Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 221 (2023) 113–120

Procedia Computer Science

www.elsevier.com/locate/procedia

10th International Conference on Information Technology and Quantitative Management

A Faster DBSCAN Algorithm Based on Self-Adaptive Determination of Parameters

Bing Ma[a], Can Yang[a], Aihua Li[a,*], Yuxue Chi[a], Lihua Chen[b]

[a]*Central University of Finance and Economics, School of management science and Engineering, Beijing 102206, P. R. China*
[b]*Digital China Group Co.,Ltd, Beijing 100190, P. R. China*

Abstract

The DBSCAN algorithm is a well-known cluster method that is density-based and has the advantage of finding clusters of different shapes, but it also has certain shortcomings, one of which is that it cannot determine the two important parameters *Eps* （neighborhood of a point） and *Mints* （minimum number of points） by itself, and the other is that it takes a long time to traverse all points when dataset is large. In this paper, we propose an improved method which is named as K-DBSCAN to improve the running efficiency based on self-adaptive determination of parameters and this method changes the way of traversing and only deals with core points. Experiments show that it outperforms DBSCAN algorithms in terms of running time efficiency.

Keywords: Data-mining; Cluster; K-DBSCAN; Self-adaptive

1. Introduction

With the development of Internet technology, a huge amount of data has been generated, which contains much valuable information. Data mining is just the process of discovering interesting and useful knowledge from a large amount of data, and in recent years, the application scenarios of data mining have been expanded and it gradually becomes a hot direction for database research [1]. Cluster analysis is an important functional module in data mining, which is a kind of unsupervised learning. The task of cluster analysis is to make the data between the same cluster as similar as possible and the data between different clusters as dissimilar as possible.

The DBSCAN algorithm [2] is a classical density-based clustering method that can find arbitrarily shaped clusters in nonspherical data regions and effectively detect noise, but before the clustering starts, two important parameters

* Corresponding author. Tel.: 86-10-62288622.
E-mail address: aihuali@cufe.edu.cn

10.1016/j.procs.2023.07.017

need to be specified, Eps and Mints, which are global variables to which the algorithm is very sensitive. The accuracy of the algorithm depends on these two important parameters [3]. Based on its own characteristics, the DBSCAN algorithm has three shortcomings: one is that when the data density is uneven, different density regions use the same parameters, if the Eps is large, the low density regions may be combined with poor quality clustering, but that the Eps is small will identify a large number of noisy points; second, the parameter dependence is high but cannot determine parameter values by itself; and third, for large-scale datasets, time performance is low [4].We addresses the latter two shortcomings to improve, other researches are mainly as follows for latter two shortcomings: first, for the adaptive determination of parameters, OPTICS algorithm [5] by sorting the data points to produce an ordered list does not directly produce clustering results shielding the parameters Eps sensitive, but still cannot work well for uneven datasets[6]. The study in [7] introduced the concept of distance distribution but required setting Mints to a fixed value of 4 and determining the value of the corresponding Eps based on the distance of two objects in the sorted dataset, and the number of clusters still need to be set in advance. Li et al. [8] introduced the concept of density threshold to determine the optimal parameters through the convergence of density thresholding, and if using the data after noise removal then the clustering accuracy can be improved. Based on the above methods of parameter determination, the adaptive method is chosen, which has the advantage of being completely free of human intervention and reflecting the statistical properties of the dataset itself.

On the other hand, for time performance studies, Zhou et al. [9,10] reduced the number of data queries by data sampling in different ways, but only selecting data, both reduced the running time but was not effective on datasets with uneven density distribution. Li et al. [11] reduced the number of scanned points by selecting the next core point to be processed by selecting a point that is close to the current core node but not marked to improve the runtime efficiency, this improved algorithm may lose data information. Li [12] roposes a neighborhood-based scanning approach that uses inequalities to filter out unnecessary distance calculations, thereby reducing the algorithm's computational effort. Hanafi [13] introduces the concept of operational datasets to increase the speed of operations by not scanning remote points to outside operational datasets with smaller data sizes.

Based on the above research, we propose an improved algorithm that can adaptively determine parameters and improve efficiency by changing the way to scan points. Compared with the traditional DBSCAN algorithm, the applicability of the improved algorithm on different datasets is investigated and illustrated.

2. The DBSCAN algorithm

The DBSCAN algorithm is a classical density-based clustering method [2] that can effectively identify noisy points. It views the clusters as the largest set of data connected by density, and the basic principle is that the points belonging to the same cluster are connected by density. The DBSCAN algorithm was first proposed by Ester et al. Each data point in a cluster with a certain radius *(Eps*) must contain at least a minimum number of points (*Mints)* in its neighbourhood. The algorithm is described as follows [14]:

2.1. Definitions

Definition 1 (*Eps*-neighborhood of a point): The circular area formed by taking a given data point p as the centre of a circle and *Eps* as the radius is called the neighbourhood for the given object p.

Definition 2 (Core point): A data point p is called a *core point* if the number of data points contained in its neighbourhood is larger than the threshold value of *Mints*.

Definition 3 (Border point): A data point p is called a border point if the number of data points contained in its neighbourhood is less than the threshold value of *Mints*.

Definition 4 (Noise point): A data point p is called a noise point if it does not belong to any of the clusters.

Definition 5 (Directly density-reachable) For the given parameter *Eps, Mints,* q is said to be directly density reachable from q to p if q is in the neighbourhood of the core point p.

Definition 6 (Density-reachable) For a given dataset if there exists a chain of data points, $p_1, p_2, \ldots, p_n$,(p_{i+1} from i directly density reachable, $i = 1, \ldots, n-1$), and q to p_n is directly density reachable, then q to p_1 is said to be density reachable.

Definition 7 (Density-connected) If the data points p and q are density reachable about the core point o, then p and q are said to be density connected about o.

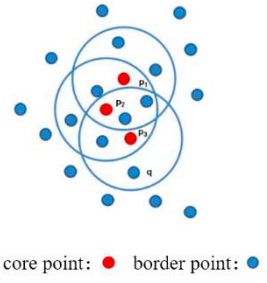

Fig. 1 Data point categories

The DBSCAN algorithm uses an iterative query to perform clustering. First, figure out all the core points in the dataset according to the given parameters *Eps* and *Mints*, which is called the core point set *N*. Then, examine a core point *o* in the dataset. If the point *o* is not labelled into a cluster, create a new cluster *C*, label all the points in the neighborhood of the point *o* and *o* into the cluster *C*. Next, continue if the point *p* is a core point, then mark all points in the neighbourhood of *p* into cluster *C*. Repeat the step of examining the remaining points in cluster *C* until no new points can be added to cluster *C*. Continue the above steps for the points in core point set *N* and not been examined until all points have been examined.

The DBSCAN algorithm continuously absorbs points with directly accessible density from a core point *p*. The different clusters are merged to one cluster if the density is accessible between them, which has the advantage that arbitrarily shaped clusters can be found and noisy points can be identified, but the time performance is low due to figuring out core points and examining all other points [15,16].

3. Improved algorithm K-DBSCAN

3.1. Adaptive parameter determination

In this paper, we use the distribution characteristics of the data itself to determine the two input parameters of the algorithm to locate the method itself. The advantage of doing so is that it can reflect the characteristics of the data as well as automatically determine the parameter values to avoid manual intervention. This method determines the *Eps, Mints* parameter columns by calculating the distance matrix as follows.

For a dataset D with n data points, the distance between each data point and its Kth nearest data point is calculated, and the n data distances are averaged to obtain the K distance average. K is taken from 1 to n, and all the K distance averages are calculated in turn to obtain a $1\ x\ n$ distance vector, which is the candidate parameter column of *Eps*. The *Eps* increases as K increases.

At the same K, for *Mints*, the number of points p_i in the neighborhood of each data point with the corresponding *Eps* as the radius is calculated, and n *Eps* are obtained and averaged, which is the corresponding to *Mints*. That is:

$$Mints = \frac{1}{n}\sum_{i=1}^{n} p_i \tag{1}$$

The two parameters are only determined by the distribution characteristics of the dataset itself, and as K increases, the number of clusters obtained will show a decreasing trend and remain stable and converge within a certain interval because *Eps* and *Mints* increase with the value of K[11]. In the case of the correct number of clusters, the larger the *Eps* and *Mints* are, the lower the noise rate and the higher the accuracy. Therefore, if the number of clusters is correct, the larger *Eps* and *Mints* are the better. Therefore, the combination of *Eps* and *Mints* corresponding to the maximum K in the stability interval is chosen as the parameter input.

3.2. K-DBSCAN

The main reason why the DBSCAN algorithm is time consuming is that all data points need to be traversed. The essence of the improved methods is to reduce the number of traversed data points, and the process of adaptively determining the optimal parameters requires several more calls to the clustering process, which consumes more time. Based on this, we proposes an improved algorithm K-DBSCAN that is very suitable for parameter adaptation. K-DBSCAN, which only deals with the core points, the steps are: 1. scan the dataset to find all core points, then remove all non-core points to reduce the dataset size; 2. consider each core point as a cluster and merge clusters with intersections; 3. points in the neighbourhood of the core point are in the same cluster as the core point. The theory foundation is that there are many neighbourhood areas that overlap between the core points that are close together. The DBSCAN algorithm can reduce the running time in two ways, and the difference compared with the traditional DBSCAN algorithm for the traversal of points is illustrated in Fig.2.

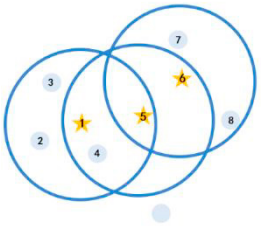

Fig.2 Comparison of ways of traversing points

In Fig.2, asterisks indicate core points, and circles indicate boundary points. For the improved K-DBSCAN algorithm, if the first traversed point is core point 1 and the next traversed point is a core point outside the neighbourhood, i.e., point 6, the intersection of the neighbourhoods of the two core points exists as core point 5, so they are grouped into one cluster *C1*. The above clearly shows that the K-DBSCAN algorithm reduces the number of point traversals compared to DBSCAN, which is the reason for the increased operational efficiency. The improved algorithm K-DBSCAN pseudo code is as follows.

K-DBSCAN *(Data, Eps, Mints)*

//all points in the dataset *Data* are initialized to unclassified

1) **for** i **from** 1 **to** len(*Data*):
2) N_i = *Data* _query（x_i, *Eps*）;
3) if N_i ≥Mints **then**
4) add x_i **to** the core points set O;
5) **end if**
6) **end for**
7) C=0;
8) //**for** the first examined core point x_1, cluster_id=C, C=1;
9) **for** x_i **in** O
10) **if** marked core point x_m **in** x_i's neighborhood then
11) cluster_id **for** x_i = cluster_id **for** x_m;
12) **else**

```
13)        cluster_id for x_m = C;
14)        C+=1;
15)    end if
16)  end for
17)  reorganize cluster；//merging clusters with duplicate core points
18)  border point; // unclassified one the same cluster with corresponding core point
19)  return cluster
```

4. Experiments and analysis of results

4.1. Experimental design

To verify the magnitude of the K-DBSCAN in time efficiency and the clustering effect over the DBSCAN algorithm, the experiments in this paper are designed as follows.

Experiment 1 Verify whether the K-DBSCAN algorithm improves time efficiency on a simple random dataset.

Experiment 2 Investigates the improvement in time efficiency and clustering results of the K-DBSCAN algorithm on complex datasets of different sizes.

Experiment 3 Investigates whether the K-DBSCAN algorithm continues to maintain its time efficiency advantage on multidimensional datasets.

The hardware environment tested are same, the algorithm implementation language is Python, and the runtime is selected as the average of ten runs. Therefore, the timestamp values used in the following experimental results are the average of the ten run times of the algorithm. The clustering results are compared using classification accuracy for labelled datasets and contour coefficients for unlabeled datasets.

4.2. Operational efficiency tests

A simple randomly generated test dataset in Python is used below for a time efficiency comparison. This generated random dataset has a total of 400 data points, which can be divided into 5 categories.

According to the parameter adaptive determination method, the optimal number of clusters and the value of the Eps and Mints combinations of the DBSCAN algorithm and the K-DBSCAN algorithm are found. It can be seen that the optimal number of clusters for the DBSCAN and K-DBSCAN algorithms is 6, which contains a class of noise points. Without considering the noise points, the number of clusters is the same as that of the original classification of the datasets, and the specific results are shown in Fig. 3. (a) and Fig. 3. (b). The corresponding optimal Eps and Mints parameter combinations for both are the same, and the optimal parameter groups is at K=77 (K indicates the column number of the parameter groups, and the following K has the same meaning). It can be seen that they are consistent for the noise points identified, but for some of the edge points clustering results are different, which is due to the different traversal methods of the two algorithms for the points.

In terms of running time, due to the small amount of data in the test dataset, the difference in running time is not significant. We use time stamps required to run the algorithms as the comparison. The DBSCAN algorithm runs with a single time stamp difference of 3.37 s, and the clustering results are 98.75% accurate. The K-DBSCAN algorithm runs with a single time stamp difference of 3.21 s, and the clustering results are 99% accurate. The running time of the improved algorithm is reduced, and the running efficiency has been improved.

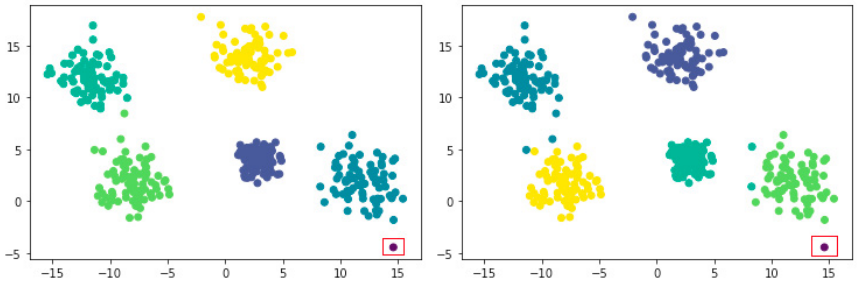

Fig. 3. (a) DBSCAN clustering results; Fig. 3. (b) K-DBSCAN clustering results.

4.3. Testing on different size datasets

To further test the efficiency of the K-DBSCAN, the next we conduct on more complex 2D datasets with different sizes. The datasets used are all unlabelled datasets, and the original data distribution is reflected in Figure 6. The D1-D6 two-dimensional datasets with volumes of 312, 572, 1024, 1744 2000 and 3444. Different datasets with density differences or with noise were used to test the efficiency, and the datasets are described in Table 1.

Table 1. Description of the dataset

Dataset name	Density differences	Noise point
D1	✓	
D2	✓	✓
D3		✓
D4		✓
D5	✓	✓
D6		✓

Table 2 shows the contour coefficients and the running time stamp differences to compare the clustering effect and running time for different datasets D1-D6. A comparison of the boosted time differences in the table below shows that the K-DBSCAN algorithm runs in less time than the DBSCAN algorithm on all tested datasets. Although the different densities have an effect on the running time of the datasets, overall, the improvement in running efficiency becomes more significant as the size of the dataset increases, and the clustering results obtained are somewhat different from those of the traditional algorithm. In terms of silhouette coefficients, the silhouette coefficients obtained by the DBSCAN algorithm and the K-DBSCAN algorithm have their own advantages in terms of performance effects on different datasets. K-DBSCAN has better clustering effects than DBSCAN on some datasets, and some clustering effects are not as good as DBSCAN. Because the parameter adaptation approach chosen in this paper requires multiple calls to the algorithm, which consumes time to grow rapidly on large-scale datasets; therefore, it is not recommended to use the combined method of this paper on large-scale datasets.

Table 2. Algorithm operation efficiency comparison table (unit: s)

Dataset	DBSCAN		K-DBSCAN		Comparison	
	Silhouette Coefficient	K-best	Silhouette Coefficient	K-best	Time difference value	Percentage of time difference
D1	-0.079	14	0.039✓	24	0.14	8.09%
D2	0.232	29	0.218	15	1.09	8.52%
D3	0.544	57	0.486	69	6.093	4.84%
D4	0.038	79	0.235✓	180	23.46	5.73%
D5	0.360	67	0.577✓	754	27.32	5.81%
D6	-0.057	255	0.0593✓	569	137.9316	2.70%

Fig. 4. (a) shows that the running time of both the DBSCAN and K-DBSCAN algorithms is not linear but rather parabolic, the figure 7-2 indicates that the running time of the algorithm is not only positively related to the size of the data volume but also that the larger the data volume is, the more significant the improvement. The Fig. 4. (a) and Fig. 4. (b) also shows that the K-DBSCAN algorithm maintains good runtime improvement as the amount of data increases, which illustrates the good scalability of the K-DBSCAN algorithm. For the first circular dataset D1, the K-DBSCAN algorithm has higher silhouette coefficients and identifies fewer noise points. For the smiley face dataset D2, although the K-DBSCAN algorithm does not have the same silhouette coefficients as the DBSCAN algorithm, both algorithms look good from the actual clustering results. And for the random dataset D5, similar results to dataset D1, the K-DBSCAN with higher silhouette coefficients and fewer noise points.

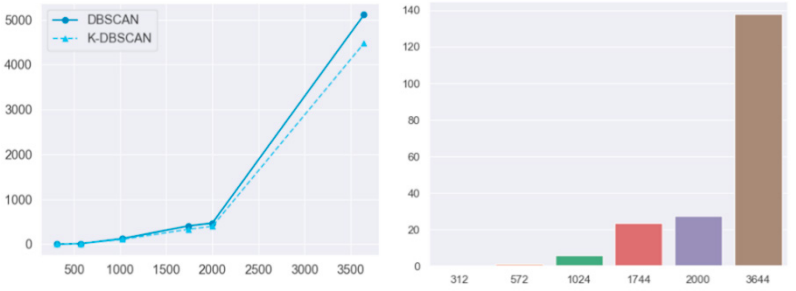

Fig. 4. (a) The running time against data size; Fig. 4. (b) The boosted time against data size.

4.4. Testing on multidimensional datasets

To investigate the clustering effect of the K-DBSCAN on different dimensional datasets, three multidimensional datasets commonly used for clustering on the UCI database are selected for comparison with the description in Table 3. The adaptive algorithm used previously to determine the optimal parameters was still used for multidimensional datasets. These datasets are often used for clustering, and the current clustering results were generally accepted.

Table 3. Description of the multidimensional number dataset

Dataset name	Data volume	Number of dimensions	Number of clusters
Iris	150	4	3
Wine	178	13	3
Glass	214	6	6

For the multidimensional dataset, the two results are very similarly in the Iris dataset, with the DBSCAN algorithm having better silhouette coefficients than the K-DBSCAN algorithm, but the DBSCAN algorithm identifies two clusters as one and a small number of noise points. For the Wine dataset, the K-DBSCAN silhouette coefficients are better than the DBSCAN algorithm, and more noise points are identified. As for Glass dataset, both algorithms will identify only one cluster, but the number of noise points identified differs, with the K-DBSCAN algorithm having better silhouette coefficients than the DBSCAN algorithm.

Importantly, the K-DBSCAN algorithm outperforms the DBSCAN algorithm in time performance on all three datasets, demonstrating that the advantage of the K-DBSCAN algorithm in time performance is maintained for multidimensional datasets. But the K-DBSCAN algorithm has not been optimized for the problem of poor clustering effect of the DBSCAN algorithm on multidimensional datasets. In general, the DBSCAN algorithm uses Euclidean distances to calculate distances between points, which leads to a decrease in the distance measure as the dimensionality of the dataset increases, which directly affects the determination of parameters.

Table 4. Comparison of running efficiency in multidimensional dataset (unit: s)

Dataset	DBSCAN			K-DBSCAN		
	Running time	Silhouette Coefficient	Number of clusters	Running time	Silhouette Coefficient	Number of clusters
Iris	0.28	0.528	3	0.26	0.446	3
Wine	0.50	0.281	4	0.46	0.287✓	4
Glass	0.69	0.618	2	0.64	0.636✓	2

5. Summary

In this paper, the way of the DBSCAN algorithm traversing data points has been changed by only examining some of the core points. By this way, we get the K-DBSCAN algorithm which improves the time efficiency and maintains this property for both density differences and multidimensional datasets. And the improvement is more obvious for larger datasets. However, the K-DBSCAN algorithm still suffers from the problem that the DBSCAN algorithm does not get the good clustering results of density differences and multidimensional datasets. In following research, the distance calculation can be changed to introduce a more suitable distance calculation method in multidimensional datasets. For datasets with density differences, data partitioning methods such as gridding and partitioning can be used to first locally cluster and then merge [17] to improve the clustering effect.

Acknowledgements

The authors also acknowledge the financial support of the NSFC (71932008, 71401188).

References

[1] Chen M S, Han J and Yu P S (1996) "Data mining: an overview from a database perspective." *IEEE Transactions on Knowledge and data Engineering* **8(6)**:866-883.

[2] Ester M, Kriegel H P and Sander J, Xu X (1996) "A density-based algorithm for discovering clusters in large spatial databases with noise." *Knowledge Discovery and Data Mining* **96(34)**:226-231.

[3] Thang T M and Kim J (2011) "The anomaly detection by using dbscan clustering with multiple parameters." *2011 International Conference on Information Science and Applications* **2011**:1-5.

[4] Qian W, Gong X and Zhou A (2003) "Clustering in very large databases based on distance and density." *Journal of Computer Science and technology* **18(1)**:67-76.

[5] Ankerst M, Breunig M M, Kriegel H P and Sander J (1999) "OPTICS: Ordering points to identify the clustering structure." *ACM Sigmod record* **28(2)**:49-60.

[6] Pingjiang F and Lindong G (2004) "Adaptive DBSCAN-based algorithm for constellation reconstruction and modulation identification." *2004 Asia-Pacific Radio Science Conference, 2004. Proceedings. IEEE* **2004**:177-180.

[7] Shi-hong Y, Ping L, Ji-dong G and Shui-geng Z (2005) "A statistical information-based clustering approach in distance space." *Journal of Zhejiang University-Science A* **6(1)**:71-78.

[8] Li W, Yan S, Jiang Y, Zhang S and Wang C (2019) "Research on method of self-adaptive determination of DBSCAN algorithm parameters." *Computer Engineering and Applications* **55(05)**:1-7.

[9] Zhou S, Zhou A, Cao J, Wen J, Fan Y and Hu Y (2000) "Combining sampling technique with DBSCAN algorithm for clustering large spatial databases." *Knowledge Discovery and Data Mining. Current Issues and New Applications: 4th Pacific-Asia Conference, PAKDD 2000 Kyoto, Japan, April 18–20, 2000 Proceedings 4. Springer Berlin Heidelberg* **2000**:169-172.

[10] Zhou H F and ZHOU Y (2012) "Density-based clustering algorithm combined with limited regional sampling." *Journal of Computer Applications* **32(08)**:2182.

[11] Shuangqing L I and Shengdi M U (2014) "Improved DBSCAN algorithm and its application." *Computer Engineering and Applications* **50(08)**:72-76.

[12] Li S S (2020) "An improved DBSCAN algorithm based on the neighbor similarity and fast nearest neighbor query." *IEEE Access* **8**:47468-47476

[13] Hanafi N and Saadatfar H (2022) "A fast DBSCAN algorithm for big data based on efficient density calculation." *Expert Systems with Applications* **203**:117501.

[14] Zhou A, Zhou S, Cao J, Fan Y and Hu Y (2000) "Approaches for scaling DBSCAN algorithm to large spatial databases." *Journal of computer science and technology* **15:**509-526.

[15] Birant D, Kut (2007) "ST-DBSCAN: An algorithm for clustering spatial-temporal data." *Data and Knowledge Engineering* **60(1)**:208–221.

[16] Jiang H, Li J, Yi S, Wang X and Hu X (2011) "A new hybrid method based on partitioning-based DBSCAN and ant clustering." *Expert Systems with Applications* **38(8)**:9373-9381.

[17] Darong H and Peng W (2012) "Grid-based DBSCAN algorithm with referential parameters." *Physics Procedia* **24**: 1166-1170.