a wavelet transform method*
  - *CLIQUE: represents a grid- and density- based approach for clustering in high-dimensional space*
- In high dimensional space
  - *More efficient than a density-based approach*
  - *Use of summarized information cause it to increasingly lose effectiveness*

## Grid-base algorithm: WaveCluster

- Methods
  - *Summarize the data by imposing a multidimensional grid structure*
  - *Use a wavelet transformation to transform the original feature space*
  - *Find dense regions in the transformed space*
- WaveCluster
  - *Wavelet transform: data converted from the spatial domain into the frequency demain*
  - *Nature clusters in the data become more distinguishable*
  - *Identify clusters by searching for dense regions in the new domain*
- Advantages of WaveCluster
  - *Emphasize regions where the points cluster*
  - *Suppress weaker information outside of the cluster boundaries*
  - *Automatically result in the removal of outliers*
  - *Computational complexity: $O(n)$, $n$ is the number of objects*
  - *Handle large data sets efficiently*
  - *Discover clusters with arbitrary shape*

# Grid-base algorithm: WaveCluster (Example)

## Summary

- Partitioning methods
  - *Examples: K-means, K-medoids, and EM*
  - *Itertive reallocation to improve the clusters' quality from an initial solution*
  - *Tend to find clusters in spherical shape and similar in szie*
  - *Useful for application: facility location, objective: minimize the distance from the objects to their cluster centers*
- Hierarchical clustering
  - *Fix the membership of a data object once it has been allocated to a cluster*
  - *BIRCH*
    - Compress them into clustering features
    - Use iterative relocation to improve the clusters quailty
  - *CHAMELEON: uses more compex criteria when mergin clusters*

## Summary

- Density-based clustering
  - *Use density of data points within a region to discover clusters*
  - *DBSCAN*
    - Requires two parameters: Eps and MinPts
    - Sensitive to Eps and MinPts
  - *DENCLUE*
    - Model overall density of a point as the sum of influence functions
    - Utilize a grid structure to handle the computation efficiently
    - Outperfrom DBSCAN in high dimensional space

- Grid-base clustering
  - *Quantize the clustering space into a finit number of cells*
  - *Identify cells that contain more than a number of points as dense*
  - *Form clusters by connecting these dense cells*
  - *More efficient than a density-based approach*
  - *Lose effectiveness as the number of dimension increase*