Mandatory 4: DBSCAN

02807 Computational Tools for Data Science

Submission opens:	**8:00 Saturday, November 10, 2018 (or sooner)**
Deadline:	**20:00 Sunday, November 18, 2018**
Individual:	**This assignment must be completed individually** (see collaboration policy on course page)

Exercise

In this mandatory assignment, you have to do your own implementation of the DBSCAN algorithm for 2D points using the Euclidean distance metric. You can either use the slides or the Wikipedia article for pseudocode (`https://en.wikipedia.org/wiki/DBSCAN`).

Your program must read the three command line arguments: filename, eps, and minPts. "eps" and "minPts" are the two parameters your DBSCAN algorithm should use. "filename" is the name of a file with the format (the same format as used on the clustering weekplan):

```
n;d;
x1;y1;
x2;y2;
...
xn;yn;
```

where n is the number of points, and d is the dimension (for this assignment it will always be 2), and $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$ are the points.

For each point in the input, you must output a single number that corresponds to the cluster that point belongs to. Noise points should be considered as belonging to cluster 0, and all real clusters should be numbered $1, 2, \ldots$.

Your program **must** process the points in the order described in the file, this will guarantee only one unique correct output exists – only this solution will be accepted (even though other clusterings might be considered correct in general).

You do not need to use any smart data structure to find neighboring points in order to solve the mandatory part of this assignment.

We strongly recommend you use your visualization and test data from the Clustering weekplan to test your solution.

Submission You must submit your solution on CodeJudge **before** the deadline (any submissions made after the deadline will **not** be counted). Your latest submission before the deadline will be used for grading thus you **must** ensure your best solution is uploaded as the latest. Submission on CodeJudge will not be available before the submission opens time stated in the top.

Grading In order to obtain full points for this assignment, you must implement all the described functions in a correct and efficient way. From submission opens until the deadline you can submit your code on CodeJudge and see how well it scores. An entirely correct solution gets 100 points. Partially correct/efficient solutions will get a score between 0 and 100. On CodeJudge you can see the results of all the tests, but all except the already provided sample test data will be hidden.

The score you obtain in this mandatory assignment will count towards your final grade.

Competition If you are up for an extra challenge, there will be a competition among those interested to make the fastest possible solution. This is completely voluntarily, and will **not** count towards your final grade – instead there will be a small prize for the fastest solution.

The problem in the competition is exactly like the mandatory part except n might be bigger, so everybody can join the competition with their solution without extra effort. We will simply sum the number of solved test cases, and in the case of a draw the sum of the running time of all passed tests will decide who got the fastest solution.

To join the competition, simply also submit your best solution to the "Competition" exercise on CodeJudge (you must still submit to the mandatory exercise as well). A live scoreboard will show the current standings on CodeJudge.