# SS-ZG548: ADVANCED DATA MINING

# 05 Incremental Clustering

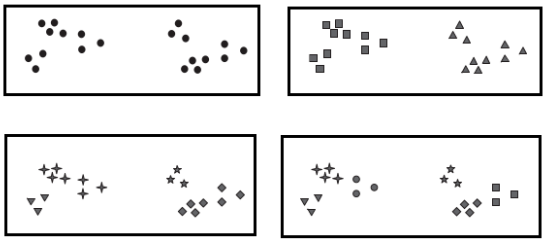
**Dr. Kamlesh Tiwari**
Assistant Professor, Department of CSIS,
BITS Pilani, Pilani Campus, Rajasthan-333031 INDIA


Aug 23, 2021 ONLINE (WILP @ BITS-Pilani July-Dec 2021)

http://ktiwari.in/adm

## Clustering

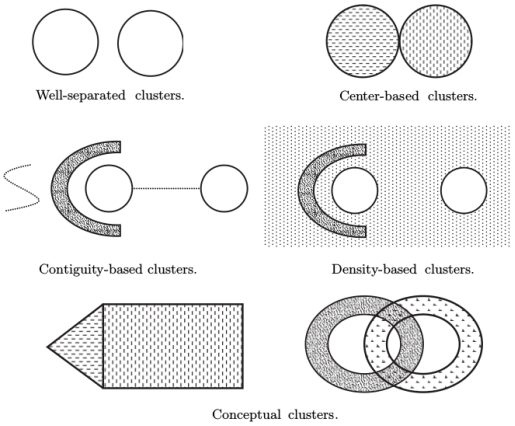
Grouping data - based on their homogeneity (similarity or closeness).



Objects within a group are similar (or related) and are different from the objects in other groups. When is it better?



## Clustering






## Clustering

- **Unsupervised** in nature (i.e. right answers are not known)
- Clustering is useful to 1) Summarization, 2) Compression, and 3) Efficiently Finding Nearest Neighbors
- **Type:**
  - Hierarchical (nested) versus Partitional
  - Exclusive versus Overlapping versus Fuzzy
  - Complete versus Partial
- **K-means:** This is a prototype-based[1], partitional clustering technique that attempts to find a user-specified number of clusters (K), which are represented by their centroids.

[1] object is closer (more similar) to a prototype



## K-means Algorithm

Number of clusters *i.e.* the value of $K$ is provided by the user

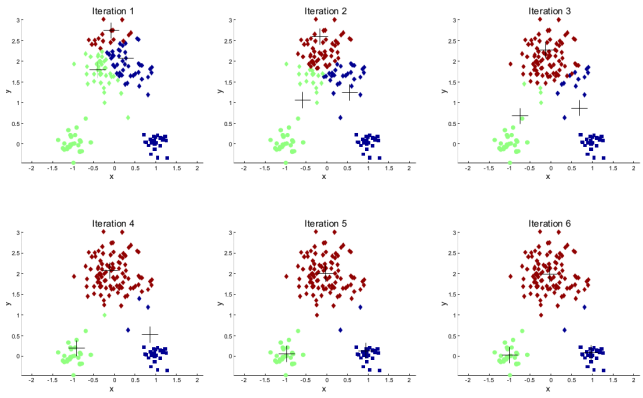
**Algorithm 1:** K-means

1. Randomly select $K$ points as centroids
2. **repeat**
3. **foreach** *datum point* $d_i$ **do**
4. Assign $d_i$ to one of the closest centroids (thereby forming $K$ clusters)
5. Recompute centroid (mean) for each cluster
6. **until** *The centroids converge*;

Closeness is measured by **Euclidean distance**, cosine similarity, correlation, Bregman divergence *etc*



## K-means in Action



## Evaluation of K-means[2]

For a given data set $\{x_1, x_2, ..., x_n\}$, let K-means partitions it in $\{S_1, S_2, ..., S_K\}$ then the objective is to minimize

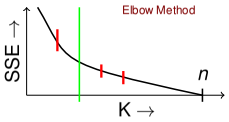
$$\underset{S}{\operatorname{argmin}} \sum_{i=1}^{K} \sum_{x \in S_i} dist^2(x, \mu_i)$$

where $\mu_i$ corresponds to $i^{th}$ centroid. $\mu_i = \frac{1}{|S_i|} \sum_{x \in S_i} x$

- Typical choice for *dist* function is Euclidean Distance

**How to proceed?**

- Choose a $K$ (How?)
  - Run K-means algorithm multiple times
  - Choose clusters corresponding to the one that minimized sum of squared error (SSE)
- If $K == n$, no error.
- Good clustering has smaller $K$




[2] Hamerly, Greg and Elkan, Charles, "Learning the k in k-means", pp 281–288, NIPS-2003



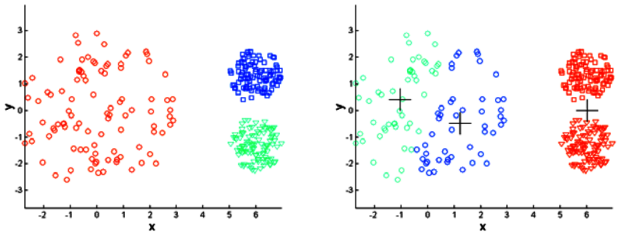
## Evaluation of K-means

- **Choosing K:** 1) Domain Knowledge, 2) Preprocessing with another algorithm, 3) Iteration on $K$
- **Initialization of Centers:** [3] 1) Random point in space, 2) Random point of data, 3) look for dense region, 4) Space uniformly in feature space
- **Cluster Quality:** 1) Diameter of cluster verses Inter-cluster distance, 2) Distance between members of a cluster and the cluster center, 3) Diameter of smallest sphere, 4) Ability to discover hidden patterns
- **Efficiently:** mini-batch K-Means

[3] *"k-means++: The advantages of careful seeding"*, David Arthur, Sergei Vassilvitskii, Stanford 2006



## Limitations of K-means

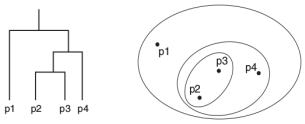


- Has problem when data has
  - Different size clusters
  - Different densities
  - Non-globular shape
- Handling Empty Clusters
- When there are outliers
- Updating Centroids Incrementally



## Other Approaches

- **Mini Batch K-Means** less computation and faster convergence
- **K-Medoids:** chooses data point as center and minimizes a sum of pairwise dissimilarities. Resistance to noise and/or outliers
- **Agglomerative Hierarchical Clustering:** repeatedly merging the two closest clusters until a single (Single Link)

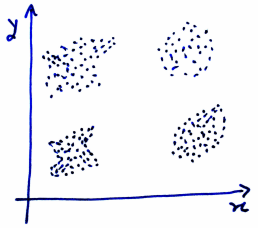



- **DBSCAN:** density-based clustering algorithm that produces a partitional clustering, in which the number of clusters is automatically determined by the algorithm.
- **More variations:** Affinity propagation, Mean Shift, Spectral Clustering, Ward hierarchical, Optics, Gaussian Mixture, Birch



## Clustering in dynamic databases

- Can we use $k$-Means clustering algorithm?




- Randomly choose $k$ data points as centroid
- Assign each data point to closest centroid
- Update centroid until converge
- Typically SSD (sum of square error) is optimized

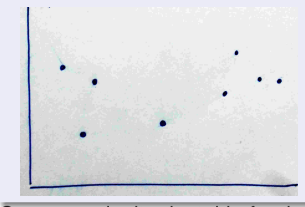
- What about PAM (partitioning around medoids)? NO
- Single/Average/Farthest link clustering?



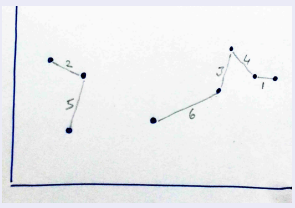
## Single Link

1. Starts with each point as cluster
2. Merges two nearest clusters $n - k$ times

Consider following data points in 2D space



Finally we get

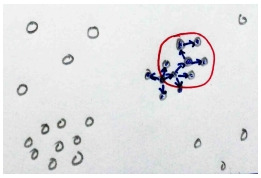


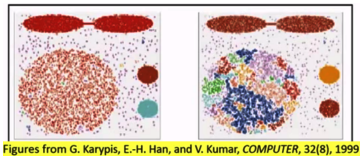

Can we make it adaptable for dynamic databases?

## DBSCAN

**DBSCAN** [4] (Density-Based Spatial Clustering of Applications with Noise) is a spatial clustering algorithm of KDD96

- Parameters (Eps/MinPts) and points (core/border/noise)
- Uses DFS





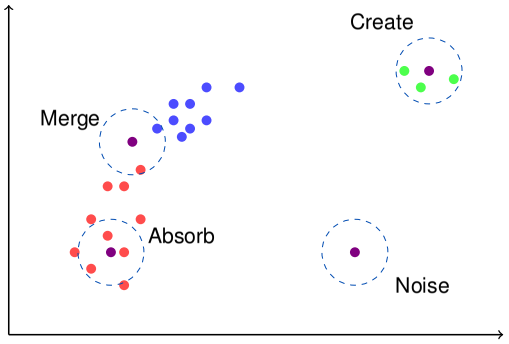
Figures from G. Karypis, E.-H. Han, and V. Kumar, *COMPUTER*, 32(8), 1999

- Advantage: clusters of arbitrary shape
- Disadvantage: Sensitive to parameters

[4] A density-based algorithm for discovering clusters in large spatial databases with noise, *M Ester, HP Kriegel, J Sander, X Xu*, KDD, 96(34), pp 226–231, 1996

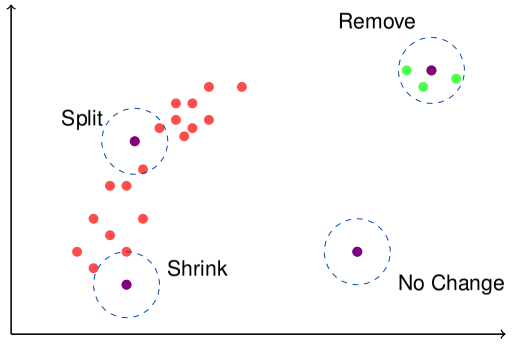


## Incremental DBSCAN (Addition)






## Incremental DBSCAN (Deletion)






## Incremental DBSCAN

- Insertion and deletion are treated separately
- Based on change in density in affected region, clusters are updated.
- Update cost is proportional to number of points in affected region that is high
- You may be doing redundant operations

**Differ update for some time**

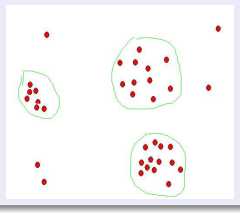
Assume periodic arrival of updates.

- Cluster new data
- Merge it with previous clusters (it is easy to see the density change)

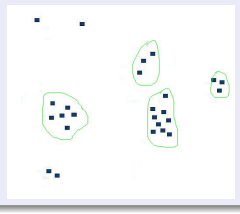


## Incremental DBSCAN

**Initial Database**



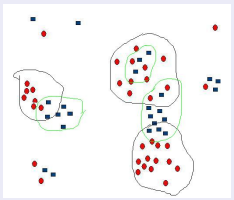
**New Date**



- Region based merging is applied

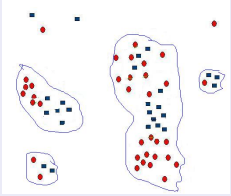


## Incremental DBSCAN



Overlapping of clusters made from original and the new data points looks as

- Point $p$ is in set of intersection $I'$ if $\exists p' \in D$ such that $p$ and $p'$ are neighbor
- It is necessary an sufficient to process all $p \in I'$
- Efficiently compute $I'$. How?



## A Question

**Example:** Assume you have a database that contains coordinates of points in a 2D plane. Now I give you one more point $P$ and ask you the following question.

Give me the point from database which is less than 3cm away from $P$.

- What approach you would follow?
- Evaluate distance from all points in database and sort
- Evaluate distance from all points in database and take minimum
- Something else?

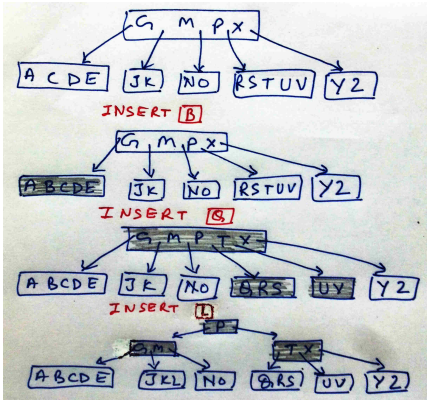


## Recall B-tree

- $k$ is a parameter specifying any non-root node can have $k-1$ to $2k-1$ number of keys
- B-Tree is a rooted tree. Every node $x$ has $n(x)$ number of keys with $key_1[x] \leq key_2[x] \leq ... \leq key_{n[x]}[x]$ and $leaf[x] = TRUE$ if it is a leaf node
- Internal node also contains $n(x)+1$ pointers $c_1[x], c_2[x], ...., c_{n[x]+1}[x]$
- For chosen $k > 1$, three exists at least $k-1$ and at most $2k-1$ keys at every node except root
- Height of tree $h \leq \log_t \frac{n+1}{2}$

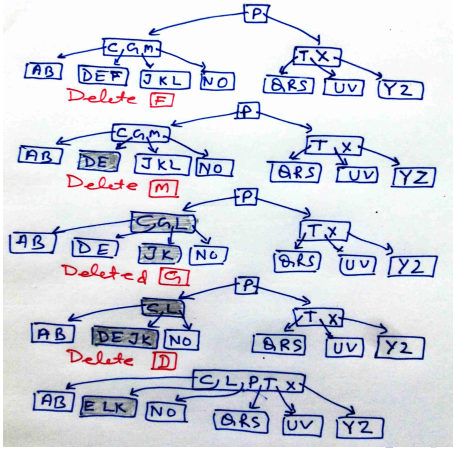


## B-tree (insertion)






## B-tree (deletion)






## Thank You!

**Thank you very much for your attention!**

**Queries ?**