Introduction to Machine Learning

Assignment 6

Mohamed Gamil (s3897605)
Sofia Teeriaho (s3887626)

October 28, 2021

Introduction

In this assignment, we are asked to implement the Density-based spatial clustering of applications with noise(DBSCAN) algorithm on a two dimensional dataset of 200 datapoints. This unsupervised clustering algorithm should group together data points that are packed "densely" and identify points that are positioned alone as "outliers". The main parameter settings for this function are MinPts(minimum points to form a cluster) and eps(the threshold value or radius when forming a cluster). We have to use the k-nearest neighbor search graph to find the most optimal eps value. We then use that eps value with different MinPts values. The most optimal setting in terms of MinPts should be found by comparing silhouette scores in a silhouette table.

Method

As previously mentioned, this algorithm takes two parameters as input, the the radius formed around a datapoint(eps) and the minimum number of points needed to form a cluster(MinPts). To be able to explain the algorithm, some terms should be introduced:
e-neighborhood: The radius formed around a datapoint -based on the eps value- and the other points it contains.
If a datapoint has a sufficient number of points in its e-neighborhood (>=MinPts) then its a **core point**.
If a datapoint has less than sufficient number of points in its e-neighborhood, but it is in the e-neighborhood of another core point, it is a **border point**.
A datapoint that is neither core or border, is labelled as an **outlier**.

It first randomly chooses an unvisited datapoint as the start. Using the eps value, a circle with a specific range is created around that datapoint and all other points in that circle would be part of the datapoint's e-neighborhood. Based on the number of points in the e-neighbourhood, it is classified as a core or outlier point. The algorithm moves to the other points in the neighbourhood, classifying them as border or core points based on their e-neighborhood. When

one of those points is a core point, its e-neighborhood is added to the original cluster. This is repeated until a connected cluster has been went through, then a new datapoint is reached and it starts a new cluster. The whole process is looped until all points are eventually visited.

To find the optimal epsilon value, we used K nearest neighbor search graphs for the three different values of k(3,4,5), k being the variable for MinPts.
To find the most optimal value for MinPts. Different values for MinPts must each have different effects on the clustering algorithm's performance. A metric to analyse the quality of the algorithm's clustering created on the data is by using the silhouette score. In basic terms it creates a comparison using mainly two things, separation and cohesion. Cohesion is a measure of similarity between a datapoint and other datapoints in its cluster. Separation is simply how different a datapoint is when compared to other datapoints in other clusters. The range of the silhouette is between the values -1 and 1. Ideally, a datapoint that "belongs" to its cluster would have a higher silhouette value. Thus, if most datapoints have a high silhouette value, that means that they are well matched to their clusters and the clustering has worked well. In order to be able to analyse this, we took the means of the silhouettes in the different cases to be able to compare them, using a table. The table displays the difference silhouette scores between different values for MinPts.

$$Silhouette_{score} = \frac{1}{n} \sum_{i=1}^{n} \frac{b_i - a_i}{max(a_i, b_i)}$$

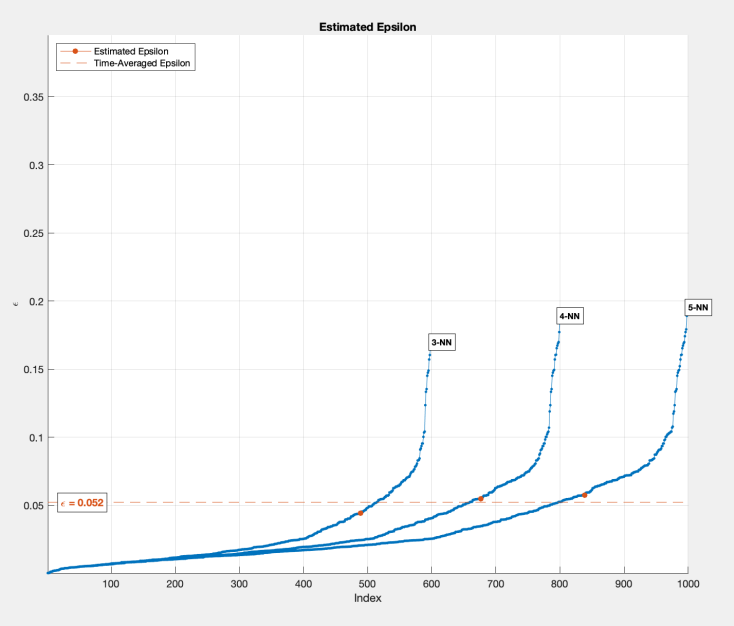
Code implementation

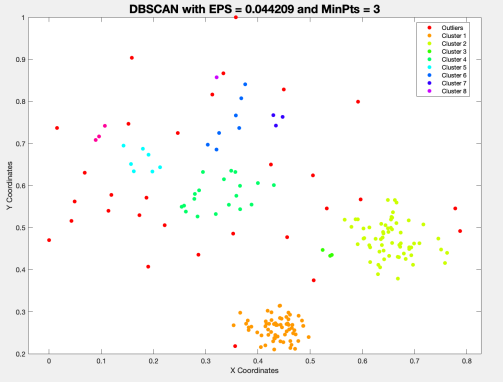
The code was implemented using the pseudocode for DBSCAN. This includes three functions `dbscanFunc`, `regionQuery` and `expandCluster`. Additional functions like `prepareData` and `findEuclidian` were created previously for other purposes but utilised in this assignment as well. `prepareData` was primarily used to add two additional columns, one for keeping track of cluster labels and the other one for keeping track of whether that point has been visited or not [1 0]. We used our function `findEuclidian` from previous assignments, however, we chose to use regular euclidian distance instead of squared euclidian distance given that `clusterDBSCAN.estimateEpsilon()` uses regular euclidean distance. Plots are printed as separate figures given that MinPts = 3, 4, 5. An average silhouette score is calculated using only inliers of the data, this means that if the a data point is labeled -1, which is an outlier label, then it is not included in the silhouette score. `data(:,3)` represents cluster labels. The scores are averaged using `mean(s)`.

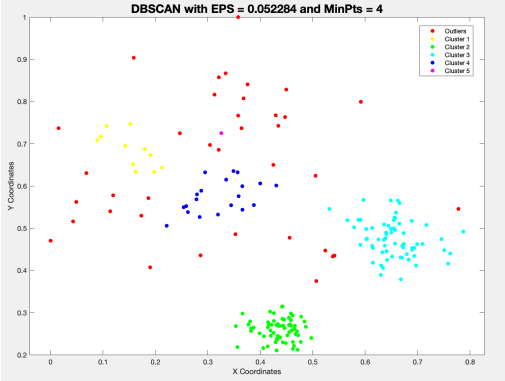
```
sData = [];
for j = 1:P-1
    if data2(j,3) ~= -1
        sData(end+1,:) = data2(j,:);
```

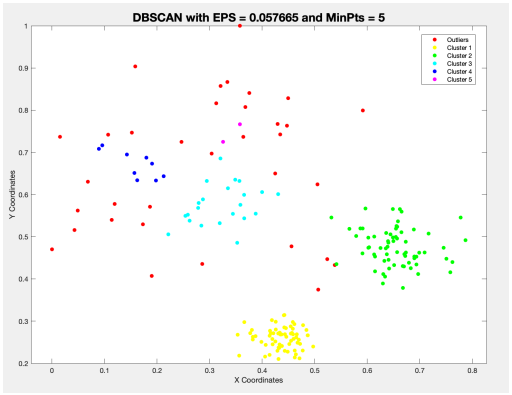
```
    end
end
% Calculate silhouette score
s = silhouette(sData(:,1:2),sData(:,3));
mean(s)
```

Results









Silhouette Score Table

The silhouette scores achieved below were obtained by using the built in function. As an advantage from the DBSCAN algorithm outlier detection feature, we were able to avoid including all noise from the silhouette score calculation. As expected, this resulted in higher scores. Each column represents the silhouette score when a certain value of MinPts and its corresponding epsilon value were applied.

MinPts =	3	4	5
Score =	0.6979	0.8793	0.8762

Discussion

We can assume that DBSCAN has certain advantages when it comes to comparing it with other algorithms such as the k-means and hierarchical clustering algorithms. The main advantage is its insensibility to noise and its ability to manage different shapes and sizes of clusters.

When agglomerative hierarchical clustering was applied to the same dataset, we achieved the following silhouette scores:

Silhouette Score Table of Hierarchical Clustering

	Single	Complete	Average	Ward
K = **2**	0.4711	0.6101	0.7207	0.7207
K = **3**	0.1669	0.5500	0.8049	0.8027
K = **4**	0.1816	0.5852	0.7789	0.7744

Thus, we could see how the DBSCAN algorithm has handled this dataset way more efficiently given that its silhouette scores are generally higher when using certain values of MinPts.

There is no notable difference between MinPts = 4 and MinPts = 5. However, the slight improvement when using MinPts = 4 fits the general rule that when a 2 dimensional dataset is used, it is best to use MinPts = 4(check below for source). Evidently, from the silhouette score the algorithm performs poorly with MinPts = 3. This could be explained as it forms more clusters since there is a lower threshold to form one. Thus, noise(outliers) are not detected and instead taken into clusters.

source:
http://www.sefidian.com/2020/12/18/how-to-determine-epsilon-and-minpts-parameters-of-dbscan-clustering/