Clustering Algorithms for Spatial Big Data

Gabriella Schoier(✉) and Caterina Gregorio

DEAMS – Department of Economic, Business, Mathematic and Statistical Sciences "Bruno de Finetti", University of Trieste, Tigor, 22, 34100 Trieste, Italy
gabriella.schoier@deams.units.it

Abstract. In our time people and devices constantly generate data. User activity generates data about needs and preferences as well as the quality of their experiences in different ways: i.e. streaming a video, looking at the news, searching for a restaurant or a an hotel, playing a game with others, making purchases, driving a car. Even when people put their devices in their pockets, the network is generating location and other data that keeps services running and ready to use. This rapid developments in the availability and access to data and in particular spatially referenced data in a different areas, has induced the need for better analysis techniques to understand the various phenomena. Spatial clustering algorithms, which groups similar spatial objects into classes, can be used for the identification of areas sharing common characteristics. The aim of this paper is to analyze the performance of three different clustering algorithms i.e. the *Density-Based Spatial Clustering of Applications with Noise* algorithm (DBSCAN), the *Fast Search by Density Peak* (FSDP) algorithm and the classic *K-means* algorithm (K-Means) as regards the analysis of spatial big data. We propose a modification of the FSDP algorithm in order to improve its efficiency in large databases. The applications concern both synthetic data sets and satellite images.

Keywords: Spatial data mining · Clustering algorithms · DBSCAN · FSDP · K-Means · Arbitrary shape of clusters · Handling noise · Image analysis

1 Introduction

The rapid developments in the availability and access to spatially referenced information in a variety of areas, has induced the need for better analysis techniques to understand different phenomena. Spatial clustering algorithms, which groups similar spatial objects into classes, can be used for the identification of areas sharing common characteristics.

Clustering is an unsupervised classification of patterns - observations, data items, or feature vectors - into groups or clusters [10]. Cluster analysis can be defined as the organization of a collection of patterns - usually represented as a vector of measurements, or a point in a multidimensional space - into clusters based on similarity [7].

The clustering problem has been considered in many contexts and by researchers in different disciplines. It is useful in several exploratory pattern-analysis, grouping, decision-making and machine-learning situations, including data mining (see e.g. [8]),

DOI: 10.1007/978-3-319-62401-3_41

spatial data mining (see e.g. [1, 4, 14]), document retrieval, image segmentation, and pattern classification.

Clustering techniques have been recognized as primary Data Mining methods for knowledge discovery in spatial databases, i.e. databases managing 2D or 3D points, polygons etc. or points in some d-dimensional feature space (see e.g. [14, 25, 26]).

The aim of this paper is to compare a classical density based algorithm i.e. *the Density-Based Spatial Clustering of Applications with Noise* algorithm (DBSCAN) (see [6, 7, 13, 22–24]) with *the Fast Search by Density Peak* (FSDP) [21] and the classic *K-means* algorithm (K-Means) for the identification of clusters in a Spatial Big Data context. As regards the FSDP algorithm a modification of this algorithm in order to improve its efficiency in the application on Big Data has been proposed. The implementation has been performed in the R language.

2 Some Observations on Spatial Big Data

Spatial data mining can be used for browsing spatial databases, understanding spatial data, discovering spatial relationships, optimizing spatial queries.

Spatial data, as other kinds of data, are becoming bigger and bigger, although since the introduction of GIS and desktop GIS in particular, GIS users and experts have become facing with the issue of managing big amount of data, even though often data were much more difficult to retrieve than today. In geographical terms, the nature of data is such that an increase of dimension of the dataset is always very possible, both in terms of the number of the records to be considered, as well as in terms of the attribute of the geographical data. Both the vector and raster data formats used in GIS analysis tend to be multidimensional, i.e., containing a quantity of elements to be considered in any form of grouping and aggregation. In any case at least two fields (if not three) are needed to store the spatial information while all the attribute data contribute to increasing the dimension of the dataset. Satellite imagery in particular represents another case, in which redundant information is also considered, as very close pixels present very little differences although weighting in the processing, storage and visualization time of the data. So compression algorithms on one side and proficient clustering tools are needed in order to extract the more precise and complete set of geographic information [15–19, 27].

In this paper, the data used for the analysis, have been bit-map images which are real examples of raster spatial analysis. In this case each image is formed by a regular grid of pixels, a color present in every cell of this grid has 24 bit (16 million of colors are possible). The units of analysis are the pixels; we consider both spatial and non-spatial attributes of these units. This analysis, called “Segmentation of images”, is essential in the manipulation, recognition and object-based analysis of multimedia resources [2, 5, 28].

3 The Methodology

In this paper the performance and the computational requirements of three spatial algorithms (see e.g. [20]) are compared in the case of application to spatial Big Data: *the Density-Based Spatial Clustering of Applications with Noise* algorithm (DBSCAN), *the Fast Search by Density Peak* (FSDP) and the classic *K-means* algorithm (K-Means).

K-means and DBSCAN are two well known clustering algorithms while the FSDP clustering algorithm is a density algorithm proposed in 2014 (see [21]).

3.1 DBSCAN Algorithm

In this section we consider clustering methods based on the notion of density. These regard clusters as dense regions of units which are separated by regions of low density (representing noise); moreover they may be used to discover clusters of arbitrary shape (see e.g. [3, 4, 10–12]).

Among these the *DBSCAN* algorithm is a locality-based algorithm relying on a density based notion of clustering. Density based methods can be used to filter out noise (outliers) and discover clusters of arbitrary shape. This algorithm judges the density around the neighborhood on an unit to be sufficiently dense if the number of points within a distance *EpsCoord* of an unit is greater than *MinPts*, in this case the unit is a *core point* otherwise is a *border point*. This algorithm has been generalized in different papers.

The key idea is that for each point of a 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 shape of a *neighborhood* is determined by the choice of a distance function for two points p and q, denoted by *dist(p, q)* The *Epscoord* neighborhood of a point q is defined by

$$N_{\varepsilon}(q) = \{q \in D | dist(p, q) \leq \varepsilon\},$$

where D is a data set of points.

A naive approach could require for each point in a cluster that there are at least a minimum number (*MinPts*) of points in an *Eps-neighborhood* of that point. However, this *ap*proach fails because there are two kinds of points in a cluster, points inside of the cluster (*core points*) and points on the border of the cluster (*border points*).

In general, an *Eps-neighborhood* of a border point contains significantly less points than an *Eps-neighborhood* of a core point. Therefore, one would have to set the minimum number of points to a relatively low value in order to include all points belonging to the same cluster. This value, however, is not characteristic for the respective cluster particularly in the presence of noise. Therefore, one has to require that for every point p in a cluster C there is a point q in C so that p is inside of the *Eps-neighborhood* of q and $N_{\varepsilon}(q)$ contains at least *MinPts* points.

This definition is elaborated in the following: a point p is *density reachable* from a point q if there is a chain of points $p_1, p_{2,\ \ldots}, p_{n-1}, p_n$ where $p_1 = p$ and $p_n = q$ such that p_i is *direct density reachable* from p_{i+1}.

Moreover a point p is *directly density reachable* from a point q if p belongs to the neighborhood of q and q is a core point.

The clustering formed from DBSCAN follows the rules written below:

1. A point can only belong to a cluster if and only if it lies within the *Epscoord*-neighborhood of some core point in the cluster.
2. A *core point o* within the *Epscoord*- neighborhood of another core point p must belong to the same cluster as p.
3. A *border point r* within the *Epscoord*- neighborhood of some core point must belong to the same cluster to at least one of the core points.
4. A border point which does not lie within the *Epscoord*- neighborhood of any core point is considered to be noise.

3.2 A Density Based Algorithm: Clustering by Search and Find of Density Peaks (FSDP)

A clustering algorithm named "Clustering by fast search and find of density peaks" (FSDP) has been proposed in 2014 by [21] this modification is for finding the centers of clusters quickly.

They propose an approach based on the idea that cluster centers are characterized by a higher density than their neighbors and by a relatively large distance from points with higher densities. This idea forms the basis of their clustering procedure in which the number of clusters arises intuitively, outliers are automatically spotted and excluded from the analysis, and clusters are recognized regardless of their shape and of the dimensionality of the space in which they are embedded.

The accuracy of their algorithm depends on the threshold, and no efficient way was given to select its suitable value, i.e., the value was suggested be estimated on the basis of empirical experience.

Our modification regards the implementation i.e. it follows the steps of the *densityClust* R package with two substantial differences: as input it takes the data matrix instead of the distance matrix in so doing the calculation of the distance between points is done during the steps of the algorithm. This avoids saving a large distance matrix that requires a huge memory usage.

3.3 K-means Algorithm

The K-means algorithm is very well known (see e.g. [7]). The algorithm allocates the data points (objects) into clusters, so as to minimize the sum of the squared distances between the data points and the center of the clusters.

The centers of the clusters are initialized by randomly selecting from the data or by fixing particular data points, then the data set is clustered in the process of assigning

each point to the nearest center. When the data set has been identified, the average position of the data points within each cluster is calculated and the cluster center then moved to the average position. This process is repeated until a condition of stopping is reached, in other words the algorithm has these steps:

Step 1: Place K points into the space represented by the objects that are being clustered. These points represent initial group centroids.

Step 2: Assign each object to the group that has the closest centroid.

Step 3: When all objects have been assigned, recalculate the positions of the K centroids.

Step 4: Repeat Steps 2 and 3 until the centroids no longer move or another stopping rule is achieved. This produces a separation of the objects into groups from which the metric to be minimized can be calculated.

The K-means algorithm requires three user-specified parameters: number of clusters K, cluster initialization, and distance metric. The most critical choice is K.

K-means is typically used with the Euclidean metric for computing the distance between points and cluster centers. As a result, K-means finds spherical or ball-shaped clusters in data. K-means with Mahalanobis distance metric has been used to detect hyperellipsoidal clusters (see [9]), but this comes at the expense of higher computational cost. In certain cases it is possible to improve the results of algorithm using the standardization of the variables.

3.4 The Implementation of the Three Algorithms Using the R Language

The datasets used were formed by 250 000 observations and 5 variables: two spatial, the x and y coordinates of the pixel, and three non-spatial, the 3 RGB color bands (red, green and blue) of the point. Before applying the algorithm, a standardization of the variables has been performed, this standardization avoid problems in the results of the implementation of both the FSDP and K-means algorithms.

As written before the *R* language has been used for the analysis. The *dbscan*, *stats* and the *densityClust R* packages have been used for analyzing the data, whereas the *raster* and *rgda*l packages have been used for managing the raster objects and the visualization of the results.

To apply the "Fast search by density peak" algorithm on big data we have created a new R implementation of the algorithm. Our implementation follows the steps of that of the *densityClust R* package with two substantial differences: as input it takes the data matrix instead of the distance matrix, so the calculation of the distance between points is done during the steps of the algorithm. This avoids saving a large distance matrix that requires a huge memory usage.

The metric used is the Euclidian one this is the same used in the classical DBSCAN and the K-means algorithms. In order to reduce computational time some of the functions have been written in C++ language, hence we used the *Rcpp* package to integrate R and C++.

The three algorithms require different parameters: number of clusters for the K-means, threshold value for the radius of the epsilon neighborhood (eps) and minimum number of points of the epsilon neighborhood (minpts) for the DBSCAN and local density (ρ) and minimum distance to higher density points (σ) thresholds for the FSDP. To choose the values of the parameters for the density-based algorithms two plots were used: the K-NN distance plot for the DBSCAN(see Fig. 1(a)) and a decision graph for the FSDP (see Fig. 1(b)). The K-NN distance plot represents the k-nearest neighbor distances. Once set the *minpts*, which can be the dimensionality of the dataset plus one or higher, eps is the first knee in the *minpts*-NN distance plot (see Fig. 1(a)).

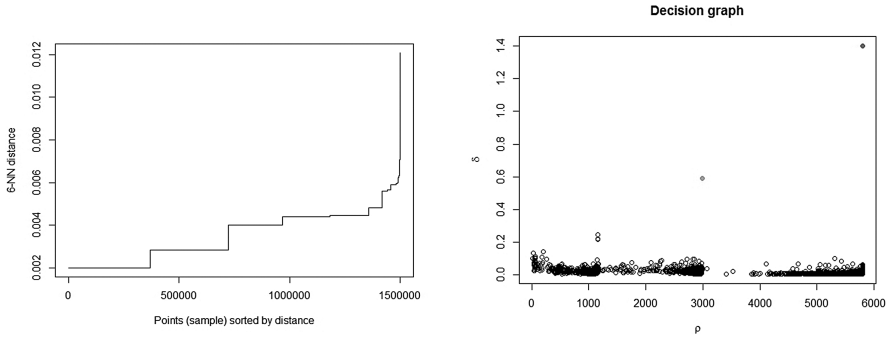

Fig. 1. (a) K-NN distance plot (b) Decision graph

In the decision graph we can see the distribution of the points relatively to the parameters ρ and σ.

The points with high σ and high ρ will be clusters centers, hence by looking at the graph we can decide which is the threshold values of ρ and σ above which the cluster centers are identified. In the figure they are the two points colored in red and green.

4 Computational Aspects

In the analysis of Big Data computational aspects are fundamental in considering which method has to be used; in particular CPU time and RAM usage have to be taking into account. As regards the choice of one of the three algorithms that have been considered, comparing the CPU times, we verified that the K-means algorithm is the most performing as in the DBSCAN and the FSDP time increases very quickly as the number of points gets larger (see Fig. 2).

An implementation of the DBSCAN has been proposed which drastically reduces the computational time needed by using the Lagrange distance instead of the Euclidian one has been proposed [23].

The use of the distance matrix instead of the data matrix in the FSDP algorithm leads to other computational aspects as regards the RAM usage: whereas the K-means and the DBSCAN require few mega-bites to perform the analysis even on very big dataset, to calculate and save the distance matrix needed to apply the FSDP algorithm

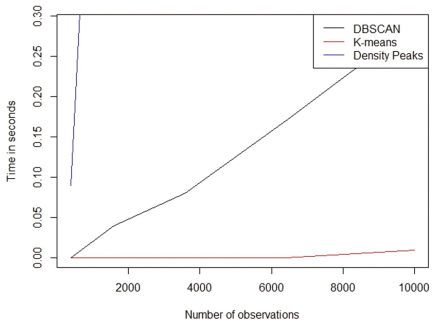

Fig. 2. CPU time

an enormous amount of RAM is required. This made it impossible to apply the FSDP algorithm to spatial Big Data. As an example, for the data set that we considered of 250 000 units demanded more than 200 Gb of RAM.

This is why we propose an R implementation of the "Fast Search by Density Peaks" (FSDP) that takes as input the data matrix and therefore allows to apply the algorithm to Big Data. Our version of the FSDP for Big Data calculates the distance between points only when is needed avoiding to save a complete distance matrix that would have more than 10 billion elements to save.

5 The Application Results and Discussion

5.1 R Implementation from "*densityclust*" Package and the New Big Data Implementation

First of all, we verify that the performance of our new Big Data implementation of FSDP in R is the same as the one of the *densityClust* R package. A dataset of 10 000 pixels has been used for this scope. The results are shown in Fig. 2. As one can see the result is the same (Fig. 3).

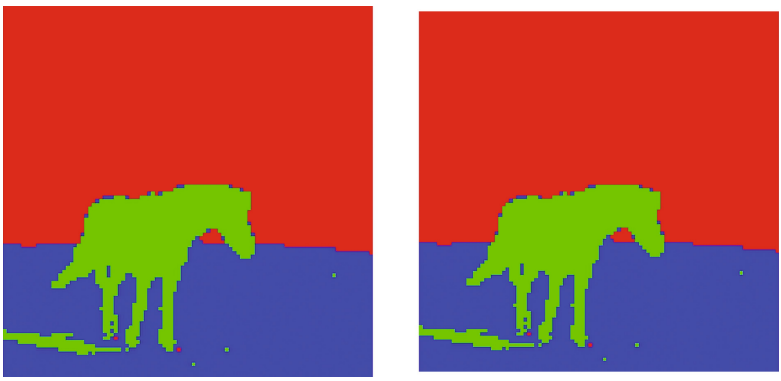

Fig. 3. Comparison of original R implementation and Big Data implementation in R

5.2 K-means, DBSCAN and Fast Search by Density Peak Clustering

To test the K-means, DBSCAN and the FSDP algorithms we consider some applications to a synthetic dataset, a digital photo and to satellite images.

In the data set presented in Fig. 4 we consider a synthetic dataset consisting in a partition in regular geometric shapes of uniform color. Each cluster has been represented by a different color. From now on the noisy points will be always colored in black.

Fig. 4. A synthetic data set

Even in this simple synthetic dataset, the DBSCAN outstands the other two algorithms. As one can see the FSDP algorithm shows the poorest results (see Fig. 5).

Fig. 5. Results of k-means, DBSCAN and Density Peak respectively

Next, we have applied the algorithms to a digital picture (Fig. 6).

This is a good example of how the k-means algorithms is not able to detect clusters of irregular shapes and on the contrary the DBSCAN is very good at it. The DBSCAN in this dataset produces a result extremely precise, i.e. the cluster representing the

Fig. 6. Digital photo

shadow of the pony. The FSDP algorithm performs slightly better than the k-means but still the result is completely satisfactory (see Fig. 7).

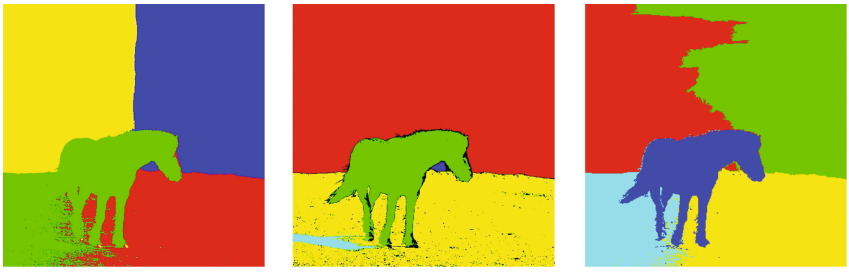

Fig. 7. Results of k-means, DBSCAN and FSDP respectively

Next applications regard data sets of satellite images. The former is a LANDSAT 8 Satellite image of the gulf of Trieste (see Fig. 8) the latter a LANDSAT 8 Satellite of the Alps (see Fig. 10).

The results of the K-means and the FSDP algorithm don't seem to differ very much, they perform very well. In this case, even though the separation line between the "sea cluster" and the "land cluster" is irregular the K-means is able to distinguish between the two areas and so do the SFDP. However even in this case, the DBSCAN performance is the most convincing because it is the only one what recognizes as noise the clouds in the right upper corner (see Fig. 9).

This is a much more challenging satellite image than the previous one so it is not surprising that the k-means fails to recognize clusters of so irregular shape. In this last example also the results of the DBSCAN algorithm are not perfect, but still better than the FSDP and the k-means algorithms (see Figs. 10 and 11).

Fig. 8. LANDSAT 8 Satellite image -Trieste gulf http://landsatlook.usgs.gov/viewer.html

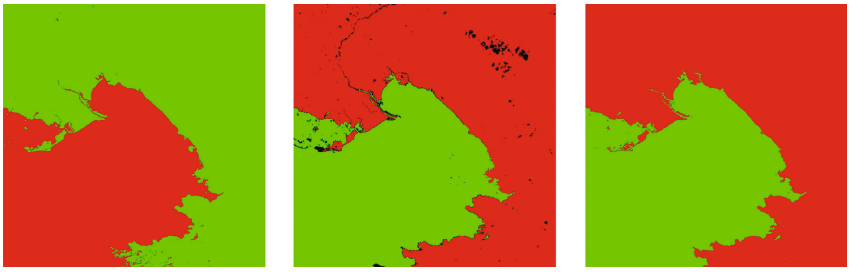

Fig. 9. Results of k-means, DBSCAN and Density Peaks respectively

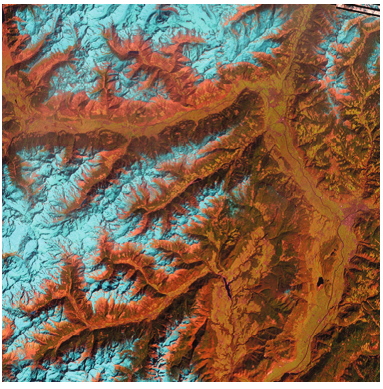

Fig. 10. Image LANDSAT 8 Satellite – Alps http://landsatlook.usgs.gov/viewer.html

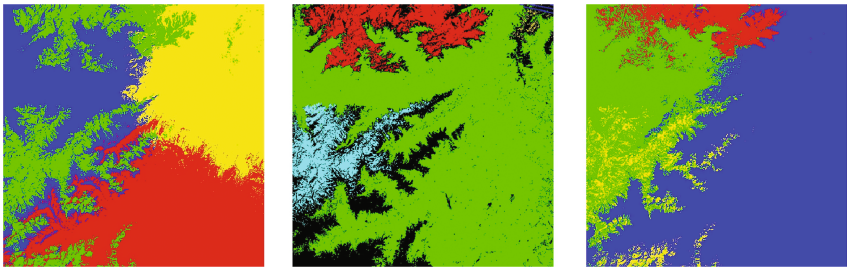

Fig. 11. Results of k-means, DBSCAN and Density Peak respectively

6 Conclusions

In this paper three way of clustering methods applied to identify homogeneous areas in a Spatial Big Data context are compared: *the Density-Based Spatial Clustering of Applications with Noise* algorithm (DBSCAN), *the Fast Search by Density Peak* (FSDP) and the classic *K-means* algorithm (K-Means).

As regards the FSDP algorithm a modification of this algorithm, in order to improve its efficiency in the application on Big Data, has been proposed. The implementation has been performed using the R language.

The applications regards both synthetic and real datasets. The spatial clustering analysis allowed to obtained good bit-map images and good representation of satellite images.

The main advantage of K-means algorithm is its simplicity and speed which allows it to run on large datasets. One problem of the application of the K-means is the necessity of knowing *a prori* the number of clusters. Other problem regard the identification of noise (outliers) and the discovering of clusters of arbitrary shape.

DBSCAN is robust enough to identify clusters in noisy data, requires just a few parameters and is mostly insensitive to the ordering of the points in the database. This algorithm is efficient even for very large spatial databases, discovers clusters of arbitrary shape and does not need to know the number of clusters in the data a priori, as opposed to K-means. Our modification of the FSDP algorithm regards the implementation with two substantial differences: as input it takes the data matrix instead of the distance matrix; this avoids saving a large distance matrix that requires a huge memory usage.

References

1. Bailey, T.C., Gatrell, A.C.: Interactive Spatial Data Analysis. Addison Wesley Longman, Edinburgh (1996)
2. Bedard, Y.: Beyond GIS: Spatial On-line Analytical Processing and Big Data. University of Maine (2014). http://umaine.edu/scis/files/2014/09/Beyond-GIS-Spatial-On-Line-Analytical-Processing-and-Big-Data-Yvan-Bedard.pdf

3. Chen, Y., Suel, T., Markowetz, A.: Efficient query processing in geographic web search engines. In: SIGMOD 2006, Chicago, Illinois, USA, 27–29 June 2006 (2006). http://cis.poly.edu/suel/papers/geoquery.pdf
4. Cressie, N.: Statistics for Spatial Data. Wiley, London (1993)
5. Cugler, D.C., Dev, O., Evans, M.R., Shekhar, S., Medeiros, C.B.: Spatial Big Data: Platforms, Analytics, and Science. http://www.spatial.cs.umn.edu/geojournal/2013/geojournal.pdf. Accessed 22 Sept 2016
6. El-Sonbaty, Y., Ismail, M.A., Farouk, M.: An Efficient density-based clustering algorithm for large databases. In: Proceedings of the 16th IEEE International Conference on Tods with Artificial Intelligence (ICTAI) (2004)
7. Ester, M., Kriegel, H.P., Sander, J., Xiaowei, X.: A density-based algorithm for discovering clusters in large spatial databases with noise. In: Proceeding of the 2nd International Conference on Knowledge Discovery and Data Mining, pp. 94–99 (1996)
8. Fayyad, U., Piatesky–Shapiro, G., Smyth, P.: From Data Mining to Knowledge Discovery in Databases (1996). http://www.kdnuggets.com/gpspubs/aimag-kdd-overview-1996-Fayyad.pdf
9. Fahad, A., Alshatri, N., Tari, Z., Alamri, A., Khalil, I., Zomaya, A.Y., Foufou, S., Bouras, A.: A survey of clustering algorithms for big data: taxonomy and empirical analysis. IEEE Trans. Emerg. Topics Comput. **2**(3), 267–279 (2014)
10. Han, J., Kamber, M., Tung, A.K.H.: Spatial Clutering Methods in Data Mining: A Survey (2001). ftp://ftp.fas.sfu.ca/pub/cs/han/pdf/gkdbk01.pdf
11. Hemalatha, M., Naga Saranya, N.: A recent survey on knowledge discovery in spatial data mining. Int. J. Comput. Sci. Issues **8**(3), 473–479 (2011)
12. Jan, A.K.: Data clustering. 50 years beyond K-means. Pattern Recogn. Lett. **31**(8), 651–666 (2010)
13. Khan, K., Rehman, S.U., Fong, S., Sarasvady, S.: DBSCAN: past, present and future. In: The Fifth International Conference on the Applications of Digital Information and Web Technologies, February 2014, pp. 232–238 (2014)
14. Koperski, K., Han, J., Adhikary, J.: Mining Knowledge in Geographical Data (1998). ftp://ftp.fas.sfu.ca/pubcs/han/pdf/geo_survey98.pdf.
15. Lee, J., Kang, M.: Geospatial big data: challenges and opportunities. Big Data Res. **2**(2), 74–78 (2015)
16. Liu, J., Li, J., Li, W., Wu, J.: Rethinking Big Data: a review on the data quality and usage issues. PRS J. Photogrammetry Remote Sens. **115**, 134–142 (2016)
17. Mao, J., Jain, A.K.: A self-organizing network for hyper-ellipsoidal clustering (HEC). IEEE Trans. Neural Netw. **7**(1), 16–29 (1996)
18. Mennis, J., Guo, D.: Spatial Computing. "Spatial Data Mining". https://www.youtube.com/watch?v=sZeb93O_z2w&list=PLN5UPhO05nn8WE4ZbzUwUhzq_p2XChK6r&index=3. Accessed 22 May 2016
19. Mohebi, A., Aghabozorgi, S., Wah, T.Y., Herawan, T., Yahyapour, R.: Iterative Big Data clustering algorithms: a review. Soft. Pract. Exp. **46**(1), 107–129 (2016)
20. Pragati, S., Hitech, G.: A review of density-based clustering in spatial data. Int. J. Adv. Comput. Res. **2**(5), 210–213 (2012)
21. Rodriguez, A., Laio, A.: Clustering by fast search and find of density peaks. Science **344** (6191), 1492–1496 (2014)
22. Sander, J., Ester, M., Kriegel, H.P., Xiaowei, X.: Density-based clustering in spatial databases: the algorithm GDBSCAN and its applications (1999). http://www.dbs.informatik.uni-muenchen.de/Publikationen/

23. Schoier, G., Bato, B.: A modification of the DBSCAN algorithm in a spatial data mining approach. In: Meeting of the Classification and Data Analysis Group of the SIS, CLADAG 2007, Macerata, 12–14 September 2007, pp. 395–398. EUM, Macerata (2007)
24. Schoier, G., Borruso, G.: A methodology for dealing with spatial big data. Int. J. Bus. Intel. Data Min. **12**(1), 1–13 (2017)
25. Steinbach, M., Ertöz, L., Kumar V.: The Challenges of Clustering High Dimensional Data (2003). http://www-users.cs.umn.edu/~kumar/papers/high_dim_clustering_19.pdf
26. Xu, R., Wunsch II, D.: Survey of Clustering Algorithms (2005). http://ieeexplore.ieee.org/iel5/72/30822/01427769.pdf
27. Wang, S., Yuan, H.: Spatial data mining in the context of big data. In: 2013 International Conference on Parallel and Distributed Systems, December 2013, pp. 486–491 (2013)
28. Ye, Q., Gao, W., Zeng, W.: Color image segmentation using density-based clustering. In: 2003 IEEE International Conference on Acoustics, Speech, and Signal Processing, vol. 3, p. III-345 (2003)