

Analysis and Extensions of Popular Clustering Algorithms

Renáta Iváncsy, Attila Babos, Csaba Legány

Department of Automation and Applied Informatics
and HAS-BUTE Control Research Group
Budapest University of Technology and Economics
Goldmann Gy. tér 3, H-1111 Budapest, Hungary
e-mail: {renata.ivancsy, babos.attila, csaba.legany}@aut.bme.hu

Abstract: Clustering is one of the most important research areas in the field of data mining. Clustering means creating groups of objects based on their features in such a way that the objects belonging to the same groups are similar and those belonging in different groups are dissimilar. In this paper the most representative algorithms are described and categorized based on their basic approach. Three of them, namely, the partitioning-based k-means, the hierarchical bottom-up and the density-based DBScan algorithms are analyzed based on their clustering quality and their performance.

Keywords: Clustering, Data mining, k-means, DBScan

1 Introduction

Clustering objects means to categorize them based on their feature vectors. For this reason a distance measure on these features has to be defined. The objects that are near to each other belong to the same cluster, and those that are far from each other belong to different clusters. The result of the clustering depends strongly on the definition of the distance-function.

Data mining means discovering hidden information in large amount of data. Clustering is used in many research areas like machine learning, object recognition, information retrieval etc. However, in case of large number of objects data mining solutions have to be used to obtain an efficient algorithm.

This paper analyzes the most popular clustering algorithm in the field of data mining. It categorizes them regarding their fundamental approach and describes three of the most popular algorithms in detail. Based on experimental results the different algorithms are compared regarding their clustering quality and their performance.

The organization of the paper is as follows. Section 2 offers a brief survey about the clustering algorithms. In this section three of the most popular algorithms are described in detail that forms the basis of our experiments. In Section 3 the enhanced version of the density-based DBScan algorithm is introduced. Experimental results and the analysis are shown in Section 4. In this section both the quality and the performance of the algorithms are analyzed.

2 Overview of the Clustering Algorithms

Clustering is a process of discovering groups of objects such that the objects belonging to the same group are similar in a certain manner, and the objects belonging to different groups are dissimilar. The main problems one faces when creating a clustering algorithm are the following:

- The objects can have hundreds of attributes that have to be taken into consideration for clustering. One of the key issues is how to reduce this number to achieve an efficient algorithm.
- The type of the attributes can be diverse, and not only numerical attributes has to be handled.
- Because of the first two problems defining a similarity function between the objects is not a trivial task. Many features and many types of attributes have to be handled efficiently.
- The main feature of clustering in a data mining application is the high number of objects that have to be clustered. Thus the processing time or the memory requirement of the algorithm can be huge, that has to be reduced using some heuristics.
- Validating the resulting clusters is also a hard task. In case of low dimensionality, when the clusters can be represented visually, the validation can be made by a human, but in case having large number of objects with high dimensionality statistical methods have to be used and indices have to be defined which can be computationally expensive.

There are many algorithms in the literature that deal with the problem of clustering large number of objects. The different algorithms can be classified regarding different aspects. One of the key issues, which determines also another features of the algorithm, is the basic approach of the clustering algorithm.

The aim of the **partition-based algorithms** is to decompose the set of objects into a set of disjoint clusters where the number of the resulting clusters is predefined by the user. The algorithm uses an iterative method, and based on a distance measure it updates the cluster of each object. It is done until any changes can be made. The most representative partition-based clustering algorithms are the k-

means and the k-mediod, and in the data mining field the CLARANS [1]. The advantage of the partition-based algorithms that they use an iterative way to create the clusters, but the drawback is that the number of clusters has to be determined in advance and only spherical shapes can be determined as clusters.

Hierarchical algorithms provide a hierarchical grouping of the objects. There exist two approaches, the bottom-up and the top-down approach. In case of bottom-up approach, at the beginning of the algorithm each object represents a different cluster and at the end all objects belong to the same cluster. In case of top-down method at the start of the algorithm all objects belong to the same cluster which is split, until each object constitute a different cluster. The steps of the algorithms can be represented using a dendrogram. The resulting clusters are determined by cutting the dendrogram by a certain level. A key aspect in these kind of algorithms is the definition of the distance measurements between the objects and between the clusters. Many definitions can be used to measure distance between the objects, for example Euclidian, City-Block, Minkowski and so on. Between the clusters one can determine the distance as the distance of the two nearest objects in the two clusters, or as the two furthest or as the distance between the medioids of the clusters. The drawback of the hierarchical algorithm is that after an object is assigned to a given cluster it cannot be modified later. Furthermore, like in partition-based case, also only spherical clusters can be obtained. The advantage of the hierarchical algorithms is that the validation indices (correlation, inconsistency measure), which can be defined on the clusters, can be used for determining the number of the clusters. The best known hierarchical clustering methods are CHAMELEON [2], BIRCH [3] and CURE [4].

Density-based algorithms start by searching for core objects, and they are growing the clusters based on these cores and by searching for objects that are in a neighborhood within a radius ϵ of a given object. The advantage of these type of algorithms is that they can detect arbitrary form of clusters and it can filter out the noise. DBSCAN [5] and OPTICS [6] are density-based algorithms.

Grid-based algorithms use a hierarchical grid structure to decompose the object space into finite number of cells. For each cell statistical information is stored about the objects and the clustering is achieved on these cells. The advantage of this approach is the fast processing time that is in general independent of the number of data objects. Grid-based algorithms are STING [7], CLIQUE [8] and WaveCluster [9].

Model-based algorithms use different distribution models for the clusters which should be verified during the clustering algorithm. A model-based clustering method is MCLUST [10].

Fuzzy algorithms suppose that no hard clusters exist on the set of objects, but one object can be assigned to more than one cluster. The best known fuzzy clustering algorithm is FCM (Fuzzy C-MEANS) [11].

Hereinafter the three algorithms are described in detail, which are the focus of our analytical experiments. These algorithms are the partition-based k-means, the hierarchical bottom-up and the density-based DBScan algorithms.

2.1 The K-means Algorithm

The k-means algorithm is a partition-based algorithm that takes the number of the clusters as input parameter. The algorithm organizes the objects into exactly k partitions where each partition represents a cluster.

The k-means algorithm works as follows. First of all the algorithm randomly selects k of the objects. Each selected object represents a single cluster, and because in this case only one object is in the cluster, this object represents the mean or center of the cluster. As the second step the algorithm assigns each objects to exactly one cluster based on a distance measure. Each object is assigned to those cluster that is the nearest to the given object. In such a way a starting clustering is achieved. In the further steps the algorithm iteratively improves the quality of the clustering by relocation of the objects.

The second step is that the algorithm calculates the mean for each cluster, and this mean value will represent the given cluster. For each of the objects the distances are checked, and the object is placed into another cluster if the given object is closer to the other cluster than to the cluster it is already. This process iterates until the criterion function converges. Typically, the squared-error criterion is used. This criterion tries to make the resulting k clusters as compact and as separate as possible.

The advantage of the algorithm is its favorable execution time. Its drawback is, however, that the user has to know in advance how many clusters are searched for. Other drawback is that it can find only spherical clusters and it cannot handle noise. Furthermore the result of the clustering algorithm depends on the object selection at the beginning of the algorithm.

2.2 Bottom-up Hierarchical Algorithm

The Bottom-up algorithm is a fundamental hierarchical algorithm, and as it is shown in its name, it starts the clustering by placing each object in different clusters and then merges them into larger and larger clusters until all of the objects are in the same cluster. The user can give a termination condition to the algorithm, in this case the algorithm does not reach the state where each object is in the same cluster but it terminates earlier.

The algorithm merges two clusters if they are the most similar among the clusters. The similarity can be defined in several ways and from different aspects. A similarity function has to be defined between the objects and also between the

clusters that is mostly a distance function. In most cases the distance function is chosen out of the following list:

- Euclidian distance
- Standart euclidian distance
- Mahalanobis distance
- City Block distance
- Minkowski distance

The definition of the distance measure affects the result of the clustering significantly. Another important aspect is the distance definition between two clusters. The following definitions are used:

- The distance of two clusters equals to the distance of the two objects in the different clusters that are the nearest to each other (nearest neighbor).
- The distance of two clusters equals to the distance of the two objects in the different clusters that are furthest from each other (furthest neighbor).
- The distance of two clusters equals to the distance of the means of the two clusters.
- The distance of two clusters equals to the distance of the center of the two clusters.

During the merging of the clusters a binary hierarchical tree is built that can be represented with a dendrogram. The number of the clusters can be determined by an input parameter of the algorithm, or by calculating the cophenetic correlation coefficient and the inconsistency quotient.

The drawback of the algorithm is that, like the k-means algorithm, only spherical clusters can be determined. Furthermore after an object is assigned to a given cluster it cannot be modified later which makes the algorithm less robust.

2.3 DBScan Algorithm

The DBScan algorithm is a fundamental density-based clustering algorithm. Its advantage is that it can discover clusters with arbitrary shapes. The algorithm typically regards clusters as dense regions of objects in the data space which are separated by regions of low density objects. The algorithm has two input parameters, ϵ and MinPts. For understanding the process of the algorithm some concepts and definitions has to be introduced.

The naïve definition of dense objects is as follows. The objects of a cluster are spread dense if in the neighborhood within a radius ϵ of each object at least MinPts other object exist.

- The neighborhood within a radius ε of a given object is called the **ε -neighborhood** of the object.
- If the ε -neighborhood of an object contains at least a minimum number, MinPts, of objects then the object is called a **core object**.
- Given a set of objects, we say that an object p is **directly density-reachable** from object q if p is within the ε -neighborhood of q , and q is a core object.
- An object p is **density-reachable** from object q with respect to ε and MinPts in a set of objects, if there is a chain of objects p_1, \dots, p_n , $p_1 = q$ and $p_n = p$ such that p_{i+1} is directly density-reachable from p_i .
- An object p is **density-connected** to object q with respect to ε and MinPts in a set of objects, if there is an object o such that both p and q are density-reachable from o with respect to ε and MinPts.

The DBScan algorithm works as follows. It checks the ε -neighborhood of each object of the dataset, and if in this area more objects than MinPts exist then it is called a core object. Each cluster is grown from a core object by collecting those points that are directly density-reachable from the core point. The algorithm terminates if no more points exist that can be assigned to a cluster. Those objects are treated as noise that could not be assigned to any cluster during the algorithm.

The advantage of this algorithm is that it can find arbitrary shapes of clusters and it exploits the benefit of the natural approach clustering those objects together that forms a dense region. Furthermore it can handle noise as well.

3 Enhancement of the DBScan Algorithm

In order to enhance the performance of the DBScan algorithm [12] suggest using R^* trees for determining the ε -neighborhood of an object. The R^* tree is similar to the B-tree in a certain manner, namely, in both trees the distance equals between the root and the leaves, and the number of the children of each node is limited. This feature ensures that the processing time on this tree is logarithmic.

For each node belongs an encapsulating region that contains the children of the node. To determine the ε -neighborhood of an object its encapsulating region has to be determined, and the tree has to be traversed from the children of the object to the leaves.

The following criteria are important to consider when creating a tree:

- Keeping the number of the branches as low as possible.
- Keeping the tree as compact as possible.
- Keeping the distances in each dimension as equal as possible.

These criteria are in contradiction, thus there exists no optimal solution for creating an R^* tree. The complexity of creating the tree equal to $O(n \cdot \log n)$ and the complexity of using the tree is $O(n)$.

4 Experimental Results

In our experiments four algorithms were implemented, namely, k-means, bottom-up, DBScan and DBScan with R^* tree structure. The simulations were executed on a Pentium 4 CPU, 2.26GHz, and 1GB of RAM computer. The different algorithms were implemented in C# using .NET Framework v.1.1. The experimental results were performed on sythetic dataset created by a data generator.

4.1 Quality Analysis

In this section the quality of the clustering algorithms are analyzed. Here a definition of a good clustering is needed, which cannot be defined in an exact mathematical way. In our experiments we assumed that in two dimensional objects the quality of a clustering can be determined with human observation the best. It means that a clustering is said to be good if it seem good fot a human observer.

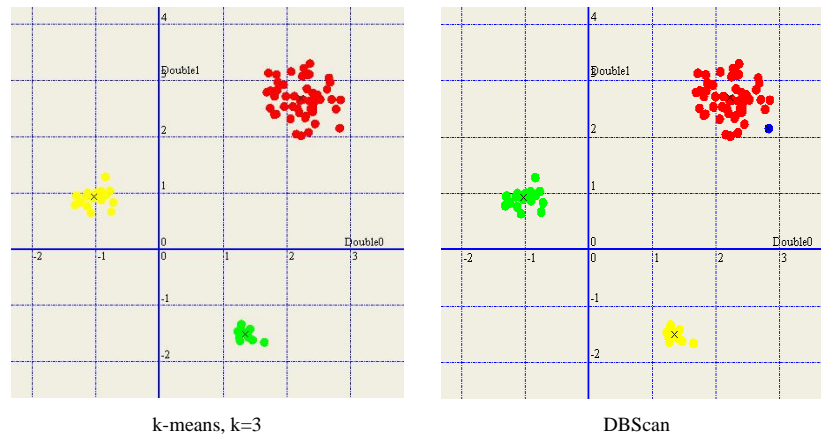


Figure 1

Clustering Gaussian-distribution-based objects

For analysing the quality of the clustering five representative configuration of objects were used:

- noise (randomly placed points)

- Points having Gaussian-distribution
- Points having a form of embedded rings
- Arbitrary shapes

Circles with noise Because of space limitations not all cases are displayed and explained only some of them. Figure 1 shows the clustering results of the k-mean and the DBScan algorithms on an object space having Gaussian-distribution. The center of each cluster is depicted with a cross.

It can be seen well that there exist a point (its coordinates are (2.8, 2.1)) that is handled as noise in case of DBScan algorithm, and that is merged into a cluster in case of the k-means algorithm. Of course the fact whether an object is handled as noise or not depends on the input parameters of the DBScan algorithm.

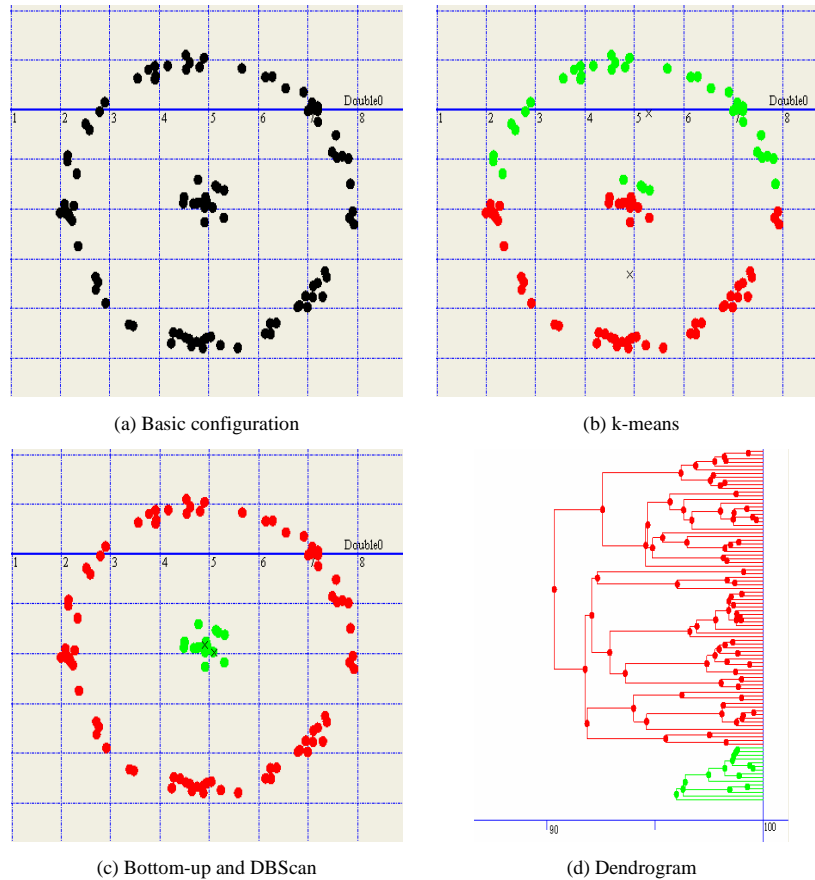


Figure 2

Basic object configuration of embedded rings and the result of clustering in case of all the algorithms

Figure 2 (a) shows the basic object configuration in case of embedded rings. Based on the human taste this could be clustered into two clusters the one would be the outer ring and the second the circle in the middle. In Figure 2 (b) the clustering results of the k-means algorithm is depicted. It can be observed that this result does not match our expectations. The two clusters are split with a horizontal line instead of having two embedded rings as clusters. However, the results of the bottom-up and the DBScan algorithms, depicted in Figure 2 (c), discover the right clusters. This can be explained with the fundamental approach of these two algorithms. The hierarchical algorithm finds these clusters if the distance between the clusters is defined as the nearest-neighbor. The dendrogram for the bottom-up algorithm is shown in Figure 2 (d). The density-based DBScan algorithm discovers the right clusters because it uses the density-reachable definition. The two core points are found in the two different rings, and the clusters are grown from this core points using the direct density-reachable definition.

4.2 Performance Analysis

The performance analysis were made on object spaces having different numbers of objects. The objects have Gaussian-distribution such that both the k-means, the bottom-up and the DBScan algorithms provide about the same result. The number of the objects varies from 100 to 100,000. Table 1 shows the number of steps that the various algorithms have to be done. In order to better compare the different algorithms they step numbers as a function of the number of objects are depicted in Figure 3.

Table 1
Comparing the number of steps of the different algorithms

Number of points	k-means	Bottom-up	DScan (without R*)	DBScan (with R*)
100	0,2	0,08	0,2	0,2
1000	5	1	5	1
2000	12	901	5	2
5000	50	5985	30	4
10000	240	10325	132	9

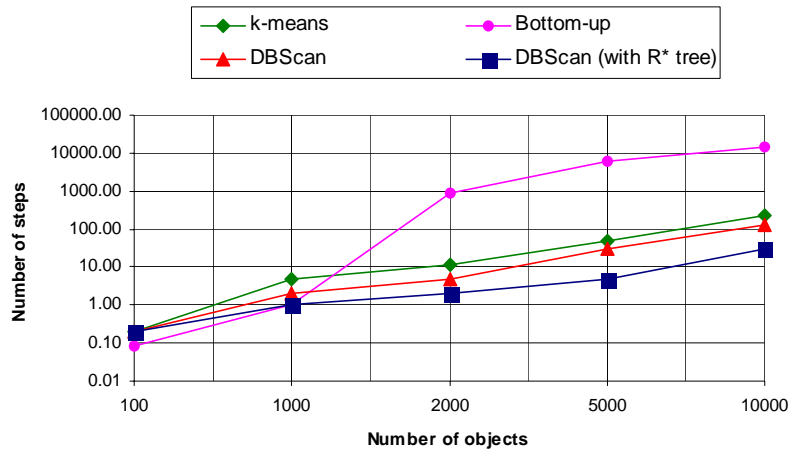


Figure 3
Number of steps of the algorithms in logarithmical scale

When analyzing the results of the performance measurements, the conclusion can be drawn that in case of small number of objects the most efficient algorithm is the bottom-up algorithm. However, this algorithm is proven the least efficient in case of large number of objects. This can be explained by the complexity of the algorithm that is $O(n^3)$. In case of great number of objects the most efficient algorithms are the DBScan and the DBScan enhanced with the R* tree structure.

Conclusions

This paper deals with the problem of discovering groups of objects in such a way that given a similarity measure the objects belonging to the same cluster are similar and those belonging to different clusters are dissimilar. The different clustering algorithms were ranged into categorized. Three of the main algorithms were implemented, and their behaviors were analyzed based on our experimental results. The two basic aspect of the analysis were the quality of the resulting clusters and the performance of the algorithms.

Acknowledgement

This work has been supported by the Mobile Innovation Center, Hungary, by the fund of the Hungarian Academy of Sciences for control research and the Hungarian National Research Fund (grant number: T042741).

References

- [1] R. T. Ng and J. Han, "Clarans: A method for clustering objects for spatial data mining," *IEEE Transactions on Knowledge and Data Engineering*, vol. 14, no. 5, pp. 1003-1016, 2002

- [2] G. Karypis, E.-H. S. Han, and V. K. NEWS, "Chameleon: Hierarchical clustering using dynamic modeling," *Computer*, vol. 32, no. 8, pp. 68-75, 1999
- [3] T. Zhang, R. Ramakrishnan, and M. Livny, "BIRCH: an efficient data clustering method for very large databases," pp. 103-114, 1996
- [4] S. Guha, R. Rastogi, and K. Shim, "Cure: An efficient clustering algorithm for large databases," in *SIGMOD 1998, Proceedings ACM SIGMOD International Conference on Management of Data, June 2-4, 1998, Seattle, Washington, USA* (L. M. Haas and A. Tiwary, eds.), pp. 73-84, ACM Press, 1998
- [5] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise.," in *KDD*, pp. 226-231, 1996
- [6] M. Ankerst, M. M. Breunig, H.-P. Kriegel, and J. Sander, "Optics: ordering points to identify the clustering structure," in *SIGMOD '99: Proceedings of the 1999 ACM SIGMOD international conference on Management of data, (New York, NY, USA)*, pp. 49-60, ACM Press, 1999
- [7] W. Wang, J. Yang, and R. Muntz, "Sting: A statistical information grid approach to spatial data mining," 1997
- [8] R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan, "Automatic subspace clustering of high dimensional data for data mining applications," pp. 94-105, 1998
- [9] G. Sheikholeslami, S. Chatterjee, and A. Zhang, "WaveCluster: A multi-resolution clustering approach for very large spatial databases," in *Proc. 24th Int. Conf. Very Large Data Bases, VLDB*, pp. 428-439, 24-27, 1998
- [10] C. Fraley and A. Raftery, "Mclust: Software for model-based cluster and discriminant analysis," 1999
- [11] J. C. Bezdek, R. Ehrlich, and W. Full, "Fcm: Fuzzy c-means algorithm," *Computers and Geoscience*, vol. 10, no. 2-3, pp. 191-203, 1984
- [12] N. Beckmann, HP. Kriegel, R. Schneider, B. Seeger: The R*-tree: An Efficient and Robust Access Method for Points and Rectangles, *In Proceedings of the 1990 ACM SIGMOD International Conference on Management of Data, Atlantic City, New Jersey, US*, 1990. pp. 322-331