Enhanced clustering

Gianluca Bortoli
DISI - University of Trento
Student id: 179816
gianluca.bortoli@studenti.unitn.it

Martin Brugnara
DISI - University of Trento
Student id: 182904
mb@disi.unitn.eu

ABSTRACT

Keywords

Big data, Data mining, Clustering, Streaming, Parallel computation

1. INTRODUCTION

The clustering problem consists in grouping together data items that are "similar" to each other such that the inter-group similarity is high, while the intra-group one is low.

This challenge has received a lot of attention over the years. According to Jane and K. [8], the first specific study appeared in 1954 [4] and by now thousends of solutions have been proposed.

Data clustering has been used in many different disciplines, such as data mining [7], statistics [14, 1] and machine learning. The most common usages aim to gain insight to data (underlying structure) and for summarizing it through cluster prototypes (compression).

The concept of similarity varies a lot in the different contexts it can be applied. For example the Euclidean distance (L2) can be used when dealing with continuous values or the Jaccard similarity index, which computes similarity for generic sets of elements. Nonetheless, the underlying algorithm is agnostic with respect to the similarity measure that is applied to compute a distance between the elements in the data. Clustering can be also viewed as identifying the dense regions of the probability density of the data source [3].

The literature suggests two different approaches: *partitional* and *hierarchical*.

The first strategy needs some parameters to be set and known in advance. For example the *k-means*, which is one of the most popular and adopted algorithm, requires the number of cluster to be found (K).

The latter can be implemented both in a top down (divisive) or a bottom up (agglomerative) manner. Initially, the divisive algorithm treats all data as a single big cluster and later splits it until every object is separated [10]. On the contrary, the agglomerative starts considering each "element" as a *singleton* (a cluster composed of one element). Next, the most similar clusters are collapsed together until only one big cluster remains. Implicitly the merging order defines a clear hierarchy among the intermediate representations (dendrogram).

Clearly, both the above mentioned approaches to the clustering problem have their disadvantages. The partitional methods require prior knowledge on the data distribution, while the hierarchical ones imply the user interaction to decide the dendrogram's cut height. For a complete list of the various clustering tequnique flavours refer to Jan *et al.* review [8]. However, a solution that does not suffer from those is still an open challenge.

In this work we propose a completely autonomous system which merges the two strategies to overcome their weaknesses, meaning that it satisfies the following *Data Mining Desiderata*:

1. **streaming**: require one scan of the database, since reading from secondary memory is still the most costly I/O operation. Moreover the analysis can be stopped and restarted without having to re-process the whole data ("stop and resume" support). This property adds the capability to incorporate additional data with existing model efficiently (incremental computation).

2. **on-line "anytime" behaviour**: a "best" answer is always available at any time during the computation phase.

3. **limited memory**: the tool must work within the bounds of a given amount of main memory (RAM).

2. RELATED WORK

The most popular and simplest partitional algorithm is **k-means** [12]. Like every other solution belonging to this class, it requires the number of clusters to be found (k) to be known a-priori. Unfortunately, there exists no mathematical formula to compute such parameter in advance, requiring the test to be run multiple times with different values in order to find the best solution according to some criterion (*e.g.* the Schwarz Criterion [13]). This algorithm is based on the notion of distance and it is usually employs the Euclidean one. The resulting division into clusters can be also seen as a lossy compression of the points towards the centroids, which are points identifying the clusters. The main idea behind the k-means consists in minimizing an objective function. Usually the Mean Squared Error (MSE) is chosen, where the error is defined as the distance between each point and the centroid of the cluster it is assigned to. This process is iterative; initially k points are identified to be the centroids, then all the points are assigned to the nearest centroid (locally minimazing the MSE) and finally the centroids are recomputed as the barycenter of the clusters. The procedure continues until the convergence of the centroids' locations. A noteworthy aspect is that the bootstrap phase, namely the initial centroids identification, highly influences the outcome.

Different centroids usually lead to different results, since the algorithm is designed to find a local optimum. Several options for the bootstrap have been proposed like the one from Bradley and Fayyad [2]. They suggest to run the k-means M times using any initial centroid selection strategy on M different subsets of the initial data. After that, an optimal grouping of the $M \times k$ centroids identified in the previous runs has to be found. Given the small set size a brute force approach is a reasonable option. Finally the "real" k-means will use those centroids as the initial ones.

Using a distance as a similarity measure implies that the clusters will have a spherical shape. It follows that the algorithm performs best when the input data have normally distributed features values.

Despite these disadvantages, many variants and optimizations have been proposed both by the industrial and academic communities [9, 11, 5].

Another important clustering algorithm is the Density-based spatial clustering of applications with noise, more commonly known as **DBSCAN** [6]. As the name suggests, it is a density-based approach to the problem, meaning that it groups together points with many others in the neighborhood and penalizes points in low density areas (outliers). It relies on two user provided parameters namely *minPts* and ϵ. The *minPts* variable represents the minimum number of points that must lie in a circle of radius ϵ (neighborhood).

DBSCAN exploits the *density-reachability* to define three classes of points:

- *core*: set of points that have at least *minPts* neighbors.
- *reachable*: set of points that are in the neighborhood of a *core* point, but are not *core* points themselves.
- *outlier*: set of points that have less than *minPts* points in their neighborhood.

3. PROBLEM DEFINITION

4. SOLUTION

5. CONCLUSIONS AND FUTURE WORK

6. REFERENCES

[1] J. D. Banfield and A. E. Raftery. Model-based gaussian and non-gaussian clustering. *Biometrics*, pages 803–821, 1993.

[2] P. S. Bradley and U. M. Fayyad. Refining initial points for k-means clustering. In *ICML*, volume 98, pages 91–99. Citeseer, 1998.

[3] P. S. Bradley, U. M. Fayyad, C. Reina, et al. Scaling clustering algorithms to large databases. In *KDD*, pages 9–15, 1998.

[4] J. Durbin and A. Stuart. Callbacks and clustering in sample surveys: An experimental study. *Journal of the Royal Statistical Society. Series A (General)*, 117(4):387–428, 1954.

[5] C. Elkan. Using the triangle inequality to accelerate k-means. In *ICML*, volume 3, pages 147–153, 2003.

[6] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Kdd*, volume 96, pages 226–231, 1996.

[7] U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy. *Advances in knowledge discovery and data mining*, volume 21. AAAI press Menlo Park, 1996.

[8] A. K. Jain. Data clustering: 50 years beyond k-means. *Pattern recognition letters*, 31(8):651–666, 2010.

[9] T. Kanungo, D. M. Mount, N. S. Netanyahu, C. D. Piatko, R. Silverman, and A. Y. Wu. An efficient k-means clustering algorithm: Analysis and implementation. *IEEE transactions on pattern analysis and machine intelligence*, 24(7):881–892, 2002.

[10] L. Kaufman and P. J. Rousseeuw. *Finding groups in data: an introduction to cluster analysis*, volume 344. John Wiley & Sons, 2009.

[11] A. Likas, N. Vlassis, and J. J. Verbeek. The global k-means clustering algorithm. *Pattern recognition*, 36(2):451–461, 2003.

[12] J. MacQueen et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA., 1967.

[13] G. Schwarz et al. Estimating the dimension of a model. *The annals of statistics*, 6(2):461–464, 1978.

[14] H. C. Tijms. *Stochastic models: an algorithmic approach*, volume 303. John Wiley & Sons Inc, 1994.