Determining the Eps Parameter of the DBSCAN Algorithm

Artur Starczewski[1](✉) and Andrzej Cader[2,3]

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

[2] Information Technology Institute, University of Social Sciences, 90-113 Łódź, Poland
acader@san.edu.pl

[3] Clark University, Worcester, MA 01610, USA

Abstract. Clustering is an attractive technique used in many fields and lots of clustering algorithms have been proposed so far. The Density-Based Spatial Clustering of Applications with Noise (DBSCAN) is one of the most popular algorithms, which has been widely applied in many different applications. This algorithm can discover clusters of arbitrary shapes in large datasets. However, the fundamental issue is the right choice of two input parameters, i.e. radius *eps* and density threshold $MinPts$. In this paper, a new method is proposed to determine the value of *eps*. The suggested approach is based on an analysis of the sorted values of the distance function. The performance of the new approach has been demonstrated for several different datasets.

Keywords: Clustering algorithms · DBSCAN · Data mining

1 Introduction

Data clustering is one of the most important approaches used to discover naturally occurring structures in a dataset. Clustering is a process, which refers to grouping objects into meaningful clusters so that the elements of a cluster are similar, whereas they are dissimilar in different clusters. Nowadays, a variety of large collections of data are created. This brings a great challenge for clustering algorithms, so new different clustering algorithms and their configurations are being intensively developed, e.g. [12–15,26]. Data clustering is applied in many areas, such as biology, spatial data analysis, business, and others. It should be noted that there is no a clustering algorithm, which creates the right data partition 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 results of data clustering. So far, a number of authors have proposed different cluster validity indices or modifications of existing ones, e.g., [11,25,29–31].

L. Rutkowski et al. (Eds.): ICAISC 2019, LNAI 11509, pp. 420–430, 2019.
https://doi.org/10.1007/978-3-030-20915-5_38

Among clustering algorithms four categories can be distinguished: partitioning, hierarchical, grid-based and density-based clustering. For example, the well-known partitioning algorithms are, e.g. *K-means*, *Partitioning Around Medoids* (*PAM*) [6,34] and *Expectation Maximization* (*EM*) [19], whereas the hierarchical clustering includes agglomerative and divisive approaches, e.g. the *Single-linkage*, *Complete-linkage* or *Average-linkage* or *DIvisive ANAlysis Clustering* (*DIANA*) [20,24]. Then, the grid-based approach includes methods such as e.g. the *Statistical Information Grid-based* (*STING*) or *Wavelet-based Clustering* (*WaveCluster*) [21,28,32]. The next category of clustering algorithms is the density-based approach. The *Density Based Spatial Clustering of Application with Noise* (*DBSCAN*) is the most famous density-based algorithm [10]. It can discover clusters of an arbitrary shape and size, but is rarely used to cluster multidimensional data due to so-called "*curse of dimensionality*". Consequently, it has many extensions, e.g. [5,7,8,18,27,33]. However, the *DBSCAN* also requires two input parameters, i.e. radius *eps* and density threshold *MinPts*. Determination of these parameters is crucial to the right performance of this clustering method. Especially the *eps* radius is very difficult to be determined correctly. Recently, new concepts have been proposed for the determining of these input parameters [16]. It is important to note that clustering methods can be used during a process of designing various neural networks [1–3], fuzzy and rule systems [4,9,17,22,23].

In this paper, a new approach to determining the *eps* radius is proposed. It is based on an analysis of a *knee*, which appears in the sorted values of the distance function used in the dataset. 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 on datasets. Finally, Sect. 5 presents conclusions.

2 The Concept of the DBSCAN Clustering 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* radius and the density threshold *MinPts*. To understand the basic concept of the algorithm several terms should be explained. Let us denote the *eps* radius by ε and a dataset by X, where a point $p \in X$. The ε 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 ε-*neighborhood* of point $p \in X$ is called $N_\varepsilon(p)$ and is defined as follows: $N_\varepsilon(p) = \{q \in X | dist(p,q) \leq \varepsilon\}$, where $dist(p,q)$ is a distance function between p and q.

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

Definition 2: A point p is *directly density-reachable* from a point q with respect to ε and the $MinPts$ when q is a *core point* and p belongs to the *ε-neighborhood* of q.

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

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

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

Definition 5: A point p is *density-connected* to a point q with respect to the ε and the $MinPts$ when there is a point o such that p and q are *density-reachable* from the point o.

Definition 6: A cluster C with respect to the ε 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 ε and the $MinPts$, then $q \in C$.
2. $\forall p, q \in C$: p is *density-connected* to q with respect to the ε and the $MinPts$.

The DBSCAN algorithm creates clusters according to Definition 6. At first, a point p is selected randomly and if $|N_\varepsilon(p)| \geq MinPts$ than the 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 cluster found. On the other hand, if $|N_\varepsilon(p)| < MinPts$, then the 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 Explanation of the New Approach to Determine the Eps Parameter

As mentioned above the right choice of the *eps* (ε) parameter is a fundamental issue for a high performance of the DBSCAN. However, it is a very difficult

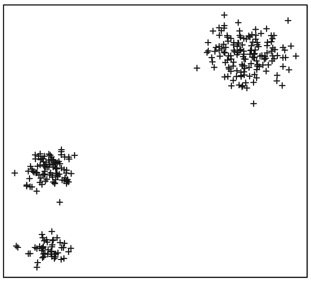

Fig. 1. An example of 2-dimensional dataset consisting of three clusters.

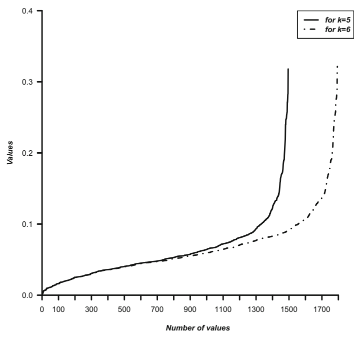

Fig. 2. Sorted values of the k_{dist} function with respect to k = 5 and k = 6 for the dataset.

task. One of the most popular ideas bases on a distance function which computes a distance between each point $p \in X$ and its k-th nearest neighbor. This function can be denoted by k_{dist}. It requires an input parameter k, which is the number of the nearest neighbors of the point p. For instance, Fig. 1 shows an example of a 2-dimensional dataset consisting of three clusters. The clusters contain 150, 100 and 50 elements, respectively. The k_{dist} function can be used in the dataset, and then the results are sorted in an ascending order. Figure 2 presents the sorted results for two values of the k parameter, i.e. k = 5 and k = 6. It can be observed that the plots include a "*knee*", where the distances change significantly. Moreover, the number of calculated distances for k=6 is greater than for k = 5, because additional distances have to be calculated. The fundamental issue is the appropriate determining of a *threshold point*, which defines the maximal k_{dist} value in the clusters of a dataset. All the points with k_{dist} values higher than the maximal k_{dist} value are considered to be a noise. It can be noted that the *threshold point* refers to the "*knee*" but it is very difficult to determine this point correctly. To solve this problem, a new method, which

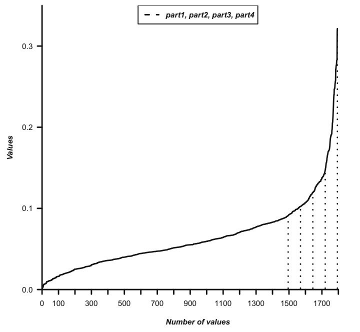

Fig. 3. Partition of the *range* for four equal parts: *part*1, *part*2, *part*3 and *part*4.

consists of a few steps, is proposed. Let us denote a set of all the sorted values of k_{dist} function for the X dataset by V_{sdist}. The ε (*eps*), mentioned above, depends on the *threshold point* occurring in the dataset for the given k number of the nearest neighbors of the point. It should be noted that in Fig. 2 the values of the k_{dist} function increase abruptly when they are to the right of the "*knee*". This means that there are points beyond the *threshold point* of the dataset and they can be interpreted as a 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. Among the sorted values, it is possible to determine a *range*, which indicates the *knee* more precisely. It can be defined by v_{start} and v_{stop} points as follows:

$$\begin{aligned} v_{start} &= |V_{sdist}| - |X| \\ v_{stop} &= |V_{sdist}| \end{aligned} \tag{1}$$

where $|V_{sdist}|$ is the number of the elements of the V_{sdist} and $|X|$ is the number of the elements of the X. Furthermore, the *range* is divided into four equal parts, i.e. *part*1, *part*2, *part*3 and *part*4. The size of such part is equal to $|X|/4$. For example, Fig. 3 shows the partition of the *range* for four equal parts. Next, for each of the parts the average values are calculated as follows:

$$\begin{aligned} S_{v1} &= \frac{1}{n}\sum_{i=v1}^{n} k_{dist}(i) \\ S_{v2} &= \frac{1}{n}\sum_{i=v2}^{n} k_{dist}(i) \\ S_{v3} &= \frac{1}{n}\sum_{i=v3}^{n} k_{dist}(i) \\ S_{v4} &= \frac{1}{n}\sum_{i=v4}^{n} k_{dist}(i) \end{aligned} \tag{2}$$

where n is the number of the k_{dist} values occurring in each part and v_j is the start point of the jth part, $j = 1..4$. These start points are defined as follows: $v1 = v_{start}$, $v2 = v1 + n$, $v3 = v2 + n$ and $v4 = v3 + n$. Next, the "*knee*" can be analyzed by these calculated averages values. First, three factors a, b and c are computed. They can be expressed as follows:

$$a = \frac{S_{v2}}{S_{v1}} \quad b = \frac{S_{v3}}{S_{v2}} \quad c = \frac{S_{v4}}{S_{v3}} \tag{3}$$

These factors play a key role in the analysis of the "*knee*". For instance, when the values of the k_{dist} increase very slowly in *part*1, *part*2 and *part*3, the average values S_{v1}, S_{v2}, S_{v3} also do not change significantly. Thus, the values of a and b are almost equal. Furthermore, if values of the k_{dist} function increase abruptly in *part*4, then parameter c will have a large value. In this case, the ε should equal S_{v4} because there is a high probability that the S_{v4} refers to the *threshold point* occurring in this dataset so it can be expressed as:

$$\begin{gathered} if(a \approx b) \ \wedge \ (c \geq T) \ \ then \\ \varepsilon = S_{v4} \end{gathered} \tag{4}$$

where T is a *constant value* and is determined experimentally ($T = 1.4$). On the other hand, when $c < T$, the values of the k_{dist} increase slowly in $part4$ and the "*knee*" can be quite wide, so smaller values of the k_{dist} function refer to the *threshold point*. In this case, the ε should be calculated with respect to $part2$, $part3$ and $part4$ so it is defined as follows:

$$\begin{gathered} if(a \approx b) \ \wedge \ (c < T) \ \ then \\ \varepsilon = \frac{S_{v2} + S_{v3} + S_{v4}}{3} \end{gathered} \tag{5}$$

Next, when the values of the k_{dist} increase significantly in $part1$, $part2$ and $part3$, the values of a and b are different, i.e. the b is much bigger than the a factor. It means that the *threshold point* refers to the values from $part3$ and $part4$. Thus, the ε can be defined as follows:

$$\begin{gathered} if(a \neq b) \ \ then \\ \varepsilon = \frac{S_{v3} + S_{v4}}{2} \end{gathered} \tag{6}$$

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

4 Experimental Results

In this section, several experiments have been conducted on 2-dimensional artificial datasets using the original $DBSCAN$ algorithm. This algorithm is one of most popular clustering methods, because it can recognize clusters with arbitrary shapes. Artificial datasets include clusters of various sizes and shapes. The first

Table 1. A detailed description of the artificial datasets

Datasets	No. of elements	Clusters
Data 1	75	4
Data 2	500	4
Data 3	700	6
Data 4	700	3
Data 5	900	4
Data 6	500	4
Data 7	700	2
Data 8	2300	7
Data 9	3000	3

parameter k ($MinPts$) equaled 6 in all the experiments. Such value of the k guarantees that the algorithm does not create clusters of too low a density threshold. To automatically determine the radius ε, the new approach described above is used. Moreover, the evaluation of the accuracy the DBSCAN algorithm is conducted by visual inspection. It can be noted that this algorithm is rarely used to cluster multidimensional data due to the so-called "*curse of dimensionality*". Recently, however, a modification of this algorithm has been proposed to solve this problem [7].

4.1 Datasets

In the conducted experiments nine 2-dimensional datasets are used. Most of them come from the *R* package. The artificial data are called *Data* 1, *Data* 2, *Data* 3, *Data* 4, *Data* 5, *Data* 6, *Data* 7, *Data* 8 and *Data* 9, respectively.

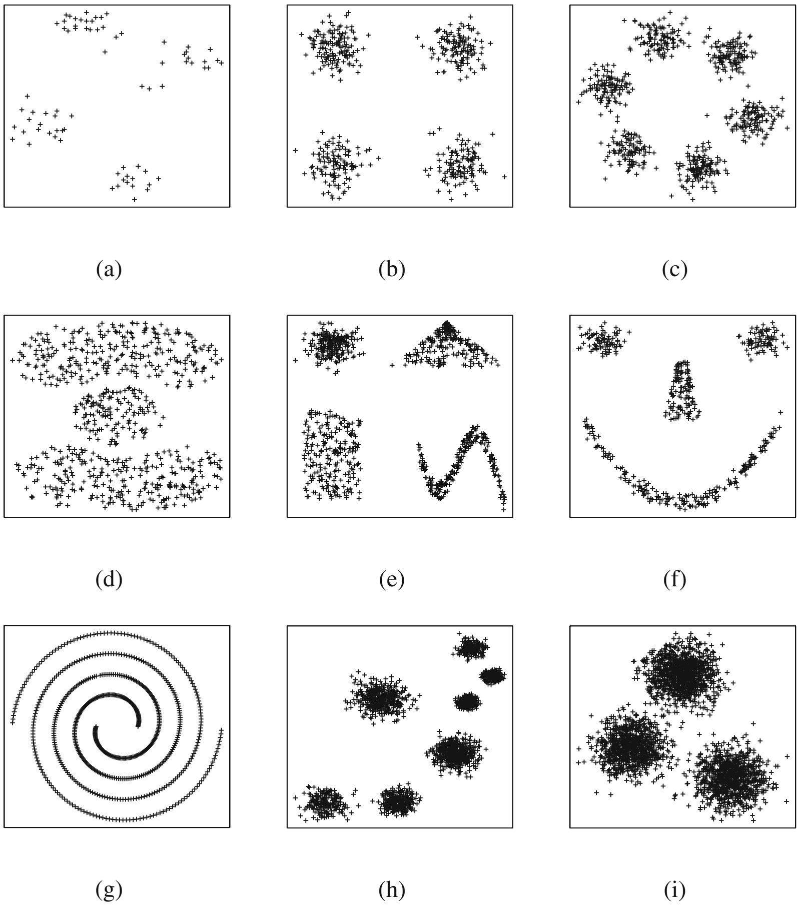

Fig. 4. Examples of 2-dimensional artificial datasets: (a) *Data* 1, (b) *Data* 2, (c) *Data* 3, (d) *Data* 4, (e) *Data* 5, (f) *Data* 6, (g) *Data* 7, (h) *Data* 8 and (i) *Data* 9.

They consist of various number of clusters, i.e. 2, 3, 4, 6, and 7 clusters. The scatter plot of these data is presented in Fig. 4. As it can be observed on the plot, the distances between the clusters are very different and some clusters are quite close. Generally, the clusters are located in different areas and some of the clusters are very close and others quite far. For instance, in *Data* 5 the elements create Gaussian, square, triangle and wave shapes, *Data* 6 consists of 2 Gaussian eyes, a trapezoid nose and a parabola mouth (with a vertical Gaussian one) and *Data* 7 is so-called the spirals problem, where points are on two entangled spirals. Moreover, the sizes of the clusters are different and they contain a various number of elements. Table 1 shows a detailed description of these datasets used in the experiments.

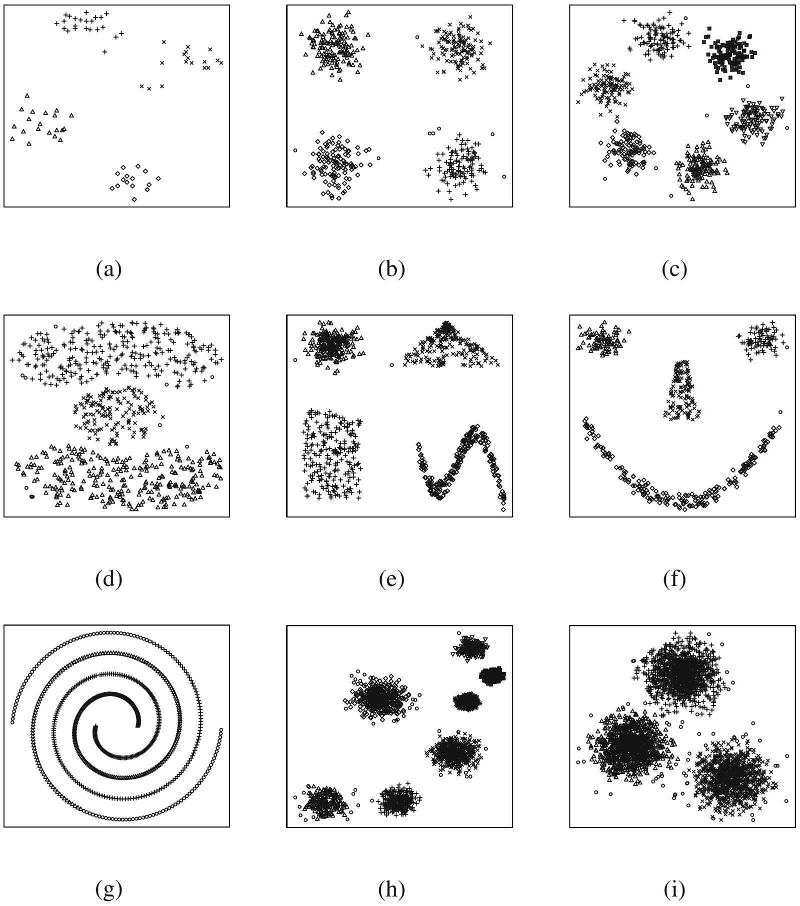

Fig. 5. Results of the *DBSCAN* clustering algorithm for 2-dimensional datasets: (a) *Data* 1, (b) *Data* 2, (c) *Data* 3, (d) *Data* 4, (e) *Data* 5, (f) *Data* 6, (g) *Data* 7, (h) *Data* 8 and (i) *Data* 9

4.2 Experiments

The experimental analysis is designed to evaluate the performance of the new method to automatically specify the ε parameter. As mentioned above, this parameter is very important for the $DBSCAN$ algorithm to work correctly and it is usually determined by visual inspection of the sorted values of the k_{dist} function. On the other hand, the new approach described in Sect. 3 allows us to determine this parameter in an automatic way. In these experiments the nine 2-dimensional datasets used are called $Data$ 1, $Data$ 2, $Data$ 3, $Data$ 4, $Data$ 5, $Data$ 6, $Data$ 7, $Data$ 8 and $Data$ 9 datasets. It needs to be noted that the value of the k ($MinPts$) parameter equals 6 in all the experiments. Then, when the ε parameter is specified by the new method, the $DBSCAN$ algorithm can be used to cluster these datasets. Figure 5 shows the results of the $DBSCAN$ algorithm, where each cluster is marked with different signs. The data elements classified as the *noise* are marked with a circle. It should be noted that the new approach provides correct values of the ε in all the experiments. Thus, despite the fact that the differences of distances and shapes between clusters are significant, all the datasets are clustered correctly by the $DBSCAN$. Moreover, the number of data elements classified as noise in all the datasets is small.

5 Conclusions

In this paper a new method is proposed for computing the ε parameter for the DBSCAN algorithm. This method uses the k_{dist} function, which computes the distance between each point $p \in X$ and its kth nearest neighbor. The fundamental issue is to correctly determine the *threshold point*, which defines the maximal k_{dist} value in the dataset clusters. To solve this problem, first, the new method finds out the *region* of the k_{dist} values creating the *knee* and divides it into four parts. Next, average values of these parts are determined. This makes it possible to calculate the right value of ε. 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 automatic way to compute ε is useful and easy. All the presented results confirm a very 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., Wilamowski, B.M.: Parallel learning of feedforward neural networks without error backpropagation. In: Rutkowski, L., Korytkowski, M., Scherer, R., Tadeusiewicz, R., Zadeh, L.A., Zurada, J.M. (eds.) ICAISC 2016. LNAI, vol. 9692, pp. 57–69. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-39378-0_6

3. Bilski, J., Kowalczyk, B., Grzanek, K.: The parallel modification to the Levenberg-Marquardt algorithm. In: Rutkowski, L., Scherer, R., Korytkowski, M., Pedrycz, W., Tadeusiewicz, R., Zurada, J.M. (eds.) ICAISC 2018. LNCS, vol. 10841, pp. 15–24. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-91253-0_2
4. Bologna, G., Hayashi, Y.: Characterization of symbolic rules embedded in deep DIMLP networks: a challenge to transparency of deep learning. J. Artif. Intell. Soft Comput. Res. **7**(4), 265–286 (2017)
5. Boonchoo, T., Ao, X., Liu, Y., Zhao, W., He, Q.: Grid-based DBSCAN: indexing and inference. Pattern Recogn. **90**, 271–284 (2019)
6. Bradley, P., Fayyad, U.: Refining initial points for K-Means clustering. In Proceedings of the Fifteenth International Conference on Knowledge Discovery and Data Mining, pp. 9–15. AAAI Press, New York (1998)
7. 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)
8. Darong, H., Peng, W.: Grid-based DBSCAN algorithm with referential parameters. Phys. Proc. **24**(Part B), 1166–1170 (2012)
9. D'Aniello, G., Gaeta, M., Loia, F., Reformat, M., Toti, D.: An environment for collective perception based on fuzzy and semantic approaches. J. Artif. Intell. Soft Comput. Res. **8**(3), 191–210 (2018)
10. 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)
11. Fränti, P., Rezaei, M., Zhao, Q.: Centroid index: cluster level similarity measure. Pattern Recogn. **47**(9), 3034–3045 (2014)
12. Gabryel, M.: The bag-of-words method with different types of image features and dictionary analysis. J. Univ. Comput. Sci. **24**(4), 357–371 (2018)
13. Gabryel, M.: Data analysis algorithm for click fraud recognition. In: Damaševičius, R., Vasiljevienė, G. (eds.) ICIST 2018. CCIS, vol. 920, pp. 437–446. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-99972-2_36
14. Gabryel, M., Damaševičius, R., Przybyszewski, K.: Application of the bag-of-words algorithm in classification the quality of sales leads. In: Rutkowski, L., Scherer, R., Korytkowski, M., Pedrycz, W., Tadeusiewicz, R., Zurada, J.M. (eds.) ICAISC 2018. LNCS, vol. 10841, pp. 615–622. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-91253-0_57
15. 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, ICDM 2004, pp. 403–406. IEEE (2004)
16. Karami, A., Johansson, R.: Choosing DBSCAN parameters automatically using differential evolution. Int. J. Comput. Appl. **91**, 1–11 (2014)
17. Liu, H., Gegov, A., Cocea, M.: Rule based networks: an efficient and interpretable representation of computational models. J. Artif. Intell. Soft Comput. Res. **7**(2), 111–123 (2017)
18. Luchi, D., Rodrigues, A.L., Varejao, F.M.: Sampling approaches for applying DBSCAN to large datasets. Pattern Recogn. Lett. **117**, 90–96 (2019)
19. Meng, X., van Dyk, D.: The EM algorithm - an old folk-song sung to a fast new tune. J. Roy. Stat. Soc. Ser. B (Methodol.) **59**(3), 511–567 (1997)
20. Murtagh, F.: A survey of recent advances in hierarchical clustering algorithms. Comput. J. **26**(4), 354–359 (1983)
21. Patrikainen, A., Meila, M.: Comparing subspace clusterings. IEEE Trans. Knowl. Data Eng. **18**(7), 902–916 (2006)

22. Prasad, M., Liu, Y.-T., Li, D.-L., Lin, C.-T., Shah, R.R., Kaiwartya, O.P.: A new mechanism for data visualization with TSK-type preprocessed collaborative fuzzy rule based system. J. Artif. Intell. Soft Comput. Res. **7**(1), 33–46 (2017)
23. Riid, A., Preden, J.-S.: Design of fuzzy rule-based classifiers through granulation and consolidation. J. Artif. Intell. Soft Comput. Res. **7**(2), 137–147 (2017)
24. Rohlf, F.: Single-link clustering algorithms. In: Krishnaiah, P.R., Kanal, L.N. (eds.) Handbook of Statistics, vol. 2, pp. 267–284 (1982)
25. Sameh, A.S., Asoke, K.N.: Development of assessment criteria for clustering algorithms. Pattern Anal. Appl. **12**(1), 79–98 (2009)
26. Serdah, A.M., Ashour, W.M.: Clustering large-scale data based on modified affinity propagation algorithm. J. Artif. Intell. Soft Comput. Res. **6**(1), 23–33 (2016). https://doi.org/10.1515/jaiscr-2016-0003
27. 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)
28. 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)
29. Shieh, H.-L.: Robust validity index for a modified subtractive clustering algorithm. Appl. Soft Comput. **22**, 47–59 (2014)
30. Starczewski, A.: A new validity index for crisp clusters. Pattern Anal. Appl. **20**(3), 687–700 (2017)
31. Starczewski, A., Krzyżak, A.: A modification of the Silhouette index for the improvement of cluster validity assessment. In: Rutkowski, L., Korytkowski, M., Scherer, R., Tadeusiewicz, R., Zadeh, L.A., Zurada, J.M. (eds.) ICAISC 2016. LNCS, vol. 9693, pp. 114–124. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-39384-1_10
32. 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)
33. 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)
34. Zalik, K.R.: An efficient K-Means clustering algorithm. Pattern Recogn. Lett. **29**(9), 1385–1391 (2008)