Survey on Clustering Algorithms for Large Dataset

Daniele Bacarella
University of Uppsala
Email: Daniele.Bacarella.3279@student.uu.se

***Abstract*—During the last decade the size of datasets has increased and with it also the dimensionality of each multimedia object stored. Cluster analysis attempts to find natural clusters of similar objects and the complexity of the task depends on many factors that can be the numbers of objects to cluster and their representation in the dataset just to cite some. Many clustering algorithms have been developed so far, each of which tries to beat the previous ones when applied on large dataset. The aim of this survey is to inspect some clustering algorithms for large datasets showing when possible a comparison between them.**

I. Introduction

Cluster analysis, can be defined as the process of organizing objects (patterns) into groups (clusters) whose members are among their "peers". So, patterns of a cluster have a strong similarity between them and at the same time an inferior similarity with the patterns of other clusters. The definition provided, although seemingly simple and intuitive, opens up a range of operational issues related to the definition and measurement of similarity between objects, as well as on methods and implementations of clustering algorithms using said similarity. It is important to point out right now what are the new application scenarios of cluster analysis in order to understand the renewed interest that has been developing in recent years around these activities, which are typical in statistics and hence extensively studied. Recently, clustering techniques have been applied to: the context of image segmentation, pattern recognition, optical character recognition and information retrieval, with particular regard to the development of the web. It should also be noted as the vastness of the data often implies the impossibility of having a prior set of predefined classes, leading consequently to consider the unsupervised cluster analysis, seen as the task to find hidden structures in unlabeled data, as an activity. The latter observation has important implications in the evaluation of clustering algorithms as related to their ability to know and correctly estimate the initial parameters with which these algorithms work. Here we will go through some clustering algorithms showing for each of them how they work and the principles on which they are based on. In order, the algorithms shown here are: CLARANS, BIRCH, DBSCAN, DENCLUE and CHAMELEON.

II. CLARANS

It belongs to the family of so called Centroid-based clustering algorithms. A cluster is represented by a representative point (centroid) which may not belong to the dataset. Points are assigned to the nearest centroid according to a distance/similarity function. Such centroid can be either calculated as the mean point and so not always, representing an actual point in the dataset, or the nearest data point to the mean one (centroids are hence forced to be members of the dataset). K-Medoids that uses the latter approach is less sensitive to the outliers and clusters found do not depend on the point examination order but unfortunately it does not scale well. CLARANS [1] has been designed in such a way to address the scalability problem. The principle around the algorithm is that instead of clustering all the dataset, here only a sample of it is clustered. The algorithm CLARANS in fact is based on two existing well-known algorithms in cluster analysis called PAM and CLARA which are here briefly introduced.

PAM

PAM's approach (Partitioning Around Medoids) attempts to find a medoid (most representative object) for each of K clusters, where K is fixed. Concept of representative object of a cluster here is intended to be the object which is most centrally located. Once the medoids have been selected, in order to group the rest of the objects we say that the object O_i belongs to the cluster represented by the medoid O_j if $d(O_i, O_j) = min_{O_m} d(O_i, O_m)$, where $d()$ is the distance function and with $min_{O_m} d(O_i, O_m)$ is meant to be the minimum distance over all the medoids. The K medoids are initially chosen arbitrarily in the dataset, after the first step they are recalculated performing a sort of swapping between the current medoids and the rest of the objects checking whether this would bring to a better quality of the clustering, measured as the average dissimilarity between an object and the medoid of its cluster. The algorithm ends when no swap bring further improvements.

CLARA

CLARA on the other side, acts similarly as the previous algorithm but it relies on sampling. Instead of finding a representative object looking on the entire data set, it picks a sample, applies PAM and finds the medoids there. The intention is to provide a similar efficacy as it would actually consider the entire dataset and also improving the efficiency. The proper approximation of the selected medoids is possible only if the the drawn sample are chosen in a sufficiently random way. Comparing the complexity of these two methods, it is clear that PAM is absolutely inefficient on large dataset since that each iteration of the algorithm has a cost quadratic on the size of the dataset, more in details, given n the size of the dataset and k the number of medoids: $O(k(n-k)^2)$

CLARA, running PAM on each sample have a complexity of: $O(k(40+k)^2 + k(n-k))$ where the size of the sample is $40 + 2k$.

For more details on the algorithms see [1] and [5]

Contributions of CLARANS

CLARANS can be seen as a relaxation of the sampling approach of CLARA. Given n the number of objects and k the number of medoids to find, the search task is interpreted as a graph $G_{n,k}$ of nodes, where each node represents a set of objects, namely the medoids. So the set of nodes of the graph is $\{\{Om1...Omk\} \| Om1...Omk$ are objects of the dataset$\}$ and so each one represents a clustering. Now let Sa be the set of the selected objects and with $G_{sa,k}$ the subgraph consisting of objects as subset of S_a. The main difference here between CLARA and CLARANS is that while in the first the search is restricted to a pre designed subgraph, in the latter the subgraph is redesigned on each search. This improves the efficacy of CLARANS since a more wide set of neighbors are examined and still is more efficient than CLARA because it still checks only a portion of the neighbors instead of all of them. For what concerns the tuning of the parameters, two of them affect the efficiency of the algorithms: *maxneighbor*, which is the max number of neighbors to examine and *numlocal*, the number of local minima to obtain. It is easy to see that the higher is the value of *maxneighbor*, the more CLARANS resembles PAM.

III. BIRCH

Balanced Iterative Reducing and Clustering using Hierarchies, BIRCH [2] is a density-based method designed to overcome the problem related to the dataset size. The clustering algorithms shown so far perform several scans over the entire dataset causing a high I/O overhead. BIRCH needs only a single scan (just few more in case of refining), representing so a valuable algorithm that fits well for large database. Also it exploits dataset where the objects are not uniformly distributed over the space, treating dense areas as one eliminating hence the outliers. A Clustering Feature CF, is a compact data format for points in a cluster, it is a vector $CF = (N, LS, SS)$, where N is the number of objects in the cluster, LS is the linear sum of the objects and SS is the squared sum. Exploiting the *Additivity Theorem* also the CF of two clusters merged is obtained as follows: $CF1+CF2 = (N1+N2, LS1+LS2, SS1+SS2)$. The data structure used to store the CFs is a *CF tree*, a height balanced tree that grows dynamically as the data is scanned.

Two parameters describe the tree that are the branching factor B and a memory threshold T which the tree size depends on.

Fundamentally, BIRCH algorithm consists of 4 phases, see Figure 1, here briefly described:

- **Phase 1**: All the objects are scanned and at the same time an initially CF tree is built. It starts using a low threshold and then it put the objects into the tree, in case it runs out of memory the tree is re built into a smaller one by increasing the threshold . Here is important to pick a good initial threshold in order to get a smaller number of re builts as possible but still it is hard to get.
- **Phase 2**: The tree is condensed into a smaller one based on a desirable length. This is considered optional, the aim here is to reduce the size of the tree so that the next phase gets a better performance.
- **Phase 3 (Global Clustering)**: During the phase, leaf nodes are clustered according to their CF values by running a known clustering algorithm, more precisely an agglomerative hierarchical clustering algorithm.
- **Phase 4 (Local Clustering)**: Finally here it is possible to refine the clustering obtained in the previous phase. This phase is optional and requires more scans of the dataset to get a more accurate set of centroids and hence a more accurate objects re-distribution.

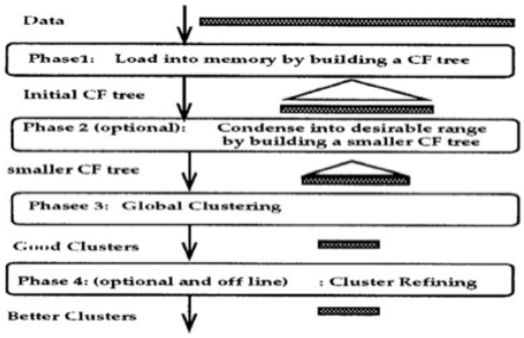

Fig. 1: BIRCH Algorithm's phases - (zhang96 [2])

Parameters Tuning

The selection of parameters affects the performance, the *Initial threshold* for instance, affects the running time during the building of the CF tree. Also, BIRCH has a good balancing between the size of the dataset and the runtime providing a good quality clustering without affecting much the runtime. In fact, in case the size of the tree would increase, the time needed to rebuild the tree would increase as well, but since it is done in memory this does not affect much the runtime. In case sufficient memory is not available, the accuracy can be improved during Phase 4.

IV. DBSCAN

DBSCAN (Density-Based Spatial Clustering of Applications with Noise) [4] is a density-based clustering which clusters objects situated in a region with a certain degree of density while all the other objects in the more sparse regions are considered as noise or outliers. The density-based notion of a cluster is characterized by a 'core' point, a distance value from it and also by a minimum number of objects in the neighborhood that is a area where the radius is the distance value. Given the distance value ε and a point p, the set of objects within a certain area is called ε-Neighborhood of p and the minimum number of points is indicated with *MinPts*.

Point types

Depending on the number of points in the neighborhood a point p is referred as:

- *Core Point*: a point with a value of ε-Neighborhood equal or greater than *MinPts*.
- *Border Point*: a point contained in the ε-Neighborhood of a core point but it is not a core point.
- *Outlier*: all the others.

Concepts

To better understand DBSCAN it is also important to know the concept of *density-reachability* that is a cluster model which DBSCAN is based on. More specifically, the clusters are defined as *Density-Connected Sets* wrt. *MinPts* and ε.

The concept of density-reachability is defined by the following three notions:

- *Directly Density-Reachable*: a point p is directly density-reachable from q if the latter is a core point and p is contained in its ε-Neighborhood, see Figure 2 for an example. This does not say anything about the nature of the point p and so the relation is considered not symmetric. The only case where the relation is symmetric is when both p and q are core points.

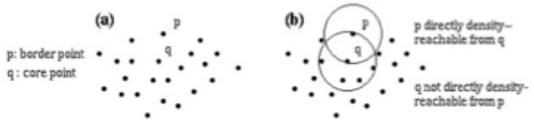

Fig. 2: Directly Density-Reachable - (Ester, Kriegel, Sander, Xu '96 [4])

- *Density-Reachable*: a point p is density-reachable from q if there exists a chain of points P_1, .. , P_n where $P_1 = q$ and $P_n = p$ and each P_{i+1} point is directly density-reachable from P_i, see Figure 3 for an example.

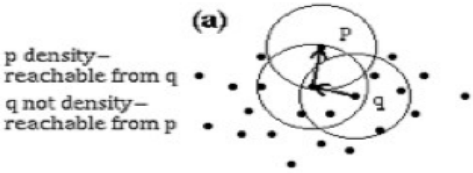

Fig. 3: Density-Reachable - (Ester, Kriegel, Sander, Xu '96 [4])

- *Density-Connected* : a point p is density connected to a point q if there is a point o such that both p and q are density-reachable from o, see Figure 4 for an example.

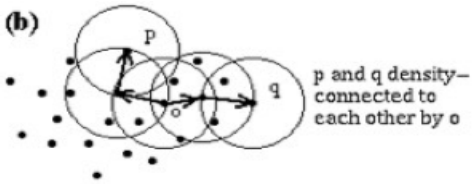

Fig. 4: Density-Connected - (Ester, Kriegel, Sander, Xu '96 [4])

Complexity

For each point DBSCAN determines its ε-Neighborhood and checks it against the MinPts value. To perform this task efficiently it uses index structures such as a R* tree, where its height is on the order of $O(\log n)$, where n is the size of the dataset, and so since DBSCAN needs only one scan over the dataset, the complexity of this algorithm is $O(n \log n)$.

Parameters Tuning

Determining good parameters ε and *MinPts* here as in all the other methods is crucial to get a quality clustering. If the dataset contains many actual clusters that differ in density then choosing a tight ε might prevent a less dense cluster from being found.

A heuristic developed to choose the values for ε and *MinPts* is based on the distance of the points to their k-neighbor, *k-dist*. Given a k, for each point in the dataset its *k-dist* is calculated and then sorted in descending order. The graph of this mapping in fact gives an idea of the density distribution in the dataset.

This graph has been called *sorted k-dist graph*, Figure 5 gives an idea of such graph with $k = 4$.

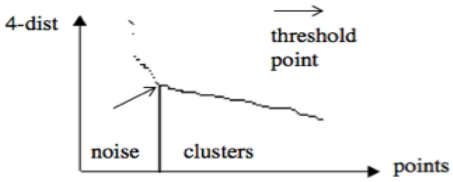

Fig. 5: Sorted 4-dist graph - (Ester, Kriegel, Sander, Xu '96 [4])

Performance

DBSCAN performs well with arbitrarily shaped clusters, it does not require the knowledge of the number of clusters and it has the notion of noise, Figure 6 shows three clusterings on datasets containing clusters of diverse shapes and sizes.

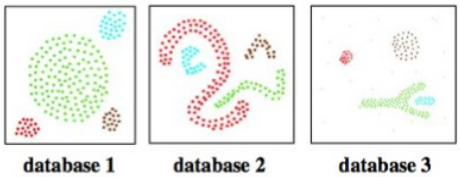

Fig. 6: DBSCAN Clusterings - (Ester, Kriegel, Sander, Xu '96 [4])

Nonetheless it has some problems on handling clusters of different density, in Figure 7 the two small clusters have a greater density than the big one in the center and some objects are left out of it.

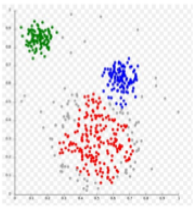

Fig. 7: DBSCAN and clusters different in density - (https://en.wikipedia.org/wiki/File:DBSCAN-Gaussian-data.svg)

V. DENCLUE

It represents an improvement of DBSCAN algorithm. Existing algorithms are somewhat limited on handling multimedia databases because of the presence of high-dimensional objects. DENCLUE [6] seems to work better than the previous algorithm by introducing a new clustering model with a strong mathematical basis that enables a compact mathematical description of clusters in high-dimensional dataset. The idea here is that each object y, belonging to a d-dimensional feature space F^d, can 'influence' the other objects in the neighborhood and so be described by the *influence function* $f_B^y \colon F^d \to \mathbb{R}_0^+$, defined in terms of a basic influence function f_B. In particular for the definition of specific influence functions we need a distance function $d \colon F^d \times F^d \to \mathbb{R}_0^+$ which determines the distance of two d-dimensional feature vectors.

$$f_B^y(x) = f_B(x, y) \text{ where } x \in F^d$$

Examples of basic function influence f_B are:

1. Square Wave Influence Function

$$f_{Square}(x, y) = \begin{cases} 0 & d(x, y) > \sigma \\ 1 & otherwise \end{cases}$$

2. Gaussian Influence Function

$$f_{Gauss}(x, y) = e^{-\frac{d(x,y)^2}{2\sigma^2}}$$

Where the parameter σ describes the influence of a data point in the database. Figure 8 shows an example with both Square Wave and Gaussian as basic influence function:

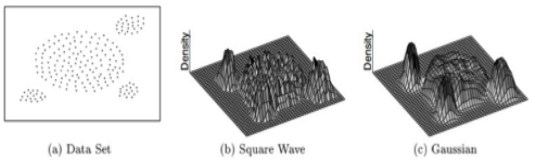

Fig. 8: Square Wave and Gaussian distance functions - (A. Hinneburgm D.A. Keim [6])

The *density distribution function* is the sum of the influence functions of all objects. Given N objects described by a set of feature vectors $D = \{x_1, ..., x_N\} \subset F^d$:

$$f_B^D(x) = \sum_{i=1}^{N} f_B^{xi}(x)$$

The authors defined two notions of clusters, *center-defined clusters* and *arbitrary-shape clusters*. Both use the notion of *density-attractor*. In mathematical terms, it represents the local maxima of the overall density function (density distribution function), an example, in Figure 9.

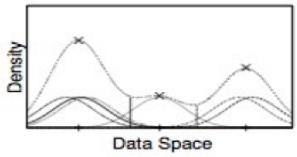

Fig. 9: Density-Attractors shown in the graph as crosses - (A. Hinneburgm D.A. Keim [6])

Center-Defined Clusters

A center-defined cluster with density-attractor $x*$, $f_B^D(x*) > \xi$ is the subset of the database which is density-attracted by $x*$. The parameter ξ is defined by the authors with the expression of how many clusters we are intended to get and more formally it describes when a density-attractor is significant.

Arbitrary-Shape Clusters

An arbitrary-shape cluster consists of a set of center-defined clusters which are linked by a path with a significance ξ. For the formal definition of the two above see [6].

As always with algorithms that require parameters, the goodness of these affects the quality of the clustering. Follow two examples that show how the values of ξ and σ give very different results, respectively in Figure 10 and Figure 11.

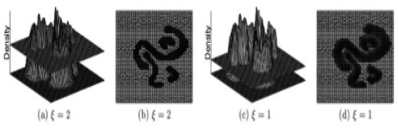

Fig. 10: Arbitrary-Shape Clusters for ξ =2 and ξ =1 - (A. Hinneburgm D.A. Keim [6])

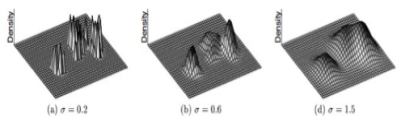

Fig. 11: Center-defined Clusters for different σ (0.2, 0.6, 1.5) - (A. Hinneburgm D.A. Keim [6])

Noise-Invariance

The resistance to the presence of noise of DENCLUE is stated as lemma, it in fact says that the number of density-attractors in the dataset without noise $D = D_c$ and in the one with noise $D = D_c + D_n$ is the same and the probability that the density-attractors remain identical approaches 1 for $||D_n|| \to \infty$. In short the density-attractors do not change as the noise level increases.

Idea of Proof

The *Idea of Proof* provided basically partitions the density function into, clusters and noise:

$$f^D(x) = f^{D_c}(x) + f^N(x)$$

where the density function of noise approximates a constant.

DENCLUE Algorithm

The algorithm relies on two things: a *Local Density Function* and a *CubeMap Data Structure*.

Given the function $near(x) = \{x_1 \in D | d(x, x_1) \leq k\sigma\}$ where k is a constant, the Local Density Function $f^D(x)$ is defined as follows:

$$f^D(x) = \sum_{z \in near(x)} f_B^z(x)$$

A *CubeMap* is a data structure based on d-dimensional hypercubes for storing the data and efficiently determining the density function.

The algorithm is composed by 2 steps:

- the first one applies a treatment on data with the aim to speed up the next step, in particular it constructs a map of the relevant portion of data to efficiently access the neighboring points of each point.
- the second step calculates the local density functions and from them the density-attractors with a method called *hill-climbing*.

Complexity

The first step requires a scan over the entire dataset so its complexity is $O(|D|)$; the second step performs again a scan and builds the tree of the populated map, so the complexity in the worst case is $O(|D| + |D| * log(|D|))$.

The overall complexity is hence $O(|D| * log(|D|))$.

Performance

The authors provide a performance comparison between DENCLUE and DBSCAN setting algorithms parameters with values such that they give similar results. It turns out that DENCLUE performs 45 times faster than DBSCAN on a dataset of 100 thousand points, results are plotted in Figure 12 and Figure 13.

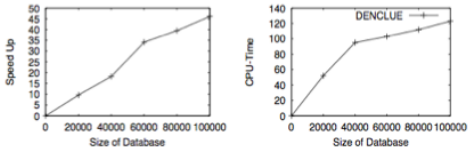

Fig. 12: Performance of DENCLUE - (A. Hinneburgm D.A. Keim [6])

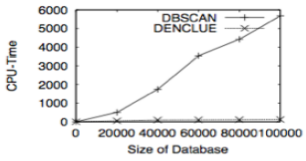

Fig. 13: Comparison of DENCLUE and DBSCAN - (A. Hinneburgm D.A. Keim [6])

VI. CHAMELEON

CHAMELEON [3] is a clustering algorithm that explores graph partitioning and dynamic modeling in hierarchical clustering. It can be applied to all types of data, as a similarity function can be constructed. In fact the algorithm merges two clusters according to the aggregate interconnectivity and closeness, more precisely two clusters are merged if the interconnectivity and closeness between them are high relative to the internal interconnectivity and closeness of objects within the clusters. It uses a k-nearest neighbor graph to represent the objects, capturing dynamically in this way the concept of neighborhood.

CHAMELEON has been designed to address the weakness of two similar algorithms: CURE and ROCK. The first one measures the similarity between two clusters based just on the similarity of the closest pair of the representative points belonging to different clusters while CHAMELEON considers the information about the aggregate interconnectivity of objects in the two clusters. ROCK, on the other side, while considering the interconnectivity of two clusters, ignores information about the closeness of those (it does not consider the value of the stronger edges across clusters) but only considers the aggregate interconnectivity across the pairs of clusters. Figure 14 shows an overview of the clustering's approach used by CHAMELEON.

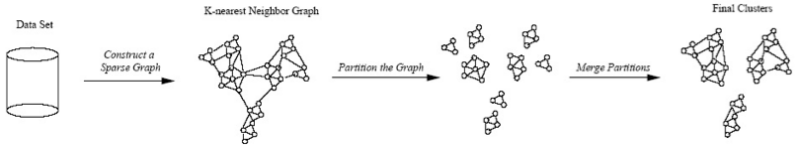

Fig. 14: CHAMELEON Overall Framework - (Karypis G., Eui-Hong Han, Kumar V [3])

Finally, the algorithm can be divided into three parts:

- Builds the k-Nearest Neighbor Graph from the similarity matrix.

- Partitions the graph to find an initially set of sub-clusters.
- Merges the clusters by their relative interconnectivity RI and relative closeness RC.

CONCLUSION

There are many aspects that contribute to the quality of the clustering pattern of the product. It must be first recognized that all algorithms based on density present the obvious advantage of not requiring a priori the number of clusters that prevents from getting disastrous results even on datasets with very simple structure of the cluster due to an erroneous estimate of that number. Furthermore, the goodness of the resulting clusters does not depend neither on the order in which points are processed, nor any type of initial choice (see k-means algorithm). These features are inherited from the fact that not the distance between the patterns is considered but their density, also the most important feature is that it follows from the ability to recognize clusters of arbitrary shape. In conclusion, one can predict that the combination of methods based on density, which possess the intrinsic characteristic of providing very high quality results, with the implementation techniques developed for other algorithms, consider the map of DENCLUE or use of BIRCH, constitutes a valid approach for the development of future algorithms that combine the best efficiency and accuracy.

REFERENCES

[1] R. Ng and J. Han, *Efficient and effective clustering method for spatial data mining*. VLDB'94 (CLARANS).

[2] T. Zhang, R. Ramakrishnan, and M. Livny, *BIRCH: an efficient data clustering method for very large databases.*, SIGMOD'96 (BIRCH).

[3] Karypis G., Eui-Hong Han, Kumar V, *Chameleon: hierarchical clustering using dynamic modeling.*, Harlow, England: Addison-Wesley, 1999 (CHAMELEON).

[4] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. *A density-based algorithm for discovering clusters in large spatial databases*, KDD'96 (DBSCAN).

[5] L. Kaufman and P.J. Rousueeuw., *Finding Groups in Data: an Introduction to Cluster Analysis*, John Wiley and Sons(PAM,CLARA), 1990.

[6] A. Hinneburgm D.A. Keim, *An Efficient Approach to Clustering in Large Multimedia Databases with Noise* , (DENCLUE).