Requirement of Clustering Algorithms
- minimum requirement of domain knowledge
- discovery of clusters with arbitrary shape with good efficiency on large databases

DBSCAN - designed to discover clusters of arbitrary shape relying on density based notion of clusters.
- Requires only one i/p parameter.

Spatial Database is large ⇒ automated knowledge discovery.
We solve the class task of class identification.

Two types of clustering Algorithms - Partitioning and Hierarchical.

Partitioning Algorithms partitions the Database D of n objects into k clusters, k being the i/p parameter ∴ some domain knowledge is required. Usually follows the iterative strategy to partition, where each cluster is determined by it's gravity center (KNN) or the point nearest to it's center (K-medoid algorithms)

Shape of each cluster in a partitioning algorithm is convex which is very restrictive.
eg. CLARANS

Hierarchical Algorithms create a hierarchical decomposition of D, which is represented by a dendogram, a tree which keeps splitting the D into smaller subsets until each subset contains only one object, each node of the tree is a cluster.
Leaves to Root → (agglomerative approach)
Root to Leaves → (divisive approach

eg. Ejcluster, critical distance, $O(n^2)$

A Density Based Notion of Clusters

Database D ∈ k-dimentional space S.

Key-idea – For each point of a cluster the neighbourhood of a given radius has to contain a minimum number of points. Shape of the cluster depends upon the distance func^n chosen, dist(p, q), between two points p and q.

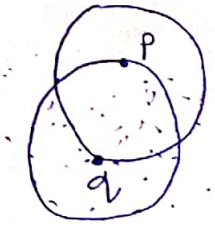
(Eps-neighborhood of a point) – the Eps-neighborhood of a point p denoted by $N_{eps}(p)$ is defined as

$$N_{eps}(p) = \{ q \in D \mid dist(p,q) \leq Eps \}$$

Naive approach would be to set MinPts constant but this method fails ∵ we have two kinds of points in the cluster core and border, core pts contains more points in their neighborhood than the border points.

- A point p is "directly density-reachable" from a point q wrt Eps, MinPts if
  i) $p \in N_{eps}(q)$
  ii) $|N_{eps}(q)| \geq MinPts$ (core point condition)

This condition is ~~sym~~ ~~sym~~ symmetric iff they belong to core points.



p is border point
q is core point
q is not directly density-reachable from p.

- A point p is "density-reachable" from a point q wrt Eps and MinPts if there is a chain of points $p_1, \ldots, p_n$ s.t. $p_1 = q$, $p_n = p$ & p_{i+1} is directly density-reachable from p_i

Transitive but not symmetric but is symmetric for core points.

- A point p is density-connected to a point q wrt Eps and MinPts if there is a point O s.t. both p and q are density-reachable from O wrt Eps and MinPts.

Symmetric Relation. ~~Re~~ Reflexive for density reachable points.

- Cluster is a set of density connected points which is maximal wrt density reachability.

Noise is simply the set of points in D not belonging to any of its clusters

- Let D be a database of points. A cluster C wrt Eps and MinPts is a non-empty subset of D satisfying the following condition-
  1) $\forall p, q$: if $p \in C$ and q is density-reachable from p wrt Eps and MinPts, then $q \in C$ (Maximality)
  2) $\forall p, q \in D$: p is density connected to q wrt Eps and MinPts (Connectivity)

- Let $C_1, \ldots, C_k$ be the clusters of a ~~point~~ Database D, wrt Eps_i and $MinPts_i$ $i = 1, \ldots, k$, then Noise is the set of points in the database D not belonging to any cluster C_i ; i.e. noise = $\{p \in D \mid \forall i : p \notin C_i\}$

Cluster C contains atleast MinPts.

Given Eps, MinPt (i) choose arbitrary point in the database following the core point condition i.e. the "seed" (ii) ~~R~~ Retrieve all the pts that are directly reachable from ~~pt~~ seed.

Lemma - if $p \in D$ s.t. $|N_{Eps}(p)| > MinPts$ and $O = \{O \in D \mid$ s.t. O is density reachable from p wrt Eps and MinPts$\}$ then O is a cluster wrt Eps and MinPts.

Lemma 2: Let C be any cluster in D and let $p \in C$ st $|N_{Eps}(p)| \geq MinPts$. Then C equals to the set $O = \{o \mid o$ is density-reachable from p wrt MinPts and Eps$\}$

i.e Cluster C is uniquely determined by any of its core points

DBSCAN - Density Based Spacial Clustering of Applications with Noise.

Uses global value of Eps and Minpts which is determined by the least dense cluster in the database D.

Algorithm

Starts with an arbitrary point $p \in D$ and retrieves all the points which are density reachable from p wrt Eps and MinPts. if p is core pt this procedure produces a cluster o/w algorithm moves to other pts $\in D$.

Distance between two sets of pts S_1 and S_2 be defined as $dist(S_1, S_2) = \min\{dist(p,q) \mid p \in S_1, q \in S_2\}$, then the two sets of points having at least the density of the thinnest cluster will be seperated from each other iff distance betn them is ~~Separated by~~ Eps Larger than eps

```
DBSCAN (SetOfPoints, Eps, MinPts)
//SetOfPoints is unclassified
    ClusterID := NextID(Noise);
    For i from 1 to SetOfPoints.size Do
        Point := SetOfPoints.get(i);
        If Point.CLID = UNCLASSIFIED THEN
            If ExpandCluster(SetOfPoints, Point, ClusterID, Eps, MinPts)
                ClusterID := NextID(ClusterID)
            End If
        EndIf
    EndFor
End; //DBSCAN
```

```
ExpandCluster (SetOfPoints, Point, CId, Eps, MinPts): Boolean;
    Seeds: SetOfPoints.regionQuery (Points, Eps);
    If seeds.size < MinPts Then  // no core point
        SetOfPoint.changeCID (point, NOISE);
        Return False;
    Else // all points in the seed are density reachable from Point
        SetOfPoints.changeCIds (seeds, CId);
        seeds.delete (Point);
        While seeds <> Empty Do
            currentP := seeds.first();
            result := SetOfPoints.regionQuery (currentP, Eps);
            If result.size >= MinPts Then
                For i from 1 to result.size Do
                    resultP := result.get (i)
                    If resultP.CId IN (UNCLASSIFIED, NOISE) Then
                        If resultP.CId = UNCLASSIFIED Then
                            seeds.append (resultP);
                        EndIf;
                        SetOfPoints.changeCId (resultP, CId)
                    EndIf; // Unclassified or Noise.
                EndFor;
            EndIf; // Result Size >= MinPts
            seeds.delete (currentP);
        EndWhile; // seed <> Empty
        Return True;
    EndIf;
End;
```

SetOfPoints.regionQuery (Point, Eps) – returns eps neighbourhood of points in SetOfPoints as a list of points. (eg use R* trees)

Average runtime complexity of RegionQuery is O(logN) and for each of the n points we have at most on regionquery

∴ Average Runtime complexity of DBSCAN is O(NlogN)

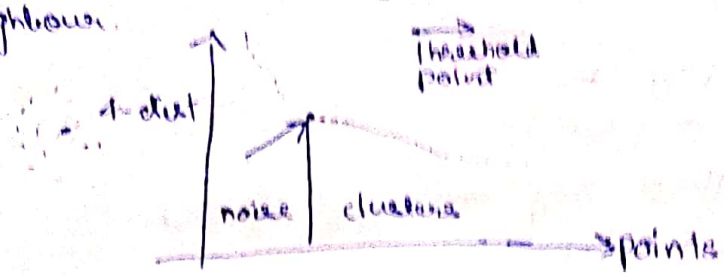
If p is a border point of two clusters C_1 and C_2, p is assigned to the cluster discovered first.

- Determining the parameters Eps and MinPts

(of the smallest cluster in the database)

d = distance of a point p to its k^{th} nearest neighbour, then d-neighbourhood of point p contains k+1 points for almost all points p.

k-dist : D → ℝ mapping each point to the k^{th} distance from its k^{th} nearest neighbour.



By inspection MinPts parameter is set to 4 for Databases.

Steps to computing Eps and MinPts =

- System computes and displays 4-dist graph for the Database
- System receives a proposed threshold pt p for the % of noise entered
- Either accepts or selects another point in the D as the threshold value.

Eps = 4-dist(p)