**Lab 6**

**Implementation of DBSCAN**

## Introduction

## Density-based spatial clustering of applications with noise (DBSCAN) is a data clustering algorithm proposed by Martin Ester, Hans-Peter Kriegel, Jörg Sander and Xiaowei Xu in 1996. It is a density-based clustering non-parametric algorithm: given a set of points in some space, it groups together points that are closely packed together (points with many nearby neighbors), marking as outliers points that lie alone in low-density regions (whose nearest neighbors are too far away). DBSCAN is one of the most common clustering algorithms and also most cited in scientific literature.

## Clustering is an essential technique in machine learning and is used widely across domains and industries (think about Uber’s route optimization, Amazon’s recommendation system, Netflix’s customer segmentation, and so on). Clustering is an unsupervised learning technique where we try to group the data points based on specific characteristics. There are various clustering algorithms with K-Means and Hierarchical being the most used ones. Some of the use cases of clustering algorithms include:

* Document Clustering
* Recommendation Engine
* Image Segmentation
* Market Segmentation
* Search Result Grouping
* and Anomaly Detection.

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. 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 and **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. 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**.

**Reachability and Connectivity**

These are the two concepts that you need to understand before moving further. Reachability states if a data point can be accessed from another data point directly or indirectly, whereas 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**

## ****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 w ill be taken as noise. Whereas, if chosen too big then various small clusters will merge into a big cluster, and we will lose details.