Production Systems and Information Engineering
Volume 6 (2013), pp. 15-26

EFFICIENCY ANALYSIS OF QUALITY THRESHOLD CLUSTERING ALGORITHMS

LÁSZLÓ BEDNARIK
University of Miskolc, Hungary
Department of Information Technology
laszlobednarik@gmail.com

LÁSZLÓ KOVÁCS
University of Miskolc, Hungary
Department of Information Technology
kovacs@iit.uni-miskolc.hu

[Received January 2012 and accepted September 2012]

Abstract. An important aspect of clustering is to provide a good intra-cluster similarity. Most of the traditional methods do not consider this aspect and they generate weak clusters from this viewpoint. The paper presents a survey of the two dominant candidates for quality threshold clusterings, the QT and BIRCH methods. Besides the analysis, a new variant of BIRCH method which can provide a better performance is proposed.

Keywords: clustering, quality threshold clustering, BIRCH, genetic algorithm

1. Introduction

Data clustering is an important and widely used technique in data analysis and data mining. The goal of clustering is to split the set of elements into subsets where the elements of the same group are more similar to each other than the elements from different groups. The process of clustering usually includes the following steps: select an appropriate representation form of objects; define an appropriate similarity function; determine the appropriate clustering algorithm and parameters; execute the algorithm and interpretat the results. As the clustering problem uses an unsupervised learning algorithm, there are many subjective parameters in the process. One of the key parameters is the aspect of similarity: at which value of similarity can the objects be assigned to the same cluster. There are many standard methods in the literature for clustering. The most widely used standard methods are hierarchical clustering [1], partitioning clustering [2], hybrid method [3], incremental or batch methods, monothetic vs. polythetic methods [4], crisp and fuzzy clustering [8].

Although the literature on clustering is very rich, there is an aspect that has not attracted much interest in recent years. The aspect in question is the similarity threshold within a cluster. Based on the informal definition of clustering, the elements within a cluster should be more similar to each other than the elements of different clusters. Considering the standard methods it can be seen that these methods do not fulfill this requirement at an optimal level. The HAC method [9], for example, can generate arbitrary large clusters where the distance between two elements of the same cluster may be much higher than the distance related to an element of another cluster (see Fig.1).

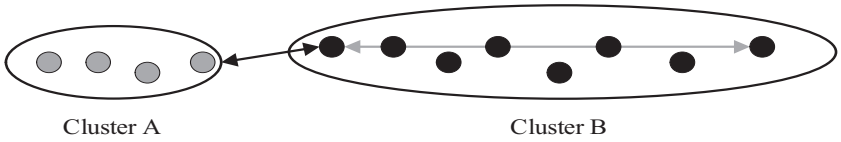

Figure 1. Inter-cluster similarity is higher than intra-cluster similarity

The majority of standard methods use a greedy approach to build up a cluster: while the distance between the candidate and the current cluster is below a threshold value, the cluster is extended with new elements. The threshold criteria are usually independent of the actual size of the current cluster. As a result, the similarity between two points of the same cluster may be arbitrarily low.

The goal of the investigation is to find clustering methods preserving the distance. In order to meet this requirement, the following two constraints are defined on the clustering model:

- for every object pair with a smaller distance than a threshold, there exists a cluster containing both elements of the pair.
- for every object pair with a larger distance than a threshold, there is no cluster containing both elements of the pair.

The first criterion ensures that in examining the target clusters, all object pairs similar to each other can be found. The second constraint says that a cluster does not contain dissimilar objects, it contains only similar objects.

In the literature, there are two dominant variants providing an intra-cluster similarity: the Quality Threshold method and the BIRCH method. The first variant can provide a higher level of quality but it requires a higher execution cost. The second one is a good and robust solution in the case of huge data sets. The paper provides a comparison of these methods and suggests some modifications on the BIRCH algorithm to create an improved intra-cluster quality for large databases.

2. Overview of clustering algorithm with quality threshold

2.1. QT algorithm

The QT (Quality Threshold) clustering method [11] ensures that the distance between any two elements within a cluster should be below a given threshold. The algorithm uses two input parameters: the first parameter is the maximum distance diameter and the second is the minimum cluster size. The diameter is defined in the following way:

$$d = \max_{i,j} \left\{ \sqrt{(x_i - x_j)^2} \right\}. \tag{2.1}$$

The size of the cluster denotes here the number of elements within the cluster. The main steps of QT clustering are the following:

1. Generating candidate clusters for each element where the candidate cluster is built up with a greedy algorithm. Taking an element y, the cluster contains all elements closer to y than the maximum radius.

2. The candidate cluster with the maximum size is selected as a true cluster. The elements of this cluster are removed from the pool, the membership of the remaining candidate clusters is updated.

3. If the largest remaining cluster has a greater size than the minimum limit, go to step 2, otherwise terminate the algorithm.

The QT algorithm generates non-overlapping clusters where some elements remain outside of clusters as outliers. The output of clustering is a set of clusters of limited diameter, thus the similarity values between the elements are above a given threshold.

Considering the execution cost, the algorithm contains three embedded loops. The outer loop runs on the selection of real clusters, the loop in the middle runs on the candidate clusters and the inner loop runs over the elements to generate the candidate clusters. The cost value can be given by

$$O\left(\frac{N^2}{L} D\right),$$

where

N : number of elements in the data set,

L: average size of a cluster,

D: the cost of distance calculation.

Besides the base variant in the literature, there can be found some improved versions using some special techniques such as parallelism [12].

2.2. BIRCH algorithm

The BIRCH (Balanced Iterative Reducing and Clustering using Hierarchies) algorithm [5] is of great importance as a fast and efficient pre-clustering method. Each resulting cluster is represented by one single point for the task-specific further clustering algorithms. The BIRCH algorithm guarantees that the distance of points in sets created is smaller than a threshold given in advance. The threshold is the most important input parameter of the BIRCH algorithm. By decreasing the threshold it can be guaranteed that points in sets are close enough to the centre of the set. Therefore the quality of clustering can be increased.

The structural information of the BIRCH algorithm is stored in Clustering Feature (*CF*) data triplet. The strength of *CF* is to the calculation of core clustering features using only simple arithmetic operations without analysing the individual elements. The structure *CF* is given by the following formula:

$$CF = \left\{ N, \overrightarrow{LS}, SS \right\}.$$

where

N: number of points in the set represented by the given *CF*,

$\overrightarrow{LS}$: d-dimensional linear sum of points in the set,

SS: d-dimensional square sum of points in the set.

The BIRCH algorithm organizes the points to be clustered in a balanced B+ tree. Each leaf-node stores points (or sets of points). Moreover, each leaf-node contains a pointer to the following node and a pointer to the preceding node in the tree to reach them fast. During the insertion operation, if the quality threshold condition does not hold, then, points in the given leaf-node are reassigned into two subsets. In the split operation the two furthest points of the set are used as nuclei of the new clusters. Then all the other points in the original set are moved into the leaf-node whose nucleus is closer to it.

Another core parameter of the algorithm is the branching factor. If the number of child nodes of a given non-leaf node exceeds the maximal branching factor, then the set is split into two parts. The two child nodes which are the furthest from each other are selected as the nuclei of the new sets. Afterwards all the other child nodes of the given non-leaf node move into the non-leaf node whose nucleus is closer.

If B denotes the maximal number of child-nodes of non-leaf nodes, M denotes the maximal size of the tree, then the cost function of the method can be given as [10]:

$$O(d \cdot N \cdot B \cdot (1 + \log_B M)) \quad .$$

Considering all components of the algorithm (e.g. reconstruction), the total time cost of the BIRCH algorithm can be calculated by the following formula [10]:

$$O(d \cdot N \cdot B \cdot (1 + \log_B M) + \log_2 \frac{N}{N_T} \cdot d \cdot (ES - 1) \cdot B \cdot (1 + \log_B M)) \quad . \qquad (2.2)$$

3. Optimal selection of parameters for BIRCH algorithm

Test results of clustering on the base BIRCH algorithm reveal that the time requirement of the procedure considerably depends on parameters of the threshold value and of the maximal branching factor. The test results show that the dominating factor is the threshold value, while the cost is considerably influenced also by the maximal branching factor *CF* of the tree. Tests performed also revealed that if the threshold value parameter is lower than the optimal value then the number of sets resulting by BIRCH algorithm is increased exponentially. The exponential increase in the number of sets results in an exponential increase in the cost of the post-processing algorithms. If the threshold value parameter is larger than the optimal value, then the number of points put into a cluster is increased, which requires a continuously increasing extra cost during insertion and reconstruction operations. If the CF value is too low, the depth of the tree increases exponentially. Administrating the increased number of nodes (creation, decomposition, memory usage, etc.) causes an increased total cost.
The goal function *(C)* is a function of two parameters: the threshold value *(T)* and the maximal branching factor *(B)*: *C=f(T,B)*. The best known representatives of local searching algorithms are: hill-climbing search [6], simulated annealing [7], local beam search, and genetic algorithm. In order to perform parameter optimization we have developed a combination of genetic algorithm and hill-climbing search algorithm.

The formal description of the applied hill-climbing search algorithm is as follows:

1. The actual point is a point given by randomly chosen coordinates *(T,B)*.
2. Determining the cost belonging to the actual point by carrying out the clustering procedure with parameters represented by the actual point.

3. Determining the neighbours of the actual point by increasing and decreasing the coordinates of the actual point by step ε.
4. Determining the costs belonging to the neighbours of the actual point by carrying out the clustering procedure for each neighbour of the actual point with parameters represented by the given neighbour.
5. If the cost of the neighbour having the smallest cost around the actual point is not smaller than the cost of the actual point, then terminate the algorithm.
6. Let the actual point be the point represented by its neighbour with the smallest cost. Take step 3.

The formal description of the algorithm reveals that in order to determine the optimal parameters the same clustering task needs to be performed repeatedly with different parameters. In a minimal case this requires performing the clustering task four times. Hence it is obvious that a clearance of the cost of clustering can only be expected when a relatively great number of samples with the same features are clustered. By denoting the cost of clustering without optimization by S, and the cost of clustering with optimization by O and the cost of finding the optimum by M, then after the i^{th} clustering the cost of clustering without optimizing is $i \cdot S$, and the cost of optimizing carried out with parameters T, B obtained by optimization is: $i \cdot O + M$. The gain of optimization can be given as

$$j \cdot S - (j \cdot O + M) \,. \tag{3.1}$$

To show the results of the simple hill-climbing method, a test was performed on a word-set containing 10,000 words from the Computer science book written by Gábor Kovács (Hungarian Electronic Library). The measured cost values are shown in Figure 2.

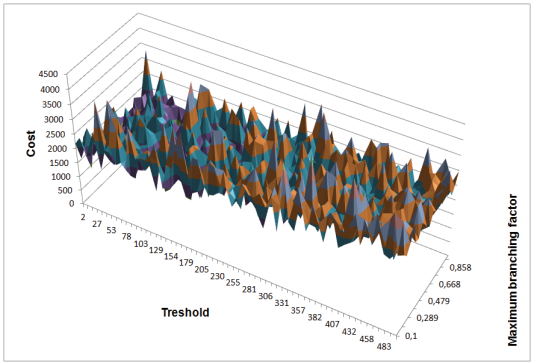

Figure 2. The value of cost calculated at each point of the domain in the example considered

The quality threshold values ran in the interval [0.1, ..., 1] with distance 0.05. For the *CF* parameter, 60 values at an equal distance from each other in interval [2, ..., 500] were chosen. As Figure 2 shows, there are many similar local optimum positions in the problem domain. In order to find a good approximation of the global optimum, the simple hill-climbing method was extended with a stochastic element, namely with a genetic algorithm module.

A group of discrete candidate points existing simultaneously creates a population. Each discrete point of the starting population is generated by coordinates assigned randomly. Each generated discrete point is the starting point of a hill-climbing search algorithm. After the local optimization process, the coordinates of each discrete point of the population are modified to the coordinates of the local minimum obtained by the search. Therefore each discrete point of the population gets into a local minimum. According to the cost of the discrete points of the population obtained in this way, the average cost that describes the population can be determined. If the average cost of the n^{th} population already exceeds the average cost of the population *(n-1),* then the algorithm terminates and the best discrete point in the population *(n-1)* is considered to be the best solution obtained by the optimization. If the n^{th} population is the first population or its average cost is smaller than or equal to the average cost of the population *(n-1),* then according to the discrete points of the n^{th} population a new population *(n+1)* is generated.

The algorithm uses the usual genetic operators such as selection, crossover and mutation, to build up the next generation. During the selection operation, a number of the best discrete points in the n^{th} population are placed into the new population without any change. For the generation of the rest of the points, the following possibilities are carried out:

- Crossover of points: denoting the coordinates of the selected discrete points by (t_i,b_i), (t_j,b_j), then the coordinates of the new point are (t_i,b_j).
- Mutation: denoting the coordinates of the chosen point by (t_i,b_i), the coordinates of the new point are $(t_i+\varepsilon_1,b_i+\varepsilon_2)$.
- A new point is generated randomly, independently of the coordinates of the selected elements.

The optimal values of parameters *T* and *B* are given by the coordinates of the point representing the lowest cost of the previous population.

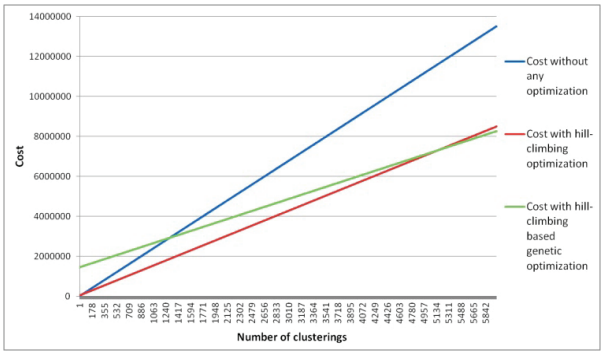

Figure 3. Dependence of costs obtained with and without optimization on the number of clustering

4. Analysing test results of clustering after BIRCH

During the test of the k-means post-clustering method after the BIRCH pre-clustering, we analysed the relationship between the following six parameters:

- the maximal distance (threshold) between points in the leaf-nodes of the BIRCH tree,
- the maximal branching factor of non-leaf nodes of the BIRCH tree,
- the expected number of clusters in a k-means algorithm based on the BIRCH algorithm,
- the number of clustering points,
- the number of alignments of points to be clustered (focus points),
- the dispersion of normal distribution of points to be clustered around focus points.

The first two parameters have a direct effect on the behaviour of the BIRCH algorithm. With the third one the number of clusters expected from the k-means can be set. The last three parameters relate to the problem domain. During tests always only one of the parameters was changed in order to be able to show its effect on the features of the algorithm. The following features were investigated:

- the time requirement of clustering (together with the k-means algorithm),
- the depth of the BIRCH tree,
- the number of leaf-nodes of the BIRCH tree (the number of sets created by the BIRCH algorithm),
- the number of steps in re-arranging the k-means clustering based on the BIRCH algorithm.

Results of the tests according to the above can be seen in the diagrams below.

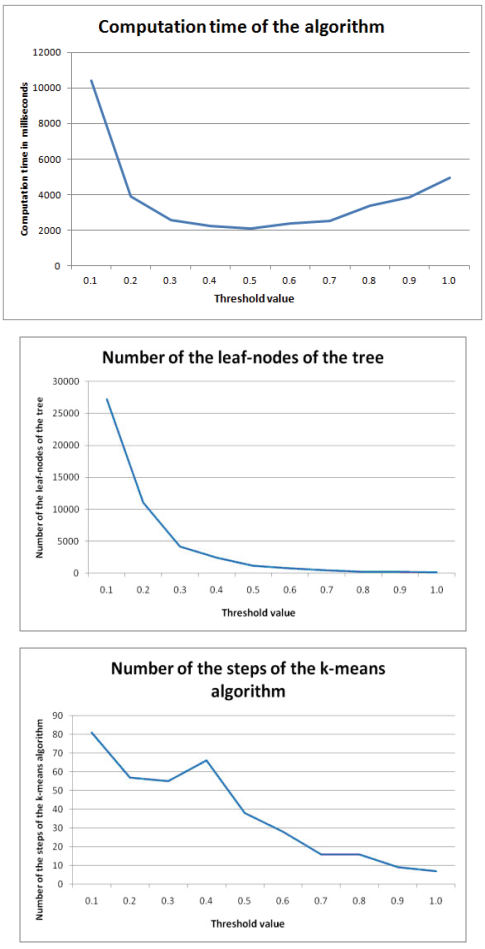

Figure 4. Analysing the effect of the threshold value

According to the test results, we can see that the number of leaf-nodes decreases exponentially with the increase of the threshold value. Therefore the k-means algorithm needs to cluster considerably fewer sets, hence the time requirement of clustering also decreases exponentially. By increasing the value of the threshold above a certain level, too many points are gathered in leaf-nodes and during the separation of sets a considerable number of points needs to be analysed. This effect can be shown in the increase of time consumption.

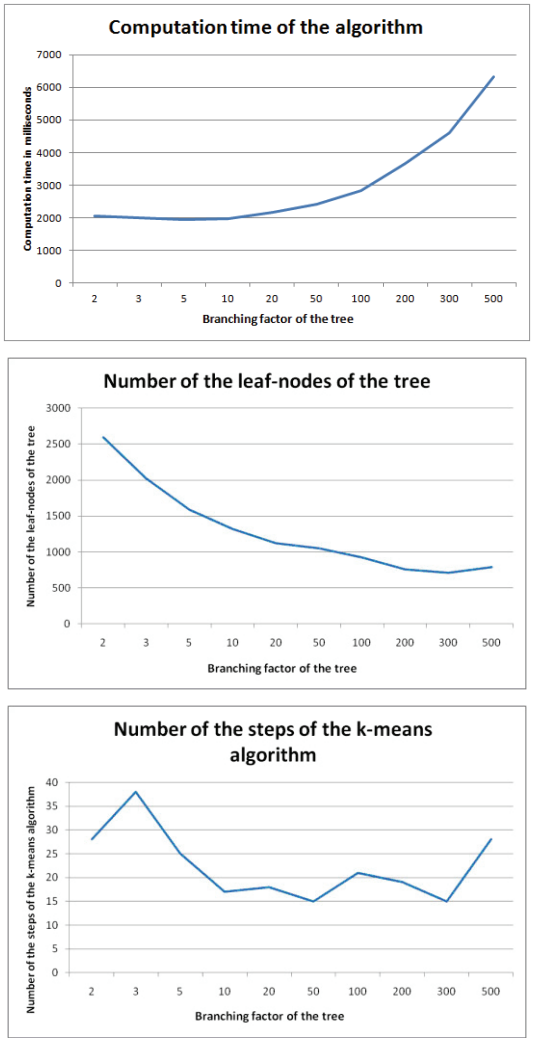

Figure 5. Analysing the effect of the maximal branching factor

By increasing the branching factor of the tree, the depth of the tree decreases exponentially. As in the case of a larger number of branchings, a considerably larger number of child nodes must be processed. This procedure results in an obvious increase in the rate of speed. Also, it is obvious that changing the branching factor does not have any effect on the function of the k-means algorithm.

5. Conclusions

An important aspect of clustering is to provide a good intra-cluster similarity. The BIRCH algorithm is a standard tool used to perform pre-clustering on large data sets using a quality-threshold constraint on the clusters. As the efficiency of the method depends on such parameters as cluster threshold and branching factor, the optimization of the parameters is a crucial step to reduce the costs of clustering operations. The proposed method, hill-climbing with genetic algorithm can be used efficiently to optimize the cluster threshold and branching factor parameters of the BIRCH algorithm. The test results show that a significant cost reduction can be achieved using the proposed optimization method.

Acknowledgements

This research was carried out as part of the TAMOP-4.2.1.B-10/2/KONV-2010-0001 project with support by the European Union, co-financed by the European Social Fund.

REFERENCES

[1] CRACRAFT, J., DONOGHUE, M.: *Assembling the tree of life: Research needs in phylogenetics and phyloinformatics*. Report from NSF Workshop, Yale University, July 2000.

[2] DAY, W.: Complexity Theory: *An Introduction for Practitioners of Classification, Clustering and Classification*, World Scientific Publ., 1992.

[3] MURTY, M., KRISHNA, G.: *A computationally efficient technique for data clustering*, Pattern Recognition, Vol. 12., 1980, pp. 153-158.

[4] SALTON, G.: *Developments in automatic text retrieval*, Science, Vol. 253., 1991, pp. 974-980.

[5] TIAN, Z., RAGHU R., MIRON L.: *BIRCH: A New Data Clustering Algorithm and Its Applications*, 1997.

[6] BART, S., CARLA P. G.: *Hill-climbing Search*, Cornell University, Ithaca, New York, USA, (Jan 15, 2006), pp. 333-336.

[7] SCOTT, K., CHARLES, D. G., MARIO, P. V.: *Optimization by Simulated Annealing*, *Science*, New Series, Vol. 220, No. 4598. (May 13, 1983), pp. 671-680.

[8] RUSPINI, E.: *A new approach to clustering*, Information Control, Vol. 15., 1969, pp. 22-32.

[9] MANNING, C., RAGHAVAN, P., SCHÜTZE, H.: *Introduction to Information Retrieval*, Cambridge University Press, 2008.

[10] TIAN, Z., RAGHU R., MIRON L.: *BIRCH: An Efficient Data Clustering Method for Very Large Databases.* Proc. of the 1996 ACMSIGMOD International Conference on Management of Data, Montreal, Canada, pp. 103–114, June 1996.

[11] HEYER, L., RAMAKRISHANAN, R., LIVNY, M.: *BIRCH: An Efficient Data Clustering Method for Very Large Databases*, Genome Research, Vol 9., 1999, pp. 1106-1115.

[12] MOCIAN, H.: *Survey of Distributed Clustering Techniques*, Internal Research Project, http://www.horatiumocian.com/papers/Distributed_Clustering_Survey.pdf, 2009.