6 Practical: Density-based clustering and outlier detection

For this assignment, you will use the same dataset used in assignment 5.

Implement the Density-based spatial clustering of applications with noise (DBSCAN) algorithm as introduced and discussed in class, using simple (squared) Euclidean distance. Consider different settings for the parameter *minPts=3,4,5*. (Parameter *eps* will be optimised in the experiments.)

Your code should have the following structures:

Pseudo code for DBSCAN

```
def  DBSCAN (D, eps, MinPts)
    C = 0
    for each unvisited point P in dataset D do
      mark P as visited
      NeighborPts = regionQuery(D, P, eps)
      if size (NeighborPts) < MinPts then
            mark P as NOISE
      else
            C = next cluster
            expandCluster(P, NeighborPts, C, eps, MinPts)
      end if
    end for
def expandCluster (P, NeighborPts, C, eps, MinPts)
    add P to cluster C
    for each point P' in NeighborPts do
      if P' is not visited then
            mark P' as visited
            NeighborPts' = regionQuery(P', eps)
            if size(NeighborPts')≥ MinPts then
                  NeighborPts = NeighbotPts joined with NeighborPts'
            end if
      end if
      if P' is not yet member of any cluster then
            add P' to cluster C
      end if
    end for
def regionQuery(D, P, eps)
    N= empty list
    for each point P" in dataset D do
      if Distance (P", P) ≤ eps then
            N = N ∪ P"
      end if
    end for
    return N that is all points within P's eps-neighborhood (including P)
```

Report

You should hand in a structured report comprising:

- **(1 point)** An **Introduction** section that describes your assignment.
- **(3 points)** A **Methods** section in which you explain the DBSCAN algorithm in a general manner. You need to implement the DBSCAN algorithm yourself. Code and implementation itself will also be taken into account for the grading of this section.
- **(4 points)** A **Experimental results** section in which you provide both qualitative and quantitative results.

  For the qualitative results, you should include the following (in total, four figures):

  - A figure displaying three plots obtained using *k*-nearest neighbour search and mark one possible elbow point on each plot to indicate the threshold value (*t*) for parameter *eps*. Here we set *k=3,4,5*. You can use the build-in function for *k*-nearest neighbour search/graph [3].
  - A figure displaying the resulting clusters and outliers (*minPts*=*k*, *eps*=*t*).

  For the quantitative results, you should provide a table listing the computed silhouette score for different parameter setting of *eps*=*t* and *minPts*=*3,4,5* (in total, 3 silhouette scores). You can use the same build-in function used in Assignment 5.
- **(2 points)** A **Discussion** section that includes your observations on both qualitative and quantitative results, and conclusions for the best choice of parameter settings.

Bonus (suggestions)

2 points max. in total:
You are given a so-called 'Thyroid disease data set' (`http://odds.cs.stonybrook.edu/thyroid-disease-dataset/`). Outliers are indicated with label 1 (others with label 0 are inliers).

- Please apply DBSCAN on this data set with different settings for the parameters (You can set 3 different values for the parameter *minPts* and then apply the *k*-nearest neighbour search to get the optimal *eps*).
- Use the provided labels to compare your results with the ground truth (i.e. the known outliers) and report the measures, such as, Precision, Recall, and F-score.

[3] For MATLAB users, you can use the function *clusterDBSCAN.estimateEpsilon(X,MinNumPoints,MaxNumPoints)* to get the *k*-nearest neighbour graph with different *k* in one go. For Python users, function *sklearn.neighbors.NearestNeighbors(n_neighbours=k)* is helpful. Once you obtain the distance matrix with the specified *k*, you need to sort it in descending or ascending order and plot the curves.