DBSCAN

Mannem Charan AI21BTECH11019

Abstract

This report consists of my basic understanding of one of the popular Ml methods "DB-SCAN".

1 DBSCAN

DBSCAN is an unsupervised learning algorithm and used for clustering problems.Every clustering method follows the same approach i.e., it will try group the **similar** data points together as clusters.Although we have *k*-means clustering algorithm to do clustering , sometimes it fail to show the results that we need and DBSCAN can be used as a replacement for *k*-means.

2 DBSCAN vs *k*-means

To understand where *k*-means fails to do the job let us consider the dataset shown in fig 2.1, The *k*-means algorithm which works on distance between the points identifies

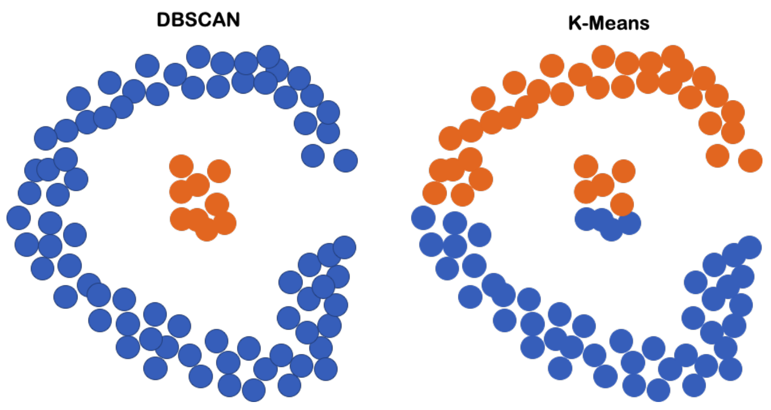

Fig. 2.1: DBSCAN vs K-means

the clusters w.r.t the mean points so sometimes it may not be similar to how a human eye clusters the data.In 2.1, DBSCAN able to mimic how a human eye identifies the clusters.So for this type of data it is best to use DBSCAN.DBSCAN works based on the densities of the points.

3 How DBSCAN works?

DBSCAN is a density-based clustering algorithm and the clusters are made sequentially one after other.To understand that let us take some unclustered data as shown in fig 3.2. Now one thing that we can do is count the number of points close to each point.

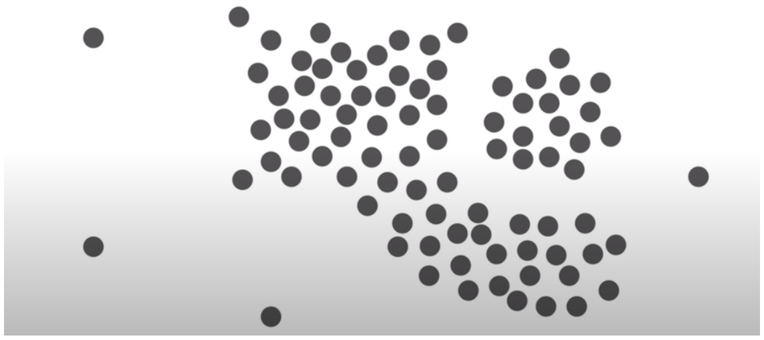
Fig. 3.2: unclustered data

In DBSCAN,we will do it by taking a small circle around the data point and the data points which are inside/crossing this circle are considered to closer to the given data point.In this case the red point is close to 8 data points in fig 3.3 And the radius of circle

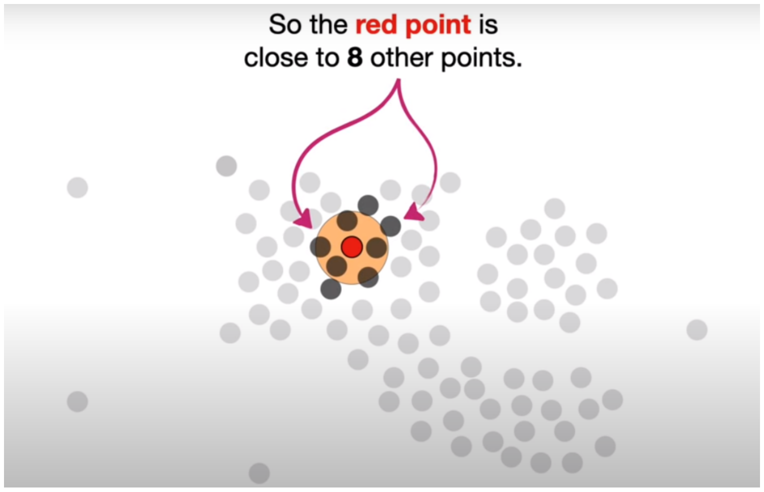

Fig. 3.3

used for finding the number of points closer to given data point is user defined known as "**eps**"(ϵ). It is one of the tuneable parameters in DBSCAN. And like this we will see

the neighbourhood of each point and count the no. of points **closer** to each point.
Then we define something called core point which has neighbouring points greater than certain threshold.This threshold is known as "**minPts**". This threshold decides whether a neighbourhood around data point is "dense" enough.This parameter is also user-defined and tuneable.Using this parameter, we will find all core points in the data.In this case we took threshold as 4 points and the core points of the data are shown in fig 3.4,
And remaining data points are non-core points.After that we randomly pick a core point

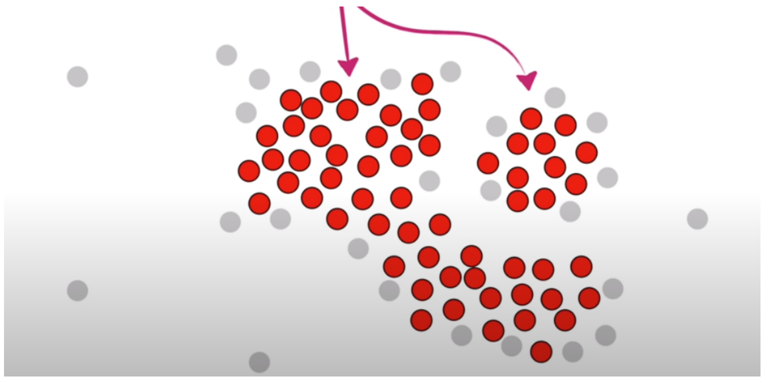
Fig. 3.4: Core points marked as red dots

and assign it to the first cluster.And then we will assign all the core points which are closer the given core point to the first cluster.Similarly we will add all core points closer to the points in the cluster and so on.Like that the cluster will grow eventually but at each step we are only adding the core points to the cluster (atleat for now) neglecting the non-core points.After we add all core points to the cluster we then add the non-core points which are **closer** to the points of cluster.Note that only core points are responsible for growing the cluster, non-core points are added to cluster depending on how close they are with core points.And like that we will identify the first cluster as shown in fig 3.5. Similarly you will form the second cluster by taking the core point that is not included in first cluster.Overall the data set after clustering look like fig 3.6, It has been grouped into two clusters and with some outliers.And this ends the DBSCAN algorithm.

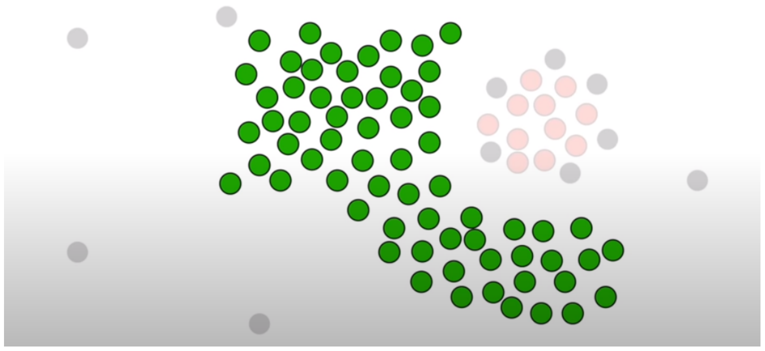

Fig. 3.5: First cluster

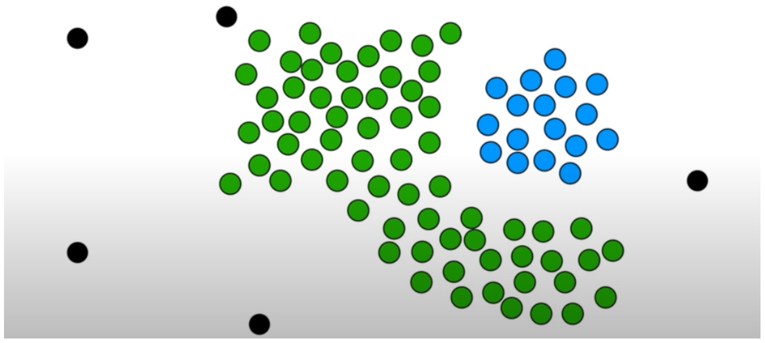

Fig. 3.6: Clustered data