A Novel Approach to Determining the Radius of the Neighborhood Required for the DBSCAN Algorithm

Artur Starczewski(✉)

Institute of Computational Intelligence, Częstochowa University of Technology,
Al. Armii Krajowej 36, 42-200 Częstochowa, Poland
artur.starczewski@pcz.pl

Abstract. Data clustering is one of the most important methods used to discover naturally occurring structures in datasets. One of the most popular clustering algorithms is the Density-Based Spatial Clustering of Applications with Noise (DBSCAN). This algorithm can discover clusters of arbitrary shapes in datasets and thus it has been widely applied in many different applications. However, the DBSCAN requires two input parameters, i.e. the radius of the neighborhood (*eps*) and the minimum number of points required to form a dense region (*MinPts*). The right choice of the two parameters is a fundamental issue. In this paper, a new method is proposed to determine the radius parameter. In this approach the distances between each element in the dataset and its k-th nearest neighbor are used, and then in these distances abrupt changes in values are identified. The performance of the new approach has been demonstrated for several different datasets.

Keywords: Clustering algorithms · DBSCAN · Data mining

1 Introduction

Clustering refers to grouping objects into meaningful clusters so that the elements of a cluster are similar, whereas they are dissimilar in different clusters. Data clustering is a very useful technique used in many fields, such as biology, spatial data analysis, business, and others. Moreover, clustering methods can be used during the process of designing various neural networks [1,2], fuzzy and rule systems [7,9,20,29], and creating some algorithms for the identification of classes [12]. A variety of large collections of data brings a great challenge for clustering algorithms, so a lots of new different clustering algorithms and their configurations are being intensively developed, e.g. [11,13,14]. It should be noted that there is no clustering algorithm which creates the right clusters for all datasets. Moreover, the same algorithm can also produce different results depending on the input parameters applied. Therefore, cluster validation should be also used to assess the results of data clustering. So far, a number of authors have proposed different cluster validity indices or modifications of existing ones, e.g., [10,23,26,27]. Generally, clustering algorithms can be divided into four categories including partitioning, hierarchical, grid-based, and density-based clustering.

L. Rutkowski et al. (Eds.): ICAISC 2021, LNAI 12854, pp. 358–368, 2021.
https://doi.org/10.1007/978-3-030-87986-0_32

For example, the well-known partitioning algorithms are, e.g. *K-means*, *Partitioning Around Medoids* (*PAM*) [4,32] and *Expectation Maximization* (*EM*) [18], whereas the hierarchical clustering includes agglomerative and divisive approaches, e.g. the *Single-linkage*, *Complete-linkage* or *Average-linkage* or *DIvisive ANAlysis Clustering* (*DIANA*) [19,22]. Then, the grid-based approach includes methods such as e.g. the *Statistical Information Grid-based* (*STING*) or *Wavelet-based Clustering* (*WaveCluster*) [21,25,30]. The next category of clustering algorithms is the density-based approach. The *Density-Based Spatial Clustering of Application with Noise* (*DBSCAN*) is the most well-known density-based algorithm [8]. However, it is seldom used to cluster multi-dimensional data, but now the original DBSCAN has also many various extensions, e.g. [3,5,6,17,24,31]. This algorithm requires two input parameters, i.e. the *eps* and *MinPts*. Determination of these parameters is very difficult, but the right choice of those parameters is a fundamental issue. In literature, some methods have been proposed to determine these parameters, e.g. [16].

In this paper, a new approach to determining the *eps* parameter is proposed. It is based on an analysis of abrupt changes in the distances between each element of the dataset and its *k*-th nearest neighbor. This paper is organized as follows: Sect. 2 presents a detailed description of the *DBSCAN* clustering algorithm. In Sect. 3 the new method to determine the *eps* radius is outlined while Sect. 4 illustrates experimental results obtained on datasets. Finally, Sect. 5 presents conclusions.

2 The Description of the DBSCAN Algorithm

In this section, the basic concept of the *DBSCAN* algorithm is described. As mentioned above, it is a very popular algorithm because it can find clusters of arbitrary shapes and requires only two input parameters, i.e. the *eps* (the radius of the neighborhood) and the *MinPts* (the minimum number of points required to form a dense region). To understand the basic concept of the algorithm several terms should be explained. Let us denote a dataset by X, where point $p \in X$. The *eps* is usually determined by the user and the right choice of this parameter is a key issue for this algorithm. The *MinPts* is the minimal number of neighboring points belonging to a so-called *core point*.

Definition 1. The *eps-neighborhood* of point $p \in X$ is called $N_{eps}(p)$ and is defined as follows: $N_{eps}(p) = \{q \in X | dist(p,q) \leq eps\}$, where $dist(p,q)$ is a distance function between p and q.

When a number of points belonging to the *eps-neighborhood* of p is greater or equal to the *MinPts*, p is called the *core point*.

Definition 2. Point p is *directly density-reachable* from point q with respect to *epsilon* and the *MinPts* when q is a *core point* and p belongs to the *eps-neighborhood* of q.

When point p is *directly density-reachable* from point q and a number of points belonging to the *eps-neighborhood* of p is smaller than the *MinPts*, p is called a *border point*.

Definition 3. Point p is a *noise* if it is neither a *core point* nor a *border point*.

Definition 4. Point p is *density-reachable* from point q with respect to the *eps* and the *MinPts* when there is a chain of points $p_1, p_2, ..., p_n$, $p_1 = q$, $p_n = p$, so that p_{i+1} is *directly density-reachable* from p_i

Definition 5. Point p is *density-connected* to a point q with respect to the *eps* and the *MinPts* when there is point o, so that p and q are *density-reachable* from point o.

Definition 6. Cluster C with respect to the *eps* and the *MinPts* is a non-empty subset of X, where the following conditions are satisfied:

1. $\forall p, q$: if $p \in C$ and q is *density-reachable* from p with respect to the *eps* and the *MinPts*, then $q \in C$.
2. $\forall p, q \in C$: p is *density-connected* to q with respect to the *eps* and the *MinPts*.

The DBSCAN algorithm creates clusters according to Definition 6. At first, point p is selected randomly and if $|N_{eps}(p)| \geq MinPts$, then point p will be the *core point* and will be marked as a new cluster. Next, the new cluster is expanded by the points which are *density-reachable* from p. This process is repeated until no more cluster are found. On the other hand, if $|N_{eps}(p)| < MinPts$, then point p will be considered as a new *noise*. However, this point can be included in another cluster if it is *density-reachable* from some *core point*.

3 The New Approach to Determining the Radius of the Neighborhood

The right choice of the *eps* (the radius of the neighborhood) is a key issue for the right performance of the DBSCAN algorithm. It is a very difficult task and usually, a distance function is used to solve this problem. This distance function is denoted by the k_{dist} and it calculates distances between each element of the X dataset and its k-th nearest neighbor. The number of the nearest neighbors is the k parameter. For instance, Fig. 1 shows an example of a 2-dimensional dataset consisting of four clusters. The clusters contain 138, 119, 127, and 116 elements, respectively. First, the k_{dist} function is used to determine all distances between each element of the X dataset and its k-th

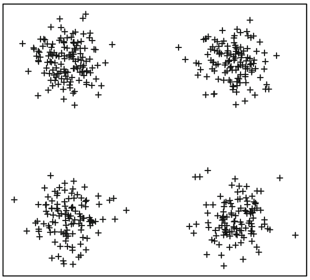


Fig. 1. An example of a 2-dimensional dataset consisting of four clusters.

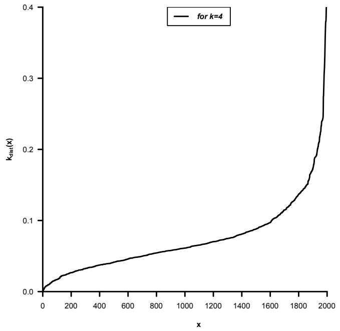


Fig. 2. Sorted values of the k_{dist} function with respect to $k = 4$ for the example dataset.

nearest neighbors. Next, the results are sorted in an ascending order. Figure 2 presents the sorted results for the k=4. It can be observed that there is a sharp change of the distances along the distance curves, i.e. values of the distances increase significantly. This place is called the "*knee*" and it can be used to determine the right value of the *eps* parameter of the DBSCAN algorithm. However, even when the "*knee*" is found correctly, the determination of the *eps* parameter is still difficult.

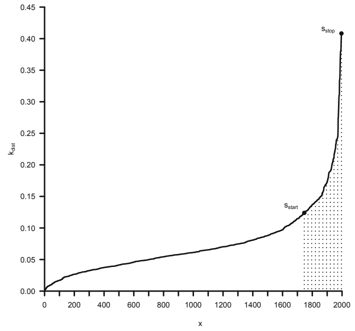


Fig. 3. Sorted values of the k_{dist} function with respect to $k = 4$ and ten identical intervals between the s_{start} and s_{stop} points.

A new approach is proposed to solve this problem. This method is a modification of the approach presented in the article [28] and it consists of a few steps. Let us denote a set of all the sorted values of k_{dist} function by S_{dist} for the X dataset. As mentioned above, the *eps* parameter depends on the "*knee*" occurring in the sorted distances (in an ascending order) for the given k parameter. It should be noted that in Fig. 2 the values of the k_{dist} function increase very abruptly when they are to the right of the "*knee*". This means that there are elements of the dataset located outside clusters and they can be interpreted as noise. Moreover, the "*knee*" usually appears at the end of the sorted values of the k_{dist} function and its size depends on the properties of the dataset.

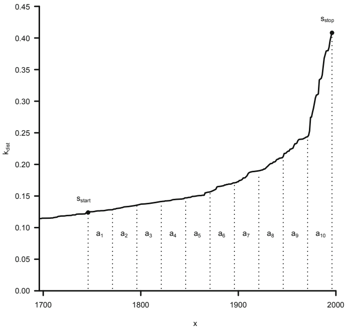


Fig. 4. Ten identical intervals between the s_{start} and s_{stop} points: $a1, ..., a10$.

Among the sorted values, it is possible to define a range, which indicates the *knee* more precisely. It can be determined by the s_{start} and s_{stop} values which are calculated as follows:

$$\begin{aligned} s_{start} &= |S_{dist}| - |X| \\ s_{stop} &= |S_{dist}| \end{aligned} \tag{1}$$

where $|S_{dist}|$ is the number of the elements of S_{dist} and $|X|$ is the number of the elements of the X dataset. Furthermore, the $s_{stop} - s_{start}$ range is divided into ten equal parts, i.e.: $a1$, $a2$, ... $a10$. The size of such part is $n = |X|/10$. For example, Fig. 3 shows the sorted values of the k_{dist} function with respect to $k = 4$ and ten identical intervals between the s_{start} and s_{stop} points. Moreover, in Fig. 4 the ten identical intervals between the s_{start} and s_{stop} points are presented more precisely. Next, for the first seven intervals, the arithmetic means are calculated which is expressed as follows:

$$v_i = \frac{k_{dist}(x_i) + k_{dist}(x_i + n)}{2} \tag{2}$$

where $i = 1, ..., 7$, n is a constant value and it equals the number of $k_{dist}(x_i)$ values occurring in each part, i.e.: $a1, ..., a10$. x_i is the parameter of $k_{dist}(x_i)$ function (see Fig. 4) and so x_1 equals the component x of the s_{start} point located at the start of the a_1 interval. Furthermore, x_2 is equal to $x_1 + n$, x_3 is equal to $x_2 + n$, and so on. However, for intervals $a8$, $a9$ and $a10$, the arithmetic means are calculated as follows:

$$v_j = \frac{k_{dist}(x_j) + k_{dist}(x_{10})}{2} \tag{3}$$

where $j = 8, 9$ and 10. x_{10} equals component x of the s_{stop} point located at the end of the a_{10} interval. In Fig. 5 the average values of all the intervals are presented. These average values (see Eq. (2) and Eq. (3)) are used to calculate the *eps* parameter. It should be noted that the "*knee*" can be analyzed by these calculated average values. First, two factors $v_{7:1}$ and $v_{8:7}$ are computed and they can be expressed as follows:

$$v_{7:1} = \frac{v_7 - v_1}{2} \qquad v_{8:7} = \frac{v_8 - v_7}{2} \tag{4}$$

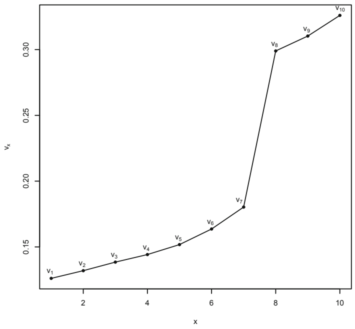


Fig. 5. Average values calculated for all intervals (i.e. $a1, ..., a10$).

These factors play a key role in the analysis of the "*knee*". For instance, when the values of k_{dist} increase very slowly in $a1, ..., a7$, average value $v_{7:1}$ does not change significantly, either. It means that the "*knee*" can be quite wide. Furthermore, if the values of the k_{dist} function increase abruptly in $a8, ..., a10$, then average values $v_{8:7}$ will have a large value. Thus, these factors can be used to calculate the *eps* parameter and it can be expressed as follows:

$$eps = v_{8-7} - v_{7-1} \tag{5}$$

It should be noted that v_{8-7} is close to the right value of the *eps* parameter. However, the v_{7-1} is very important because it shows how the distances increase in the "*knee*" and it is used to correct the value of v_{8-7} (see Eq. (5)).

In the next section, the results of the experimental study is presented to confirm the effectiveness of this new approach.

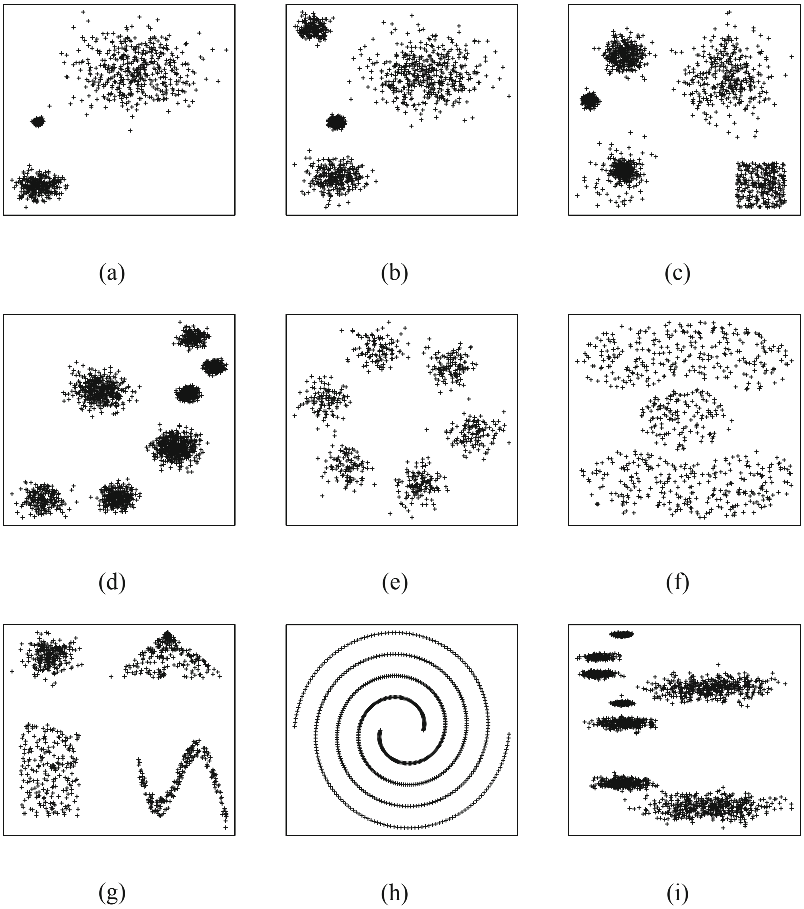
4 Experimental Study

In this section, several experiments have been conducted on 2-dimensional artificial datasets using the original *DBSCAN* algorithm. It should be noted that this algorithm can recognize clusters with arbitrary shapes. In these conducted experiments are used artificial datasets that include clusters of various sizes and shapes. Moreover, parameter k (i.e. *MinPts*) is equal to 4 in all the experiments and the visual inspection is used for the evaluation of the accuracy of the DBSCAN algorithm. The described new method is used to automatically determine the *eps* parameter.

It can be noted that the DBSCAN algorithm is rarely used to cluster multidimensional data due to the so-called "*curse of dimensionality*". However, different modifications of this algorithm have been proposed to solve the problem (e.g. [5]).

4.1 Datasets

Nine 2-dimensional datasets are used in the experiments and they are called *Data* 1, *Data* 2, *Data* 3, *Data* 4, *Data* 5, *Data* 6, *Data* 7, *Data* 8 and *Data* 9, respectively. Table 1



(a) (b) (c)

(d) (e) (f)

(g) (h) (i)

Fig. 6. Examples of 2-dimensional artificial datasets: (a) *Data* 1, (b) *Data* 2, (c) *Data* 3, (d) *Data* 4, (e) *Data* 5, (f) *Data* 6, (g) *Data* 7, (h) *Data* 8 and (i) *Data* 9.

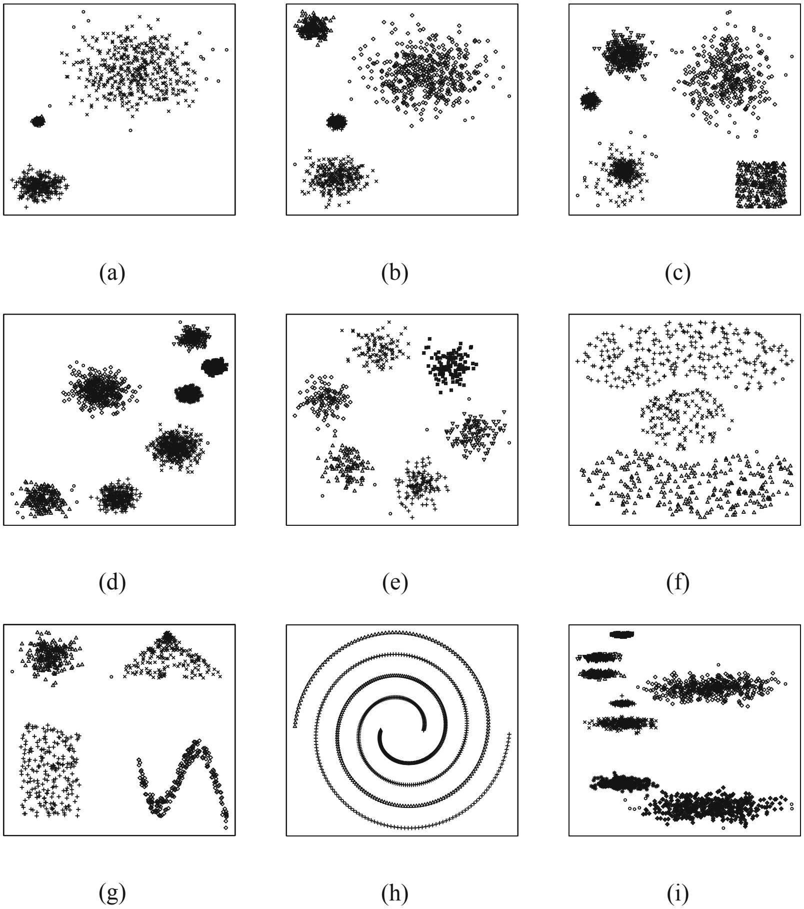
shows a detailed description of these datasets. It should be noted that they contain varied numbers of elements and clusters (from 2 to 8 clusters). Moreover, the shapes and sizes of the clusters are also different. In Fig. 6, these datasets are presented. It can be observed that the distances between the clusters are very different, some of the clusters are very close and others quite far. For instance, in *Data* 3 the elements create five clusters with different sizes, *Data* 7 contains elements which create Gaussian, square, triangle, and wave shapes, and *Data* 8 is so-called the spirals problem, where points are on two entangled spirals.

Table 1. A detailed description of the artificial datasets

Datasets	No. of elements	Clusters
Data 1	1000	3
Data 2	1200	4
Data 3	1800	5
Data 4	2300	7
Data 5	700	6
Data 6	700	3
Data 7	900	4
Data 8	700	2
Data 9	2400	8

4.2 Experiments

In this section, the evaluation of the performance of the new method to automatically specify the *eps* parameter is presented. As mentioned above, the *eps* parameter is very important because the *DBSCAN* algorithm bases on this parameter to create the right clusters. It is usually determined by visual inspection of the sorted values of the k_{dist} function. On the other hand, the new method described in Sect. 3 allows us to determine this parameter in an automatic way. The second parameter of the *DBSCAN*, i.e. the *MinPts* is also important but it is often chosen experimentally. In all the conducted experiments the *MinPts* equals 4. Such a choice of the parameter guarantees the creation of various clusters with different numbers of elements. In these experiments the 2-dimensional artificial datasets are used, i.e.: *Data* 1, *Data* 2, *Data* 3, *Data* 4, *Data* 5, *Data* 6, *Data* 7, *Data* 8 and *Data* 9 sets. Thus, when the *eps* parameter is specified by the new method, the *DBSCAN* is used to cluster the artificial datasets. Figure 7 shows the results of the *DBSCAN* algorithm, where each cluster is marked with different signs. It should be noted that despite the fact that the differences of distances and shapes between clusters are significant, all the datasets are clustered correctly by the clustering algorithm. Moreover, the data elements classified as the *noise* are marked with a circle, and their number is small in all the datasets.



(a) (b) (c)

(d) (e) (f)

(g) (h) (i)

Fig. 7. Results of the *DBSCAN* clustering algorithm for 2-dimensional datasets: (a) *Data* 1, (b) *Data* 2, (c) *Data* 3, (d) *Data* 4, (e) *Data* 5, (f) *Data* 6, (g) *Data* 7, (h) *Data* 8 and (i) *Data* 9

5 Conclusions

In this paper, a new method is proposed to determine the *eps* parameter of the DBSCAN algorithm. This method bases on the k_{dist} function, which computes the distance between each element of a dataset and its *k*th nearest neighbor. Furthermore, the calculated distances are sorted in an ascending order to find out the *knee*. Next, the distances creating the *knee* are divided into several intervals and they are used to calculate the mean values (see Eq. (4)). This makes it possible to calculate the right value of the *eps* parameter. In the conducted experiments, several 2-dimensional datasets were used, where the number of clusters, sizes, and shapes varied within a wide range. From the perspective of the conducted experiments, this method for computing *eps* is very useful. All the presented results confirm a high efficiency of the newly proposed approach.

References

1. Bilski, J., Smoląg, J.: Parallel architectures for learning the RTRN and Elman dynamic neural networks. IEEE Trans. Parallel Distrib. Syst. **26**(9), 2561–2570 (2015)
2. Bilski, J., Kowalczyk, B., Marchlewska, A., Zurada, J.M.: Local levenberg-marquardt algorithm for learning feedforward neural networks. J. Artif. Intell. Soft Comput. Res. **10**(4), 299–316 (2020)
3. Boonchoo, T., Ao, X., Liu, Y., Zhao, W., He, Q.: Grid-based DBSCAN: indexing and inference. Pattern Recogn. **90**, 271–284 (2019)
4. Bradley P., Fayyad U.: Refining initial points for k-means clustering. In: Proceedings of the Fifteenth International Conference on Knowledge Discovery and Data Mining, New York, AAAI Press, pp. 9–15 (1998)
5. Chen, Y., Tang, S., Bouguila, N., Wanga, C., Du, J., Li, H.: A fast clustering algorithm based on pruning unnecessary distance computations in DBSCAN for high-dimensional data. Pattern Recogn. **83**, 375–387 (2018)
6. Darong H., Peng W.: Grid-based dbscan algorithm with referential parameters. Phys. Procedia **24**, Part B, 1166–1170 (2012)
7. Dziwiński, P., Bartczuk, Ł, Paszkowski, J.: A new auto adaptive fuzzy hybrid particle swarm optimization and genetic algorithm. J. Artif. Intell. Soft Comput. Res. **10**(2), 95–111 (2020)
8. Ester, M., Kriegel, H.P., Sander, J., Xu, X.: A density-based algorithm for discovering clusters in large spatial databases with noise. In: Proceeding of 2nd International Conference on Knowledge Discovery and Data Mining, pp. 226–231 (1996)
9. Ferdaus, M., Anavatti, S.G., Garratt, M.A., Pratama, M.: Development of C-means clustering based adaptive fuzzy controller for a flapping wing micro air vehicle. J. Artif. Intell. Soft Comput. Res. **9**(2), 99–109 (2019)
10. Fränti, P., Rezaei, M., Zhao, Q.: Centroid index: cluster level similarity measure. Pattern Recogn. **47**(9), 3034–3045 (2014)
11. Gabryel, M.: Data analysis algorithm for click fraud recognition. Commun. Comput. Inf. Sci. **920**, 437–446 (2018)
12. Gałkowski, T., Krzyak, A., Filutowicz, Z.: A new approach to detection of changes in multidimensional patterns. J. Artif. Intell. Soft Comput. Res. **10**(2), 125–136 (2020)
13. Grycuk, R., Najgebauer, P., Kordos, M., Scherer, M., Marchlewska, A.: Fast image index for database management engines. J. Artif. Intell. Soft Comput. Res. **10**(2), 113–123 (2020)
14. Hruschka E.R., de Castro L.N., Campello R.J.: Evolutionary algorithms for clustering gene-expression data, In: Fourth IEEE International Conference on Data Mining, 2004, ICDM 2004, pp. 403–406. IEEE (2004)
15. Jain, A., Dubes, R.: Algorithms for clustering data. Prentice-Hall, Englewood Cliffs (1988)
16. Karami, A., Johansson, R.: Choosing DBSCAN parameters automatically using differential evolution. Int. J. Comput. Appl. **91**, 1–11 (2014)
17. Luchi, D., Rodrigues, A.L., Varejao, F.M.: Sampling approaches for applying DBSCAN to large datasets. Pattern Recogn. Lett. **117**, 90–96 (2019)
18. Meng X., van Dyk D.: The EM algorithm - An old folk-song sung to a fast new tune. J. Royal Stat. Soc. Series B (Methodological) **59**(3), 511–567 (1997)
19. Murtagh, F.: A survey of recent advances in hierarchical clustering algorithms. Comput. J. **26**(4), 354–359 (1983)
20. Nowicki, R., Grzanek, K., Hayashi, Y.: Rough support vector machine for classification with interval and incomplete data. J. Artif. Intell. Soft Comput. Res. **10**(1), 47–56 (2020)
21. Patrikainen, A., Meila, M.: Comparing subspace clusterings. IEEE Trans. Knowl. Data Eng. **18**(7), 902–916 (2006)

22. Rohlf, F.: Single-link clustering algorithms. In: Krishnaiah, P.R., Kanal, L.N. (eds.), Handbook of Statistics, vol. 2, pp. 267–284 (1982)
23. Sameh, A.S., Asoke, K.N.: Development of assessment criteria for clustering algorithms. Pattern Anal. Appl. **12**(1), 79–98 (2009)
24. Shah, G.H.: An improved dbscan, a density based clustering algorithm with parameter selection for high dimensional data sets. In: Nirma University International Engineering, (NUiCONE), pp. 1–6 (2012)
25. Sheikholeslam, G., Chatterjee, S., Zhang, A.: WaveCluster: a wavelet-based clustering approach for spatial data in very large databases. Int. J. Very Large Data Bases **8**(3–4), 289–304 (2000)
26. Shieh, H.-L.: Robust validity index for a modified subtractive clustering algorithm. Appl. Soft Comput. **22**, 47–59 (2014)
27. Starczewski, A.: A new validity index for crisp clusters. Pattern Anal. Appl. **20**(3), 687–700 (2017)
28. Starczewski, A., Cader, A.: Determining the Eps parameter of the DBSCAN algorithm. In: Rutkowski, L., Scherer, R., Korytkowski, M., Pedrycz, W., Tadeusiewicz, R., Zurada, J.M. (eds.) ICAISC 2019. LNCS (LNAI), vol. 11509, pp. 420–430. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-20915-5_38
29. Starczewski, J., Goetzen, P., Napoli, C.: Triangular fuzzy-rough set based fuzzification of fuzzy rule-based systems. J. Artif. Intell. Soft Comput. Res. **10**(4), 271–285 (2020)
30. Wang, W., Yang, J., Muntz, R.: STING: a statistical information grid approach to spatial data mining. In: Proceedings of the 23rd International Conference on Very Large Data Bases, VLDB 1997, pp. 186–195 (1997)
31. Viswanath, P., Suresh Babu, V.S.: Rough-dbscan: a fast hybrid density based clustering method for large data sets. Pattern Recogn. Lett. **30**(16), 1477–1488 (2009)
32. Zalik, K.R.: An efficient k-means clustering algorithm. Pattern Recogn. Lett. **29**(9), 1385–1391 (2008)