DBSCAN Clustering Algorithm

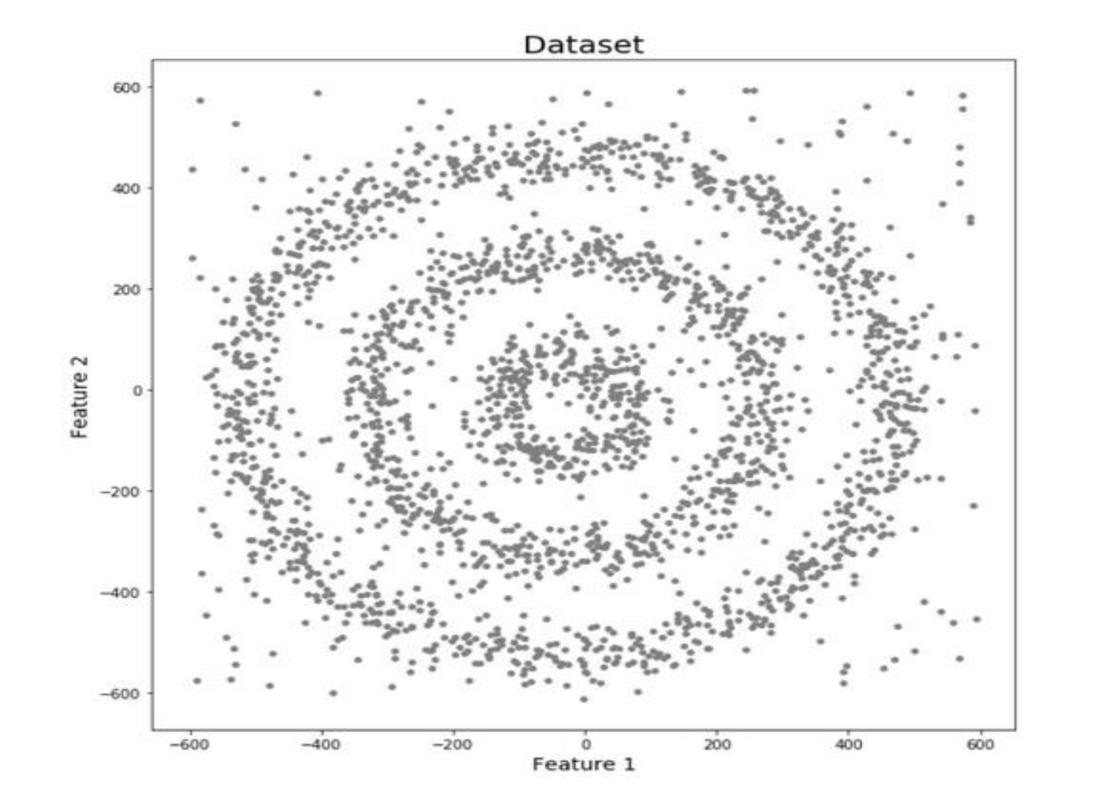
Contents

Introduction

- Clustering is an unsupervised learning method that divides data points into specific groups, such that data points in a group have similar properties than those in other groups.
- There are different approaches and algorithms to perform clustering tasks which can be divided into three sub-categories:
- Partition-based clustering: E.g. k-means, k-median
- Hierarchical clustering: E.g. Agglomerative, Divisive
- Density-based clustering: E.g. DBSCAN

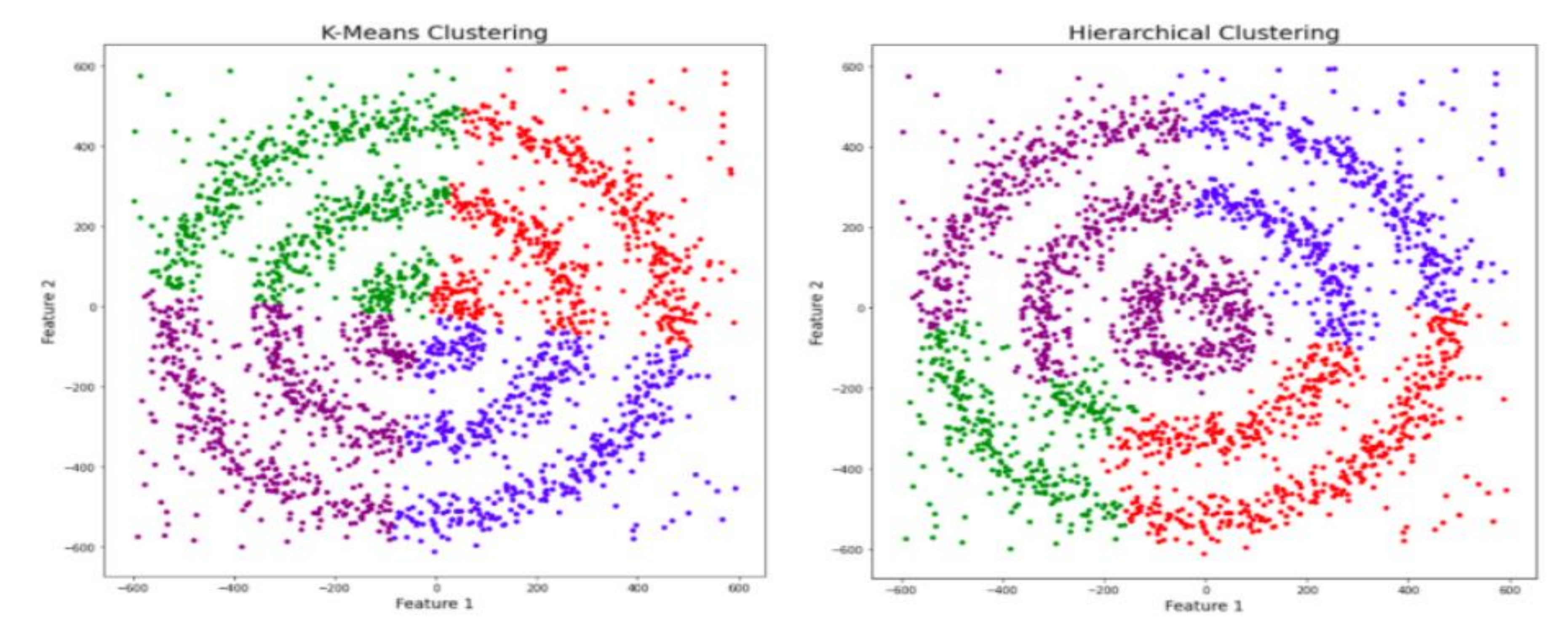
Why do we need DBSCAN Clustering?

- K-Means and Hierarchical Clustering both fail in creating clusters of arbitrary shapes. They are not able to form clusters based on varying densities. That’s why we need DBSCAN clustering.
- Let’s try to understand it with an example. Here we have data points densely present in the form of concentric circles:
- We can see three different dense clusters in the form of concentric circles with some noise here.



Why do we need DBSCAN Clustering?

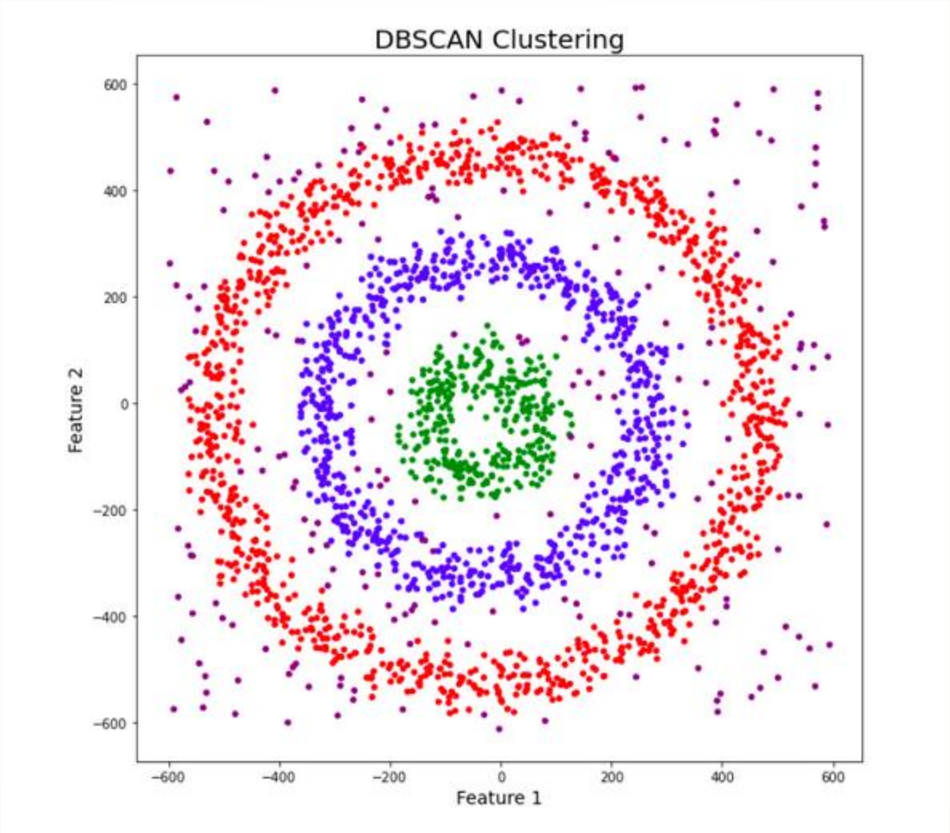
- Now, let’s run K-Means and Hierarchical clustering algorithms and see how they cluster these data points.



- this data contains noise too, therefore, I have taken noise as a different cluster which is represented by the purple color.
- Sadly, both of them failed to cluster the data points. Also, they were not able to properly detect the noise present in the dataset.

Why do we need DBSCAN Clustering?

- let’s take a look at the results from DBSCAN clustering.



- DBSCAN is not just able to cluster the data points correctly, but it also perfectly detects noise in the dataset.

What Exactly is DBSCAN Clustering?

- **DBSCAN** stands for **D**ensity-**B**ased **S**patial **C**lustering of **A**pplications with **N**oise.
- It groups 'densely grouped' data points into a single cluster.
- It can identify clusters in large spatial datasets by looking at the local density of the data points.
- **The most exciting feature of DBSCAN clustering is that it is robust to outliers**.
- It also does not require the number of clusters to be told beforehand, unlike K-Means, where we have to specify the number of centroids.
- DBSCAN requires only two parameters: *epsilon* and *minPoints*.
- ***Epsilon*** is the radius of the circle to be created around each data point to check the density
- ***minPoints*** is the minimum number of data points required inside that circle for that data point to be classified as a **Core** point.
- In higher dimensions the circle becomes hypersphere, *epsilon* becomes the radius of that hypersphere, and *minPoints* is the minimum number of data points required inside that hypersphere.

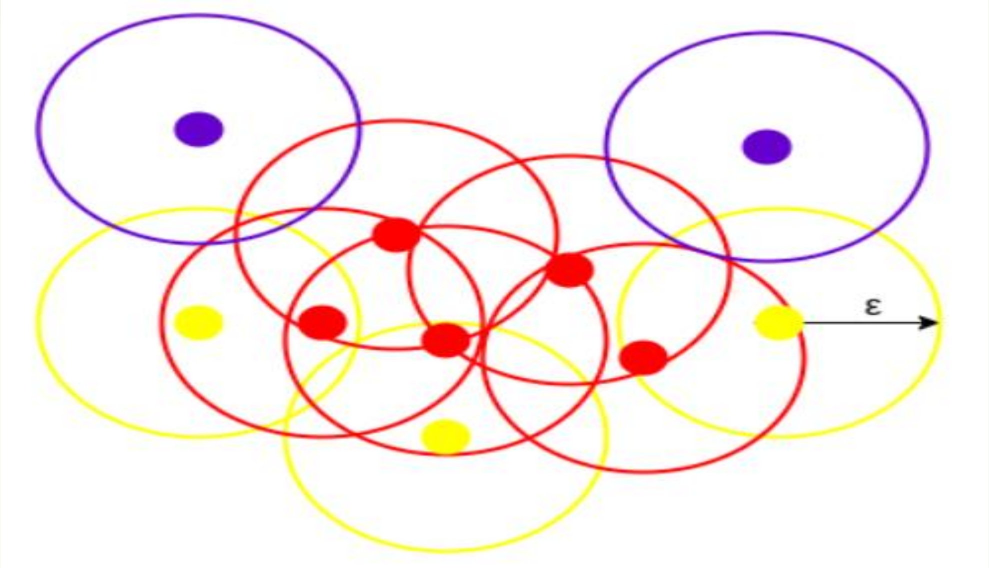
What Exactly is DBSCAN Clustering?

- Let's understand it with the help of an example.
- Here, we have some data points represented by grey color.

- Let's see how DBSCAN clusters these data points.
- DBSCAN creates a circle of *epsilon* radius around every data point and classifies them into **Core** point, **Border** point, and **Noise**.
- A data point is a **Core** point if the circle around it contains at least '*minPoints*' number of points.
- If the number of points is less than *minPoints*, then it is classified as **Border** Point, and
- if there are no other data points around any data point within *epsilon* radius, then it treated as **Noise**.

What Exactly is DBSCAN Clustering?

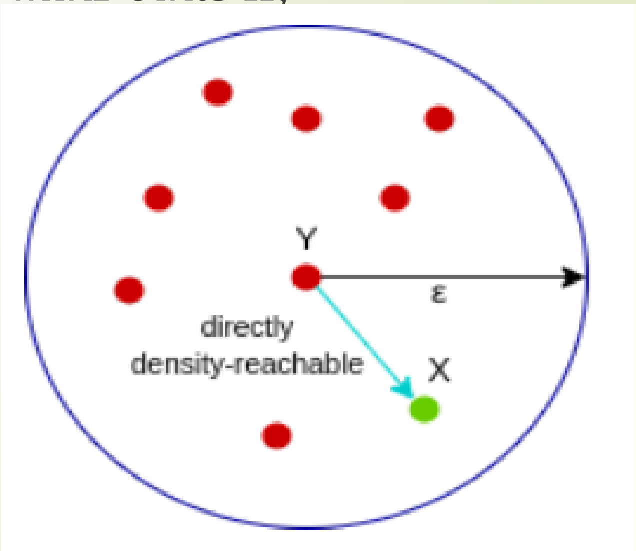
- The above figure shows us a cluster created by DBCAN with *minPoints = 3*.



- Here, we draw a circle of equal radius *epsilon* around every data point. These two parameters help in creating spatial clusters.
- All the data points with at least 3 points in the circle including itself are considered as **Core** points represented by red color.
- All the data points with less than 3 but greater than 1 point in the circle including itself are considered as **Border** points. They are represented by yellow color.
- Finally, data points with no point other than itself present inside the circle are considered as **Noise** represented by the purple color.

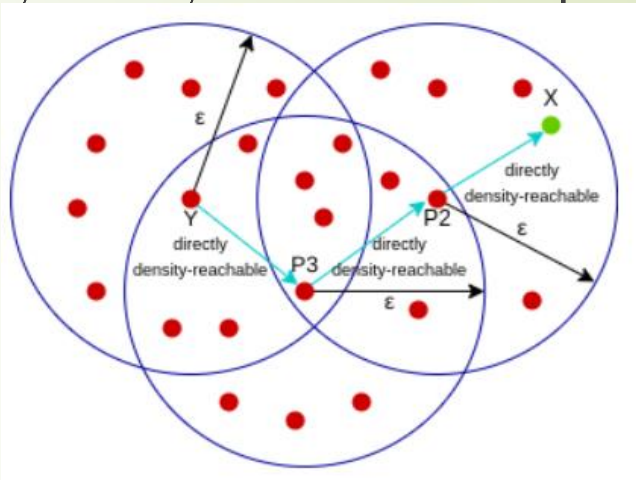
Reachability and Connectivity

- Reachability states if a data point can be accessed from another data point directly or indirectly
- Connectivity states whether two data points belong to the same cluster or not.
- In terms of reachability and connectivity, two points in DBSCAN can be referred to as:
  - **Directly Density-Reachable**
  - **Density-Reachable**
  - **Density-Connected**
- Let’s understand what they are.
- A point **X** is **directly density-reachable** from point **Y** w.r.t *epsilon*, *minPoints* if,
  1. **X** belongs to the neighborhood of **Y**, i.e, *dist(X, Y) <= epsilon*
  2. **Y** is a core point
- Here, **X** is directly density-reachable from Y**,**
- but vice versa is not valid.

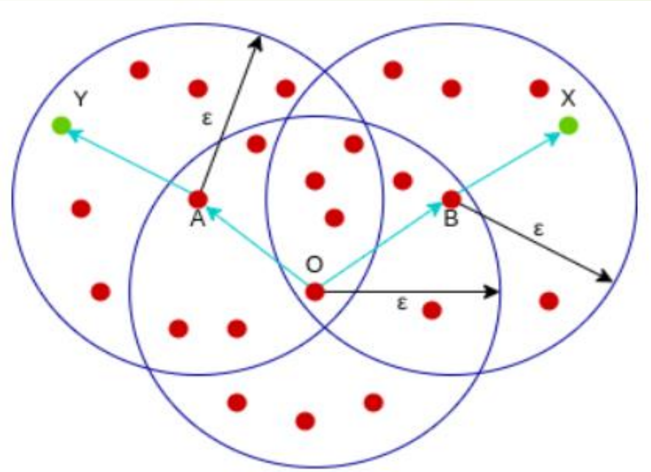


Reachability and Connectivity

- A point **X** is **density-reachable** from point **Y** w.r.t *epsilon*, *minPoints* if there is a chain of points p1, p2, p3, …, pn and p1=**X** and pn=**Y** such that pi+1 is directly density-reachable from pi.
- Here, **X** is density-reachable from **Y** with **X** being directly density-reachable from **P2**, **P2** from **P3,** and **P3** from **Y.** But, the inverse of this is not valid.



- A point **X** is **density-connected** from point **Y** w.r.t *epsilon and minPoints* if there exists a point **O** such that both **X** and **Y** are density-reachable from **O** w.r.t to *epsilon and minPoints.*
- Here, both **X** and **Y** are density-reachable from **O**, therefore, we can say that **X** is density-connected from **Y**.



Parameter Selection in DBSCAN Clustering

- DBSCAN is very sensitive to the values of *epsilon* and *minPoints*.
- Therefore, it is very important to understand how to select the values of *epsilon* and *minPoints*.
- A slight variation in these values can significantly change the results produced by the DBSCAN algorithm.
- The value of ***minPoints*** should be at least one greater than the number of dimensions of the dataset, i.e.,

$$minPoints >= Dimensions + 1$$

- It does not make sense to take ***minPoints*** as 1 because it will result in each point being a separate cluster. Therefore, it must be at least 3. Generally, it is twice the dimensions. But domain knowledge also decides its value.
- The value of ***epsilon*** can be decided from the K-distance graph.
- The point of maximum curvature (elbow) in this graph tells us about the value of *epsilon*.
- If the value of *epsilon* chosen is too small then a higher number of clusters will be created, and more data points will be taken as noise.
- Whereas, if chosen too big then various small clusters will merge into a big cluster, and we will lose details.

Algorithmic steps for DBSCAN clustering

- **Now, let’s take a look at how DBSCAN algorithm actually works. Here is the pseudo code.**
- Arbitrary select a point p
- Retrieve all points density-reachable from p based on 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

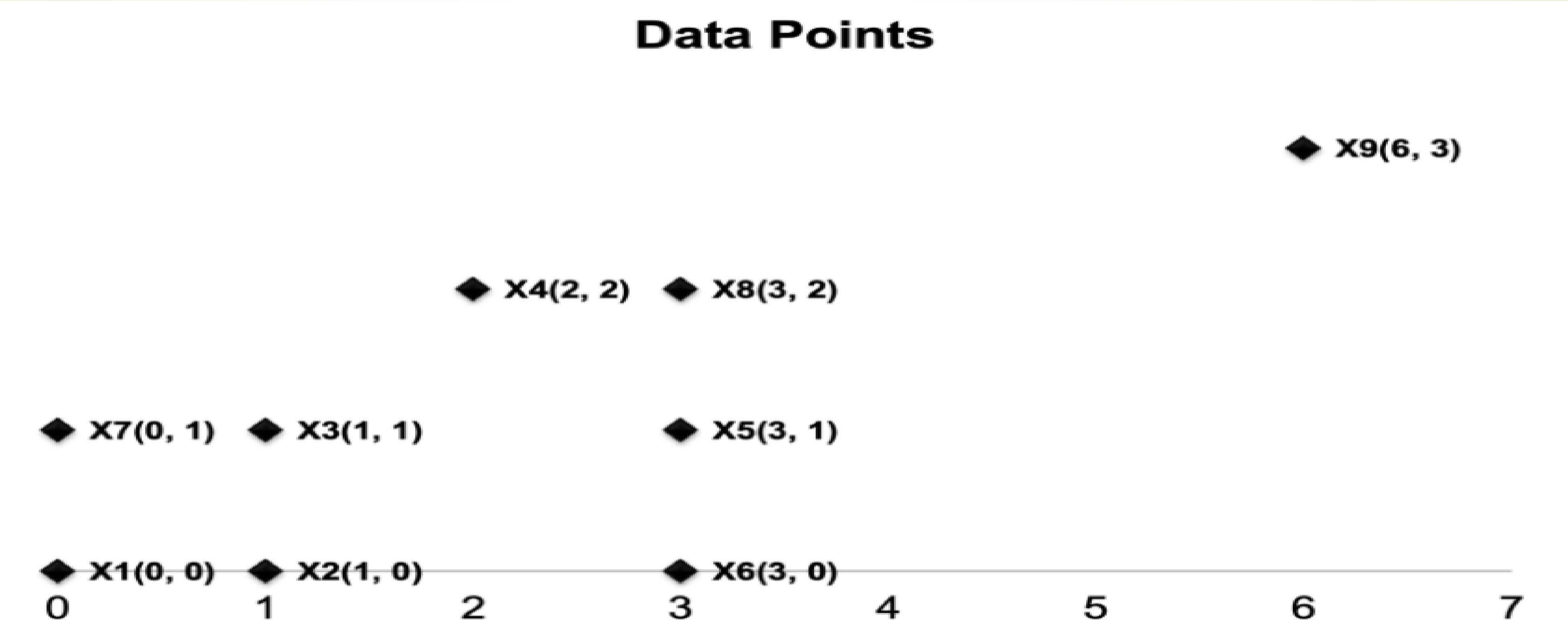
Example

- Consider the following 9 two-dimensional data points:

  x1(0,0), x2(1,0), x3(1,1), x4(2,2), x5(3,1), x6(3,0), x7(0,1), x8(3,2), x9(6,3)

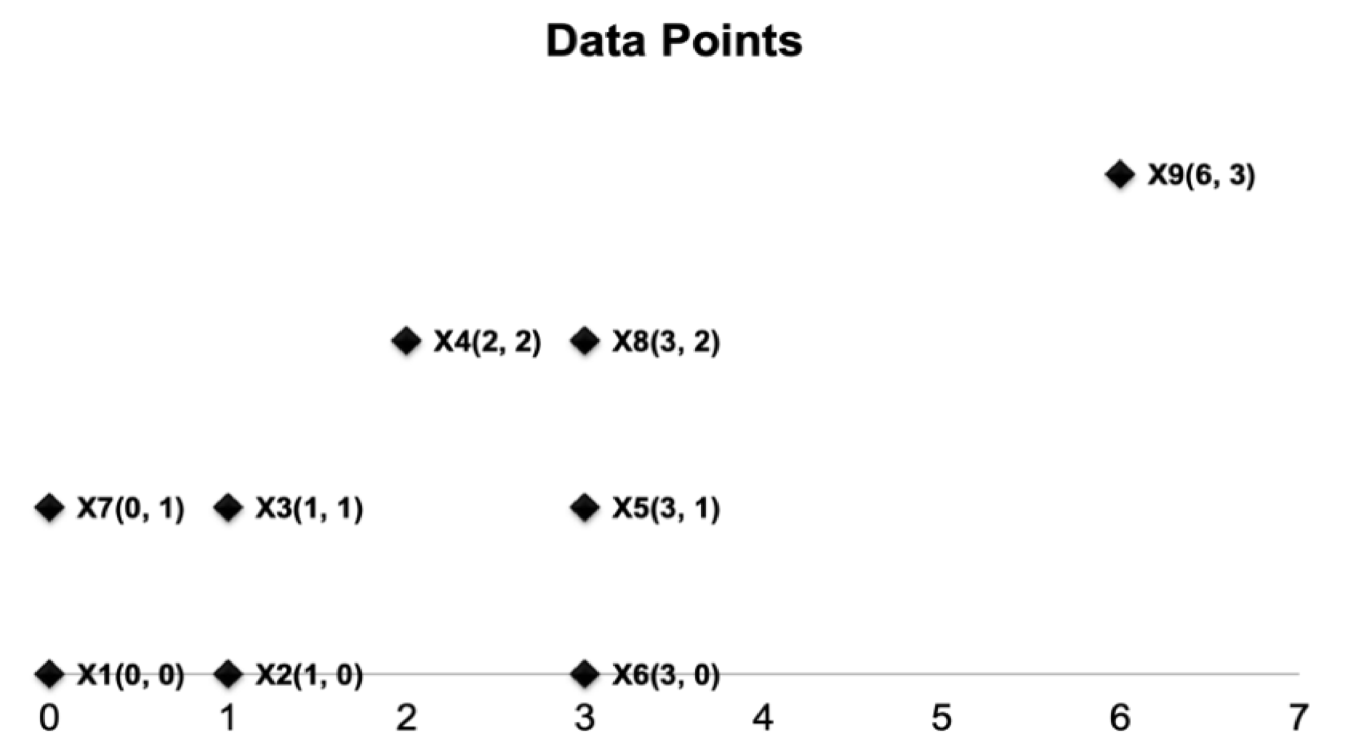
  Use the Euclidean Distance with Eps =1 and MinPts = 3. Find all core points, border points and noise points, and show the final clusters using DBCSAN algorithm.

- Lets show the result step by step



Example

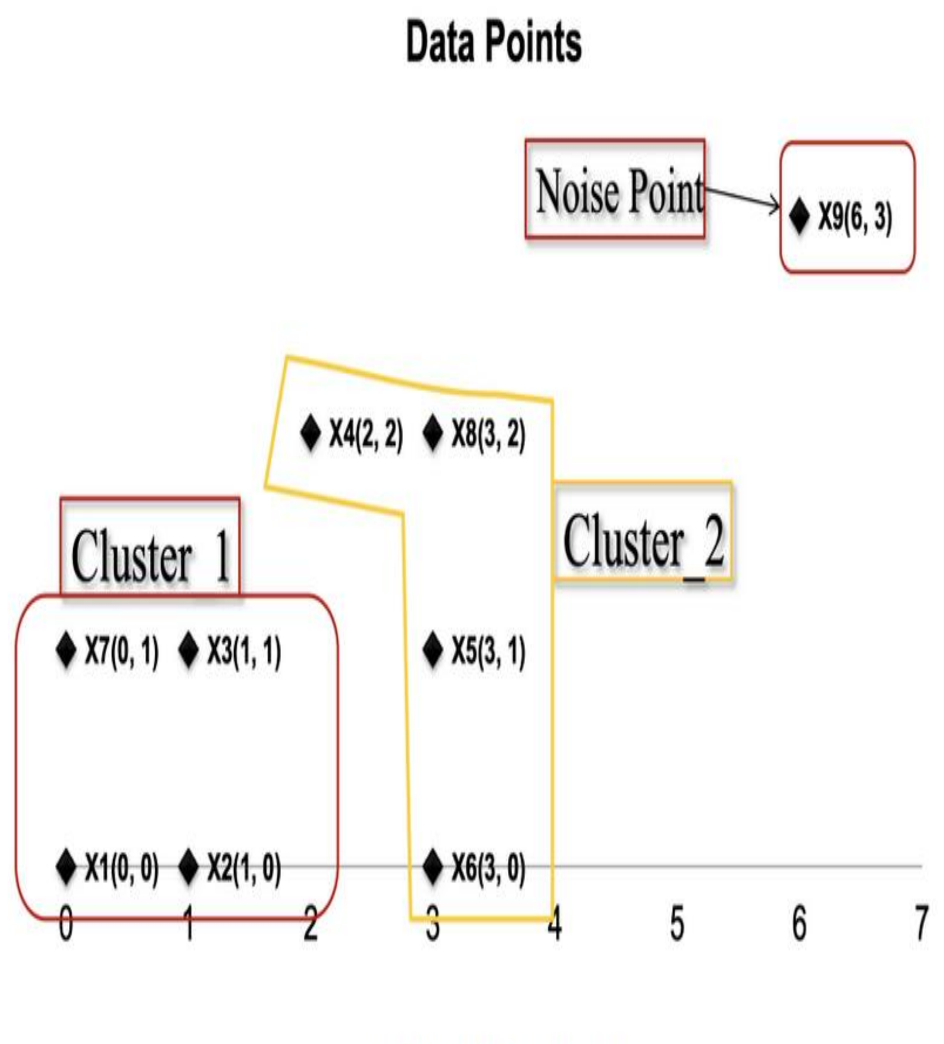
- **First, Calculate the N(p), Eps-neighborhood of point p**
- N(x1) = {x1, x2, x7}
- N(x2) = {x2, x1, x3}
- N(x3) = {x3, x2, x7}
- N(x4) = {x4, x8}
- N(x5) = {x5, x6, x8}
- N(x6) = {x6, x5}
- N(x7) = {x7, x1, x3}
- N(x8) = {x8, x4, x5}
- N(x9) = {x9}



- If the size of N(p) is at least MinPts, then p is said to be a core point. Here the given MinPts is 3, thus the size of N(p) is at least 3. **Thus core points are:{x1, x2, x3, x5, x7, x8}**
- Then according to the definition of border points: given a point p, p is said to be a border point if it is not a core point but N(p) contains at least one core point. N(x4) = {x4, x8}, N(x6) = {x6, x5}. here x8 and x5 are core points, So both **x4 and x6 are border points**.
- Obviously, the point left, **x9 is a noise point.**

Example

- **Now, let's follow the pseudo code to produce the clusters.**
- Arbitrary select a point p, now we choose x1
- Retrieve all points density-reachable from x1: {x2, x3, x7}
- Here x1 is a core point, a cluster is formed. So we have **Cluster_1**: {x1, x2, x3, x7}
- Next, we choose x5, Retrieve all points density-reachable from x5: {x4, x6, x8}
- Here x5 is a core point, a cluster is formed. So we have **Cluster_2**: {x4, x5, x6, x8}
- Next, we choose x9, x9 is a noise point, noise points do **NOT belong** to any clusters.
- Thus the algorithm stops here.



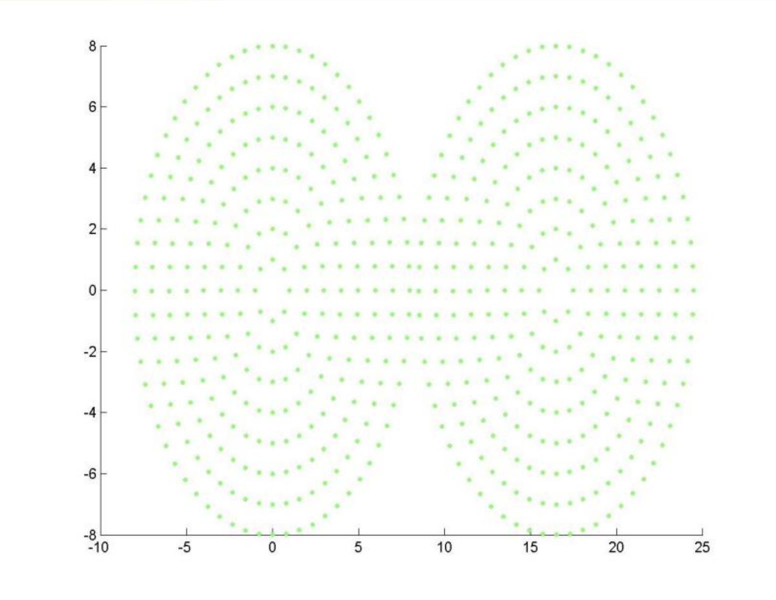
Final DBSCAN Cluster Result

Advantages

- Does not require a-priori specification of number of clusters.
- Able to identify noise data while clustering.
- DBSCAN algorithm is able to find arbitrarily size and arbitrarily shaped clusters.
- DBSCAN is robust to outliers and able to detect the outliers.

Disadvantages

- DBSCAN algorithm fails in case of varying density clusters.
- Fails in case of neck type of dataset.



- Does not work well in case of high dimensional data.

The complexity of DBSCAN Clustering Algorithm

- **Time Complexity:**

  - **Best Case:** If an indexing system is used to store the dataset such that neighborhood queries are executed in logarithmic time, we get **O(nlogn)** average runtime complexity.

  - **Worst Case:** Without the use of index structure or on degenerated data (e.g. all points within a distance less than ε), the worst-case run time complexity remains **O(n^2)**.

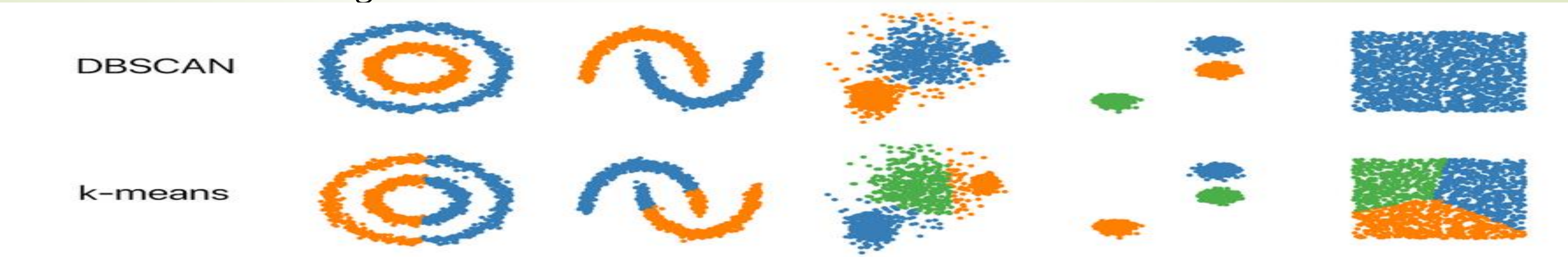
  - **Average Case:** Same as best/worst case depending on data and implementation of the algorithm.

- **Space Complexity: O(n)**

DBSCAN Vs K-means Clustering

S. No.	K-means Clustering	DBSCAN
1	Distance based clustering	Density based clustering
2	Every observation becomes a part of some cluster eventually	Clearly separates outliers and clusters observations in high density areas
3	Build clusters that have a shape of a hypersphere	Build clusters that have an arbitrary shape or clusters within clusters.
4	Sensitive to outliers	Robust to outliers
5	Require no. of clusters as input	Doesn’t require no. of clusters as input

DBSCAN also produces more reasonable results than *k*-means across a variety of different distributions. Below figure illustrates the fact:



References

- https://www.analyticsvidhya.com/blog/2020/09/how-dbscan-clustering-works/
- https://towardsdatascience.com/dbscan-clustering-explained-97556a2ad556
- https://sites.google.com/site/dataclusteringalgorithms/density-based-clustering-algorithm
- https://www.mygreatlearning.com/blog/dbscan-algorithm/

Thank You
Any Question?