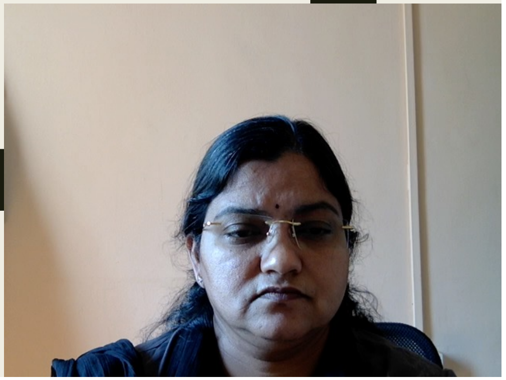
LECTURE 35

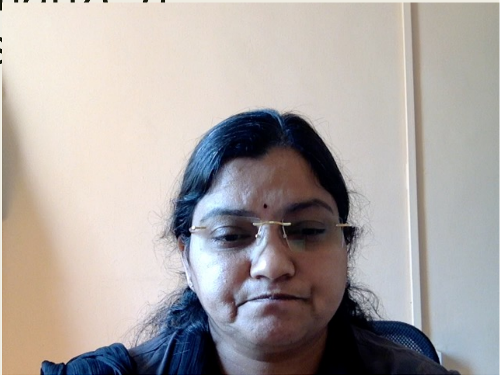
Dr.Vani V

Source : Han & Kambar , Chapter 10, Data Mining Concepts & Techniques, BIRCH & DBScan



BIRCH: Multiphase Hierarchical Clustering Using Clustering Feature Trees

- **Balanced Iterative Reducing and Clustering using Hierarchies** (BIRCH) is designed for clustering a large amount of numeric data by **integrating hierarchical clustering** (at the initial micro-clustering stage) **and other clustering methods such as iterative partitioning** (at the later macro-clustering stage).
- It overcomes the two difficulties in agglomerative clustering methods: (1) scalability and (2) the inability to undo what was done in the previous step.
- BIRCH uses the notions of clustering feature to summarize a cluster, and **clustering feature tree (CF-tree)** to repres
cluster hierarchy.



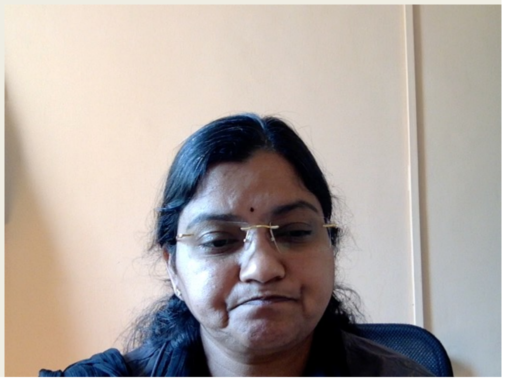
BIRCH: Multiphase Hierarchical Clustering Using Clustering Feature Trees

- The structures help the clustering method achieve good speed and scalability in large or even streaming databases.
- Make it effective for incremental and dynamic clustering of incoming objects.

Consider a cluster of n d-dimensional data objects or points. The clustering feature (CF) of the cluster is a 3-D vector summarizing information about clusters of objects. It is defined as

$$CF = \langle n, LS, SS \rangle,$$

where LS - Linear Sum of the n points
SS – Square Sum of the data points



BIRCH: Multiphase Hierarchical Clustering Using Clustering Feature Trees

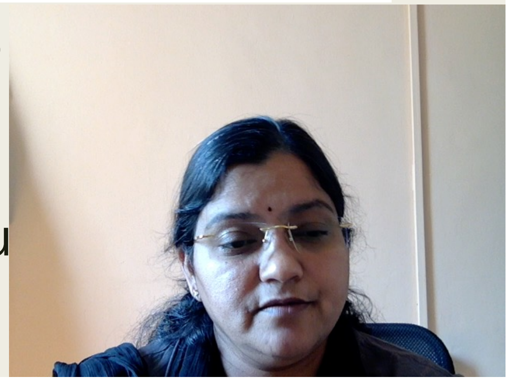
- A clustering feature is a summary of the statistics for the given cluster.
- Using a clustering feature,derive many useful statistics of a cluster.
- For example, the cluster’s centroid, x0, radius, R, and diameter, D, are

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

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

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

- Here, R -> average distance from member objects to
- D -> the average pairwise distance within a cluster.
- Both R and D reflect the tightness of the cluster arou



BIRCH: Multiphase Hierarchical Clustering Using Clustering Feature Trees

- Summarizing a cluster using the clustering feature can avoid storing the detailed information about individual objects or points.
- Only need a constant size of space to store the clustering feature. This is the key to BIRCH efficiency in space.
- Clustering features are additive. That is, for two disjoint clusters, C1 and C2, with the clustering features
  - *CF1 = {n1,LS1,SS1} and CF2 = {n2,LS2,SS2}, respectively*
- The clustering feature for the cluster that formed by merging C1 and C2 is simply

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

BIRCH : Example

- Clustering feature. Suppose there are three points, (2,5), (3,2), and (4,3), in a cluster, C1. The clustering feature of C1 is

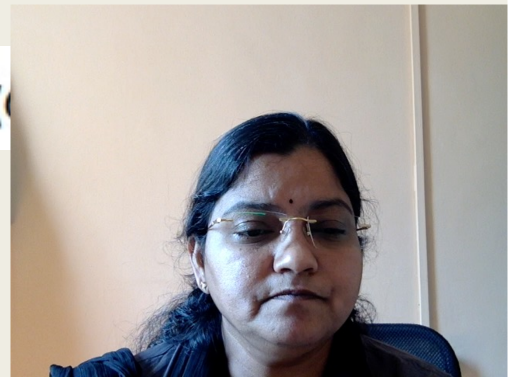
$$CF_1 = \langle 3, (2+3+4, 5+2+3), (2^2+3^2+4^2, 5^2+2^2+3^2) \rangle = \langle 3, (9,10), (29,38) \rangle.$$

- Suppose that C1 is disjoint to a second cluster, C2,

  where CF2 (3, (35,36), (417,440)).

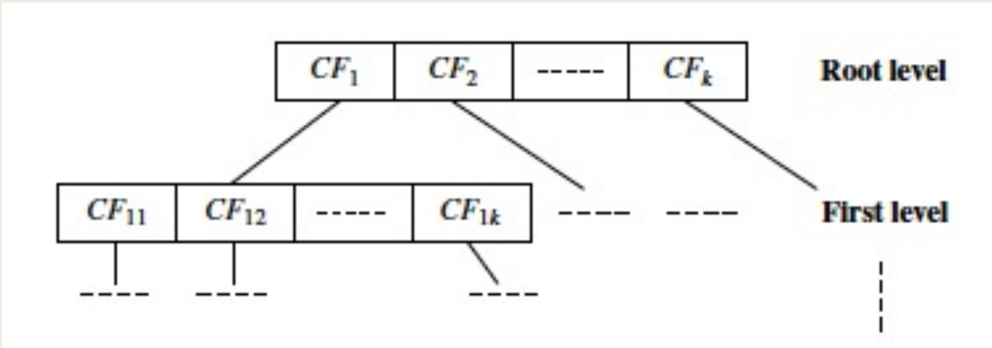
- The clustering feature of a new cluster, C3, that is formed by merging C1 and C2, is derived by adding CF1 and CF2. That is,

$$CF_3 = \langle 3+3, (9+35, 10+36), (29+417, 38+440) \rangle = \langle$$

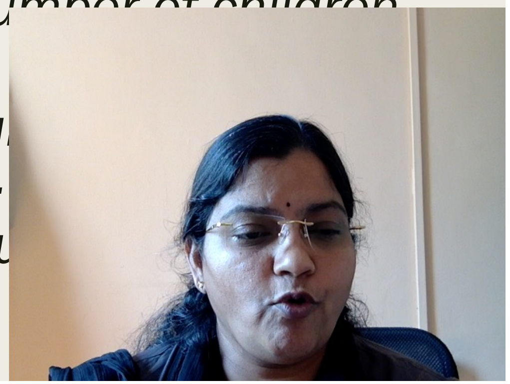


BIRCH: CF Tree

- A CF-tree is a height-balanced tree that stores the **clustering features** for a hierarchical clustering.

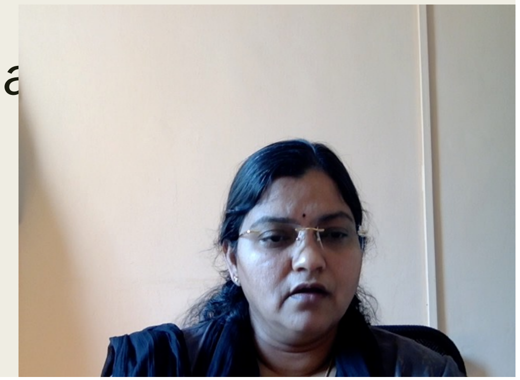


- By definition, a non-leaf node in a tree has descendants or "children."
- The non-leaf nodes store sums of the CFs of their children, and thus summarize clustering information about their children.
- A CF-tree has two parameters: branching factor, **B**, and threshold, **T.**
  - *The* ***branching factor*** *specifies the maximum nu*
    per non-leaf node.
  - *The* ***threshold parameter*** *specifies the maximu*
    subclusters stored at the leaf nodes of the tree.
  - *These two parameters implicitly control the resu*



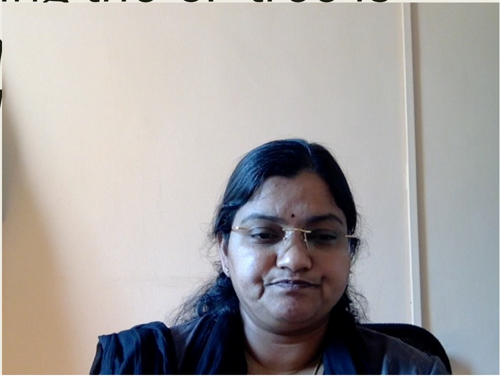
BIRCH Phases…

- Given a limited amount of main memory, an important consideration in BIRCH is to minimize the time required for input/output (I/O).
- BIRCH applies a multiphase clustering technique:
  - *A single scan of the data set yields a basic, good clustering, and*
  - *one or more additional scans can optionally be used to further improve the quality. T*
- The primary phases are
  - ***Phase 1:*** *BIRCH scans the database to build an initial in-memory CF-tree, which can be viewed as a multilevel compression of the data that tries to preserve the data's inherent clustering structure.*
  - ***Phase 2:*** *BIRCH applies a (selected) clustering a*
    cluster the leaf nodes of the CF-tree,



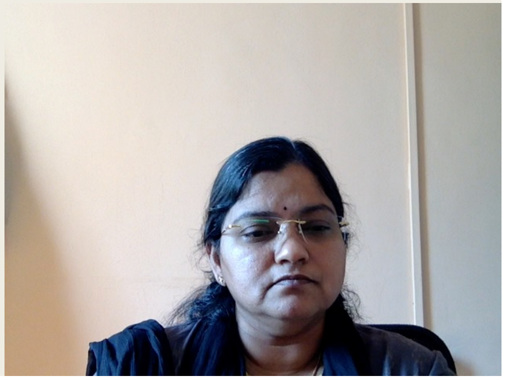
BIRCH Phases...

- For Phase 1, the CF-tree is built dynamically as objects are inserted.
  - *The method is incremental.*
  - *An object is inserted into the closest leaf entry (subcluster).*
  - *If the diameter of the subcluster stored in the leaf node after insertion is larger than the threshold value, then the leaf node and possibly other nodes are split.*
  - *After the insertion of the new object, information about the object is passed toward the root of the tree.*
  - *The size of the CF-tree can be changed by modifying the threshold.*
  - *If the size of the memory that is needed for storing the CF-tree is larger than the size of the main memory, then a value can be specified, and the CF-tree is rebuil*



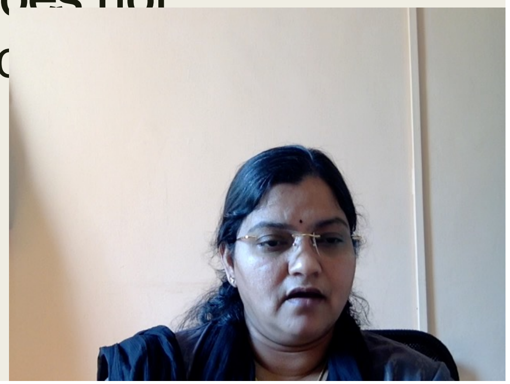
BIRCH Phases

- The rebuild process is performed by building a new tree from the leaf nodes of the old tree. Thus, the process of rebuilding the tree is done without the necessity of rereading all the objects or points.
- This is like the insertion and node split in the construction of **B+-trees.** Therefore, for building the tree, data must be read just once. Some heuristics and methods have been introduced to deal with outliers and improve the quality of CF-trees by additional scans of the data.
- Once the CF-tree is built, any clustering algorithm, such as a typical partitioning algorithm, can be used with the CF-tree in Phase 2.



How effective is BIRCH?

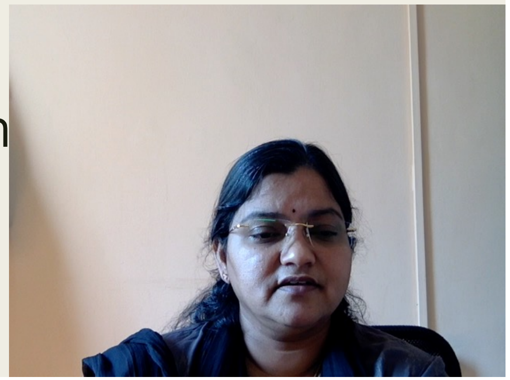
- The time complexity of the algorithm is O(n), where n is the number of objects to be clustered.
- Experiments have shown the linear scalability of the algorithm with respect to the number of objects, and good quality of clustering of the data. However, since each node in a CF-tree can hold only a limited number of entries due to its size, a CF-tree node does not always correspond to what a user may consider a natural cluster.
- If **the clusters are not spherical in shape, BIRCH does not perform well because** it uses the notion of radius
to control the boundary of a cluster.



Density-Based Methods

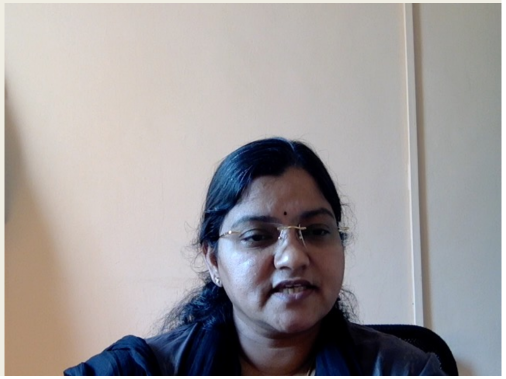
- Partitioning and hierarchical methods are designed to find spherical-shaped clusters.
- They have difficulty finding clusters of arbitrary shape such as the "S" shape and oval clusters

- Given such data, they would likely inaccurately iden
regions, where noise or outliers are included in the



Density-Based Methods

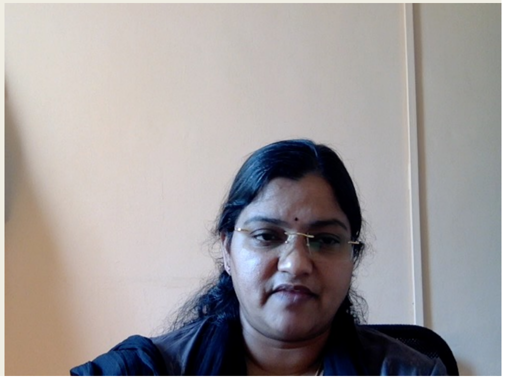
- To find clusters of arbitrary shape,
  - *model clusters as dense regions in the data space, separated by sparse regions.*
  - *This is the main strategy behind* ***density-based clustering methods, which can discover clusters of nonspherical shape.***



DBSCAN…

"How can we find dense regions in density-based clustering?"

- *The density of an object o can be measured by the number of objects close to o.*
- *DBSCAN (**Density-Based Spatial Clustering of Applications with Noise**) finds core objects, that is, objects that have dense neighborhoods.*
- *It connects core objects and their neighborhoods to form dense regions as clusters.*

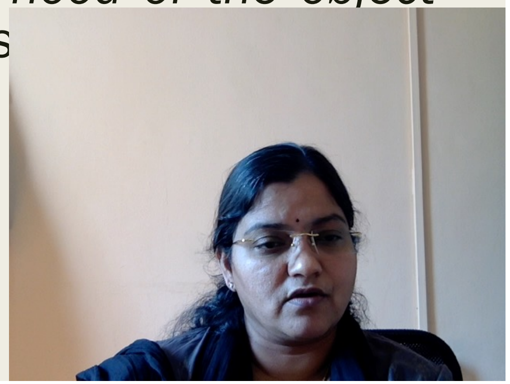


DBSCAN…

"How does DBSCAN quantify the neighborhood of an object?"

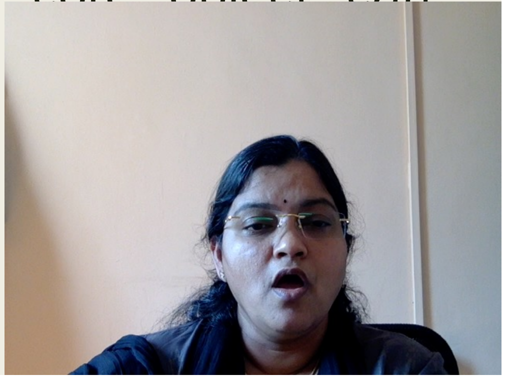
- *A user-specified parameter $\epsilon > 0$ is used to specify the radius of a neighborhood we consider for every object.*
- *The ϵ-neighborhood of an object o is the space within a radius ϵ centered at o.*
- *Due to the fixed neighborhood size parameterized by ϵ, the density of a neighborhood can be measured simply by the number of objects in the neighborhood.*
- *To determine whether a neighborhood is dense or not, DBSCAN uses another user-specified parameter, MinPts, which specifies the density threshold of dense regions.*
- *An object is a core object if the ϵ-neighborhood of the object contains at least MinPts objects. Core objects*

  dense regions.



DBSCAN…

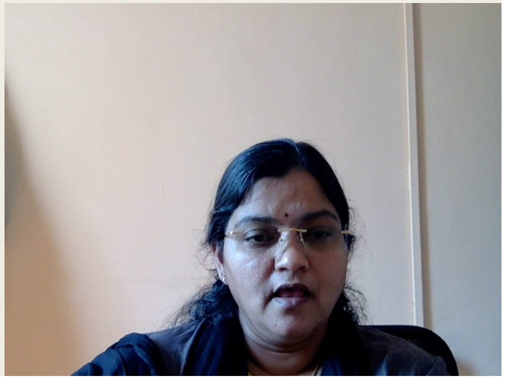
- Given a set, D, of objects, we can identify all core objects with respect to the given parameters, ϵ and MinPts.
- The clustering task is therein reduced to using core objects and their neighborhoods to form dense regions, where the dense regions are clusters.
- For a core object q and an object p, we say that p is directly density-reachable from q (with respect to ϵ and MinPts) if p is within the ϵ -neighborhood of q.
- An objectp is directly density-reachable from another object q if and only if q is a core object and p is in the ϵ -neighborhood of q.
- Using the directly density-reachable relation, a core object can “bring” all objects from its ϵ -neighborhood into a



DBSCAN…

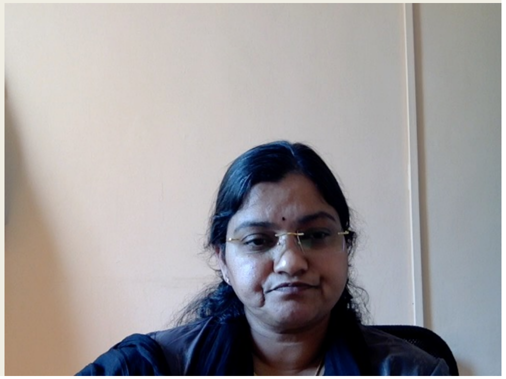
"How can we assemble a large dense region using small dense regions centered by core objects?"

- *In DBSCAN, p is density-reachable from q (with respect to ϵ and MinPts in D) if there is a chain of objects p1,…,pn, such that p1 = q, pn = p, and p_{i+1} is directly density-reachable from pi with respect to ϵ and MinPts, for $1 \leq i \leq n$, pi ∈D.*
- *Note that density-reachability is not an equivalence relation because it is not symmetric. If both o1 and o2 are core objects and o1 is density-reachable from o2, then o2 is density-reachable from o1. However, if o2 is a core object but o1 is not, then o1 may be density-reachable from o2, but not vice versa.*



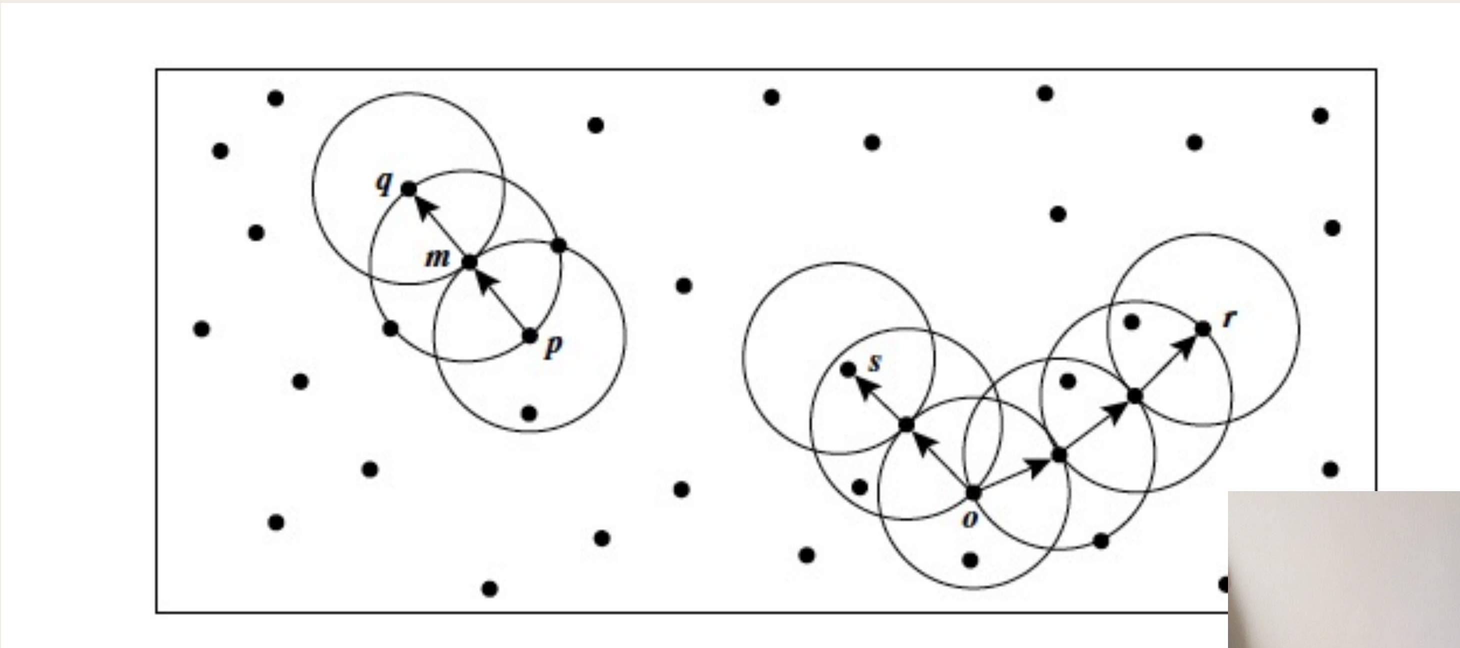
DBSCAN…

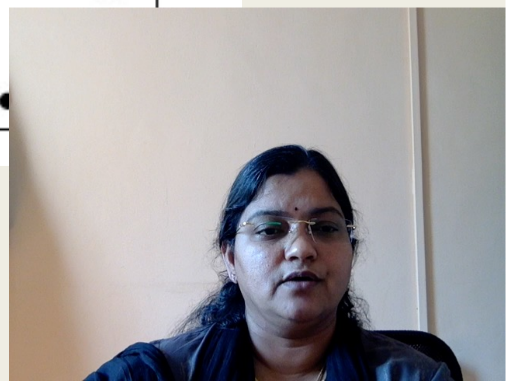
- To connect core objects as well as their neighbors in a dense region, **DBSCAN uses the notion of density-connectedness.**
- Two objects p1,p2 $\in$ D are density-connected with respect to ϵ and MinPts if there is an object q $\in$ D such that both p1 and p2 are density reachable from q with respect to ϵ and MinPts.
- **Unlike density-reachability, density connectedness is an equivalence relation.** It is easy to show that, for objects o1, o2, and o3, if o1 and o2 are density-connected, and o2 and o3 are density-connected, then so are o1 and o3.



Example: Density-reachability and density-connectivity

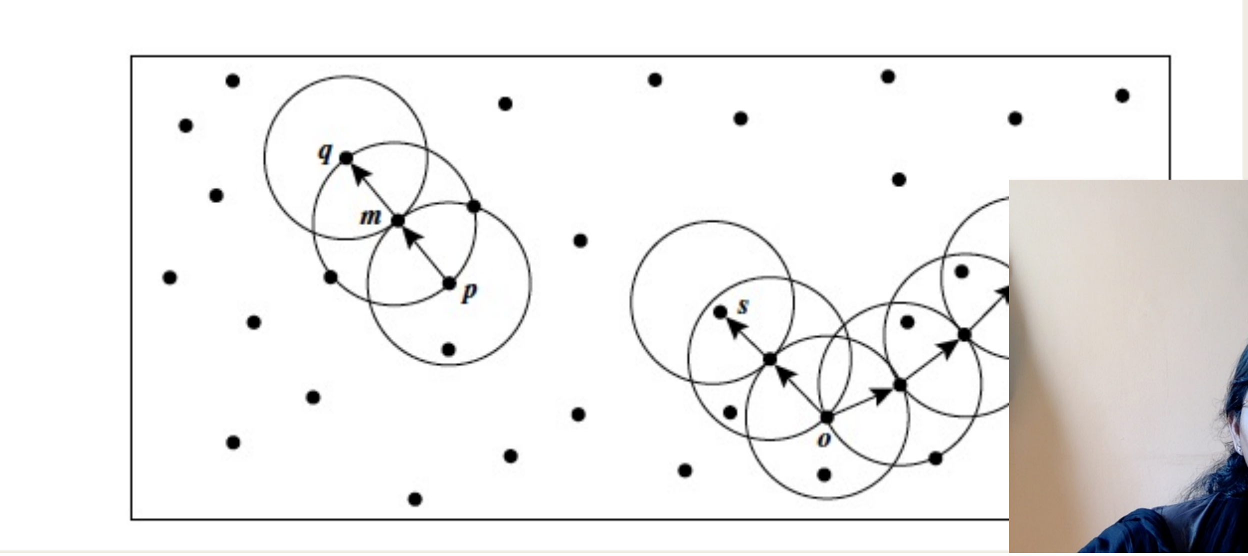
Consider Figure for a given ϵ represented by the radius of the circles, and, say, let MinPts = 3.

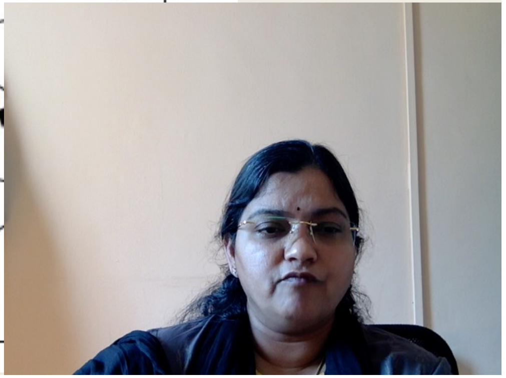




Example: Density-reachability and density-connectivity

- Of the labeled points, m,p,o,r are core objects because each is in an ϵ -neighborhood containing at least three points.
- Object q is directly density-reachable from m. Object m is directly density-reachable from p and vice versa.
- Object q is (indirectly) density-reachable from p because q is directly density reachable from m and m is directly density-reachable from p. However, p is not density reachable from q because q is not a core object.
- Similarly, r and s are density-reachable from o and o is density-reachable from r. Thus, o, r, and s are all density-connected.





DBSCAN…

"How does DBSCAN find clusters?"

- *Initially, all objects in a data set D are marked as "unvisited."*
- *DBSCAN randomly selects an unvisited object p, marks p as "visited," and checks whether the ϵ-neighborhood of p contains at least MinPts objects.*
- *If not, p is marked as a noise point. Otherwise, a new cluster C is created for p, and all the objects in the ϵ-neighborhood of p are added to a candidate set, N.*
- *DBSCAN iteratively adds to C those objects in N that do not belong to any cluster. In this process, for an object p0 in N that carries the label "unvisited," DBSCAN marks it as "visited" and checks its ϵ-neighborhood. If the ϵ-neighborhood of p0 has at leastMinPts objects, those objects in the ϵ-neighborhood of p0 are added to N.*
- *DBSCAN continues adding objects to C until C can no longer be expanded, that is, N is empty. At this time, cluster C is completed, and thus is output.*
- *To find the next cluster, DBSCAN randomly selects ar the remaining ones. The clustering process continue visited.*

Algorithm: DBSCAN: a density-based clustering algorithm.

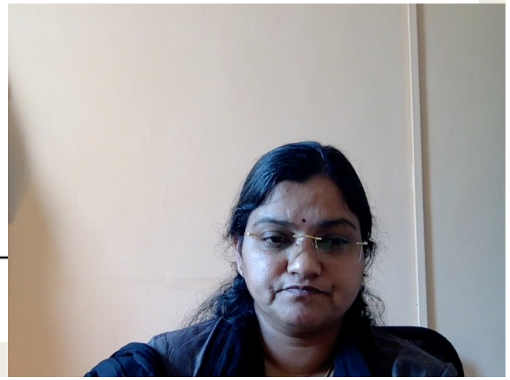
Input:

- D: a data set containing n objects,
- ϵ: the radius parameter, and
- $MinPts$: the neighborhood density threshold.

Output: A set of density-based clusters.

Method:

```
mark all objects as unvisited;
do
    randomly select an unvisited object p;
    mark p as visited;
    if the ϵ-neighborhood of p has at least MinPts objects
        create a new cluster C, and add p to C;
        let N be the set of objects in the ϵ-neighborhood of p;
        for each point p′ in N
            if p′ is unvisited
                mark p′ as visited;
                if the ϵ-neighborhood of p′ has at least MinPts points,
                add those points to N;
            if p′ is not yet a member of any cluster, add p′ to C;
        end for
        output C;
    else mark p as noise;
until no object is unvisited;
```



DBSCAN

- If a spatial index is used, the computational complexity of DBSCAN is $O(n\log n)$, where n is the number of database objects. Otherwise, the complexity is $O(n^2)$.
- With appropriate settings of the user-defined parameters, and MinPts, the algorithm is effective in finding arbitrary-shaped clusters.