Fourth Industrial Summer School

Module 4: ML

Unsupervised Learning: Clustering Algorithms

Outlines

- ✓ Density Clustering
  - ✓ DBSCAN Algorithm
  - ✓ How it works?
  - ✓ Example
  - ✓ Pros & Cons

Clustering with no prior Knowledge of K

- While k-means clustering is known for its user-friendly implementation, it can struggle with data that isn't clustered in round shapes or has a lot of outliers.

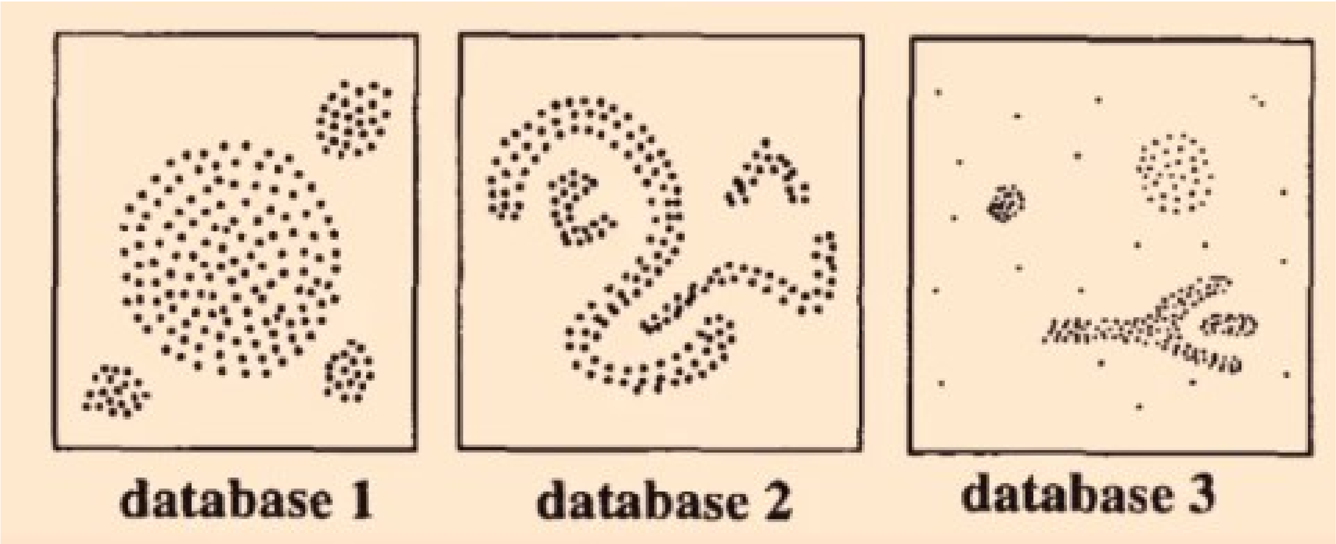
- When data isn't perfectly spherical or has outliers, a good initial step for finding clusters can be to focus on areas where there's a high density of data points.

Density
Based
Spatial
Clustering
Application
Noise

Density-based spatial clustering of applications with noise, or ***DBSCAN***, is an algorithm that groups together points in high-density, connected regions.

DBSCAN

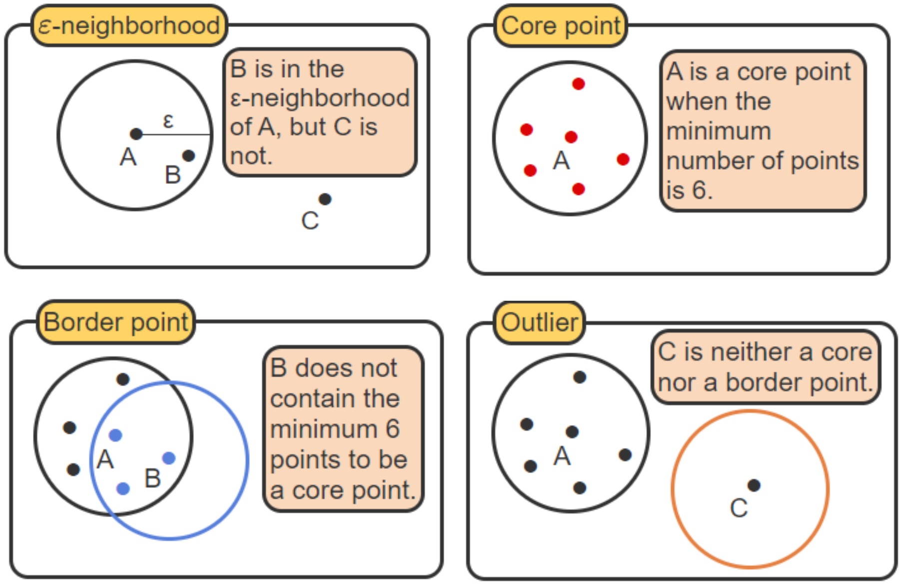
- It assumes clusters are dense regions in the data space.
- These dense regions must be separated as clusters by some lite-densely zones.
- Regions can be of any shape (i.e. no shape assumption)



DBSCAN Algorithm Terms

- DBSCAN relies on the density of points within a defined spherical neighborhood. The following terminology are, usually, used in describing the DBSCAN algorithm

1. **Eps-neighborhood:** of a point is a spherical region of **eps** radius centered at that point.
2. **A *core point*** is a point whose ε-neighborhood contains a given minimum number of points.
3. **A border point** is a point that is not a core point but is contained in an ε-neighborhood of a core point.
4. **An outlier** is a point that is neither a core point, nor a border point.

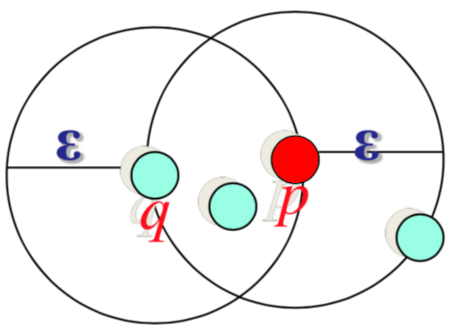


DBSCAN Algorithm Terms

- **eps-neighborhood:** is a decision parameter that is set by a user, and used to decide whether a given point belongs to a cluster.

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

where **p** is a core point, **q** is a given point, **dist(.)** is the distance function.



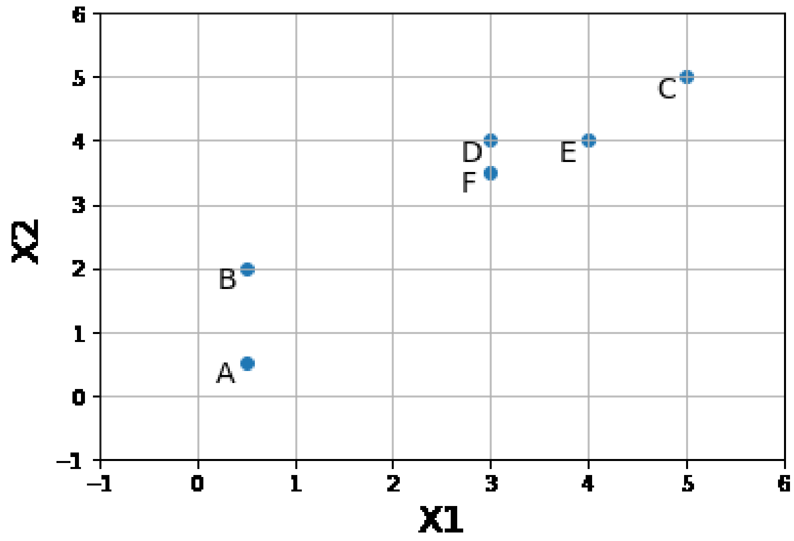
DBSCAN Algorithm

1. Mark data points as core, border, or noise points using **eps-neighborhood** and **min_samples** parameters
2. For Cluster C_i:
   1. Label all (similar) **core points** that are within eps of each other with i
   2. Assign each border point to cluster C_i with i label
   3. If a point is not in a neighborhood of any core point label as **-1** (outlier).
   4. If all points checked and labeled (stop), otherwise
   5. Repeat next C_i

Example

- A DBSCAN algorithm with (eps = 1.5, MinPts = 3) is executed on the data below. Determine for each point whether it is a core, border or noise point. What are the resulting clusters?

Pts	x1	x2
A	0.5	0.5
B	0.5	2
C	5	5
D	3	4
E	4	4
F	3	3.5



Sample Problem

- Parameters: Minpts = 3, eps= 1.5, metric function = Euclidean distance

A: Noise Point

dist(A,B)	1.50
dist(A,C)	6.36
dist(A,D)	4.30
dist(A,E)	4.95
dist(A,F)	3.91

B: Noise Point

dist(B,A)	1.50
dist(B,C)	5.41
dist(B,D)	3.20
dist(B,E)	4.03
dist(B,F)	2.92

C: Border Point

dist(C,A)	6.36
dist(C,B)	5.41
dist(C,D)	2.24
dist(C,E)	1.41
dist(C,F)	2.50

D: Core Point

dist(D,A)	4.30
dist(D,B)	3.20
dist(D,C)	2.24
dist(D,E)	1.00
dist(D,F)	0.50

E: Core Point

dist(E,A)	4.95
dist(E,B)	4.03
dist(E,C)	1.41
dist(E,D)	1.00
dist(E,F)	1.12

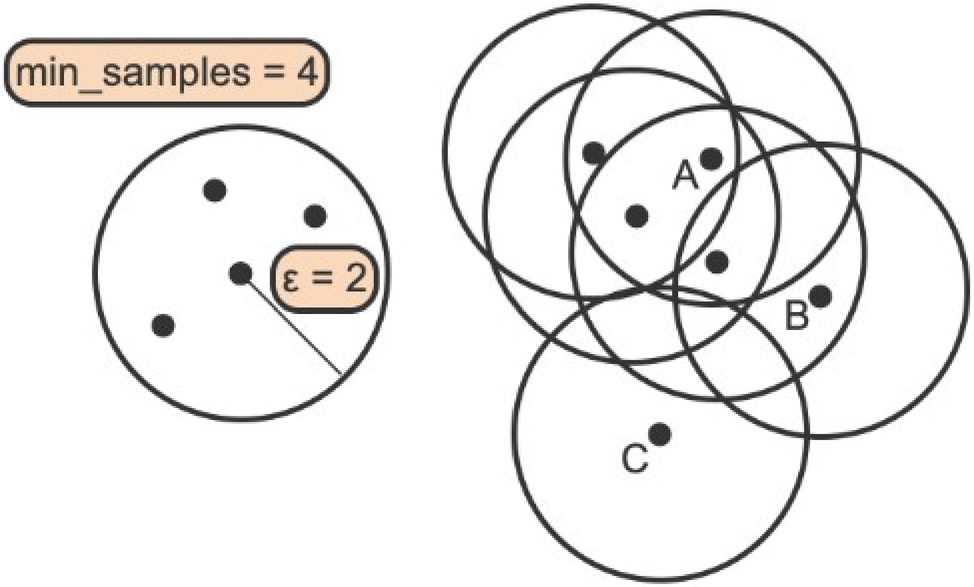
F: Core Point

dist(F,A)	3.91
dist(F,B)	2.92
dist(F,C)	2.50
dist(F,D)	0.50
dist(F,E)	1.12

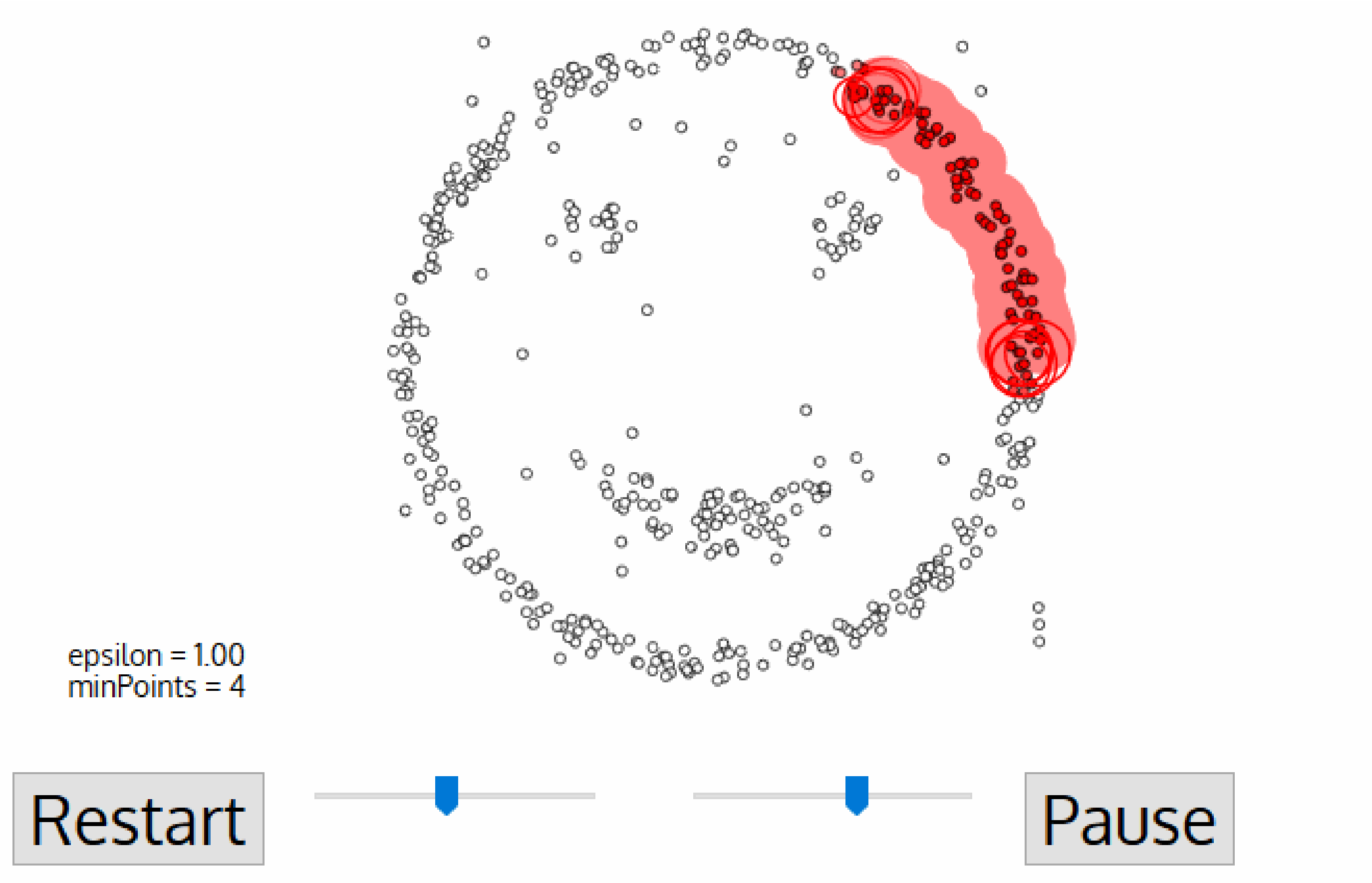
- An instance is always within neighborhood of itself.
- There can be more than one core point in a cluster
- A noise point can share neighborhood with another noise point

Question:

- Determine whether each labeled point in the below is a core point, a boundary point, or an outlier
- Given eps = 2 and the MinPts = 4.



Animation: DBSCAN



Source: https://www.naftaliharris.com/blog/visualizing-dbscan-clustering/

DBSCAN Implementation

Load the package

```
# Load the package from Scikit learn library
from sklearn.cluster import DBSCAN
```

Make an instance object

```
# Create an instance of DBSCAN
dbscan = DBSCAN(eps=0.6,
                min_samples =10,
                metric ='minkowski',
                p = 1)
```

Perform the clustering

```
dbscan.fit(X)
```

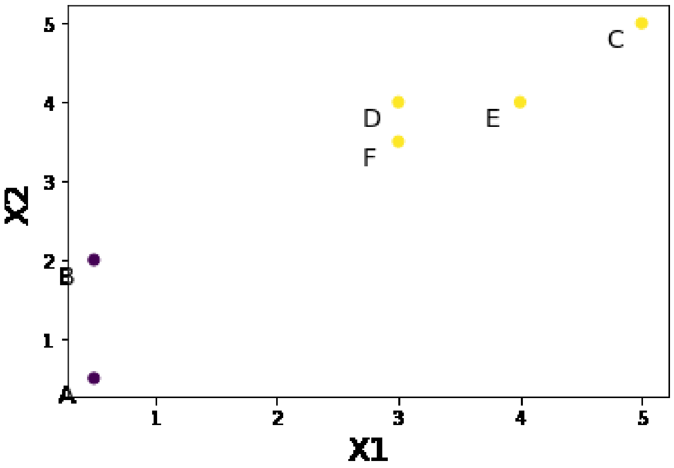
Getting number of clusters

```
if -1 in dbscan.labels_:
  Number_of_clusters = len(set(dbscan.labels_))-1
else:
  Number_of_clusters = len(set(dbscan.labels_))
```

DBSCAN Implementation

- DBSCAN returns **-1** noisy datapoints.

Pts	x1	x2
A	0.5	0.5
B	0.5	2
C	5	5
D	3	4
E	4	4
F	3	3.5




```
myX = np.array([[0.5, 0.5],[0.5, 2], [5, 5], [3, 4], [4, 4], [3, 3.5]])
```

```
dbscan1= DBSCAN(eps=1.5, min_samples=3, metric='euclidean')
dbscan1.fit(myX)
print('Noisy points:',dbscan1.labels_)

print('Core points indecies:', dbscan1.core_sample_indices_ )
print('Core points values:\n', dbscan1.components_  )
```

```
Noisy points: [-1 -1  0  0  0  0]
Core points indecies: [3 4 5]
Core points values:
 [[3.  4. ]
 [4.  4. ]
 [3.  3.5]]
```

Notes.

- **No free lunch:** we did not predefine the number of clusters, DBSCAN find them out! However, we set a number of hyperparameters to determine data clusters.

- **An issue** is how to determine **eps and minimum number of samples**.

- **Standardizing your data**: It is important to notice the scale of each feature vector, standardize your data is a good practice.

- **Attributes:** The DBSCAN model has an attribute **labels_** to access the result labels of each point**;** core points in **core_sample_indices_**.

Determine eps-neighborhood: An example algorithm

The proper choice of eps value requires domain expertise

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	*for i* *for j = 1 to n* *d(i,j) ← find distance* (x_i, x_j) *find minimum values of distances to nearest 3* *end for* *end for* *sort distances ascending and plot to find each value* *Eps corresponds to critical change in curves*

Figure 1 Pseudocode DMDBSCAN Algorithm (Elbatta 2012)

Article source: https://iopscience.iop.org/article/10.1088/1755-1315/31/1/012012/pdf

Pros and Cons

It does not require a pe-set number of clusters at all.

Works well for data with outliers

Can handle clusters of different shapes and sizes

It may fail if the data have varying densities.

Very sensitive to configuration (parameters)

It is not well scaled to high-dimensional data

[redacted]