Introduction to Machine Learning
Lab Session 6

Group 39
Lubor Budaj (s4167376) & Gasan Rzaev (s3553213)

October 2, 2022

1. Introduction

The goal of this assignment is to implement DBSCAN (Density based spatial clustering of applications with noise) algorithm using simple (squared) Euclidean distance. The data set used in this implementation is the same data set used in the previous assignment. The paramaters of the algorithm namely, $minPts = 3, 4, 5$, which stands for the minimum number of points to form a dense region , and *eps*, which stands for the radius of the neighbourhood around a data point, will be optimised in the experiments part of the implementation. For each iteration of the experiment with given parameter, we will calculate the silhouette score.

The dimensions of the data set are 200 by 2 - it contains 200 samples. Each sample contains x feature and y feature. The data is unlabeled.

In this report we will first introduce DBSCAN the Method section. Next we will provides the result of our experiments on the given data set in the Results section. Lastly, in the Discussion section we will interpret the results.

2. Method

The DBSCAN algorithm, classifies all data points in three categories, namely: **core** point, **border** point and **noise** or **outlier**. The following are the differences between the points:

- **Core point** is the point that has at least *minPts* points in its radius *eps* (ϵ).
- **Border point** is the point that has less than *minPts* points in its radius *eps*, however is in the radius of the Core point.
- **Noise or outlier** is the point that is not a core point or a border point (All the others).

Using these classifications, the algorithm visits all the data points. If they are either a core point or a border point, they are added to the clusters, otherwise they are left as an outlier.

Our implementation is based on the provided pseudo-code. We used build-in functions to estimate ϵ for given $minPts$ and to calculate the silhouette score for each combination of parameters. We haven't included the data unassigned to any cluster (noise) in the calculation of the silhouette score.

We made out implementation scalable. It's very easy to change the default parameters to produce different results. More over, our implementation works for different data sets too. We made experiments on outlier detection by DBSCAN using thyroid disease data set provided in the bonus question.

3. Results

In this section, the results of the implemented algorithm are presented in two different ways: **Qualitative** and **Quantitative** results.

Qualitative results consist of a figure displaying three plots obtained using k-nearest neighbour search, in order to get value of parameter ϵ, 3 scatter graphs showing assignment of the data points to different clusters and another figure displaying three plots obtained using k-nearest neighbour search, this time for thyroid data set.

Quantitative results will show a table listing the computed silhouette score for different values of parameter *minPts* and a table showing the results of outlier detection for thyroid data set for different values of parameter *minPts*.

3.1. Qualitative results

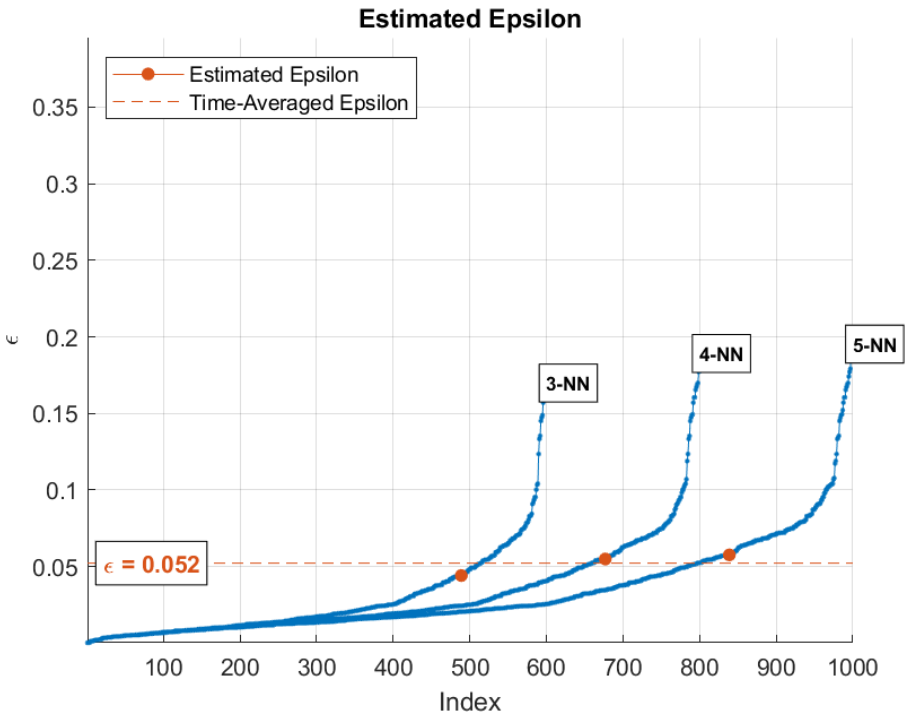

Figure 1: *K-nearest neighbour ϵ estimation*

3.2. Qualitative results

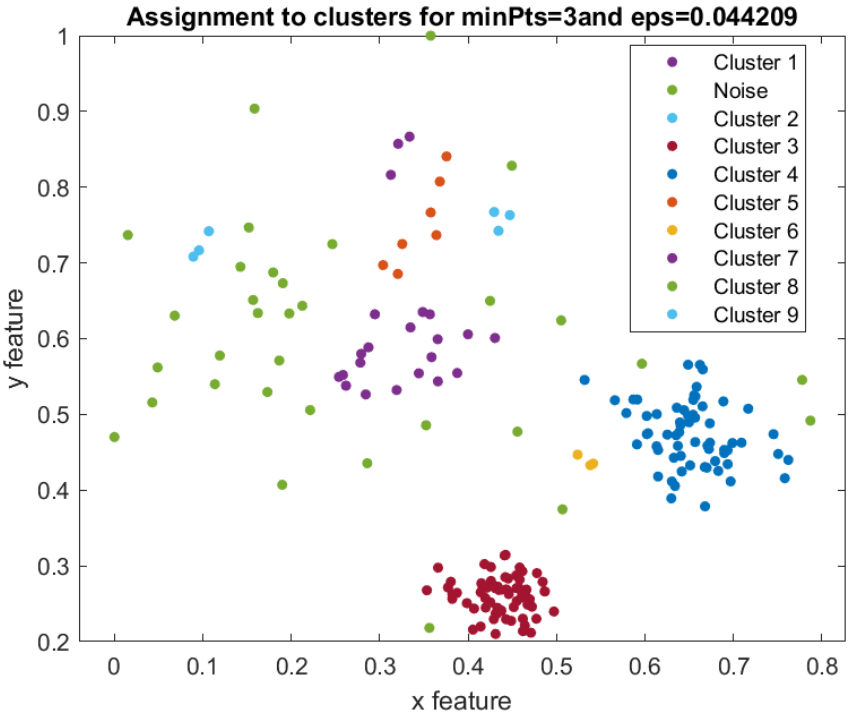

Figure 2: $\epsilon = 0.0493$

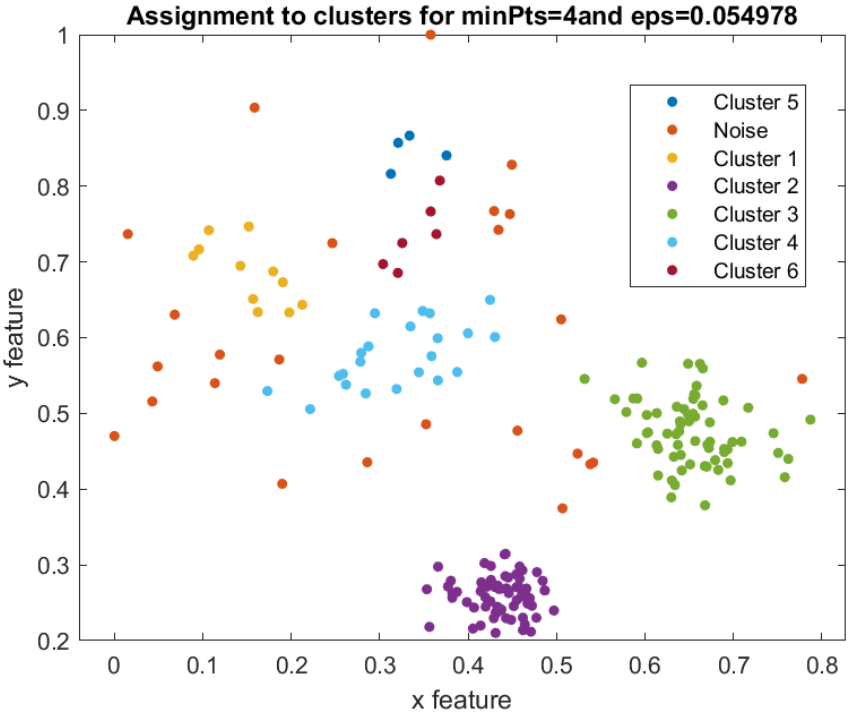

Figure 3: $\epsilon = 0.0442$

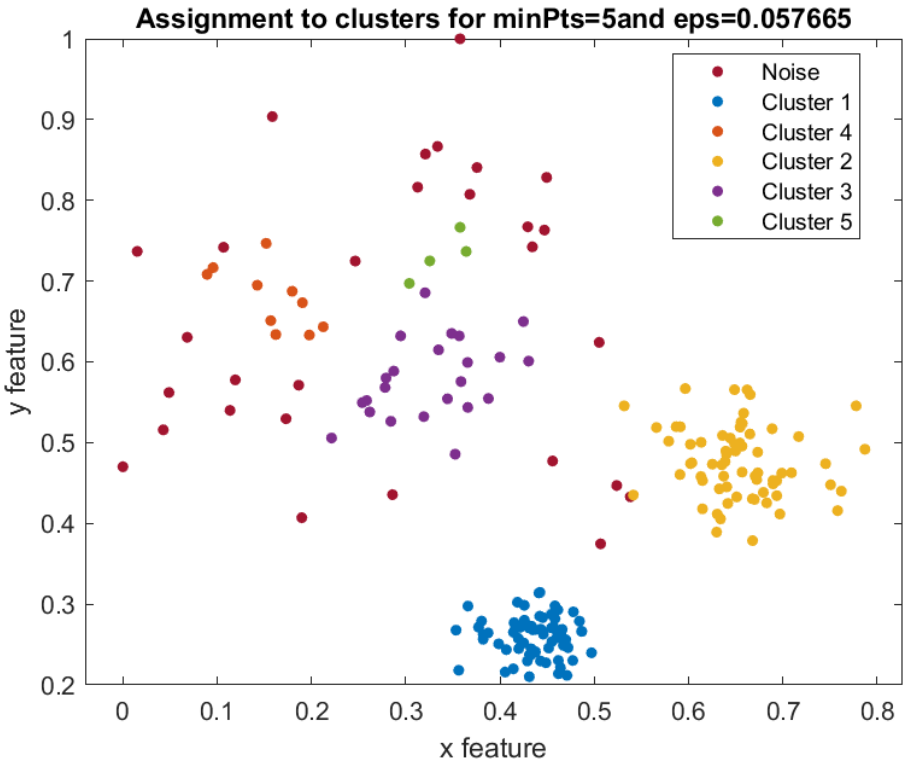

Figure 4: $\epsilon = 0.0.0550$

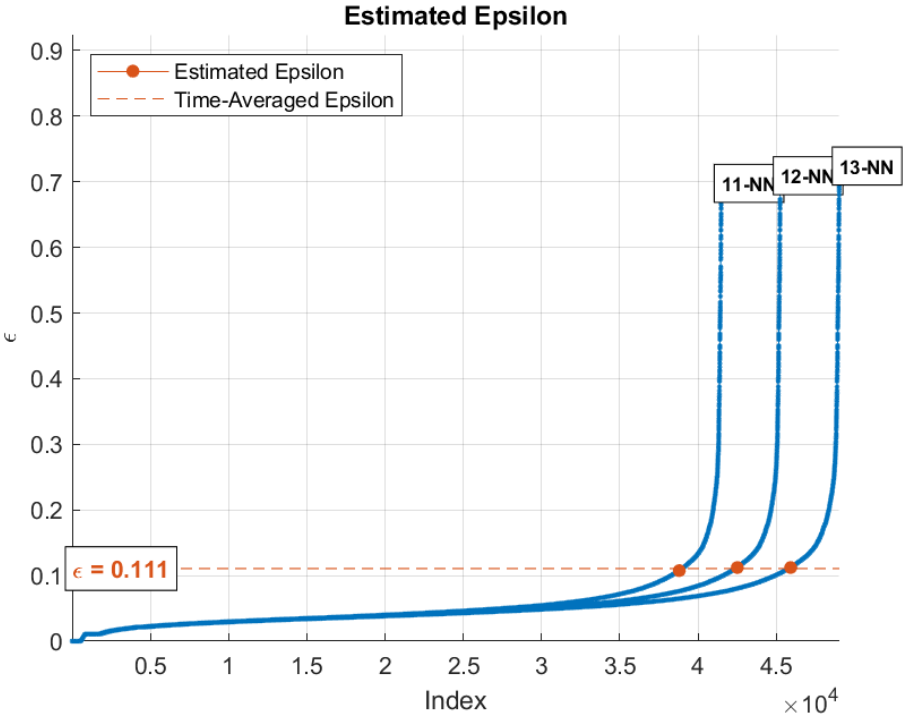

Figure 5: *K-nearest neighbour ϵ estimation for thyroid data-set*

3.3. Quantitative results

minPts	3	4	5
silhouette score	0.6947	0.8559	0.8618

Figure 6: *Table showing silhouette scores for different values of $minPts$, ϵ is set to* 0.0442, 0.055 *and* 0.0577 *respectively. The data not assigned to any cluster (noise) is not included in the data used to calculate the silhouette scores.*

minPts	11	12	13
miss-qualified proportion	0.0212	0.0207	0.0207

Figure 7: *Table showing proportion of miss-qualified samples for different $minPts$, where ϵ is set to* 0.1075, 0.1122 *and* 0.1122 *respectively for thyroid disease data set.*

4. Discussion

First we will discuss qualitative and then quantitative results.

In Figure 1 we can see the estimation ϵ using k-nearest neighbour graph for $k \in \{3, 4, 5\}$. As k increases, the estimated ϵ increases too.

In Figure 2 we can see the assignment of data points to clusters for $minPts = 3$ and $\epsilon = 0.0442$. Using these parameters in the experiment, in total we obtained 9 clusters and 27 data points were not assigned to any cluster (noise). Some clusters, such as Cluster 1, Cluster 2, Cluster 6 or Cluster 9, consist only of 3 data points. Having many small clusters, we don't expect high silhouette score using these parameters.

Next, in Figure 3 we can see the assignment of data points to clusters for $minPts = 4$ and $\epsilon = 0.055$. Using these parameters in the experiment, in total we obtained 6 clusters and 26 data points were not assigned to any cluster (noise). Compared to the previous figure (2), there is only one cluster (Cluster 5) with small number of data points - 4 data points. Other than that the assignment to cluster seem to be as expected, hence we expect high silhouette score.

Lastly, in Figure 4 we can see the assignment of data points to clusters for $minPts = 5$ and $\epsilon = 0.0577$. Using these parameters in the experiment, in total we obtained 5 clusters and 30 data points were not assigned to any cluster (noise). The changes compared to the previous figure (3 are not significant. Cluster are similar to the the previous figure, other than one cluster being noise now. There is one small cluster (Cluster 5). We expected similar silhouette score to the previous figure.

Silhouette score for each iteration of the experiment can be seen in Figure 6. The resulting silhouette score were not surprised to us, as we expected higher scores for the 2^{nd} and 3^{rd} iteration. The resulting silhouette scores are relatively high, but it is important to note that the data point not assigned to any cluster were not considered for the calculation of the scores.

4.1. Outlier detection in thyroid data set

In Figure 5 we can see the estimation ϵ using k-nearest neighbour graph for $k \in \{11, 12, 13\}$. The values of k are chosen from range given by the formula $x * 2 + -1$, where x is the number of dimensions of the data, which is 6 for this data set. The ϵ estimation produced the following values for ϵ: 0.1075, 0.1122 and 0.1122 in respective order. Compared to the previous experiments, in this one we were interested only if the data was part of any cluster or not (noise). In Figure 7

are the results of the cluster/noise classification compared to the labels provided in the data set. The proportion of miss-qualified labels is relatively low, therefore we conclude that this data set is suitable for outlier detection using DBSCAN.

5. Work Distribution

The work among the group members was distributed in a following way:

- Code: 50% done by Lubor, 50% by Gasan
- Report: 50% done by Lubor, 50% by Gasan
- Graphs: 50% done by Lubor, 50% by Gasan

The code used to complete this assignment is presented in Listing 1 in Appendix A.

A. Appendix

Listing 1: *This is the file* `code.m`

```
%import
data = readmatrix("data_clustering.csv");
% XY = importdata("thyroid.mat");
% data = XY.X;
% labels = XY.y;

%initialization of dimensions of the data
dimensions = size(data);
N = dimensions(2);
P = dimensions(1);

%set parameters
min_pts = 2 * N;
min_pts_from = min_pts - 1;
min_pts_to = min_pts + 1;

%vectors declaration
C = zeros(min_pts_to - min_pts_from + 1, P);
s_score = zeros(1, min_pts_to - min_pts_from + 1);
s_score_no_noise = zeros(1, min_pts_to - min_pts_from + 1);
eps = zeros(1, min_pts_to - min_pts_from + 1);
missqualified = zeros(1, min_pts_to - min_pts_from + 1);
cluster_data = zeros(P, N);
cluster_alloc = zeros(P, 1);

%graph for epsilon choice
figure(1);
clusterDBSCAN.estimateEpsilon(data, min_pts_from + 1, min_pts_to + 1);

for pts = min_pts_from:min_pts_to

    idx = pts - min_pts_from + 1;

    %epsilon estimation
    eps(idx) = clusterDBSCAN.estimateEpsilon(data, pts + 1, pts + 1);

    %DBSCAN
    C(idx, :) = DBSCAN(data, eps(idx), pts);

    %data manipulation for silhouette scores and figures
    lgnd = categorical(C(idx, :));
    j = 0;
    for i = 1:P
        if lgnd(i) == "-1"
            lgnd(i) = "Noise";
        else
            j = j + 1;
```

```
            cluster_data(j,:) = data(i,:);
            cluster_alloc(j) = C(idx, i);
            lgnd(i) = "Cluster " + char(lgnd(i));
        end
        if (C(idx, i) == -1 && labels(i) == 0) || (C(idx, i) > -1 &&
   labels(i) == 1)
            missqualified(idx) = missqualified(idx) + 1;
        end
    end
    missqualified(idx) = missqualified(idx) / P;

    %silhouette score
    s_score(idx) = sum(silhouette(data, C(idx, :))) / P;
    s_score_no_noise(idx) = sum(silhouette(cluster_data(1:j, :),
   cluster_alloc(1:j))) / j;

    %figures
    if N == 2
        figure(pts - min_pts_from + 3);
        color = lines(unique(C(idx, :)));
        gscatter(data(:, 1), data(:, 2), lgnd, lines);
        title("Assignment to clusters for minPts=" + int2str(pts) + "
   and eps=" + num2str(eps(idx)));
        xlabel("x feature");
        ylabel("y feature");
    end
end


function C = DBSCAN(D, eps, min_pts)
    C = zeros(1, height(D));
    i = 1;
    visited = zeros(1, height(D));
    %for each point
    for P = 1:height(D)
        %unvisited point
        if visited(P) == 0
            visited(P) = 1;
            neighbours = region_query(D, P, eps);
            %has enough neighbours?
            if length(neighbours) < min_pts
                C(P) = -1;
            else
                %make new cluster
                [C, visited] = expand_cluster(D, P, visited,
   neighbours, C, i, eps, min_pts);
                i = i + 1;
            end
        end
    end
end
```

```
function [C, visited] = expand_cluster(D, P, visited, neighbours, C, i,
    eps, min_pts)
    C(P) = i;
    P_prime = 1;
    %for each point in the neighbourhood
    while P_prime <= length(neighbours)
        %unvisited point
        if visited(neighbours(P_prime)) == 0
            visited(neighbours(P_prime)) = 1;
            neighbours_prime = region_query(D, neighbours(P_prime),
   eps);
            if length(neighbours_prime) >= min_pts
                %union of neighbours and neighbours_prime
                neighbours = unique([neighbours, neighbours_prime], '
   stable');
            end
        end
        %add to cluster if P_prime not memember of any cluster
        if C(neighbours(P_prime)) <= 0
            C(neighbours(P_prime)) = i;
        end
        P_prime = P_prime + 1;
    end
end

function N = region_query(D, P, eps)
    N = zeros(1, height(D));
    i = 1;
    for P_prime = 1:height(D)
        %is in the neighbourhood?
        if sqrt((D(P, 1) - D(P_prime, 1))^2 + (D(P, 2) - D(P_prime, 2)
   )^2) <= eps %&& P ~= P_prime
            N(i) = P_prime;
            i = i + 1;
        end
    end
    N = N(N>0);
end
```