Implementation and Evaluation of DBSCAN Algorithm

Pedro Lindeza, Marko Raguž, Rafael Belchior

June 15, 2018

Data Ming Project at Warsaw University of Technology 2017/2018

1 Task Description

This task consists of the implementation, in Java, of the clustering algorithm DBSCAN. Furthermore, 3 internal and 3 external evaluation methods are implemented in C++.

1.1 DBSCAN Algorithm

Density-based spatial clustering of applications with noise (DBSCAN) is a data clustering algorithm proposed by Martin Ester, Hans-Peter Kriegel, Jörg Sander and Xiaowei Xu in 1996. It is a density-based clustering algorithm: given a set of points in some space, it groups together points that are closely packed together (points with many nearby neighbors), marking as outliers points that lie alone in low-density regions (whose nearest neighbors are too far away). DBSCAN is one of the most common clustering algorithms and also most cited in scientific literature.[2]

2 Assumptions

Consider a set of points in some space to be clustered. For the purpose of DBSCAN clustering, the points are classified as core points, (density-)reachable points and outliers, as follows:

- A point p is a core point if at least minPts points are within distance e (e is the maximum radius of the neighborhood from p) of it (including p). Those points are said to be directly reachable from p.
- A point q is directly reachable from p if point q is within distance e from point p and p must be a core point.
- A point q is reachable from p if there is a path p1, ..., pn with p1 = p and pn = q, where each pi+1 is directly reachable from pi (all the points on the path must be core points, with the possible exception of q).
- All points not reachable from any other point are outliers. (noise)

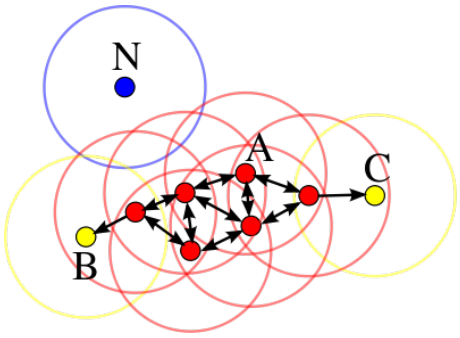


Figure 1: minPts = 4, red points are core points and N is a noise point.

3 Input

The input of my program must be a file in *.txt extension, belonging to the root directory "/datasets".

Each line of the input file correspond to a point, and its structure **must** be:

$$label\_id1,c1,...,cn$$
$$label\_idN,c1,...,cn$$

- N : total number of points and the same number of lines in the input file.
- $label\_id$: is identifier of the point. Usually a name/string.
- $c1$ .. cn : are the n values that define the point, normally integer or real.

3.1 Example of Input

```
A,0,0
B,0,1
C,0,2
...
K,12,5
L,12,6
O,12,7
```

4 Output

The output of the program will start by the number of clusters calculated following information about the input regarding the number of attributes (n in previous section) and the number of points (N also in the previous section). Then a list of points follows, in the form:

$$cluster\_number\ label\_id1\ c1\ ...\ cn$$

Where the cluster number means the reference of the cluster that point belongs in. E.g. if 2 points share the same cluster number, they both belong to the same cluster.

Finally, the output contains a list with all noise points

4.1 Example of Output

```
Clusters 2
Attributes 2
Points 18

1 A 0.0 0.0
1 B 0.0 1.0
1 C 0.0 2.0

2 K 12.0 5.0
2 L 12.0 6.0
2 O 12.0 7.0

Noise points
None
```

5 Design and Implementation

5.1 Design

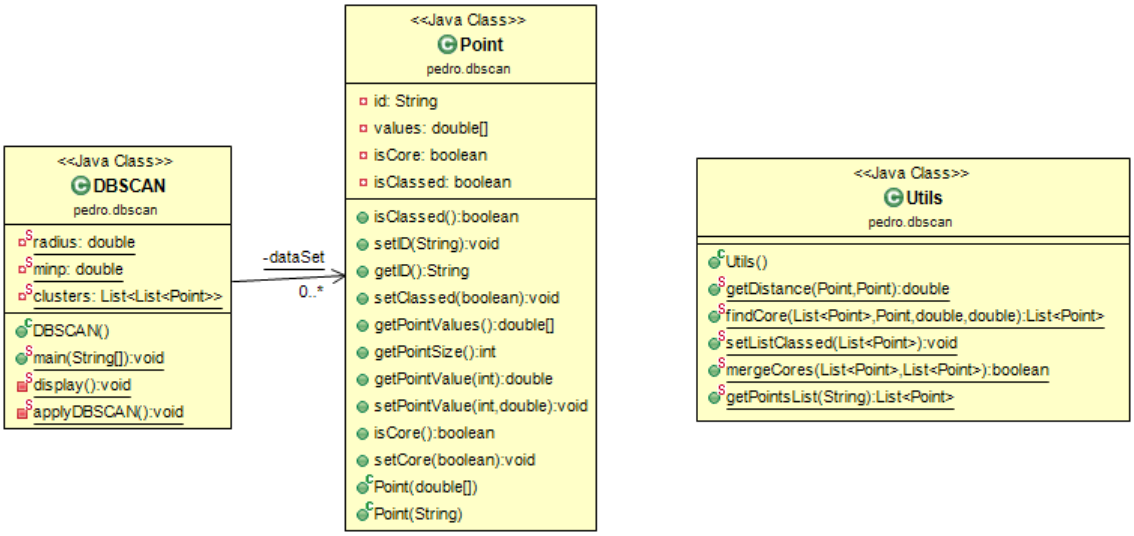


Figure 2: Class Diagram

5.2 Implementation Issues

The implementation of the algorithm is quite simple.

- Checks for every point p if they are core points, if so, a new list is generated with all points of that core circle.
- If a point is not at any of those lists is considered noise.
- Finally, the script joins the list that have points in common and defines it as a cluster.

6 User's Guide

The DBSCAN program can be found at the github repository in the link
https://github.com/pedrolindeza/dbscan

- Clone the project to your computer
- Choose the data set from */datasets/* folder. (DS from now)
- On the project's main directory run:

  ```
  >> java -jar DBSCAN.jar DS.txt radius minp > out.txt
  ```

- The output will be in the main directory as "out.txt".

The Evaluation program can be found at the github repository at the link:
https://github.com/RafaelAPB/ML_EvaluationMeasures

- Clone the project to your computer
- Choose the data sets you wish to use as input and place them on */input/* folder.
- On the project's main directory run:

  ```
  >> cmake cmake-build-release
  ```

- Build files have been written to:

  ```
  cmake-build-release
  ```

- Go to the directory where the files have been generated and run

  ```
  make
  ```

7 Execution Example

We will test the execution on a data set (test.txt) with radius = 1 and minp = 3.

```
>> java -jar DBSCAN.jar test.txt 1 3 > out.txt
```

7.1 Input

```
A,0,0
B,0,1
C,0,2
D,0,3
E,0,4
F,0,5
G,12,1
H,12,2
I,12,3
J,12,4
K,12,5
L,12,6
M,0,6
N,0,7
O,12,7
P,0,8
Q,0,9
R,1,1
```

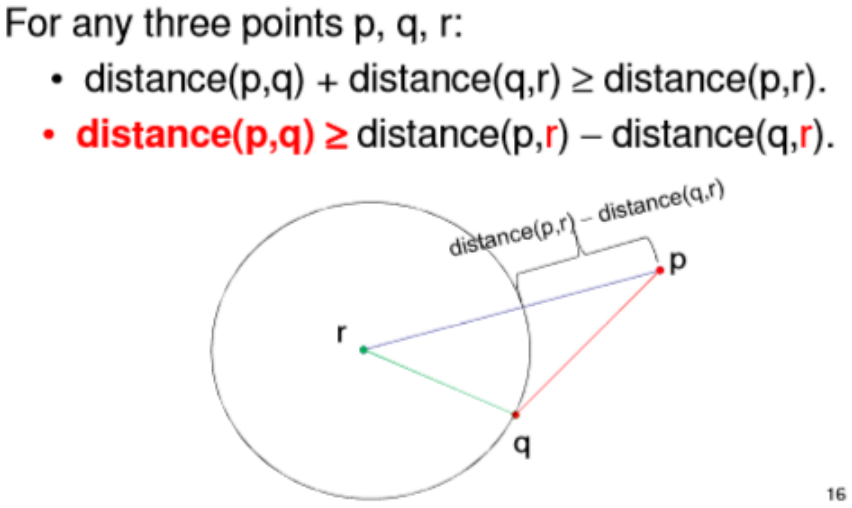
7.2 Output

```
Clusters 2
Attributes 2
Points 18

1 A 0.0 0.0
1 B 0.0 1.0
1 C 0.0 2.0
1 R 1.0 1.0
1 D 0.0 3.0
1 E 0.0 4.0
1 F 0.0 5.0
1 M 0.0 6.0
1 N 0.0 7.0
1 P 0.0 8.0
1 Q 0.0 9.0

2 G 12.0 1.0
2 H 12.0 2.0
2 I 12.0 3.0
2 J 12.0 4.0
2 K 12.0 5.0
2 L 12.0 6.0
2 O 12.0 7.0

Noise points
None
```

8 TI-DBSCAN

We will optimize the DBSCAN algorithm by reducing the number of points for which the distance has to be calculated. TI-DBSCAN algorithm is a DBSCAN algorithm with Efficient Calculation of Eps-Neighboorhoods using triangle inequality property.

8.1 Triangle inequality property

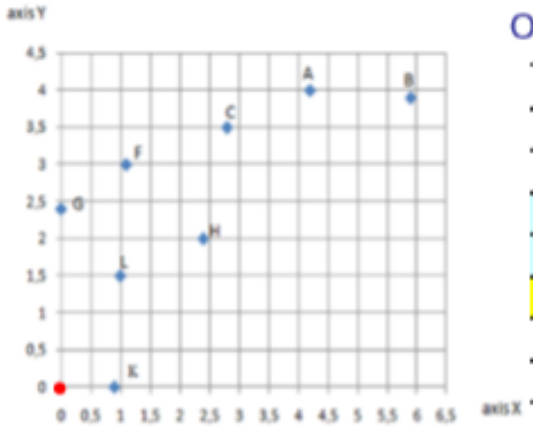


If the difference between 2 point's distances to a reference point is bigger than epsilon then we are sure that the real distance between these 2 points is equal or higher therefore it can newer be inside epsilon radius of a point.

8.2 Applying TI to DBSCAN

Firstly we calculate distance from a reference point, in our case the origin of the N-dimensional coordinate system (ie. p0=(0,0,..,0)), to all the points in the dataset. Then we sort them incrementally and save that list. When calculating the distance from a specific point we no longer calculate it between all points, but only for points for which the pessimistic distance (ie. difference between 2 point's reference distances) is not higher than radius.

We use sorted list in order to stop calculating pessimistic distance after we hit a point for which the pessimistic distance is higher than radius as show in the example below:



Ordered set of points D; Eps = 0.2

q∈D	X	Y	distance(q,r)	
K	0,9	0,0	0,9	
L	1,0	1,5	1,8	
G	0,0	2,4	2,4	$\notin N_{Eps}(F)$
H	2,4	2,0	3,1	
F	1,1	3,0	3,2	
C	2,8	3,5	4,5	
A	4,2	4,0	5,8	
B	5,9	3,9	7,1	

8.3 Implementation

Implementation issues are the same as previously written for DBSCAN with small modifications. Original DBSCAN code is modified as follows:

-Point.class has new field "distanceToOrigin" which is used to sort and compare points by distance to the origin point. It is calculated only once per point.

-Utils.findCore method has been modified to calculate real distance only to the points for which the pessimistic distance is not higher than radius and to stop calculating pessimistic distance once that difference is preceded, instead of calculating it to every point.

Full code, datasets and executable files can be found on : https://github.com/Smrtolet/TIDBSCAN

It is used in the same way as shown in DBSCAN user guide

8.4 Comments

The output of TI-DBSCAN is the same as the DBSCAN for every dataset (the ordering of the elements might be different, but the sets in the clusters are the same). What has changed is the time of execution, as it is much faster with TI implementation, specially for larger datasets as it reduces the amount of operations exponentially.

9 Clustering Evaluation

Typical objective functions in clustering formalize the goal of attaining high intra-cluster similarity (documents within a cluster are similar) and low inter-cluster similarity (documents from different clusters are dissimilar). Clustering evaluation aims to provide tools and methods in order someone to know if a clustering is good or not.

9.1 Internal Evaluation

Internal evaluation is the set of procedures that evaluates the clustering result based on the data that was clustered. Internal evaluation measures suffer from the problem that they represent functions that themselves can be seen as a clustering objective. One downfall is that by using

internal measures for evaluation, we are comparing the similarity of the optimization problems, and not necessarily how useful the clustering is.

We implemented three internal evaluation measures:

9.1.1 Davies-Boulding Index

$$DB = \frac{1}{n}\sum_{i=1}^{n}\max_{j\neq i}\left(\frac{\sigma_i + \sigma_j}{d(c_i, c_j)}\right)$$

Figure 3: Davies-Boulding formula

n - number of clusters
Cx - centroid of cluster x
Sigma - average distance of all elements in cluster x to centroid Cx
d(ci,cj) is the distance between centroids ci and cj

Since algorithms that produce clusters with low intra-cluster distances (high intra-cluster similarity) and high inter-cluster distances (low inter-cluster similarity) will have a low Davies–Bouldin index, the clustering algorithm that produces a collection of clusters with the smallest Davies–Bouldin index is considered the best algorithm based on this criterion.

9.1.2 Silhouette Index

$$Silhouette(x) = \frac{b(x) - a(x)}{\max(b(x), a(x))}$$

Figure 4:
Silhouette Index formula

a(x) – the average distance between x and other objects in a group including x
b(x) – the minimum average distance between x and the nearest group.
This index is often used to determine the optimal number of clusters. It has a value from the range <-1, 1>, where 1 means that the object is assigned to the best possible group, 0 - the object is located between two groups, and -1 - wrong assignment of the object. Furthermore: n - number

$$GSilhouette = \frac{1}{N}\sum_{i=1}^{N} Silhouette(x_i)$$

Figure 5: Dataset Silhouette Index formula

of points in the dataset

9.1.3 Calinski-Harabasz Index

`CH(k)=[B(k)/W(k)]×[(n−k)/(k−1)]}`

where n = number of data points
k = number of clusters
W(k) = within cluster variation
B(k) = between cluster variation

CH criterion is based on ANOVA ideology. Hence, it implies that the clustered objects lie in Euclidean space of scale (not ordinal or binary or nominal) variables. CH criterion is most suitable in case when clusters are more or less spherical and compact in their middle (such as normally distributed, for instance). Other conditions being equal, CH tends to prefer cluster solutions with clusters consisting of roughly the same number of objects.

9.2 External Evaluation

In external evaluation, clustering results are evaluated based on data that was not used for clustering, such as known class labels and external benchmarks. Such benchmarks consist of a set of pre-classified items, and these sets are often created by (expert) humans. In this project, we used the class labels as a comparison to how the clustering algorithm performed.

9.2.1 Purity

$$\frac{1}{N}\sum_{m\in M}\max_{d\in D}\left|m\cap d\right|$$

Figure 6: Dataset Purity formula

Purity is a measure of the extent to which clusters contain a single class. M - set of clusters
D - some set of classes D
N - data points
Note that this measure doesn't penalize having many clusters. So for example, a purity score of 1 is possible by putting each data point in its own cluster. Also purity doesn't work well for imbalanced data: if a size 1000 dataset consists of two classes, one class contains 999 points and the other has only 1 point. No matter how bad a clustering algorithm performs, it will always give a very high purity value.

9.2.2 F-Measure

$$F_\beta = \frac{(\beta^2+1)\cdot P\cdot R}{\beta^2\cdot P+R}$$

Figure 7: F-Measure formula

The F-measure can be used to balance the contribution of false negatives by weighting recall through a parameter. Furthermore, P (precision) and R (recall) are given by:

$$P = \frac{TP}{TP+FP}$$
$$R = \frac{TP}{TP+FN}$$

Figure 8: Precision and Recall formulas

9.2.3 Rand Index

here TP is the number of true positives, TN is the number of true negatives, FP is the number of false positives, and FN is the number of false negatives.

The Rand index computes how similar the clusters (returned by the clustering algorithm) are to the benchmark classifications. One can also view the Rand index as a measure of the percentage of correct decisions made by the algorithm.

$$RI = \frac{TP + TN}{TP + FP + FN + TN}$$

Figure 9: Rand-Measure formula

10 Result Evaluation

To evaluated the results on several different datasets, I will do a comparison between time and joint number of points and attributes.

10.1 Iris Dataset

- https://archive.ics.uci.edu/ml/datasets/iris
- radius = 0.5 and minp = 10

10.1.1 Output

```
Clusters 2
Attributes 4
Points 150

Time for DBSCAN:
real    0m0.625s
user    0m0.172s
sys     0m0.141s

Time for TI-DBSCAN:
real    0m0.228s
user    0m0.094s
sys     0m0.094s
```

10.2 Absenteeism Dataset

- http://archive.ics.uci.edu/ml/datasets/Absenteeism+at+work
- radius = 10 and minp = 10

10.2.1 Output

```
Clusters 26
Attributes 20
Points 740

Time for DBSCAN:
real    0m0.365s
user    0m0.500s
sys     0m0.219s

Time for TI-DBSCAN:
real    0m0.369s
user    0m0.625s
sys     0m0.109s
```

10.3 Cardio Dataset

- http://archive.ics.uci.edu/ml/datasets/Cardiotocography
- radius = 10 and minp = 10

10.3.1 Output

```
Clusters 6
Attributes 21
Points 2129

Time for DBSCAN:
real    0m2.632s
user    0m2.766s
sys     0m0.391s

Time for TI-DBSCAN:
real    0m0.719s
user    0m1.063s
sys     0m0.344s
```

References

[1] mkr_Clustering45_notes.pdf

[2] https://en.wikipedia.org/wiki/DBSCAN

[3] https://en.wikipedia.org/wiki/Cluster_analysis

[4] https://stats.stackexchange.com/questions/86645/variance-within-each-cluster

[5] https://stats.stackexchange.com/questions/97429/intuition-behind-the-calinski-harabasz-index?rq=1

[6] https://link.springer.com/article/10.1007/s40595-016-0086-9

[7] https://stats.stackexchange.com/questions/21807/evaluation-measure-of-clustering-without-having-truth-labels