Automated Determination of the Input Parameter of DBSCAN Based on Outlier Detection

Zohreh Akbari(✉) and Rainer Unland

Institute for Computer Science and Business Information Systems (ICB),
University of Duisburg-Essen, Essen, Germany
{zohreh.akbari,rainer.unland}@icb.uni-due.de

Abstract. During the last two decades, DBSCAN (Density-Based Spatial Clustering of Applications with Noise) has been one of the most common clustering algorithms, that is also highly cited in the scientific literature. However, despite its strengths, DBSCAN has a shortcoming in parameter detection, which is done in interaction with the user, presenting some graphical representation of the data. This paper introduces a simple and effective method for automatically determining the input parameter of DBSCAN. The idea is based on a statistical technique for outlier detection, namely the empirical rule. This work also suggests a more accurate method for detecting the clusters that lie close to each other. Experimental results in comparison with the old method, together with the time complexity of the algorithm, which is the same as for the old algorithm, indicate that the proposed method is able to automatically determine the input parameter of DBSCAN quite reliably and efficiently.

Keywords: Clustering · DBSCAN · Empirical rule · Machine learning · Outlier detection · Parameter determination · Unsupervised learning

1 Introduction

Machine Learning (ML) is one of the core fields of Artificial Intelligence (AI) and is concerned with the question of how to construct computer programs that automatically improve with experience [1]. Depending on the nature of the learning data available to the learning system, machine learning methods are typically classified into three main categories [2, 3]: supervised, unsupervised and reinforcement learning. In supervised learning example inputs and their desired outputs are given and the goal is to learn a general rule that maps these inputs to their desired outputs. In unsupervisaed learning, on the other hand, no labels are given to the learning algorithm, leaving it on its own to find the hidden structure of the data, e.g. to look for the similarities between the data instances (i.e. clustering [4]), or to discover the dependencies between the variables in large databases (i.e. association rule mining [5]). In reinforcement learning the desired input/output pairs are again not presented, however, the algorithm is able to estimate the optimal actions by interacting with a dynamic environment and based on the outcomes

L. Iliadis and I. Maglogiannis (Eds.): AIAI 2016, IFIP AICT 475, pp. 280–291, 2016.
DOI: 10.1007/978-3-319-44944-9_24

of the more recent actions, while ignoring experiences from the past, that were not reinforced recently.

This research focuses on the most common unsupervised learning method (i.e. cluster analysis [4, 6]), and more specifically on one of its successful algorithms the Density-Based Spatial Clustering of Applications with Noise (DBSCAN) [7]. As mentioned above, in unsupervised learning, learner processes the input data with the goal of coming up with some summary or compressed version of the data [4]. Clustering a dataset is a typical example of this type of learning. Clustering is the task of grouping a set of objects such that similar objects end up in the same group and dissimilar objects are diverted into different groups. Clearly, this description is quite imprecise and possibly ambiguous. However, quite surprisingly, it is not at all clear how to come up with a more rigorous definition [4], and since no definition of cluster is widely accepted many algorithms have been developed to suit specific domains [8], each of which using a different induction principle [9].

Due to their diversity, clustering methods are classified into different categories in the scientific literature [9–12]. However, despite the slight differences between these classifications, they all mention the DBSCAN algorithm as one of the eminent methods available. DBSCAN owes its popularity to the group of capabilities it offers [7]: (1) it does not require the specification of the number of clusters in the dataset beforehand, (2) it requires little domain knowledge to determine its input parameter, (3) it can find arbitrarily shaped clusters, (4) it has good efficiency on large datasets, (5) it has a notion of noise, and is robust to outliers, (6) it is designed in a way that it can be supported efficiently by spatial access methods such as R*-trees [13], and so on.

DBSCAN algorithm requires two input parameters, namely *Eps* and *MinPts*, which are considered to be the density parameters of the thinnest cluster acceptable, specifying the lowest density which is not considered to be noise. These parameters are hence respectively the radius and the minimum number of data objects of the least dense cluster possible. The algorithm supports the user in determining the appropriate values for these parameters offering a heuristic method, which imposes the user interaction based on some graphical representation of the data (presented in Sect. 2.2). However, since DBSCAN is sensitive to its input parameters and the parameters have significant influences on the clustering result, an automated and more precise method for the determination of the input parameters is needed.

Some notable algorithms targeting this problem are: (1) GRPDBSCAN, which combines the grid partition technique and DBSCAN algorithm [14], (2) DBSCAN-GM, that combines Gaussian-Means and DBSCAN algorithms [15], and (3) BDE-DBSCAN, which combines Differential Evolution and DBSCAN algorithms [16]. Opposed to these methods, which all intend to solve the problem using some other techniques, this paper remains with the original idea of the DBSCAN algorithm and just tries to omit the user interaction needed, allowing the algorithm to detect the appropriate value itself. This is done using some basic statistical techniques for outlier detection. Two different approaches are mentioned in this paper, which apply the concept of standard deviation to the problem of outlier detection, namely the empirical rule for normal distributions and the Chebyshev's inequality for non-normal distributions [17, 18]. This work, however, focuses mainly on the application of the empirical rule to outlier detection in

normal distributed data, and addresses the Chebyshev's inequality only as a possible solution for non-normal distributions.

The rest of the paper is organized as follows. Section 2 describes the DBSCAN algorithm and its supporting technique for the determination of its input parameters. In Sect. 3, the above mentioned statistical techniques for outlier detection are presented (i.e. the empirical rule and the Chebyshev's inequality). Section 4 describes the automated technique for the determination of the parameter *Eps*. Experimental results and the time complexity of the automated technique are then discussed in Sect. 5. Section 6 concludes with a summary and some directions for the feature researches.

2 DBSCAN: Density-Based Spatial Clustering of Applications with Noise

According to [7], the key idea of DBSCAN algorithm is that for each point of the cluster the neighborhood of a given radius has to contain at least a minimum number of points, i.e. the density in the neighborhood has to exceed some threshold. The following definitions support the realization of this idea.

Definition 1 (*Eps − neighborhood* of a point): The *Eps − neighborhood* of a point p, denoted by $N_{Eps}(p)$, is defined by $N_{Eps}(p) = \{q \in D | dist(p,q) \leq Eps\}$.

Definition 2 (directly density-reachable): A point p is directly density-reachable from a point q, w.r.t. *Eps* and *MinPts*, if

1. $p \in N_{Eps}(q)$ and
2. $\left|N_{Eps}(q) \geq MinPts\right|$

The second condition is called core point condition (There are two kinds of points in a cluster, points inside of the cluster, called core points, and points on the border of the cluster, called border points).

Definition 3 (density-reachable): A point p is density-reachable from a point q, w.r.t. *Eps* and *MinPts*, if there is a chain of points $p_1, \ldots, p_n, p_1 = q, p_n = p$ such that p_{i+1} is directly density-reachable from p_i.

Definition 4 (density-connected): A point p is density-connected to a point q, w.r.t. *Eps* and *MinPts*, if there is a point o such that both, p and q are density-reachable from o, w.r.t. *Eps* and *MinPts*.

Definition 5 (cluster): Let D be a database of points. A cluster C, w.r.t. *Eps* and *MinPts*, is a non-empty subset of D satisfying the following conditions:

1. $\forall p, q$:if $p \in C$ and q is density-reachable from p, w.r.t. *Eps* and *MinPts*, then $q \in C$. (Maximality)
2. $\forall p, q \in C$: p is density-connected to q, w.r.t. *EPS* and *MinPts*. (Connectivity)

Definition 6 (noise): Let $C_1, \ldots, C_k$ be the clusters of the database D, w.r.t. parameters Eps_i and $MinPts_i, i = 1, \ldots, k$. Then the noise is defined as the set of points in the database D not belonging to any cluster C_i, i.e. $noise = \{p \in D | \forall i{:}p \notin C_i\}$.

The following lemmata are important for validating the correctness of the algorithm. Intuitively, they state that having the parameters *Eps* and *MinPts*, a cluster can be discovered in a two-step approach. First, choose an arbitrary point from the database satisfying the core point condition as a seed. Second, retrieve all points that are density-reachable from the seed, obtaining the cluster containing the seed.

Lemma 1: Let p be a point in D and $|N_{Eps}(p)| \geq MinPts$. Then the set $O = \{o | o\, D\, and\, o\, is\, density - reachable\, from\, p,\, w.r.t.\, Eps\, and\, MinPts\}$ is a cluster, w.r.t. *Eps* and *MinPts*.

Lemma 2: Let C be a cluster, w.r.t. *Eps* and *MinPts*, and let p be any point in C with $|N_{Eps}(p)| \geq MinPts$. Then C equals to the set $O = \{o | o\, is\, density - reachable\, from\, p, w.r.t.\, Eps\, and\, MinPts\}$.

2.1 The Algorithm

The DBSCAN algorithm can be described as follows (Table 1):

Table 1. Algorithm 1: Pseudo-code of the DBSCAN

DBSCAN Algorithm (Input: *D*, *Eps*, *MinPts*)
1. While (*D* has an unclassified[a] point)
2. Select an arbitrary unclassified point *p*.
3. If *p* does not satisfy the core point condition, mark it as a noise.
4. Else retrieve all the density-reachable points from $N_{Eps}(p)$ forming a cluster containing $N_{Eps}(p)$ and mark all the member of this cluster as classified.
5. End While

[a] Note that the term unclassified here indicates that it is not determined yet if the point is a noise or not.

2.2 Determining the Parameters *Eps* and *MinPts*

DBSCAN offers a simple but effective heuristic method to determine the parameters *Eps* and *MinPts* of the thinnest cluster in the dataset. For a given k function $k - dist$ is defined from the Database D to the real numbers, mapping each point to the distance from its $k - th$ nearest neighbor. When sorting the points of the dataset in descending order of their $k - dist$ values, the graph of this function gives some hints concerning the density distribution in the dataset. This graph is called the sorted $k - dist$ graph. It is clear that the first point in the first valley of the $MinPts - dist$ graph can be the threshold point with the maximal $MinPts - dist$ value in the thinnest cluster. All points with a larger $MinPts - dist$ value are considered to be noise, and all the other points are assigned to some clusters.

DBSCAN states that according to experiments, the $k - dist$ graphs for $k > 4$ do not significantly differ from the $4 - dist$ graph and, furthermore, they need considerably

more computation. Therefore, it eliminates the parameter *MinPts* by setting it to 4 for all datasets (for 2-dimensional data). The parameter determination method also explains, that since in general, it is very difficult to detect the first valley of the $k - dist$ graph automatically, but it is relatively simple for the user to see this valley in a graphical representation, it is suggested to follow an interactive approach for determining the threshold point.

3 Statistical Techniques for Outlier Detection

The term noise in DBSCAN algorithm is equivalent to an outlier in statistics, which is an observation that is far removed from the rest of the observations [19]. One of the basic statistical techniques for outlier detection is called the empirical rule. The empirical rule is an important rule of thumb, that is used to state the approximate percentage of values that lie within a given number of standard deviation from the *mean* of a set of data if the data are normally distributed. The empirical rule, also called the 68-95-99.7 rule or the three-sigma rule of thumb states that 68.27 %, 95.45 % and 99.73 % of the values in a normal distribution lie within one, two and three standard deviations of the mean [17]. One of the practical usages of the empirical rule is as a definition of outliers as the data that fall more than three standard deviations from the norm in normal distributions [20] (Fig. 1).

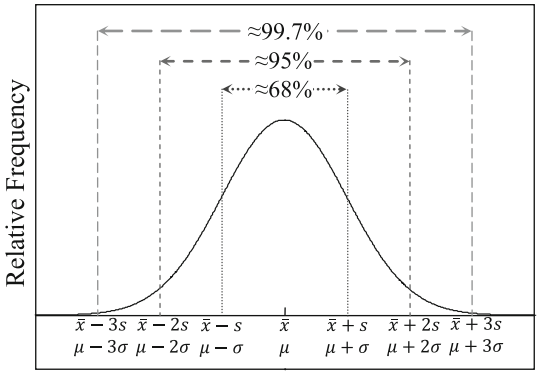


Fig. 1. The Empirical Rule [21]

If there are many points that fall more than three standard deviations from the norm, then the distribution is most likely non-normal. In this case, Chebyshev's inequality, which applies to non-normal distributions, is applicable. Chebyshev's inequality states that in any probability distribution, at least $1 - \frac{1}{k^2}$ of the values are within k standard deviations of the *mean* [17] (e.g. in non-normal distributions at least 99 % of the values lie within 10 standard deviations of the *mean*). Hence, using the Chebyshev's inequality,

the outlier can also be defined as the data that fall outside an appropriate number of standard deviations from the mean [22][1].

4 Automated Determination of the Parameter Eps

Setting the *MinPts* to 4, determining the parameter *Eps*, the algorithm is aiming a radius that covers the majority of the $4-dist$ values and stands well as a threshold for the specification of the noise values. As mentioned above, the term noise in DBSCAN algorithm is equivalent to an outlier in statistics, which is an observation that is far removed from the rest of the observations [19]. Thus, the idea here is to use statistical rules in order to find the threshold value between the accepted $4-dist$ values and the values considered for the noise points.

As mentioned above, one of the practical usages of the empirical rule is as a definition of outliers as the data that fall more than three standard deviations from the norm in normal distributions [20]. Thus, considering the $4-dist$ values, the value of parameter *Eps* can be set to their *mean* plus three standard deviations. This would cover even more than 99.73 % of the calculated $4-dist$ values, since the $4-dist$ values smaller than $mean-3\times SD$ are also covered here.

Border points and even in general, points closer to the border of the clusters usually have greater $k-dist$ values, which lead to larger *Eps* values and thus might cause two close clusters to be detected as one cluster (Since the parameter *MinPts* or k is set to 4, this problem may be caused mostly by the border points). These relatively greater $k-dist$ values, however, do not have any positive effect on the process of cluster detection, as the $k-dist$ values of the core points are actually the ones forming the right clusters and at the same time covering the border points. Figure 2 shows a case in which the $4-dist$ value of border point p is much larger than the $4-dist$ value of the core point q, which can actually cover p in its $4-dist-neighborhood$.

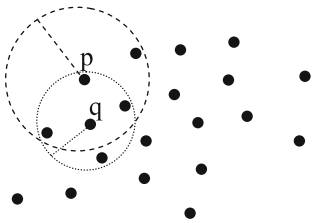


Fig. 2. $4-dist$ values for example core (q) and border point (p)

In order to eliminate the negative effect of the $k-dist$ values of the border points, the algorithm presented here considers any point with minimum $k-dist$ value which

1 This work focuses solely on the empirical rule and the normal distributions. However, the possibility of using the Chebyshev's inequality is given here, in order to show that the general idea of using outlier detection techniques for the reason of parameter determination in DBSCAN is not limited to the distribution of the data.

covers the border point in its $k-dist-neighborhood$ and replaces the $k-dist$ value of this border point with the $k-dist$ value of this core point. Thus for a given k, function $k-dists'$ is defined from the Dataset D to the real numbers, mapping each point to the $k-dist$ value of any core point, covering this point in its $k-dist-neighborhood$, with minimum $k-dist$ value. Actually, following this technique, points are considered in ascending order of their $4-dist$ values, then taking each point p, if the $4-dist'$ value for any point in its four nearest neighbors is not set so far, this value will be set to the $4-dist$ value of point p. Using this technique for each point, the $k-dist$ value of the smallest cluster, the point can join, would be considered. At the end the *mean* and the standard deviation of these $k-dist'$ values which are saved for all points are calculated and the Eps' value is set to $mean + 3 \times SD$. The following pseudo-code indicates this method (Table 2).

Table 2. Algorithm 2: Pseudo-code of the *EpsFinder*

EpsFinder **(Input: D)**
1. For each point p find the four nearest neighbors.
2. Sort the points in ascending order of theirs $4-dist$ values.
3. Following the ascending order, take each point p and if the $4-dist'$ value for any of its four nearest neighbors is not set so far, set this value to the $4-dist$ value of the point p.
4. Calculate the *mean* of the $4-dist'$ values: *mean*
5. Calculate the standard deviation of the $4-dist'$ values: *SD*
6. Set the Eps' value to $mean + 3 \times SD$.

5 Experimental Results and Time Complexity

In this section the experimental results and the time complexity of the automated technique proposed in Sect. 4 (*EpsFinder*) are discussed.

5.1 Experimental Results and Discussions

In this section, the algorithm presented in Sect. 4 is applied to some datasets. This makes the comparison between the old method and the new automated method possible. All the experiments were performed on Intel(R) Celeron(R) CPU 1.90 GHz with 2 GB RAM on the Microsoft Windows 8 platform. The algorithm and the datasets were implemented in Java on Eclipse IDE, MARS.1. Sample datasets are depicted in Fig. 3. The noise percentage for datasets 1 and 2 is 0 %, however, datasets 3 and 4 do have noise values.

In order to show the results of the clustering, each cluster is presented by a different shade of gray in Fig. 4. Noise points are marked using black color.

Figure 5 shows the sorted $4-dist'$ graphs of the sample datasets. Here, *Eps* indicates the value determined by the user, according to the visual representation of the data, and Eps' represents the value calculated automatically by the algorithm presented in Sect. 4 (*EpsFinder*).

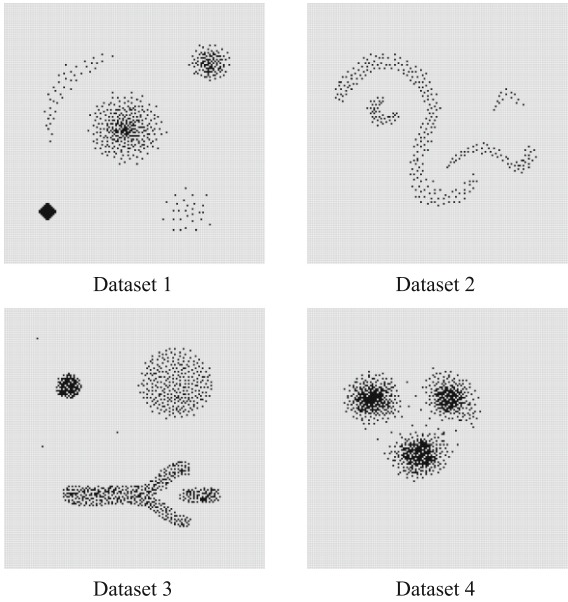


Fig. 3. Sample datasets

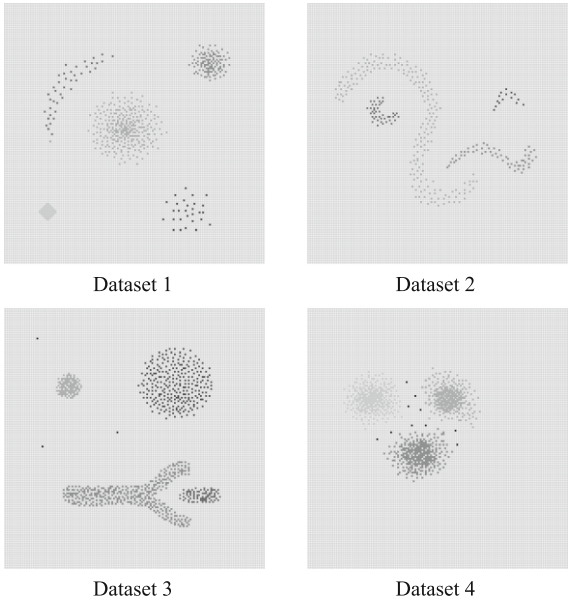


Fig. 4. Detected clusters

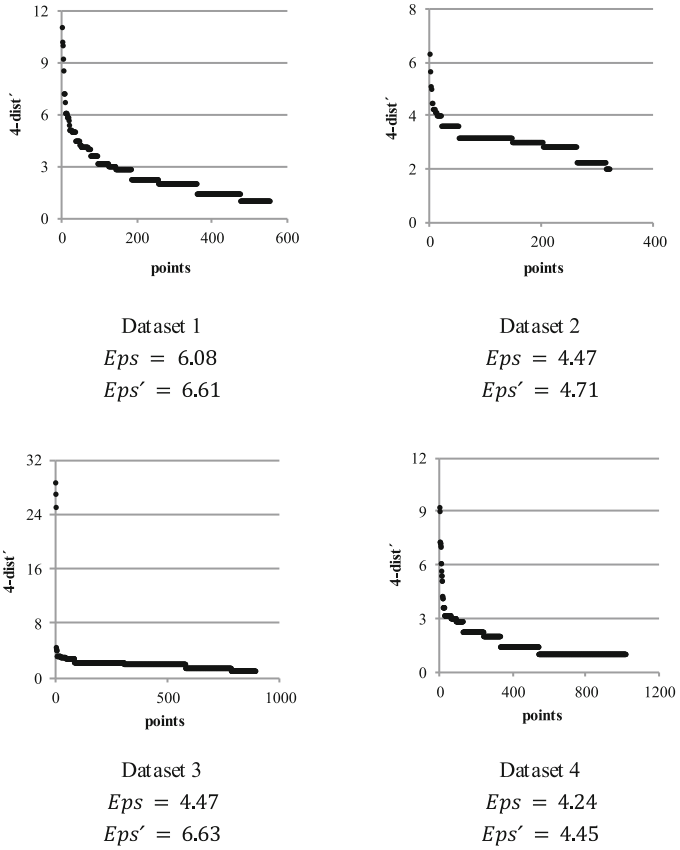
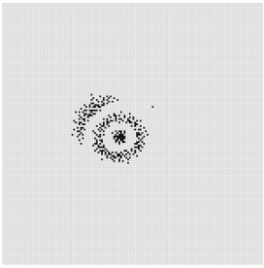


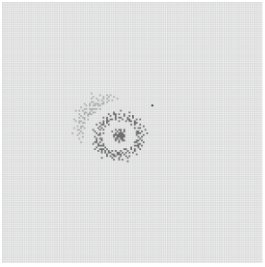
Fig. 5. Sorted $4-dist'$ graphs for sample datasets (Note that the larger difference between *Eps* and *Eps*′ for Dataset 3 is caused by the larger difference between the $4-dist'$ values of those data instances considered as noise and the rest of the data instances. This difference has no effect on the clustering result, since *Eps* and *Eps*′ are actually threshold values and since there are no data instances with $4-dist'$ values between *Eps* and *Eps*′, the clustering result would remain the same.)

In order to illustrate the problem that may occur with the $k-dist$ value of the border points (discussed in Sect. 4), dataset 5 is presented here (Fig. 6). This dataset is defined in a way that nested and very close clusters are available in it.
Result 1 in Fig. 7, indicates the clustering result according to the normal $4-dist$ values, which were considered by the old method. It is clear that the algorithm has failed to distinguish the nested clusters. Result 2 in Fig. 7, on the other hand, shows the clustering result according to the normal $4-dist'$ values. Here, the *Eps* value calculated is smaller and hence the algorithm is able to detect the nested clusters easily. Graph 1 and Graph 2 in Fig. 7 show here the $4-dist$ and $4-dist'$ values calculated using each of the techniques, together with the corresponding *Eps* and *Eps*′ values.

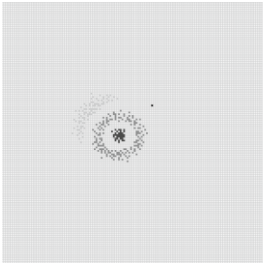


Dataset 5

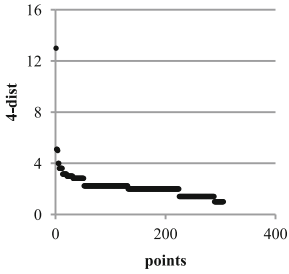
Fig. 6. Dataset 5



Result 1



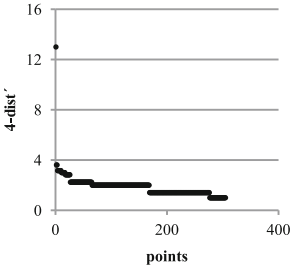
Result 2



Graph 1

$Eps = 5.10$

$Eps' = 4.82$



Graph 2

$Eps = 3.61$

$Eps' = 4.34$

Fig. 7. Different clustering results for dataset 5

It should be pointed out that even though the experiments presented here were all for 2-dimensional datasets, the idea can be applied to high-dimensional datasets as well. This is clearly possible, since the calculation of the distance between the points and the application of standard deviation remains the same for high-dimensional datasets. The only point that must be considered is that, the DBSCAN has suggested 4 as the *MinPts* value just for 2-dimensional datasets. However, as mentioned before, *Eps* and *MinPts* are the density parameters of the thinnest cluster; therefore it is always possible to

determine the *Eps* by keeping the *MinPts* parameter small enough (or even just by setting it to one). The diversity of the density may always be described with different radii containing a predefined number of points (*MinPts*).

5.2 Time Complexity

Since the algorithm needs to find the four nearest neighbors of each point in the dataset, the time complexity of the algorithm cannot be less than $O(n^2)$. Of course, since these points should have been also retrieved in the user interaction technique, and the only difference here is the calculation of the *mean* and the standard deviation, which can be done in $O(n)$, it is clear that the time complexity of the automated technique presented here, is the same as for the old method. Thus concerning the automated abilities of this technique, it is obvious that the application of this approach in the determination of the *Eps* parameter is quite reasonable.

6 Conclusion

This paper proposes a simple and effective method to automatically determine the input parameter *Eps* of DBSCAN. The work remains with the original idea of the DBSCAN algorithm and just tries to omit the user interaction needed, and allow the algorithm to detect the appropriate value itself. This is done using some basic statistical techniques for outlier detection. Two different approaches are mentioned here, which apply the concept of standard deviation to the problem of outlier detection, namely the empirical rule for normal distributions and Chebyshev's inequality for non-normal distributions. One of the practical usages of the empirical rule is as a definition of outliers as the data that fall more than three standard deviations from the norm in normal distributions. Thus, the value of parameter *Eps* can be set to *mean* plus three standard deviations. This value would cover the majority of the $k - dist'$ values and stands well as a threshold for the specification of the noise values. This work also mentioned the problem which occurs with the $k - dist$ values of the border points, and suggests a more accurate method for the determination of the values, based on which *Eps* is calculated (i.e. $k - dist'$ values). Experimental results and the time complexity of the proposed algorithm suggest that the application of this technique in the determination of the *Eps* parameter is quite reasonable. The concentration of this research was mainly on the application of the empirical rule to outlier detection in normal distributed data. The future works will have to consider the Chebyshev's inequality for possible non-normal distributions of $k - dist'$ values.

References

1. Mitchell, T.M.: Machine Learning. McGraw Hill, New York (1997)
2. Russell, S., Norvig, P.: Artificial Intelligence: A Modern Approach, 2nd edn. Prentice Hall, Englewood Cliffs (2002)
3. Hertzmann, A., Fleet, D.: Machine Learning and Data Mining Lecture Notes, CSC 411/D11, Computer Science Department, University of Toronto (2012)

4. Shalev-Shwartz, S., Ben-David, S.: Understanding Machine Learning: From Theory to Algorithm, Cambridge University Press, New York (2014)
5. Ventura, S., Luna, J.M.: Pattern Mining with Evolutionary Algorithms. Springer, Heidelberg (2016)
6. Murphy, K.P.: Machine Learning: A Probabilistic Perspective. The MIT Press, Cambridge (2012)
7. Ester, M., Kriegel, H.–P., Sander, J., Xu, X.: A density-based algorithm for discovering clusters in large spatial databases with noise. In: Simoudis, E., Han, J., Fayyad, U.M. (eds.) Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96), pp. 226–231. AAAI Press (1996)
8. Estivill-Castro, V., Yang, J.: A Fast and robust general purpose clustering algorithm. In: Pacific Rim International Conference on Artificial Intelligence, pp. 208–218 (2000)
9. Rokach, L., Maimon, O.: Clustering methods. In: Rokach, L., Maimon, O.: The Data Mining and Knowledge Discovery Handbook, pp. 321–352. Springer Science + Business Media, Inc., Heidelberg (2005)
10. Berkhin, P.: Survey of Clustering Data Mining Techniques, Technical Report, Accrue Software, San Jose, CA (2002)
11. Han, J., Kamber, M.: Data Mining Concepts and Techniques, pp. 335–391. Morgan Kaufmann Publishers, San Francisco, CA (2001)
12. Everitt, B.S., Landau, S., Leese, M., Stahl, D.: Cluster Analysis, 5th edn. Wiley, Chichester (2011)
13. Beckmann, N., Kriegel, H.-P., Schneider, R., Seeger, B.: The R*-tree: an efficient and robust access method for points and rectangles. In: Proceedings of the ACM SIGMOD International Conference on Management of Data, Atlantic City, NJ, pp. 322–331 (1990)
14. Darong, H., Peng, W.: Grid-based DBSCAN algorithm with referential parameters. In: Proceedings of the International Conference on Applied Physics and Industrial Engineering (ICAPIE-2012), Phys. Procedia, vol. 24(B), pp. 1166–1170 (2012)
15. Smiti, A., Elouedi, Z.: DBSCAN-GM: An improved clustering method based on Gaussian means and DBSCAN techniques. In: 16th International Conference on Intelligent Engineering Systems (INES), pp. 573–578 (2012)
16. Karami, A., Johansson, R.: Choosing DBSCAN parameters automatically using differential evolution. Int. J. Comput. Appl. **91**(7), 1–11 (2014)
17. Black, K.: Business Statistics: For Contemporary Decision Making (7th Edn.), Wiley, Hoboken, NJ (2011)
18. Ott, R. L., Longnecker, M.T.: An Introduction to Statistical Methods and Data Analysis (7th Edn.), Cengage Learning, Boston (2015)
19. Maddala, G.S.: Outliers. Introduction to Econometrics, 2nd edn, pp. 88–96. MacMillan, New York (1992)
20. Coolidge, F.L.: Statistics: A Gentle Introduction, p. 458. SAGE Publications, Inc., Thousand Oaks (2012)
21. Shafer, D. S., Zhang, Z.: Introductory Statistics, v. 1.0, Flatworld Knowledge, Washington, D.C. (2012)
22. Amidon, B.G., Ferryman, T.A., Cooley, S.K.: Data outlier detection using the Chebyshev theorem. In IEEE Aerospace Conference Proceedings, pp. 3814–3819 (2005)