PRIMROSE
Machine Learning Experts

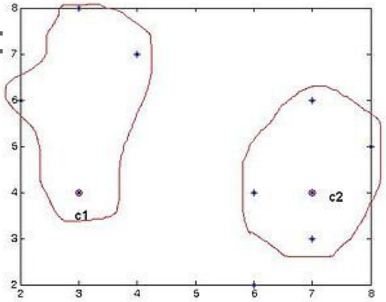
Lecture 22: Cluster analysis- Density based clustering

Overview

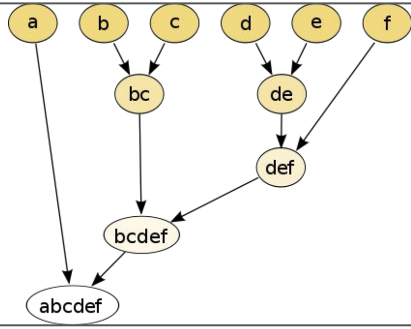
- Road map (again)
- Centroid methods disadvantages
- DBCSAN
    - Definitions
    - Algo
    - Advantages / Disadvantages
- OPTICS
    - Definitions
    - Algo

Road map

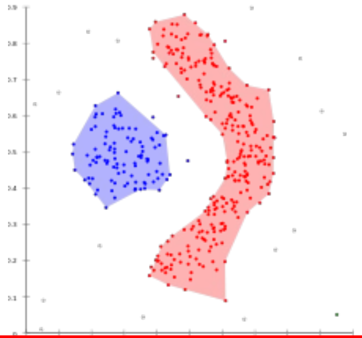
Centroid based clustering: K-means, K-medians, K-medoids



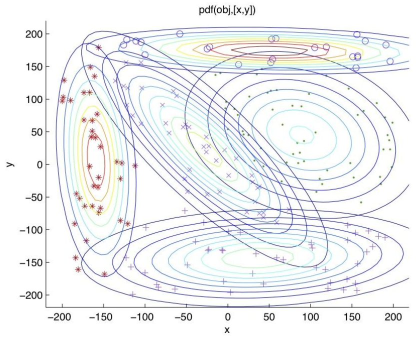
Hierarchical clustering: agglomerative/divisive (single linkage, complete linkage, UPGMA)



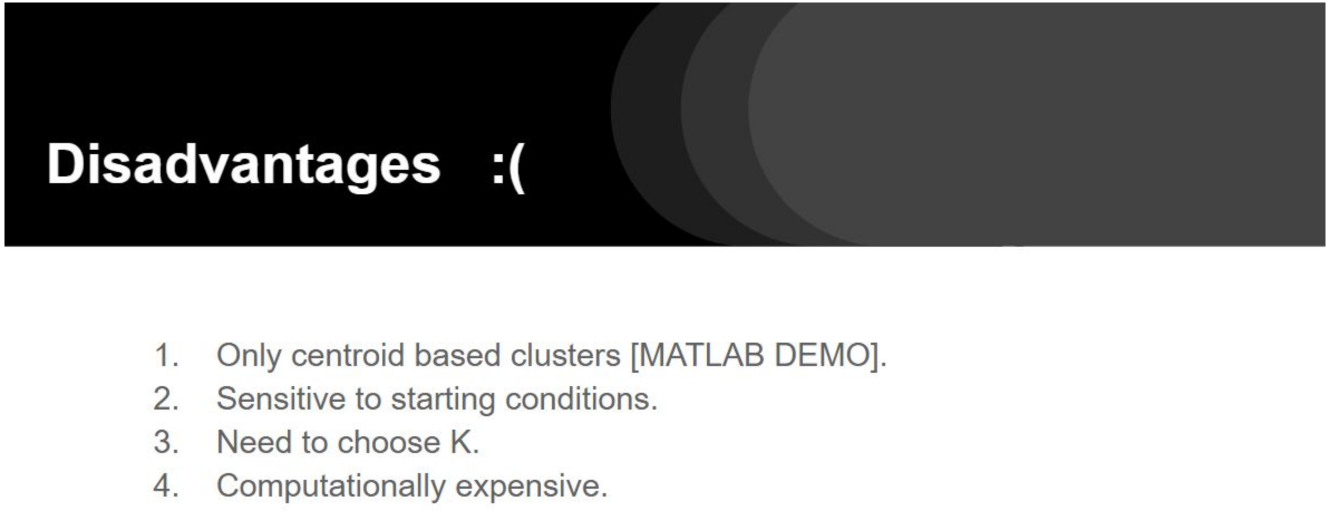
Density based clustering: DBSCAN, OPTICS.



Distribution based clustering: GMM.



K-means disadvantages- remember?



Disadvantages :(

1. Only centroid based clusters [MATLAB DEMO].
2. Sensitive to starting conditions.
3. Need to choose K.
4. Computationally expensive.

Another Kmeans disadvantage

No robustness to noise:
Every single data instance is classified to some cluster.

Moreover, this noise can shift the centroid position and damage the clustering!

DBSCAN: Density Based Spatial Clustering of Applications with Noise

Instead of distance from centroid, let us use density.

In 2014, the algorithm was awarded the test of time award (an award given to algorithms which have received substantial attention in theory and practice) at the leading data mining conference, KDD.

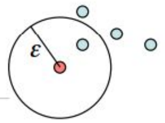
Definitions:

Requires 2 hyperparameters:

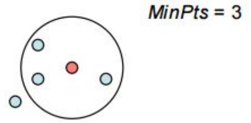
ε, *minPts*

(And the appropriate *distance* metric)

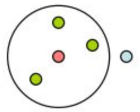
An object p is in the **ε-neighborhood** of q if the distance from p to q is less than ε.



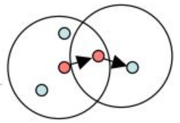
A **core object** has at least *MinPts* in its ε-neighborhood.



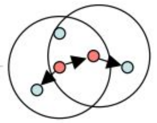
An object p is **directly density-reachable** from object q if q is a core object and p is in the ε-neighborhood of q.



An object p is **density-reachable** from object q if there is a chain of objects $p_1, \ldots, p_n$, where $p_1 = q$ and $p_n = p$ such that p_{i+1} is directly density reachable from p_i.

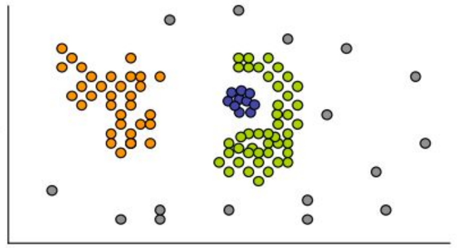


An object p is **density-connected** to object q if there is an object o such that both p and q are **density-reachable** from o.



A **cluster** is a set of density-connected objects which is maximal with respect to density-reachability.

Noise is the set of objects not contained in any cluster.

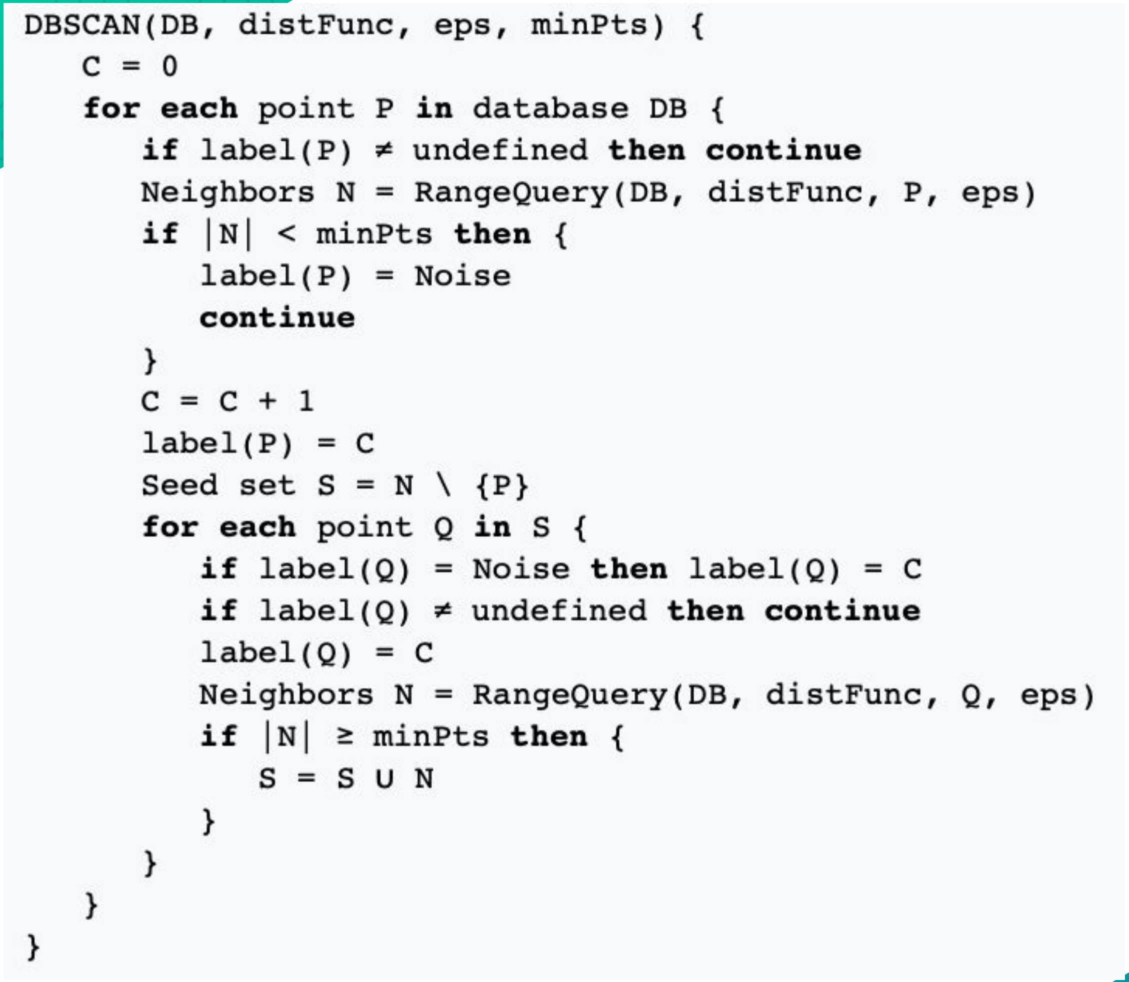


How to find all clusters

Given epsilon and minPts,

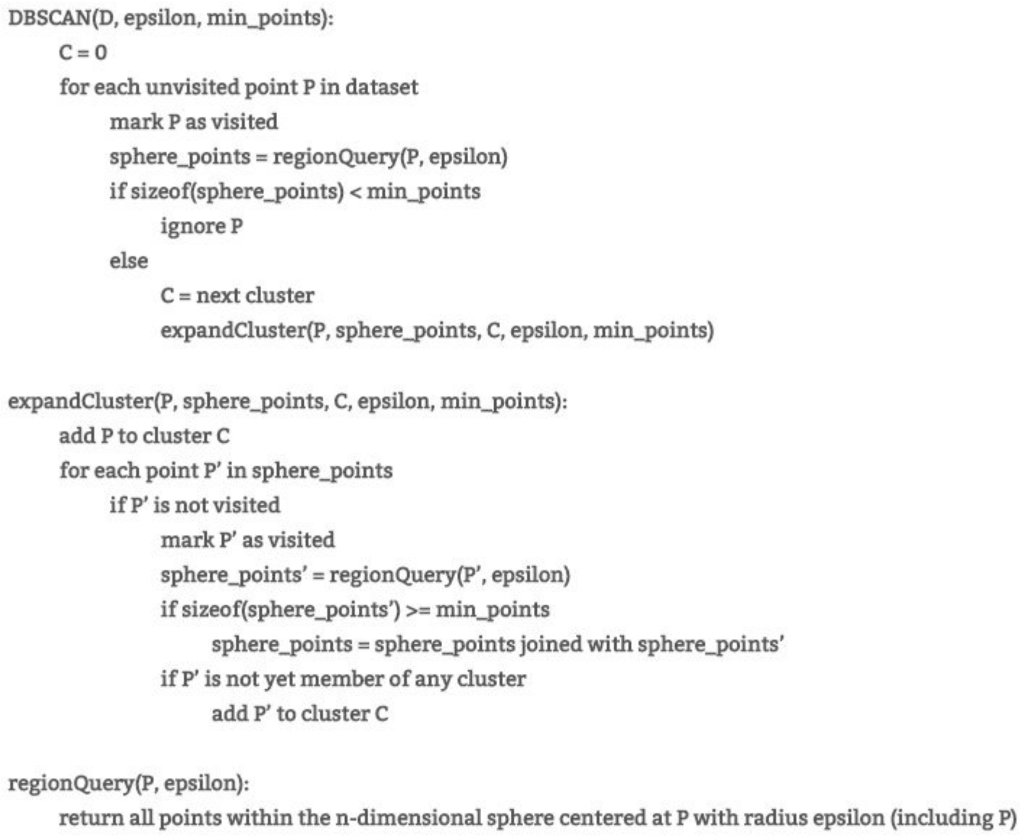
0. Start from a random instance.
1. Find epsilon-neighborhood.
2. If more than minPts this is a core, define a new cluster and add the neighborhood instances to it. (If not, move on).
3. For each instance in the cluster:
   3.1. Find epsilon-neighborhood.
   3.2. If more than minPts this is a core: Add neighborhood instances to cluster (They will be visited in the current loop!)

4. Go to 0 and repeat till you have unvisited and unclassified points



```
DBSCAN(DB, distFunc, eps, minPts) {
    C = 0
    for each point P in database DB {
        if label(P) ≠ undefined then continue
        Neighbors N = RangeQuery(DB, distFunc, P, eps)
        if |N| < minPts then {
            label(P) = Noise
            continue
        }
        C = C + 1
        label(P) = C
        Seed set S = N \ {P}
        for each point Q in S {
            if label(Q) = Noise then label(Q) = C
            if label(Q) ≠ undefined then continue
            label(Q) = C
            Neighbors N = RangeQuery(DB, distFunc, Q, eps)
            if |N| ≥ minPts then {
                S = S ∪ N
            }
        }
    }
}
```

Pseudo-code:



```
DBSCAN(D, epsilon, min_points):
    C = 0
    for each unvisited point P in dataset
        mark P as visited
        sphere_points = regionQuery(P, epsilon)
        if sizeof(sphere_points) < min_points
            ignore P
        else
            C = next cluster
            expandCluster(P, sphere_points, C, epsilon, min_points)

expandCluster(P, sphere_points, C, epsilon, min_points):
    add P to cluster C
    for each point P' in sphere_points
        if P' is not visited
            mark P' as visited
            sphere_points' = regionQuery(P', epsilon)
            if sizeof(sphere_points') >= min_points
                sphere_points = sphere_points joined with sphere_points'
            if P' is not yet member of any cluster
                add P' to cluster C

regionQuery(P, epsilon):
    return all points within the n-dimensional sphere centered at P with radius epsilon (including P)
```

[DEMO]

Notes about more clusters

- When done with one cluster, it will never be revisited again! Furthermore, no other instance will ever be classified to this cluster.
- When done with a cluster, simply go to the next unclassified instance and start a new cluster.
- If a point does not have minPts in the neighborhood, it is not necessarily noise. It may be classified to some cluster later on.
- Noise are the points left after all points were visited.
- You only have to visit each instance once!

Is DBSCAN sensitive to starting conditions?

Well, only a little bit:

A border point (non core) can be in the neighborhood of more than one core points that are not from the same cluster.
In such case, it will be classified to the cluster created first or last (depending on how algorithm was built).

Possible solution: DBSCAN* which is the same, but only core points are allowed in clusters. Border points are noise

Advantages :)

1. DBSCAN does not require one to specify the number of clusters.
2. DBSCAN can find arbitrarily shaped clusters.
3. DBSCAN has a notion of noise, and is robust to outliers.
4. DBSCAN is mostly insensitive to the ordering of the points in the database.
5. DBSCAN requires just two hyper-parameters

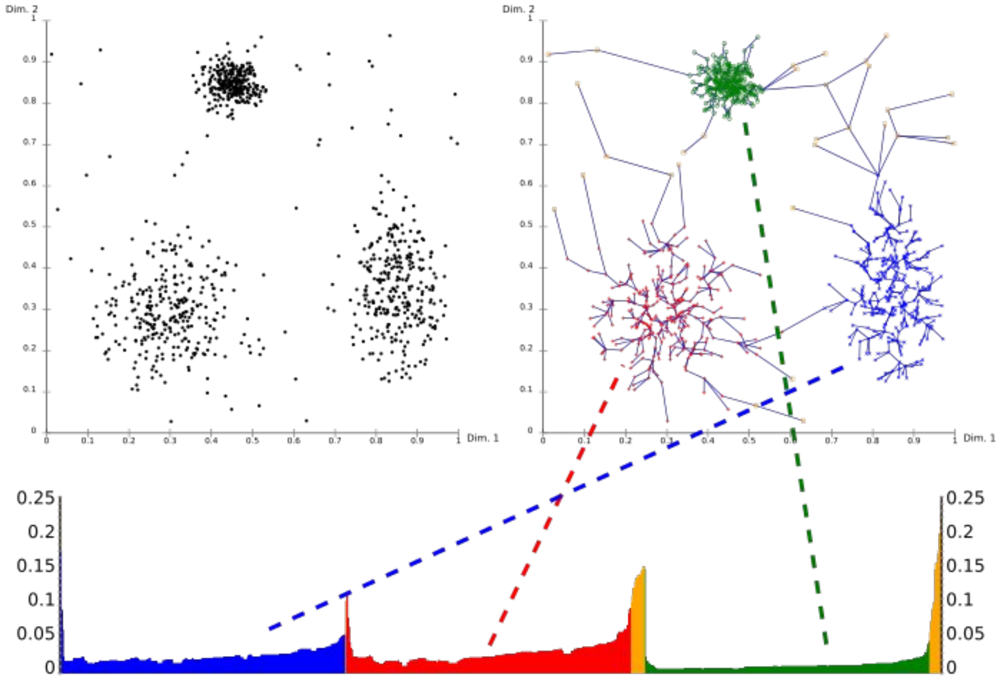
Disadvantages :(

1. DBSCAN is not entirely deterministic.
2. DBSCAN cannot cluster data sets well with large differences in densities.
3. DBSCAN has two hyper-parameters
4. DBSCAN requires computation time and memory for large amount of data

Evaluation metrics

sklearn.metrics.rand_score — scikit-learn 1.1.2 documentation for the case when we know the true labels.

OPTICS - next presentation


Dim. 2
Dim. 1
Dim. 2
Dim. 1