PROJECT REPORT

The Implementation of DBSCAN and DBSCANRN with Tanimoto Similarity

"Data Mining"

Rauzan Sumara

Introduction

Clustering is one of the most important tasks of both artificial intelligence and data mining. That is a way to group a set of data points in a way that similar data points are grouped together. Therefore, clustering algorithms look for similarities or dissimilarities among data points. Clustering is an unsupervised learning method so there is no label associated with data points. The algorithm tries to find the underlying structure of the data. There are different approaches and algorithms to perform clustering tasks which is of them is algorithms based on density clustering. Based on discussed in the lecture, we will implement algorithms of Density-Based Spatial Clustering of Applications with Noise (DBSCAN) and Density-based spatial clustering using reverse nearest neighbors (DBSCRN).

However, noted during project consultations, there are some inconsistencies in the paper presenting the pseudo-code of the DBSCRN algorithm [1]. For this reason, we will implement some modification of DBSCRN (which will be called DBSCANRN) instead of DBSCRN. In DBSCANRN, we will keep core point and non-core points definitions as given in the pseudo-code of DBSCRN. Namely, a point is considered as a DBSCRN (and by this DBSCANRN) non-core point if the number of their reverse k-nearest neighbors is less than k (lines 2-3 in the DBSCRN pseudo-code). Otherwise, it is considered as a core point (line 5 in the DBSCRN pseudo-code). We implement DBSCANRN in a way similar to NBC, except that instead of adding k+-nearest neighbors of a core point to a cluster, its reverse k-nearest neighbors will be added to the cluster.

Similarity Measures

The Tanimoto similarity is commonly used in bio-informatics or biology, and information retrieval to identify neighborhoods of sufficiently similar items or the *k* most similar things represented by real-valued vectors. The triangle inequality condition is frequently used to rapidly discover vectors that may belong to the targeted neighborhood of a given vector for metrics such as the Euclidean distance. However, the Tanimoto similarity and Tanimoto dissimilarity do not satisfy the triangle inequality property for real-valued vectors [2].

Considering this, another way to find a neighborhood with regard to Tanimoto similarity among real-valued vectors has been explained in [2]. We will adopt it, using bounds on vector lengths to calculate Tanimoto similarity ε-neighborhoods.

Property. Let u and v be non-zero vectors, $\alpha = \frac{1}{2}\left(\left(1+\frac{1}{\varepsilon}\right)+\sqrt{\left(1+\frac{1}{\varepsilon}\right)^2-4}\right)$, and $\varepsilon \in (0,1]$. If $T(u,v) \geq \varepsilon$, then $|v| \in \left[\frac{1}{\alpha}|u|, \alpha|u|\right]$. Their usefulness with an example illustrated in [2].

Datasets

We have 5 datasets:

1. the toy dataset used in the class to present the execution of the NBC clustering algorithm (slide 51)
2. dim512 from *http://cs.joensuu.fi/sipu/datasets/*
3. complex9 from https://github.com/deric/clustering-benchmark/tree/master/src/main/resources/datasets/artificial
4. luto-t7-10k from *https://github.com/deric/clustering-benchmark/tree/master/src/main/resources/datasets/artificial*
5. letter from *https://github.com/deric/clustering-benchmark/tree/master/src/main/resources/datasets/real-world*

Table 1. Used Datasets

No.	Dataset	Size	Dimension	Cardinality
0	lacture	12	2	<NA>
1	dim512	1024	512	16
2	complex9	3031	2	9
3	cluto t7 10k	10000	2	10
4	letter	20000	16	26

measure labeled dataset based on size (num of observation), dimension (num of features), and cardinality (num of unique class). We also visualize the datasets in which datasets that have more than 2 dimensions will be transformed using principal component analysis (PCA) into 2 principal components, so that we can graph into 2-dimensional space. The graph of the datasets as follow:

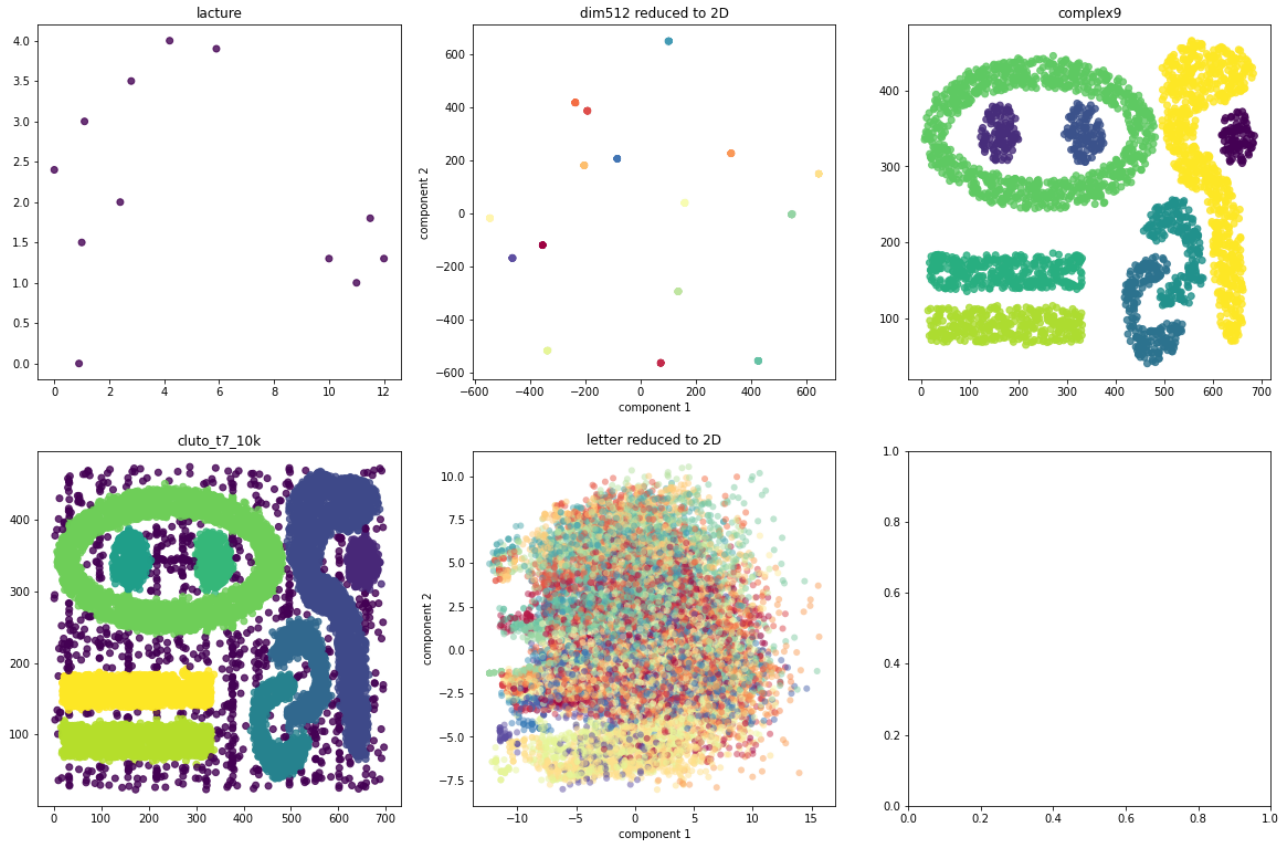


Figure 1. datasets

User's Manual

To run the program, open the Jupyter Notebook file named “DAMI_Project.ipynb”, from this link: *https://github.com/rauzansumara/dbscan-and-dbscanrn-tanimoto-similarity*. This interactive Jupyter Notebook saved the results from previous run by us. Users can modify and run each cell from top to bottom to view the result of the code. Note that this action will replace previous output plots. The environment and library requirements as follow:

Requirements
```

```
python==3.8.5
numpy==1.21.4
pandas==1.3.5
sklearn==0.24.2
matplotlib==3.3.2
```
```

Experimental Results

We conducted our experiment using python programming language. Once you open the project folder, you will be presented with the following directory structure:

```
$ tree-project-folder
.
├── datasets
│   ├── cluto-t7-10k.arff
│   ├── complex9.arff
│   ├── dim512.pa
│   ├── dim512.ts
│   ├── dim512.txt
│   ├── lecture.csv
│   └── letter.arff
├── images
│   ├── datasets.png
│   ├── dbscan_cluto_t7_10k.png
│   ├── dbscan_complex9.png
│   ├── dbscan_dim512.png
│   ├── dbscan_lacture.png
│   ├── dbscan_letter.png
│   ├── dbscanrn_cluto_t7_10k.png
│   ├── dbscanrn_complex9.png
│   ├── dbscanrn_dim512.png
│   ├── ..............
│   └── optdbscanrn_letter.png
├── docs
│   └── documentation [report]
├── save_file
│   ├── DEBUG_dbscanrn_cluto_t7_10k_D2_R10000_k10.csv
│   ├── DEBUG_dbscanrn_complex9_D2_R3031_k9.csv
│   ├── DEBUG_dbscanrn_dim512_D512_R1024_k4.csv
│   ├── DEBUG_dbscanrn_lacture_D2_R12_k2.csv
│   ├── DEBUG_dbscanrn_letter_D16_R20000_k17.csv
│   ├── DEBUG_dbscanrn_cluto_t7_10k_D2_R10000_k10.csv
│   ├── ..............
│   ├── STAT_dbscan.csv
│   ├── STAT_dbscanrn.csv
│   └── STAT_Opt_dbscanrn.csv
├── DAMI_project.ipynb
```

The *datasets* directory contains the data used. The *images* directory contains visualization graph of each experiment. We also have *docs* directory providing report file, and the most significant things are the *save_file* directory storing the DEBUG, OUTPUT, and STAT files. The results returned by a clustering algorithm for a given dataset and parameter values are saved in in 3 files:

DEBUG files - a file that contains important information for each point in the dataset on a separate line. Those are,

point id, minEps, maxEps, |Rk+NN|, identifiers if k+NN, identifiers of Rk+NN

where:

- maxEps - Eps value calculated based on first k candidates for k+NN of the point
- minEps - the minimal value of Eps within which real k+NN of the point was found

OUT files - an output file that contains the following information on a separate line for each point in the dataset:

point id, x,y,...., TypeId, ClId

where:

- point id - the position of the point in the dataset (before possible sorting),
- x, y, ... - dimension values
- TypeId - either a core point (denoted by 1), or a border point (denoted by 0), or a noise point (denoted by -1)
- ClId - a cluster identifier (which is a natural number) or -1 in the case of noise points.

STAT files - a file with the following statistics:

Dataset, Size, Dimension, Cardinality, ClId, Core_Point, Border_Point, Noise_Point, Reading_Time, kNN/rkNN_Time, Clustering_Time, Save_Time, Total_Time, Purity_Index, Rand_Index, DaviesBouldin_Index, Silhouette_Score

Implementation of DBSCAN

In this section, we did some set of experiments using DBSCAN algorithm as follow:

- lacture dataset - DBSCAN(lacture, eps=0.80, MinPts=4)
- dim512 dataset - DBSCAN(dim512, eps=0.80, MinPts=5)
- complex9 dataset - DBSCAN(complex9, eps=0.9995, MinPts=5)
- cluto_t7_10k dataset - DBSCAN(cluto_t7_10k, eps=0.99975, MinPts=6)
- letter dataset - DBSCAN(letter, eps=0.99975, MinPts=7)

Once running the code, we obtained STAT file with following results:

Table 2. STAT file of the DBSCAN experiment

Properties	Dataset				
	lecture	dim512	complex9	cluto_t7_10k	letter
Size	12	1024	3031	10000	20000
Dimension	2	512	2	2	16
Cardinality	<NA>	16	9	10	26
ClId	2	16	49	77	16
Core_Point	5	1024	2160	7065	156
Border_Point	3	0	306	793	0
Noise_Point	4	0	565	2142	19844
Reading_Time (seconds)	0.000992	0.388037	0.021002	0.074027	0.292731
FindingNN Clustering__Time (seconds)	0.001001	18.52627	53.54661	638.7607	3226.501
Save_Time (seconds)	0.002001	0.075028	0.011002	0.032009	0.200052
Total_Time (seconds)	0.003994	18.98933	53.57861	638.8667	3226.994
Purity_Index	1.000000	1.000000	0.892775	0.8256	0.04845
Rand_Index	0.272727	1.000000	0.900759	0.902848	0.052849
DaviesBouldin_Index	1.905921	0.020462	1.258448	2.696523	1.056649
Silhouette_Score	0.337355	0.98669	-0.09012	-0.30839	-0.30995

As we can see from table above, we found 2 cluster in lacture dataset. From 12 points, we got 5 core points, 3 border points, and 4 noise points. The total of running time is 0.003994 seconds. We obtained also the quality measurement such as Purity_Index, Rand_Index, DaviesBouldin_Index, and Silhouette_Score. We provided two graphs of each dataset to see the different between original clusters and predicted clusters. After running the code following results were obtained:

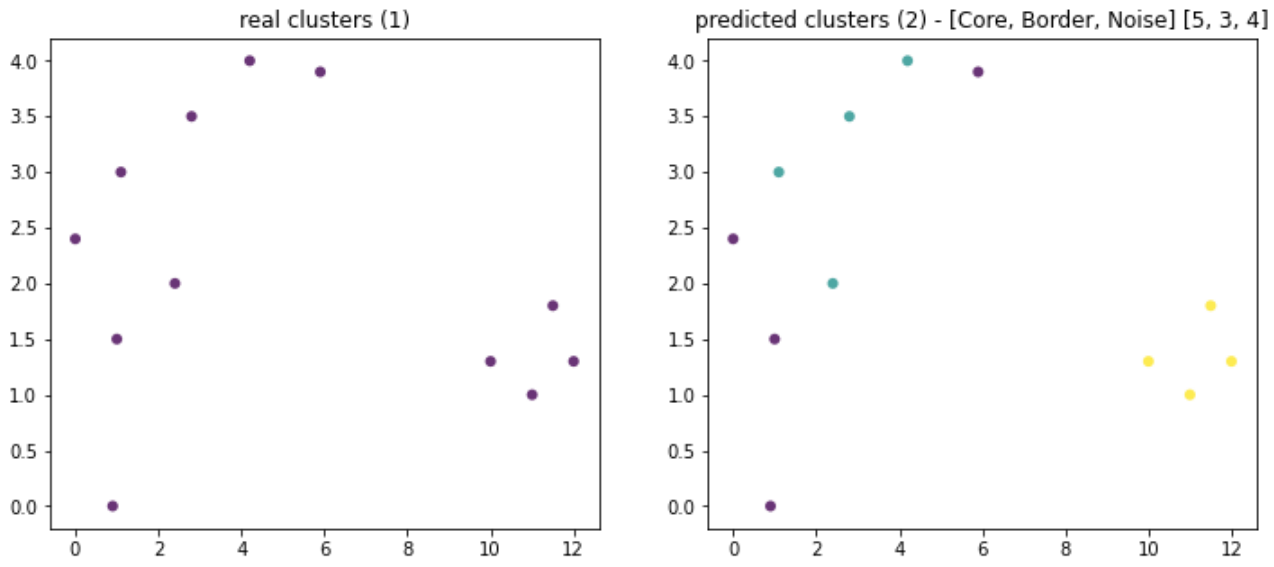


Figure 2. Visualization of Lecture Dataset (Left: Original and Right: Predicted Cluster)

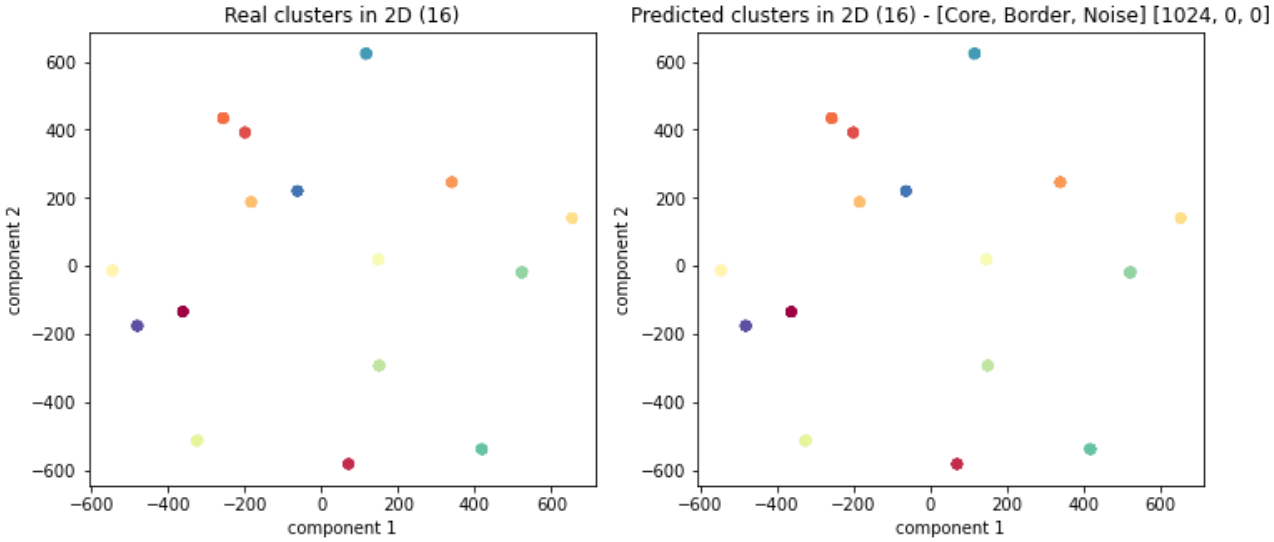


Figure 3. Visualization of dim512 Dataset (Left: Original and Right: Predicted Cluster)

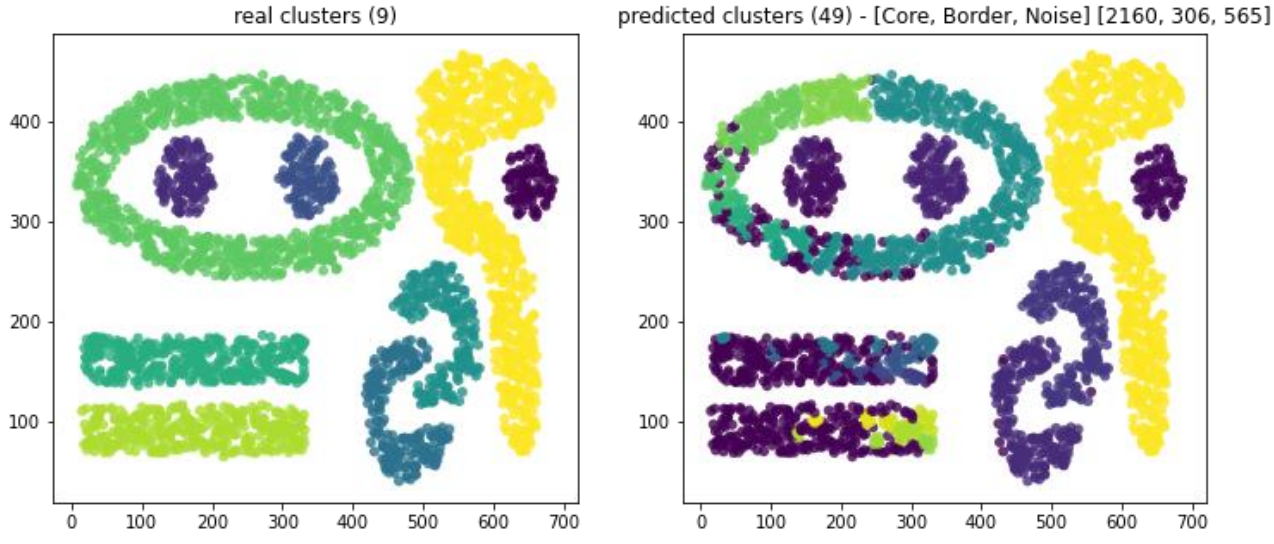


Figure 4. Visualization of complex9 Dataset (Left: Original and Right: Predicted Cluster)

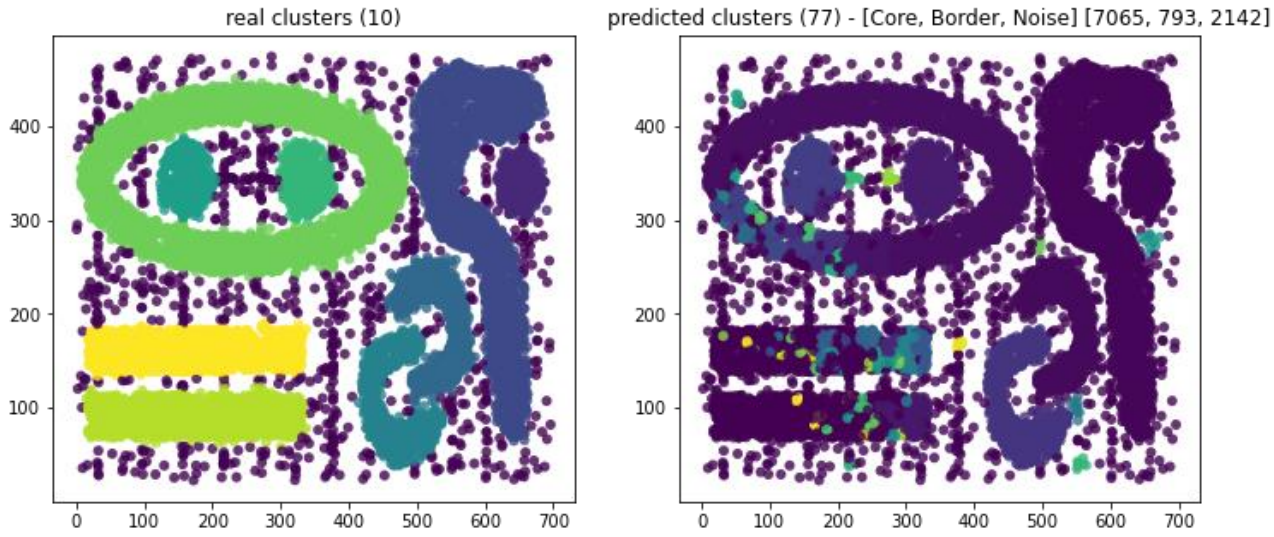


Figure 5. Visualization of cluto_t7_10k Dataset (Left: Original and Right: Predicted Cluster)

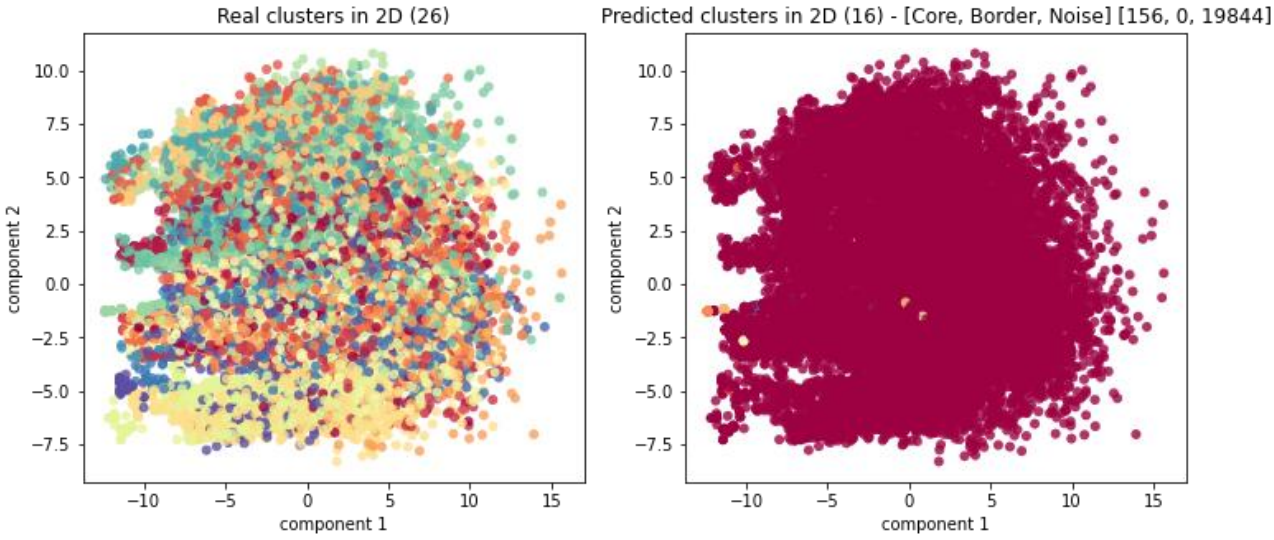


Figure 6. Visualization of letter Dataset (Left: Original and Right: Predicted Cluster)

Implementation of DBSCANRN

In Implementation of this algorithm both *non-optimized* and *optimized* version, we did some set of experiments as follow:

- lecture dataset - DBSCANRN(lecture, k=2)
- dim512 dataset - DBSCANRN(dim512, k=4)
- complex9 dataset - DBSCANRN(complex9, k=9)
- cluto_t7_10k dataset - DBSCANRN(cluto_t7_10k, k=10)
- letter dataset - DBSCANRN(letter, k=17)

Once running the code, we obtained STAT file with following results:

Table 3. STAT file of the DBSCANRN experiment with Non-Optimized Version

Properties	Dataset				
	lecture	dim512	complex9	cluto_t7_10k	letter
Size	12	1024	3031	10000	20000
Dimension	2	512	2	2	16
Cardinality	<NA>	16	9	10	26
ClId	2	27	38	61	57
Core_Point	8	209	1753	5891	9896
Border_Point	4	815	1277	4077	10098
Noise_Point	0	0	1	32	6
Reading_Time (seconds)	0.0010	0.3875	0.0210	0.0740	0.2937
kNN/rkNN_Time (seconds)	0.0040	18.7168	54.1189	604.7930	2469.4621
Clustering_Time (seconds)	0.0000	0.0010	0.0041	0.0160	0.0450
Save_Time (seconds)	0.0120	0.0855	0.0566	0.2146	0.8168
Total_Time (seconds)	0.0170	19.1908	54.2007	605.0976	2470.6177
Purity_Index	1.0000	1.0000	1.0000	0.9181	0.2775
Rand_Index	0.5152	0.9935	0.8582	0.8704	0.4423
DaviesBouldin_Index	0.3076	2.1025	0.8153	1.5622	1.2230
Silhouette_Score	0.7138	0.7086	0.2099	-0.1186	-0.1576

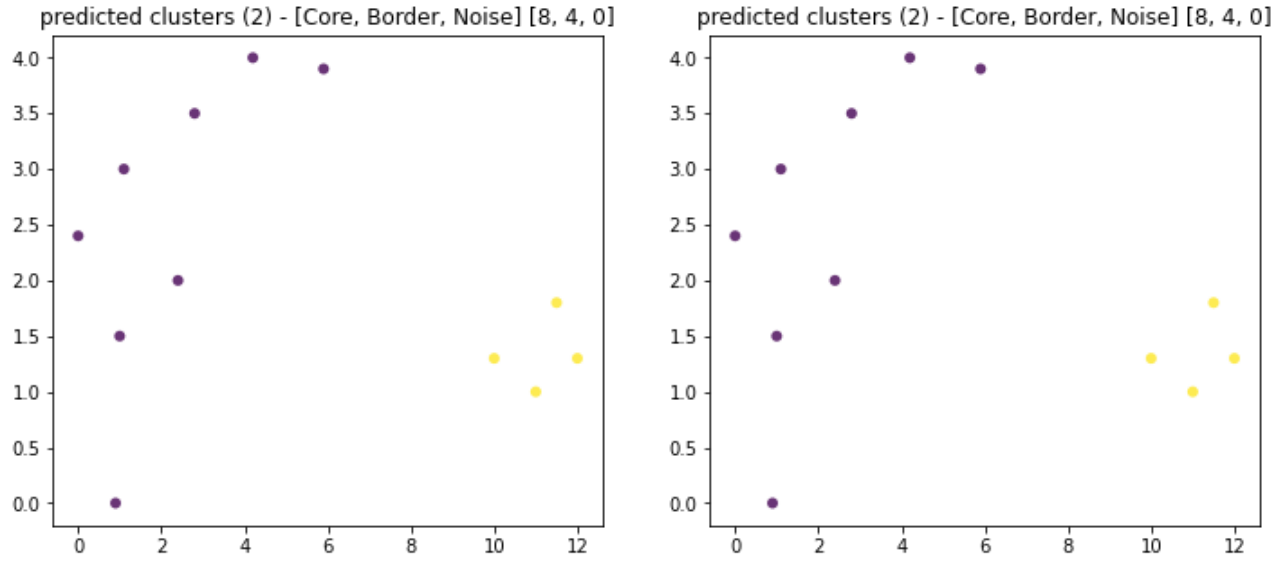
Table 4. STAT file of the DBSCANRN experiment with Optimized Version

Properties	Datasets				
	lecture	dim512	complex9	cluto_t7_10k	letter
Size	12	1024	3031	10000	20000
Dimension	2	512	2	2	16
Cardinality	<NA>	16	9	10	26
ClId	2	27	38	61	2
Core_Point	8	209	1753	5891	11614
Border_Point	4	815	1277	4077	8386
Noise_Point	0	0	1	32	0
Reading_Time (seconds)	0.0020	0.3875	0.0210	0.0730	0.2937
kNN/rkNN_Time (seconds)	0.0010	5.6236	4.1274	26.6427	1673.9906
Clustering_Time (seconds)	0.0000	0.0010	0.0050	0.0160	0.0500
Save_Time (seconds)	0.0030	0.0850	0.0648	0.2382	0.8764
Total_Time (seconds)	0.0060	6.0972	4.2182	26.9700	1675.2107
Purity_Index	1.0000	1.0000	1.0000	0.9181	0.0426
Rand_Index	0.5152	0.9935	0.8582	0.8704	0.0420
DaviesBouldin_Index	0.3076	2.1025	0.8153	1.5622	0.6251
Silhouette_Score	0.7138	0.7086	0.2099	-0.1186	0.3198

As we can see from table 3 (non-optimized version) and table 4 (optimized version), we found the same number predicted clusters and type of point (core, border or noise) on 4

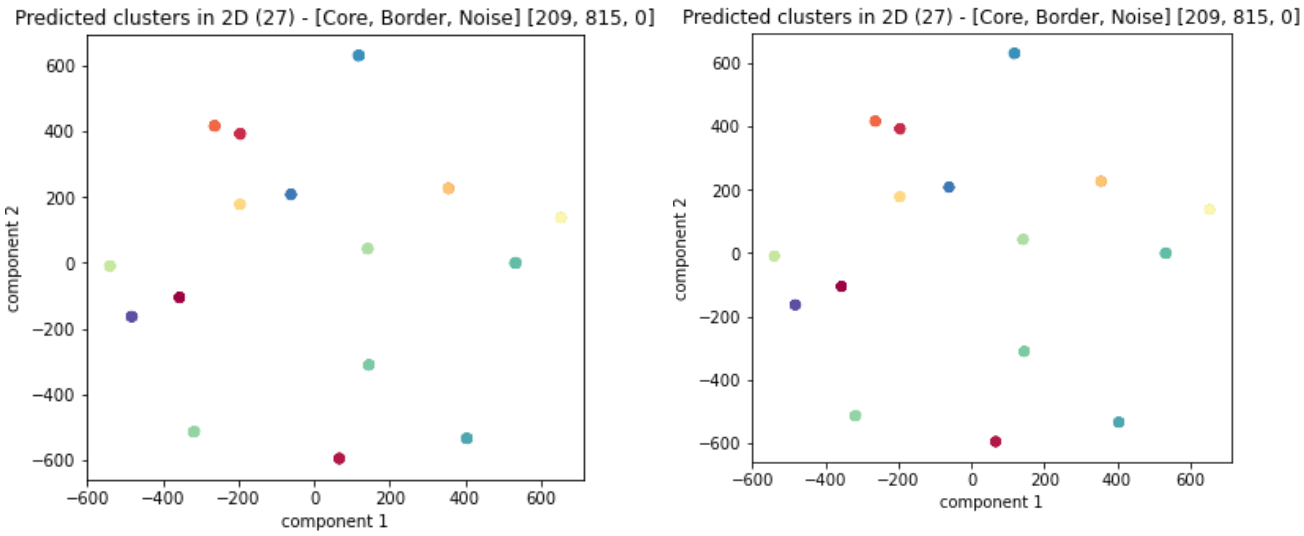
datasets, such as lecture, dim512, complex9, and cluto_t7_10k datasets between non-optimized and optimized version.

Unfortunately, we do not know why it showed different result on letter dataset between non-optimized version and optimized version. But apart from it, the comparison of DBSCANRN non-optimized and optimized version proven that the optimized version can reduce computation time up to 19 times faster than original non-optimized version. Meanwhile, the computation time does not really much different when we apply in small dataset like lecture dataset. We provided two result graphs of each version to see the different between non-optimized and optimized version. After running the code following results were obtained:



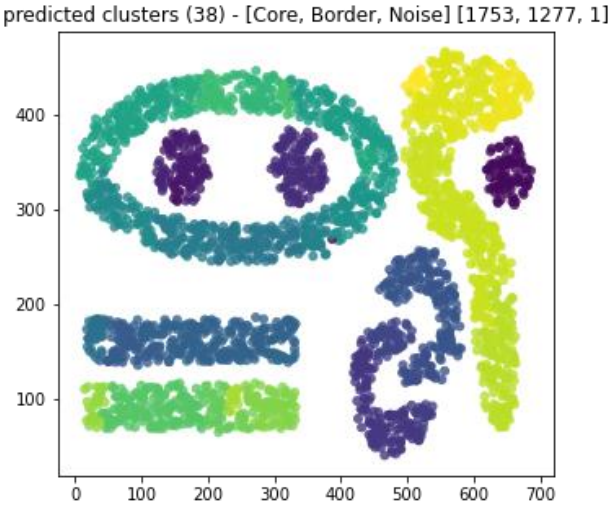
(a) Non-Optimized Version (b) Optimized Version

Figure 7. Visualization of Lecture Dataset

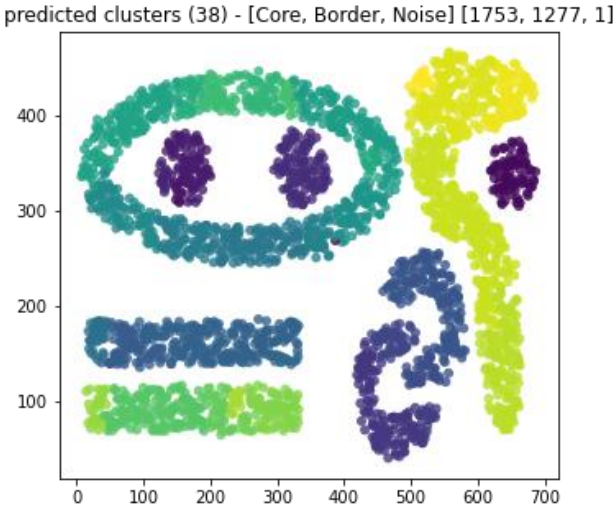


(a) Non-Optimized Version (b) Optimized Version

Figure 8. Visualization of dim512 Dataset

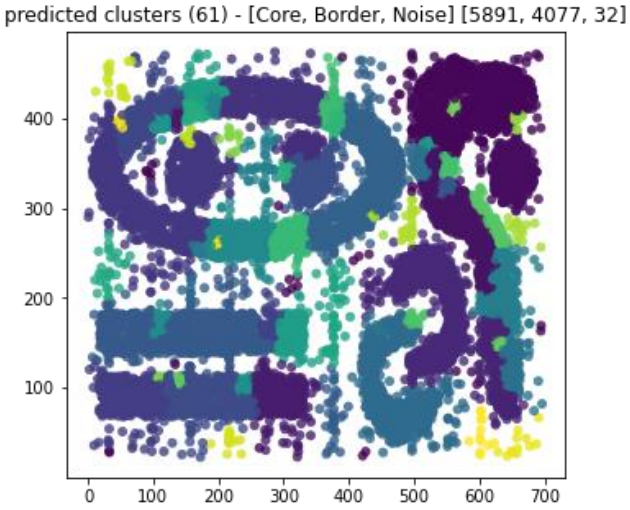


(a) Non-Optimized Version

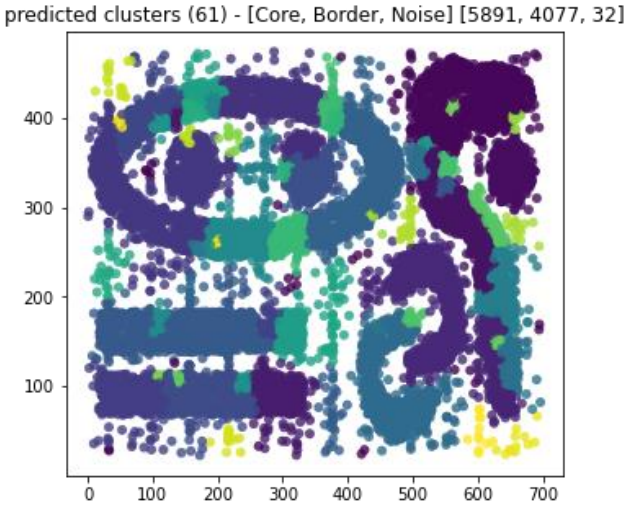


(b) Optimized Version

Figure 9. Visualization of complex9 Dataset

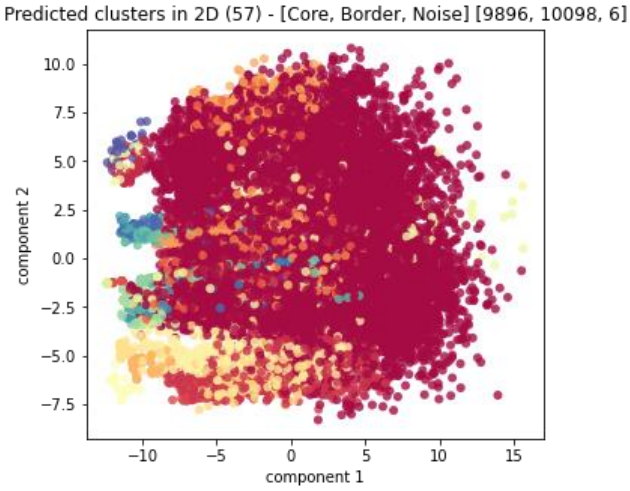


(a) Non-Optimized Version

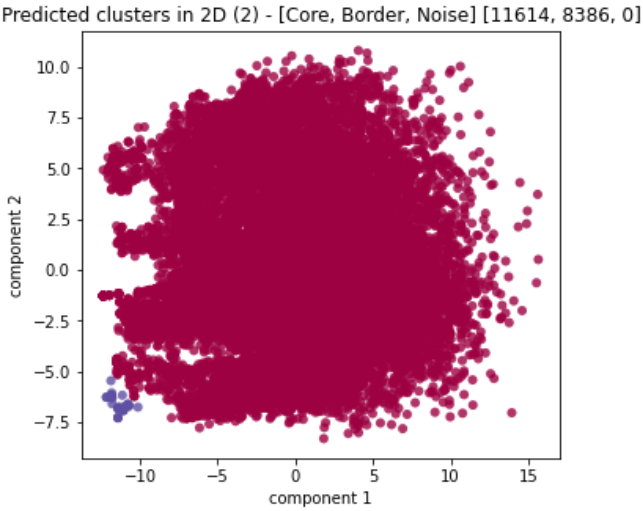


(b) Optimized Version

Figure 10. Visualization of cluto_t7_10k Dataset



(a) Non-Optimized Version



(b) Optimized Version

Figure 10. Visualization of letter Dataset

Conclusion

- in DBSCAN, computation time is affected by epsilon radius and size of dataset,
- in DBSCAN, larger *epsilon* creates more members of a cluster, while larger *minPts* usually creates a smaller number of clusters,
- when running program/code with the same parameters on DBSCANRN both versions (non-optimized and optimized version), our implementation of optimized one successfully reduced computation time up to 19 times faster compared to non-optimized version, and
- We obtained different number of found clusters only in letter dataset when comparing non-optimized and optimized version of DBSCANRN. It could be caused of code or characteristics of the data itself; we need to investigate in the future.

References

[1] Chowdhury, S. and Amorim R.C., (2019) An efficient density-based clustering algorithm using reverse nearest neighbour. The Computing Conference 2019: London

[2] Kryszkiewicz M. (2021) Determining Tanimoto Similarity Neighborhoods of Real Valued Vectors by Means of the Triangle Inequality and Bounds on Lengths.

[3] Kryszkiewicz M., Lasek P. (2010) A Neighborhood-Based Clustering by Means of the Triangle Inequality. In: Fyfe C., Tino P., Charles D., Garcia-Osorio C., Yin H. (eds) Intelligent Data Engineering and Automated Learning – IDEAL 2010. IDEAL 2010

[4] Kryszkiewicz M. (2014) Using Non-Zero Dimensions and Lengths of Vectors for the Tanimoto Similarity Search among Real Valued Vectors. Springer International Publishing Switzerland 2014