

SmallSteps: An Adaptive Distance-based Clustering Algorithm

Gy. Koch, J. Dombi*

Abstract

In this article we propose a new distance-based clustering algorithm. Contrary to coordinate-based methods which work only in metric space, distance-based clustering methods can also operate on data sets that are in similarity space, where we do not know the exact attributes of the objects only their distances from each other or the similarity/dissimilarity between them. Since this similarity is usually given by a matrix of size $\frac{n*(n-1)}{2}$ where n is the number of objects, these algorithms have at least $O(n^2)$ time complexity. One of the latest distance-based method is Chameleon which was published in 1999. According to experiences Chameleon works well only on larger data sets, but the $O(n^2)$ time complexity makes the distance-based algorithms unsuitable for huge data sets. Thus we developed a new distance-based method (SmallSteps), which can handle relatively small amount of objects too. In our solution we are looking for connected graphs which have edges with a maximum weight computed on the environments of the objects. The method is capable to detect clusters with different shapes, sizes or densities, it is able to automatically determine the number of clusters and has a special ability to divide clusters into sub-clusters.

1 Introduction

Clustering is one of the most commonly used statistical methods. It can be seen as an unsupervised machine learning process where the algorithm has to divide a set of object ($X = \{x_1, \dots, x_n\}$) into different classes ($C = \{C_1, \dots, C_k\}$) such a way that similar objects should be in the same class while dissimilars should be in different classes. These classes are called clusters. Clustering algorithms are usually used to determine the underlying structure of the object set. Often it is useful not to create statistical measures or perform tasks on the whole set of objects, but after a cluster analysing doing it to the different clusters having similar objects which gives more accurate results.

This aim is usually described as maximizing the function [6]

$$Q_S^D(X, C) = Q_S(X, C) + Q^D(X, C)$$

*University of Szeged, Hungary; e-mail: [koch, dombi]@inf.u-szeged.hu

where $Q_S(X, C)$ means the similarity between objects in the same cluster and $Q^D(X, C)$ the dissimilarity between the objects in different clusters. There are no exact mathematical formulas for $Q_S(X, C)$ or $Q^D(X, C)$ that could be acceptable for most of the cluster analysing tasks.

Dividing objects into two groups by minimizing the maximum distance in the clusters, can be done by bicoloring a maximum spanning tree in $O(n^2)$ steps. On the other hand dividing the objects into more than two clusters is an NP hard problem [1]. Although segmenting into more than two partitions can be done by sequentially dividing clusters into two, it often does not give optimal solutions and fails on very simple examples, for instance when we like to partition these objects into 3 groups (see Figure 1).



Figure 1: Problems when dividing into two groups.

Detecting the number of clusters is also a very difficult problem, most of the clustering algorithms can only divide the objects into a number of clusters given by the user. Even there are special cases when appropriate clustering is not exist or the only good clustering is to order all of the objects into one cluster (e.g. integer coordinate pairs of the 2-dimensional space).

These are the main reasons why heuristical approaches are so popular among clustering techniques.

1.1 Two Main Types of Clustering Methods

Considering practical use there are two well-separable kinds of cluster analysing methods depending on the type of problems they have to solve.

- In the first group there are the faster algorithms having $O(n)$ complexity. These methods usually take the objects as points in the d -dimensional space (if the objects have d attributes) so we will refer to this group as coordinate-based methods. These methods can handle huge datasets with hundreds of thousands or even millions of records (e.g. calls of a telephone company, web log of an on-line store, shopping transactions of a supermarket, transfers of a bank, . . . etc.). Due to their quickness (note that clustering with these methods is faster than sorting the objects or finding the two closest/most similar objects) these algorithms give a rough segmentation and mostly recognize only spherical clusters. Using this group of clustering algorithms the user usually has to give the number of clusters a priori. Most known representatives of this group are Fuzzy C-Means and Kohonen Clustering Network.

- Methods in the second group have much lower speed, they have at least $O(n^2)$ complexity. These algorithms work on the distances/dissimilarities between the objects, which explains their time complexity. Since they are much slower than the coordinate-based ones, they are only good for smaller tables and most of these algorithms have to store the distance matrix, so their memory consumption can be quite large. The advantage of these methods is that they produce much better results. They may detect clusters with arbitrary shapes or size and determine the exact number of clusters. These algorithms are very closely related to shape recognition. For example they can recognize the arcs of a detached double spiral. One of the best algorithm in this group of methods is Chameleon, which was published in 1999 [5]. Chameleon has a very powerful recognizing capabilities, it detects clusters with arbitrary shapes and determines the number of clusters needed, still it has some serious drawbacks in practical use.

To emphasize the gap between the algorithms belonging to the two different groups let us show some calculations. Consider that a clustering algorithm segments 1000 objects in the 10-dimensional space in one second. If this algorithm belongs to the first group of methods, clustering 1 million objects takes approximately 17 minutes and with single precision real number representation it requires approximately 38 megabytes of memory while if the algorithm belongs to the distance-based group clustering 1 million records takes more than 1.5 weeks and the size of the distance matrix is 1.8 terabytes.

In this paper we will propose a new cluster analysing algorithm for the second group of methods. SmallSteps is a distance-based method and overcomes the difficulties of Chameleon while it keeps all the good features of it.

2 The Way Chameleon Works

Chameleon is a new distance-based clustering method, which was published in 1999 by G. Karypis et al. [5]. It takes the objects as vertices of a graph and the distance/similarity between the objects as the weights of the edges of the graph. Chameleon has two main phases. In the first phase it creates small sub-clusters and merges them together into clusters in the second phase.

The sub-cluster creating part is done by a hypergraph-partitioning algorithm. Since partitioning a graph into a large number equally sized subgraph is an NP hard problem, Chameleon uses a heuristic technique called multilevel graph partitioning [3][4].

To merge these sub-clusters Chameleon calculates special measures. The relative closeness is responsible to merge only clusters that have uniform density among the objects in the same cluster and relative inter-connectivity is for maintaining the similar inter-connectivity in the clusters.

Chameleon can work according to two different schemes.

- In the first scheme Chameleon introduces two technical parameters as thresh-

olds. One for the relative closeness and one for the relative inter-connectivity. Pairs of clusters, whose calculated measures are above these thresholds, will be merged. Chameleon may terminate if there are no pairs of clusters whose relative closeness and relative inter-connectivity is above the thresholds or these parameters may be relaxed during the merging phase allowing Chameleon to create only one big cluster.

- According to the second scheme Chameleon uses a function to combine the relative closeness and relative interconnectivity. This function is usually has the form

$$RI(C_i, C_j) * RC(C_i, C_j)^\alpha$$

where $RI(C_i, C_j)$ and $RC(C_i, C_j)$ are the relative inter-connectivity and relative closeness between cluster C_i and C_j and α is a user specified parameter to increase the importance of one of the measures. The result of this scheme is that we get one big cluster and the order of the mergings.

The Chameleon method can handle nearly arbitrary shape of clusters and can detect quasi-automatically the number of clusters.

Experiments show that if the number of records is low, Chameleon works not well. The problem is that with the heuristical graph partitioning algorithm, Chameleon at first divides the objects into large number of relatively small sub-clusters which still have to be big enough to be able to correctly compute their internal measures (e.g. the relative inter-connectivity). If there are only small numbers of elements (i.e. less than 1000), very often one or more sub-clusters have intersections with more than one real cluster. Since it has no error correction, this means that these clusters will be connected and the algorithm cannot separate them later. Remember that methods in the distance-based group are best for relatively small tables.

Another problem with Chameleon is that it needs two technical parameters to detect the number of clusters, which are very hard to interpret for the user and in practical use it is a serious disadvantage.

Chameleon's graph partitioning algorithm is a non-deterministic procedure, which implies that the whole method is also non-deterministic.

Mainly these problems above inspired us to develop a new distance-based clustering method which is able to overcome these difficulties and gives a result of the same high quality as Chameleon does.

The developed SmallSteps

1. can be used for datasets of various size,
2. it needs no parameters, still automatically detect the number of clusters and can recognize clusters with any shape and
3. always gives the same result for a given set of objects.

3 SmallSteps: The New Procedure

3.1 The Algorithm of SmallSteps

The solution of SmallSteps, similarly to Chameleon's, comes from graph theory. The objects are considered as vertices of a graph and the distances between the objects are the weighted edges of the graph. The clusters are special connected graphs having edges with weights less than a cluster depended threshold called δ . These thresholds are recalculated continuously and unique to every cluster.

In the following part denote $|C^{(t)}|$ the number of clusters in the t^{th} iteration step, $C_k^{(t)}$ the k^{th} cluster in the t^{th} iteration step and $\delta_k^{(t)}$ its δ where $1 \leq k \leq |C^{(t)}|$. Denote $|C_k^{(t)}|$ the number of objects in cluster $C_k^{(t)}$.

SmallSteps has three main phases.

- The first phase is the initial cluster-forming phase. At the beginning of this phase ($t = 0$) all the elements belong to one cluster ($C_1^{(t)}$) and the initial delta ($\delta_1^{(t)}$) is calculated. Then in an iterative process the algorithm begins to search for special connected graphs with edge weights less than $\delta_k^{(t)}$ and the thresholds of the newly created clusters are recalculated. We repeat this process until a previously given number of iteration steps is reached. The δ 's are responsible for creating clusters of upperly bounded, nearly uniform densities, so it is very important to start every iteration with clusters having smaller δ 's otherwise neighbouring clusters with smaller densities (higher deltas) would incorporate the denser ones.

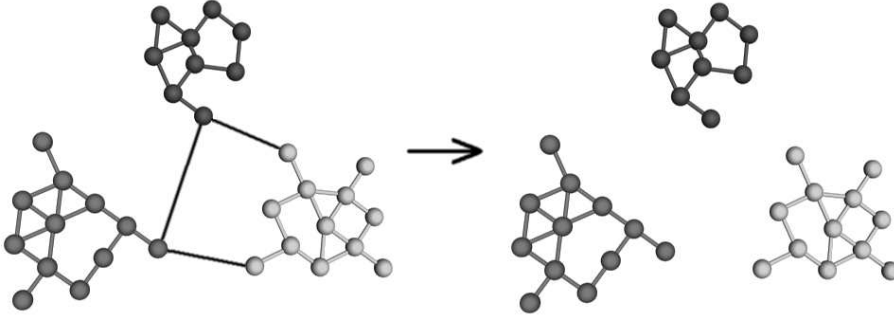


Figure 2: Applying a smaller δ , the cluster falls into sub-clusters.

The $\delta_k^{(t)}$ for the k^{th} cluster in the t^{th} iteration step is calculated based upon the average distance between the objects of the cluster $C_k^{(t)}$ and their closest

neighbours from the same cluster.

$$\delta_k^{(t)} = \frac{\sum_{x_j \in C_k^{(t)}} f\left(\frac{1}{r} \sum_{l=0}^r d(x_j, x_{j(l)})\right)}{|C_k^{(t)}|}$$

where $x_{j(l)}$ is the l^{th} closest element to x_j and $d(x_j, x_{j(l)})$ is the distance/dissimilarity between x_j and $x_{j(l)}$. r is the degree of δ , i.e. the number of neighbours involved in the calculation and f is a monotone function which determines the way the average distance is taken into consideration. In the case of Figure 4, 5 and 6 f was a linear transformation.

The degree of the δ 's (r) controls the style SmallSteps works. If only one or a few neighbours are considered then the result of SmallSteps is very close to the result of a shape recognition algorithm.

Clusters are not always segmented into sub-clusters during this first phase as shown on figure 2, but merging is also possible, still it is done very rarely.

Time complexity of this phase is $O(n^2 t_{max})$, where n is the number of objects. t_{max} is the number of cluster forming iterations and it is independent of the number of objects.

- The second phase is a merging phase. It makes decisions on merging of some of the initially formed clusters based on their delta, size and distance. Although the first phase of SmallSteps is so powerful that usually there is no need to merge clusters in this second phase, our experiments showed that these merging calculations worth to do them.

Merging objects has $O(|C^{(t_{max}+1)}|^2)$ time complexity, where $|C^{(t_{max}+1)}|$ is the number of clusters created during the first phase.

- Handling of outlier or noisy objects is done in the third phase. Datasets often have outlier or noisy objects, which not belong essentially to any clusters and could be left uncategorized, but in most cases the user want to order all of the objects to a cluster. If the user accepts outlier objects then this third phase should be skipped.

Depending on the structure of outlier elements, in SmallSteps the following two operations can be done with them:

- Let them form new clusters.
If there is a group of outlier objects that are far enough from the existing clusters, have enough elements and are close to each other to form a new connected graph with edge weights less than their special delta then these objects are allowed to create a new cluster.
- Order them to an existing cluster.

The order of outlier objects influence the result so we have developed different strategies:

- The simplest is to choose always those outlier objects that are the nearest to an existing cluster.
- We may follow a postponing strategy and choose those objects first whose classification are the easiest, i.e. who are close to a cluster but far from the other clusters.
- According to the BestFit method always those outlier objects are classified at first whose distance to a cluster best fit to the δ of the cluster. This method tries to maintain the uniform density of the clusters.

Time complexity of the last phase is $O(n_{\text{outlier}} * \max\{n_{\text{outlier}}, n_{\text{classified}}\})$, where n_{outlier} and $n_{\text{classified}}$ are the numbers of outlier and classified elements at the beginning of the third phase, respectively.

The overall time complexity of SmallSteps is

$$O\left(n^2 t_{\max} + \left|C^{(t_{\max}+1)}\right|^2 + n_{\text{outlier}} * \max\{n_{\text{outlier}}, n_{\text{classified}}\}\right) = O(n^2)$$

3.2 Handling Bounding Objects

If two clusters are in touch by only a few objects then by searching for connected graphs the algorithm will find that these clusters could form one cluster. To avoid such aggregation it is useful to find the boundary objects of the clusters and if two clusters are connected with only bounding objects then the algorithm should not merge these clusters. The algorithm can recognize the bounding object by counting their intra-cluster neighbourhood which is the number of objects from the same cluster that are closer to the object than the δ of the cluster. The bounding objects will have fewer neighbouring objects than the inner ones.

3.3 Inner Analysis of a Cluster

SmallSteps provides a useful additional feature. If the user want to analyse a cluster in more detail, he/she can specialize this cluster by segmenting it into sub-clusters. The segmentation is done by iteratively decreasing the original delta of the cluster and searching for connected graphs with edge weights less than its new delta. The specializing procedure terminates when the algorithm is able to segment the cluster into sub-clusters of acceptable size or when it turns out that such a segmentation is not possible (see Figure 3).

4 Results

The number of clusters evolves automatically during the three phases (mainly during the first phase) so no user interaction is needed for giving this number.

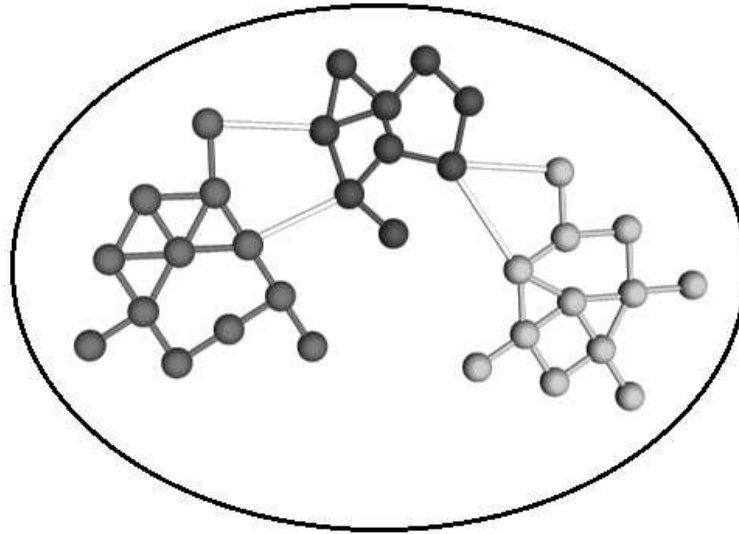


Figure 3: Segmenting a cluster into sub-clusters

Instead of relative inter-connectivity and relative closeness, SmallSteps calculates δ 's to perform cluster forming. These δ 's can be computed from only a few objects, which means that SmallSteps works well on smaller tables too.

SmallSteps can recognize clusters with different densities (see figure 4) because every cluster has its own δ and since SmallSteps searches for connected graphs it can recognize clusters with arbitrary shape or size (see figure 5).

The algorithm of SmallSteps is deterministic so it will give the same output for a given set of objects.

Since the outlier or noisy objects are handled in the last phase SmallSteps is not too sensible to this kind of objects (see figure 6).

While Chameleon is a kind of greedy, agglomerative clustering procedure and never corrects the errors made during its merging process, SmallSteps rather resembles to a divisive clustering method, but it has merging steps too and it has some error-correcting feature in all the three phases. Practical experiments showed that, among hierarchical algorithms without error correction, the divisive methods usually outperform the agglomerative ones because divisive methods need fewer steps to create the segmentation [2].

References

- [1] Charles J. Alpert, Andrew B. Kahng *Splitting An Ordering into a Partition to Minimize Diameter*

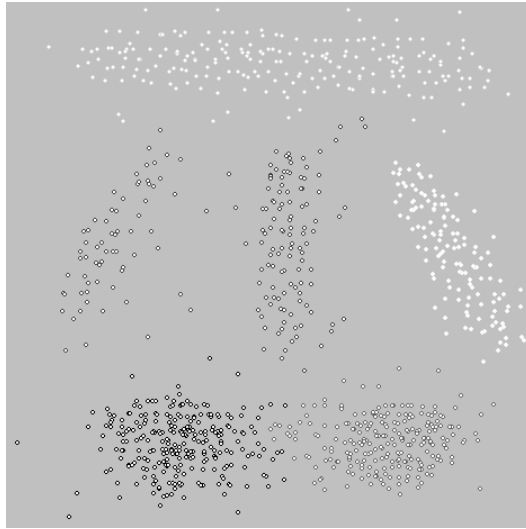


Figure 4: Detecting elliptical clusters with different densities and sizes

- [2] A. Guénoche - P. Hansen - B.Jaumard *Efficient Algorithms for Divisive Hierarchical Clustering With the Diameter Criterion*, Rutcor Research Report 16-90, May, 1990
- [3] George Karypis, Vipin Kumar *Analysis of Multilevel Graph Partitioning*
- [4] George Karypis, Vipin Kumar *Multilevel k -Way Partitioning Scheme for Irregular Graphs*, Journal of Parallel and Distributed Computing, 48(1): 96-129, 1998
- [5] George Karypis, Eui-Hong Han, Vipin Kumar *Chameleon: Hierarchical Clustering Using Dynamic Modelling*, IEEE Computer, 1999
- [6] J. W. Owsinski *Clustering - modelling, capabilities, limits, applications*, Control and Cybernetics, vol 24 (1995) No. 4

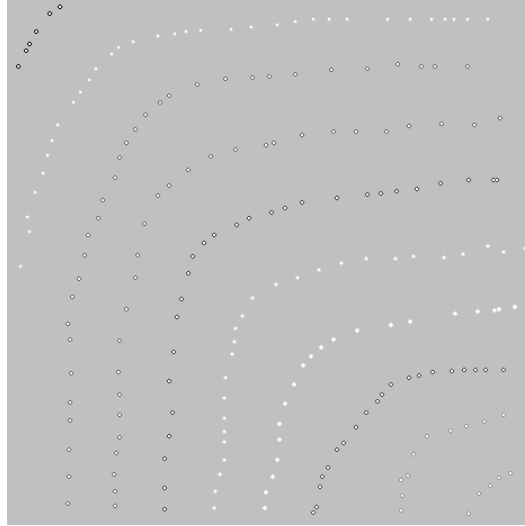


Figure 5: SmallSteps is very close to shape detection

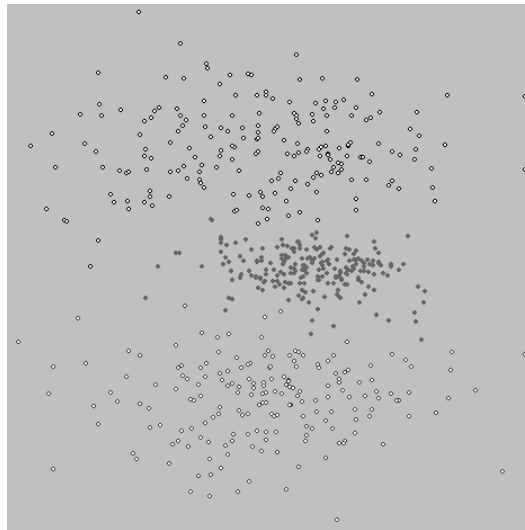


Figure 6: Three clusters with noisy elements