Clustering Algorithm (DBSCAN)

Prof Love Arora

BITS Pilani
Pilani | Dubai | Goa | Hyderabad

DBSCAN algorithm

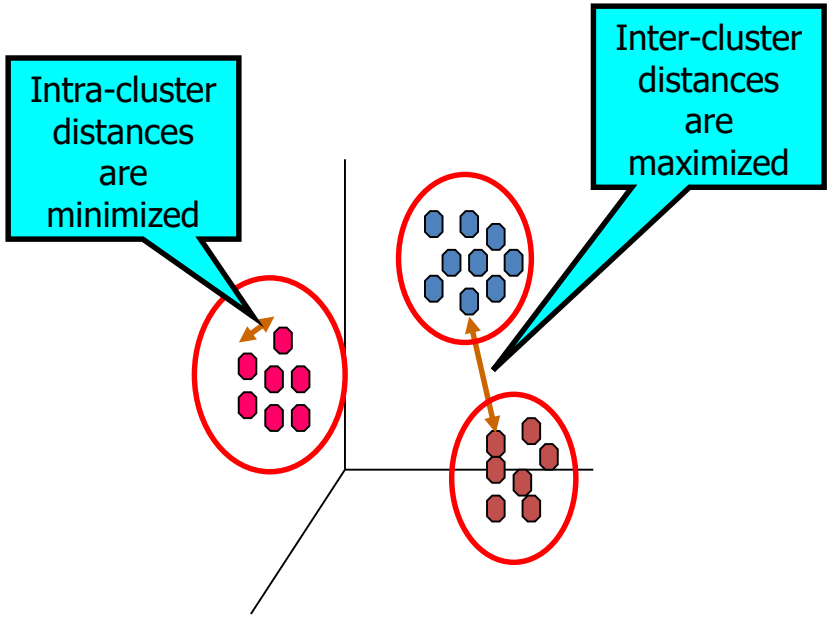
(Density-based spatial clustering of applications with noise)

Outline

- Introduction to Clustering
- Density-based Clustering
- Definitions of DBSCAN
- DBSCAN Algorithm
- Implementation in Python

What is Clustering?

- **Clustering** is an unsupervised machine learning algorithm that divides a data into meaningful sub-groups, called clusters.
- In general a grouping of objects such that the objects in a group (cluster) are similar (or related) to one another and different from (or unrelated to) the objects in other groups



Density-based Clustering Methods

- Clustering based on density (local cluster criterion), such as density-connected points

- Major features:
  - Discover clusters of arbitrary shape
  - Handle noise
  - Need density parameters as termination condition

- Several interesting studies:
  - DBSCAN: Ester, et al. (KDD'96)
  - OPTICS: Ankerst, et al (SIGMOD'99).
  - DENCLUE: Hinneburg & D. Keim (KDD'98)
  - CLIQUE: Agrawal, et al. (SIGMOD'98) (more grid-based)

DBSCAN: (Density-based spatial clustering of applications with noise)

- Proposed by Ester, Kriegel, Sander, and Xu (KDD96)
- Basic Idea: Clusters are dense regions in the data space, separated by regions of lower object density
- Discovers clusters of arbitrary shape in spatial databases with noise

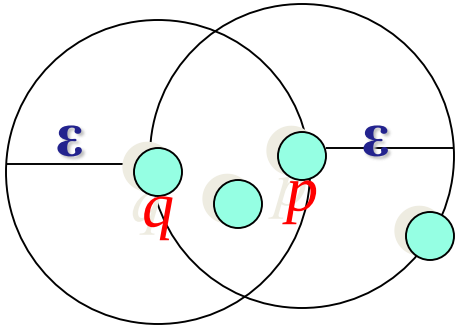
Figure 1. Sample databases

Density Based Clustering: Basic Concept

- Intuition for the formalization of the basic idea
  - For any point in a cluster, the **local point density** around that point has to exceed some threshold
  - The set of points from one cluster is spatially connected
- **Local point density** at a point ***p*** defined by two parameters
  - ε : Maximum radius of the neighbourhood of point ***p***.
  - ***MinPts*** *:* Minimum number of points in an ε - neighbourhood of that point.

ε-Neighborhood

- ε-Neighborhood – Objects within a radius of ε from an object. $N_\varepsilon(p):\{q \mid d(p,q) \le \varepsilon\}$
- "High density" - ε-Neighborhood of an object contains at least **MinPts** of objects.



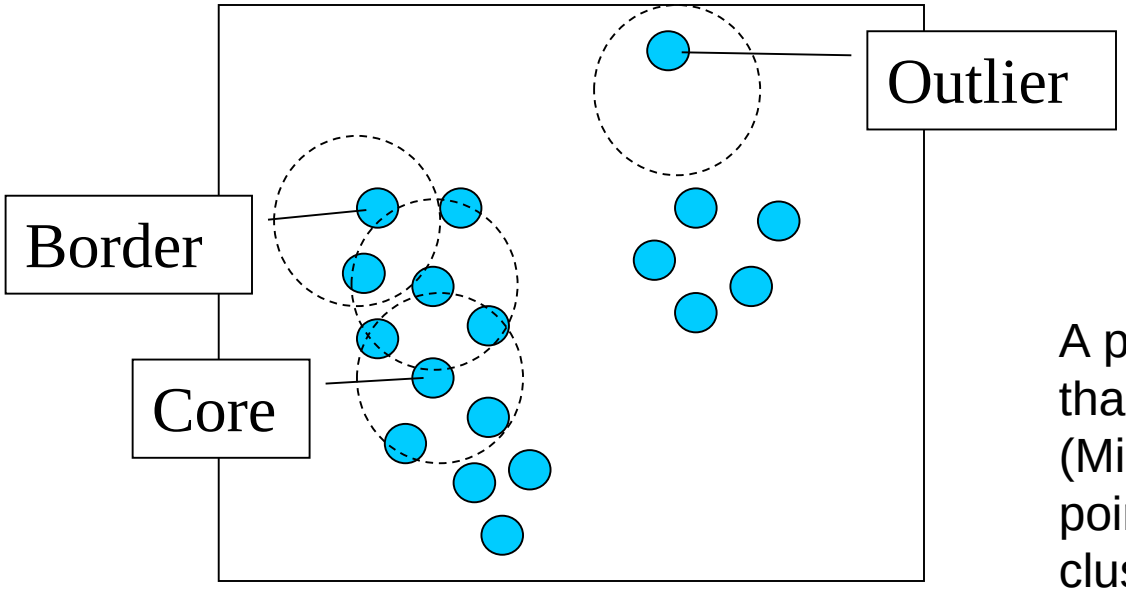
ε-Neighborhood of p

ε-Neighborhood of q

Density of p is "high" (MinPts = 4)

Density of q is "low" (MinPts = 4)

Core, Border & Outlier



ε = 1unit, MinPts = 5

Given ε and *MinPts*, categorize the objects into three exclusive groups.

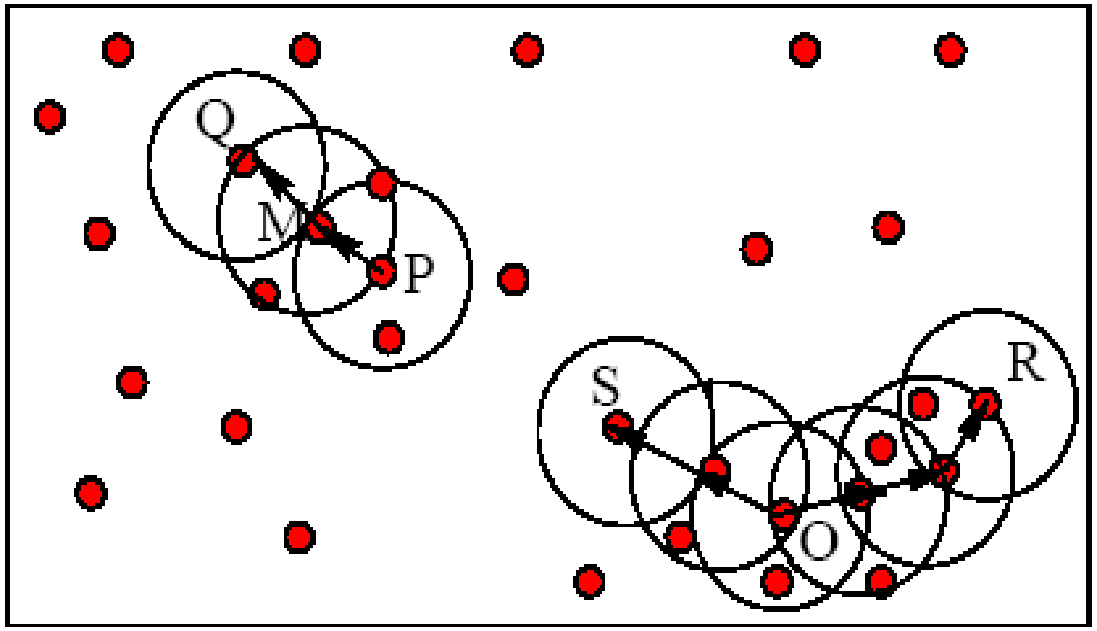
A point is a core point if it has more than a specified number of points (MinPts) within Eps These are points that are at the interior of a cluster.

A border point has fewer than MinPts within Eps, but is in the neighborhood of a core point.

A noise point is any point that is not a core point nor a border point.

Example

- **M, P, O, and R are core objects since each is in an Eps neighborhood containing at least 3 points**

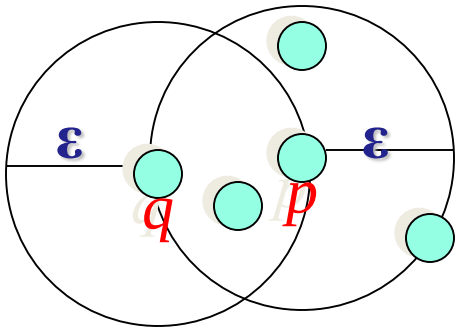


Minpts = 3

Eps=radius of the circles

Directly Density-Reachable

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

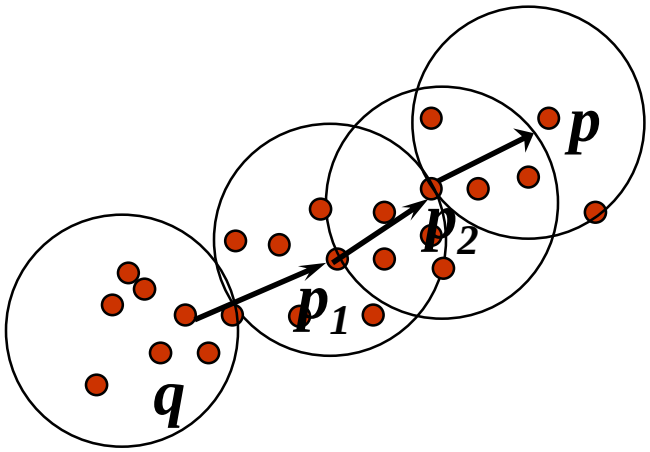


MinPts = 4

- q is directly density-reachable from p
- p is not directly density- reachable from q?
- Density-reachability is asymmetric.

Density-Reachable

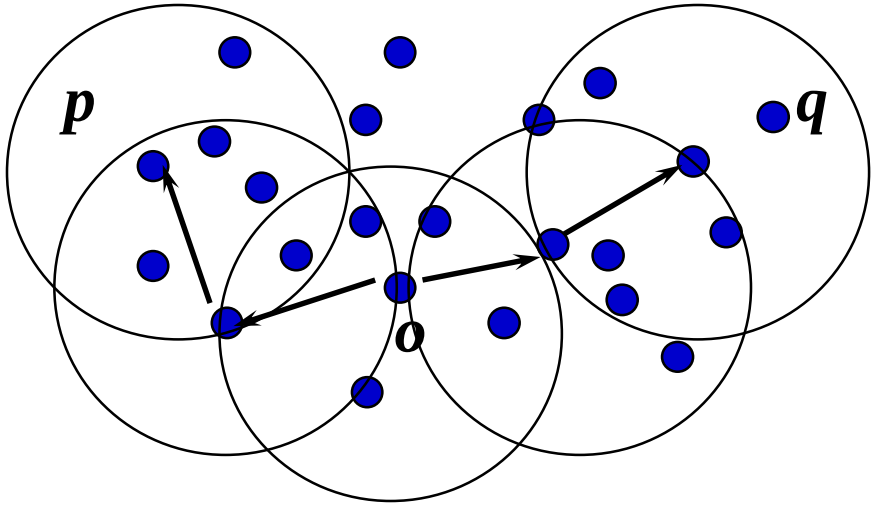
- A point *p* is density-reachable from a point *q* w.r.t. *Eps*, *MinPts* if there is a chain of points p_1, …, p_n, $p_1 = q$, $p_n = p$ such that p_{i+1} is directly density-reachable from p_i
  - A point p is directly density-reachable from p2;
  - p2 is directly density-reachable from p1;
  - p1 is directly density-reachable from q;
  - p←p2←p1←q form a chain.



- **p is (indirectly) density-reachable from q**
- **q is not density- reachable from p?**

Density-Connected

- A point ***p*** is density-connected to a point ***q*** w.r.t. Eps, MinPts if there is a point ***o*** such that both, ***p*** and ***q*** are density-reachable from ***o*** w.r.t. Eps and MinPts



DBSCAN Algorithm

Input: The data set D

Parameter: ε, MinPts

For each object p in D
    if p is a core object and not processed then
        C = retrieve all objects density-reachable from p
        mark all objects in C as processed
        report C as a cluster
    else mark p as outlier
    end if

End For

DBScan Algorithm

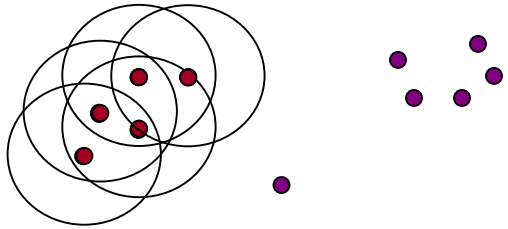
DBSCAN: The Algorithm

- Arbitrary select a point *p*
- Retrieve all points density-reachable from *p* wrt *Eps* and *MinPts*.
- If *p* is a core point, a cluster is formed.
- If *p* is a border point, no points are density-reachable from *p* and DBSCAN visits the next point of the database.
- Continue the process until all of the points have been processed.

DBSCAN Algorithm: Example

Parameter

- ε = 2 cm
- *MinPts* = 3

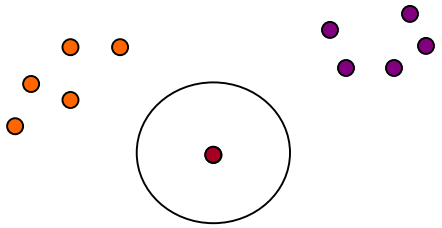


```
for each o ∈ D do
    if o is not yet classified then
        if o is a core-object then
            collect all objects density-reachable from o
            and assign them to a new cluster.
        else
            assign o to NOISE
```

DBSCAN Algorithm: Example

Parameter

- ε = 2 cm
- *MinPts* = 3

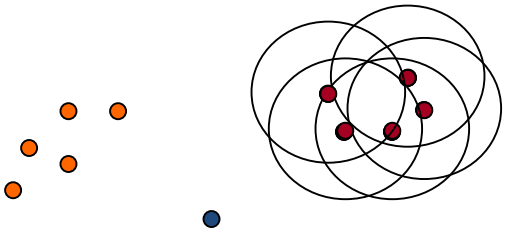


```
for each o ∈ D do
    if o is not yet classified then
        if o is a core-object then
            collect all objects density-reachable from o
            and assign them to a new cluster.
        else
            assign o to NOISE
```

DBSCAN Algorithm: Example

Parameter

- ε = 2 cm
- *MinPts* = 3



for each $o \in D$ **do**
  if o is not yet classified **then**
    if o is a core-object **then**
      collect all objects density-reachable from o
      and assign them to a new cluster.
    else
      assign o to NOISE

Determining Eps and MinPts

In layman's terms, we find a suitable value for epsilon by

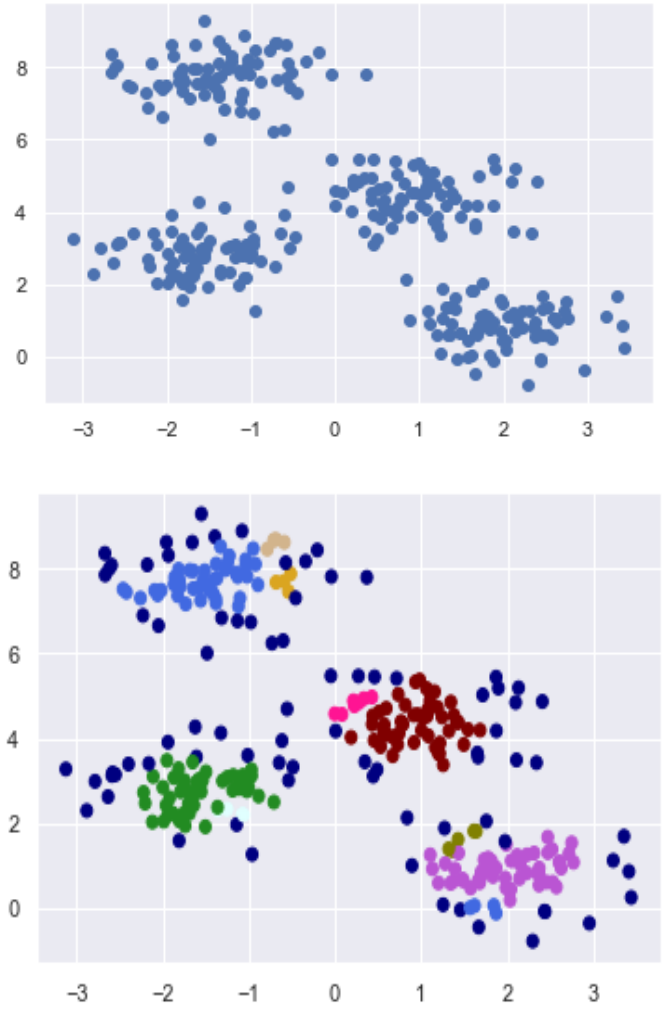
- calculating the distance to the nearest ***n*** points for each point
- sorting and plotting the results.
- Then we look to see where the change is most pronounced (think of the angle between your arm and forearm) and select that as epsilon.

Algorithm 1 The pseudo code of the proposed technique DMDBSCAN to find suitable Epsi for each level of density in data set	
Purpose	*To find suitable values of Eps*
Input	*Data set of size n*
Output	*Eps for each varied density*
Procedure	*1 for i* *2 for j = 1 to n* *3 d(i,j) ← find distance (x_i, x_j)* *4 find minimum values of distances to nearest 3* *5 end for* *6 end for* *7 sort distances ascending and plot to find each value* *8 Eps corresponds to critical change in curves*

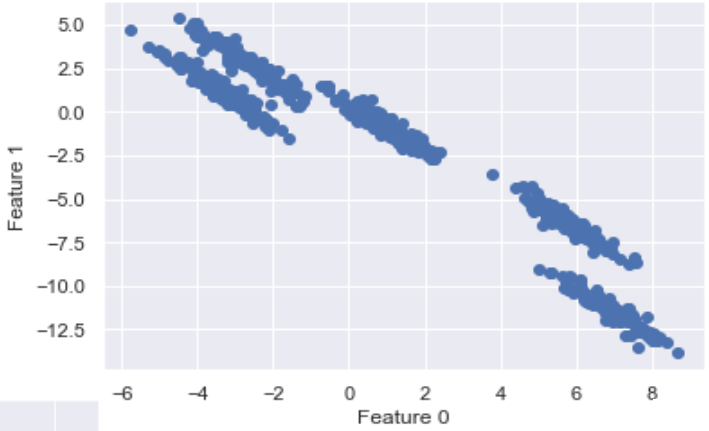
Figure 1 Pseudocode DMDBSCAN Algorithm (Elbatta 2012)

Code snippets

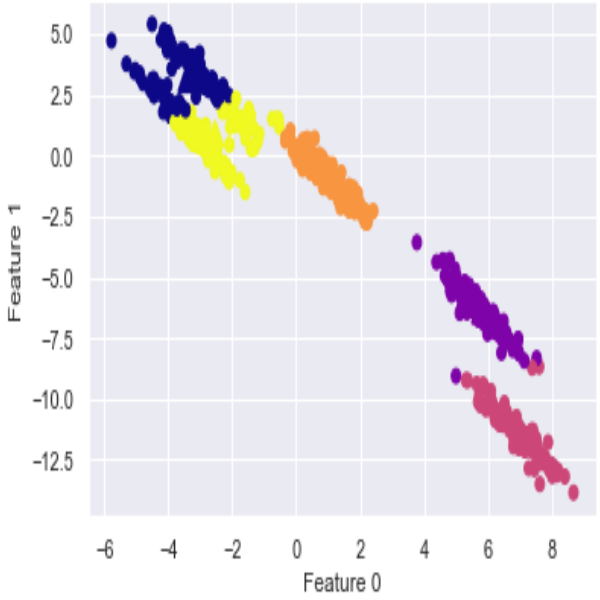
```
import numpy as np
from sklearn.datasets.samples_generator import make_blobs
from sklearn.neighbors import NearestNeighbors
from sklearn.cluster import DBSCAN
from matplotlib import pyplot as plt
import seaborn as sns
sns.set()

X, y = make_blobs(n_samples=300, centers=4,
      cluster_std=0.60, random_state=0)
plt.scatter(X[:,0], X[:,1])

m = DBSCAN(eps=0.3, min_samples=5)
m.fit(X)
clusters = m.labels_
colors = ['royalblue', 'maroon', 'forestgreen', 'mediumorchid',
      'tan', 'deeppink', 'olive', 'goldenrod', 'lightcyan', 'navy']
vectorizer = np.vectorize(lambda x: colors[x % len(colors)])
plt.scatter(X[:,0], X[:,1], c=vectorizer(clusters))
```



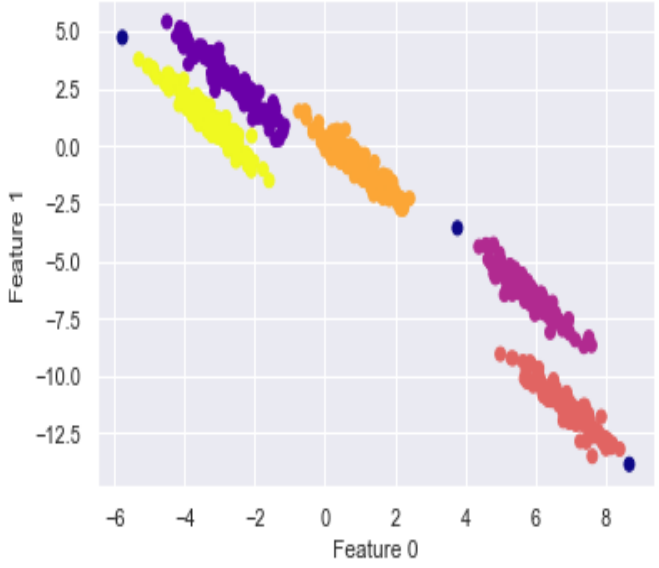
DBSCAN Vs K-Means



K-Means



DBSCAN



Thank You!!