Warsaw University of Technology
The Faculty of Electronics and Information Technology

Data Mining (EDAMI)
Project Documentation

Clustering based on density

Authors:

Adam Stelmaszczyk
Anonymous Coauthor

January 7, 2019

Contents

1 Project task

Implementation and experimental evaluation of DBSCAN [2] and DENCLUE [3] clustering algorithms.

2 Data set

For the experiments a data set with geometrical properties of wheat kernels was chosen, from the *UCI Irvine Machine Learning Repository* [1]. Kernels belong to 3 different varieties of wheat: Kama, Rosa and Canadian. Data set consists of 210 elements, 70 per variety.

2.1 Data set attributes information

Each element consists of 7 real-valued attributes, which are geometric parameters of wheat kernels, that were measured using the X-rays:

1. area A,
2. perimeter P,
3. compactness $C = \frac{4\pi A}{P^2}$,
4. length of kernel,
5. width of kernel,
6. asymmetry coefficient
7. length of kernel groove.

3 Descriptions of used algorithms

Two popular clustering algorithms were used, DBSCAN [2] and DENCLUE [3]. In this section brief descriptions of both methods are presented.

3.1 DBSCAN algorithm

DBSCAN algorithm is relatively simple algorithm controlled with 2 parameters, namely EPS and MIN_PTS. [2] Basically, we are iterating over a set of unvisited points. If we found a core point (a point which has at least

MIN_PTS in its Eps-neighbourhood), we are starting a new cluster. Looking at the neighbours of found core point we are trying to expand this new cluster as much as possible (in respect to the Eps-neighbourhood).

3.2 DENCLUE algorithm

DENCLUE algorithm is based on the idea that the influence of each data point can be modeled using a mathematical function (influence function). The overall density of the data space can be calculated as the sum of the influence function of all data points. Clusters can be determined mathematically by identifying density-attractors, which are the local maxima of the overall density function. The DENCLUE algorithm works in two steps:

1. Preclustering step, in which a map of the relevant portion of the data space is constructed. The map is used to speed up the calculation of the density function which requires to efficiently access neighboring portions of the data space.

2. Actual clustering step, in which the algorithm identifies the density-attractors and the corresponding density-attracted points.

4 Implementation details

As implementation language for the project Java was chosen. Data set is read from a `seeds_dataset.txt` text file and passed to a standard input of our bechmarking program. In the benchmark, set of objects that represent points is created. DBSCAN and DENCLUE algorithms are using that set. Other arguments that algorithms need are hard-coded. Whole program consists of five packages:

- `algorithms` In this package abstract class ClusteringAlgorithm and inherited from it classes DBSCAN and DENCLUE are located.

- `structures` Here are located basic structure classes like Cluster, Point and Points. Diagram class is shown on the picture 2.

- `scorer` This package consists of a Scorer class, which used to calculate clustering quality measure. Diagram class is shown on the picture 3.

- `visualizer` Used for the clusters visualization. Diagram class is shown on the picture 4.

- `main` Used for reading the data set from the file and bechmarking algorithms. Diagram class is shown on the picture 5.

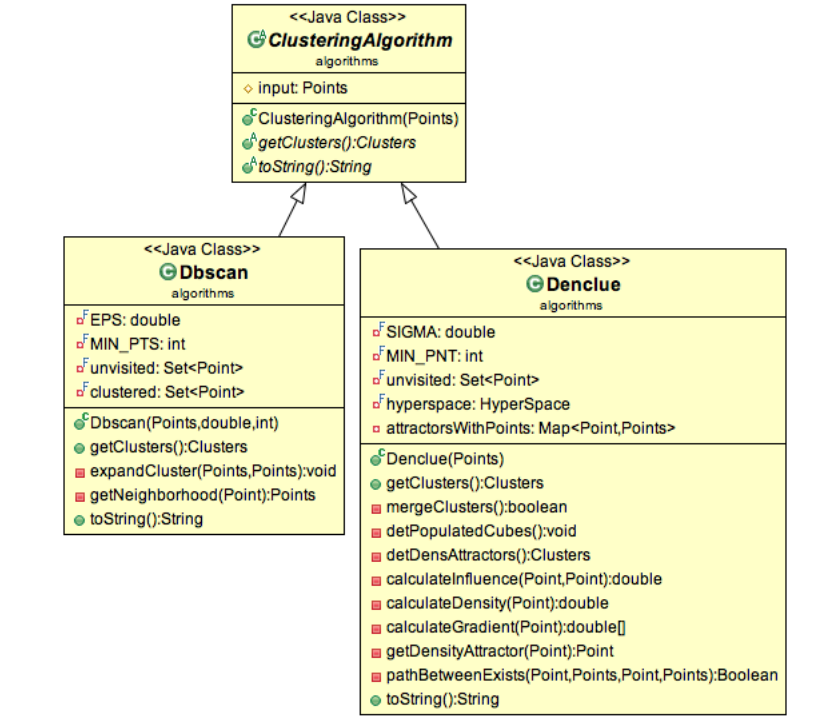

Figure 1: Class diagram for algorithms package

5 User guide

To run benchmark program please follow this 3 steps:

1. Add execution rights to 2 scripts with a command `chmod +x build.sh; chmod +x run.sh`.
2. Compile Java source code running `./build.sh`. `bin` catalog should appear.
3. Run benchmark program with `./run.sh`.

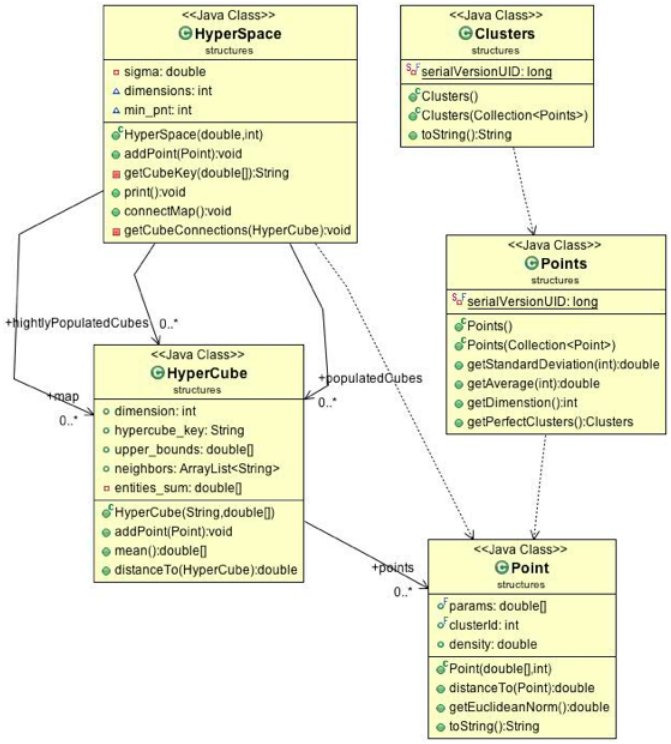

Figure 2: Class diagram for structures package

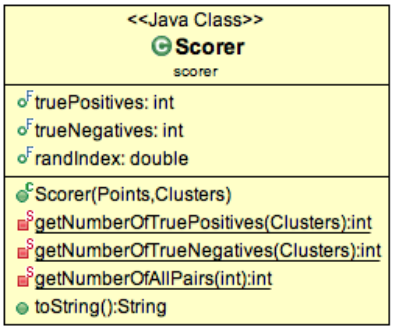

Figure 3: Class diagram for scorer package

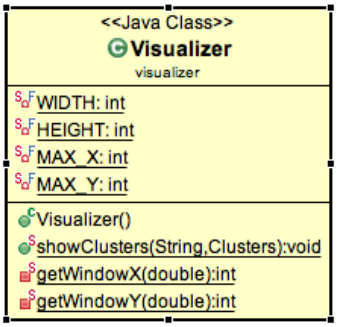

Figure 4: Class diagram for visualizer package

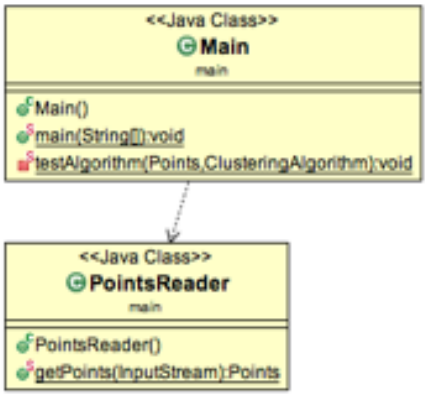

Figure 5: Class diagram for main package

6 Experimentation results and analysis

As a result of execution DBSCAN and DENCLUE algorithms two sets of clusters are generated. This clusters are represented in 2D view as a set of points, drawn as numbers (fig. 6, 7, 8). Each number is representing cluster ID. *Area* and *asymmetry coefficient* are on axes, because this attributes have the greatest standard deviations (fig. 9).

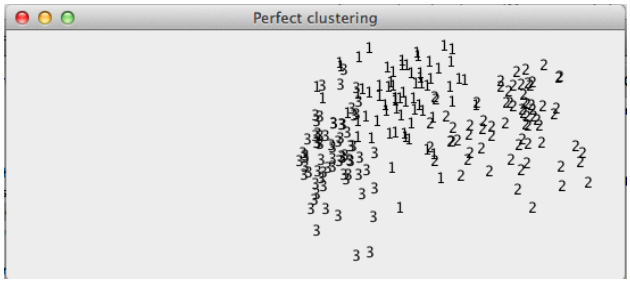

Figure 6: Perfect clustering

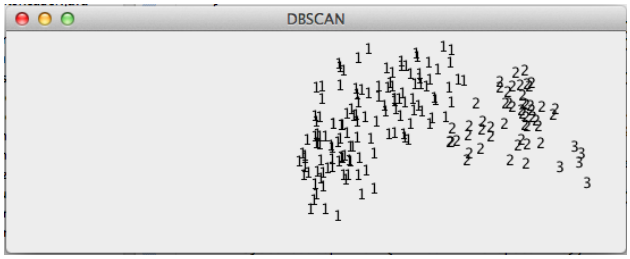

Figure 7: Clustering made by DBSCAN algorithm

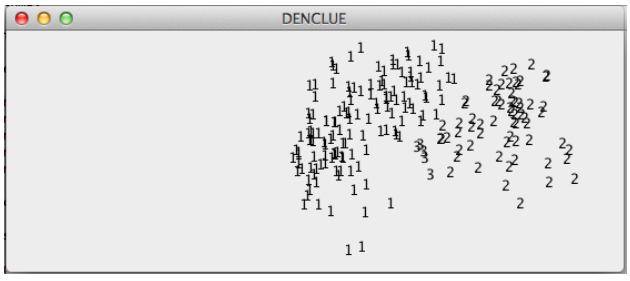

Figure 8: Clustering made by DENCLUE algorithm

Also some statistics for the running algorithms are shown, such as number of *true positives*, *true negatives* and *Rand index* [**?**] (picture 9).

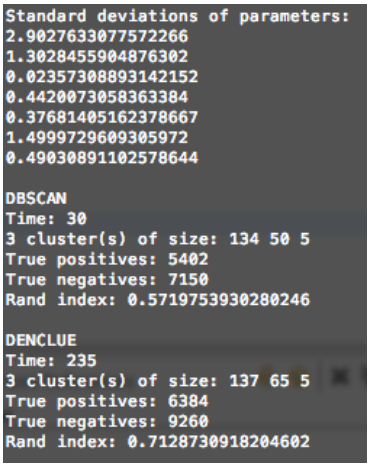

Figure 9: The output of the program

Parameters for the algorithms were chosen experimentally. The best results were achieved using:

$$EPS = 0.9, MIN\_PTS = 5$$

for DBSCAN and

$$SIGMA = 0.7, EPS = 2$$

for DENCLUE algorithm.

It's difficult to measure memory used by algorithms, because of the Java automatic garbage collection, which is implementation-specific, nondeterministatic and out of a programmer's control. The attempts to calculate free memory before and after the algorithm execution were not successful: sometimes there was no difference in the amount of free memory. Comparison of algorithms running times is presented on figure 10.

Both algorithms gave different clusterization. Quality measure (Rand index [**?**]) was always higher for DENCLUE algorithm. On the other hand, this algorithm needed more time than DBSCAN to do the same work. It should be noticed, that the chosen data set wasn't very big and this difference in time could be different if algorithms would try to deal with bigger data sets. One could come with a conclustion, that for a bigger data set DENCLUE algorithm will be much more slower than DBSCAN, but it's not completely true. Time for DBSCAN algorithm in such situation will rise quickly. DENCLUE algorithm as dividing data points to cubes, take for big calculations only highly populated cubes and other elements that were provided to faster

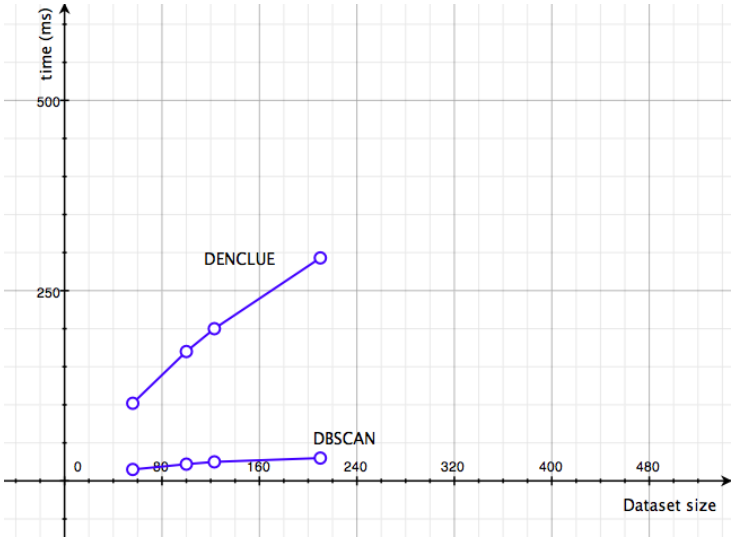

Figure 10: Comparison of DENCLUE and DBSCAN

the algorithm, will optimize it. On the tested data set DENCLUE algorithm dividing cubes for populated and highly populated wasn't a cause of some big decrease of considered cubes. There is also possible optimization for a DBSCAN algorithm, using triangle inequality (TI) property. [4] It could be a vast improvement, because now to calculate a neighbourhood all the points distances are calculated and checked. With triangle inequality most of the points would be skipped. However, even without this optimization DBSCAN was faster than DENCLUE, but less accurate, according to our quality measure.

7 Attribution

Anonymous Coauthor implemented DENCLUE algorithm, structures essential for it, tested memory and time usage and prepared majority of this documenation. Adam Stelmaszczyk implemented `scorer`, `visualizer`, `main` packages, DENCLUE algorithm, structures used in it and wrote part of this documenation.

References

[1] UC Irvine Machine Learning Repository. `http://archive.ics.uci.edu/ml/datasets/seeds`.

[2] Martin Ester, Hans-Peter Kriegel, Jorg Sander, and Xianowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, (KDD-96), 1996.

[3] Alexander Hinneburg and Daniel A. Keimm. An efficient approach to clustering in large multimedia databases with noise. 1998.

[4] Marzena Kryszkiewicz and Piotr Lasek. Ti-dbscan: Clustering with db-scan by means of the triangle inequality. 2010.