An Incremental Hierarchical Clustering

Arnaud Ribert, Abdel Ennaji, Yves Lecourtier

P.S.I. Faculté des Sciences, Université de Rouen
76 821 Mont Saint Aignan Cédex, France
Arnaud.Ribert@univ-rouen.fr

Abstract

This article describes a new algorithm to treat time incremental data by a hierarchical clustering. Although hierarchical clustering techniques enable one to automatically determine the number of clusters in a data set, they are rarely used in industrial applications, because a large amount of memory is required when treating more than 10,000 elements. To solve this problem, the proposed method proceeds by updating the hierarchical representation of the data instead of re-computing the whole tree when new patterns have to be taken into account. Memory gains, evaluated for a real problem (handwritten digit recognition) allow to treat databases containing 7 times more data than the classical algorithm.

1. Introduction

Numerous techniques of pattern recognition and image segmentation require an efficient clustering method to understand, interpret and simplify large amounts of multi-dimensional data [7][10][9][1]. Most of the time, k-means algorithm and other partitional methods are used in industrial applications [5][11][7][16]. However, they present the major drawback of requiring close-to-the-final-solution initial conditions, specially concerning the number of clusters. This a priori knowledge is rarely at the user's disposal because, if they use a clustering technique, this is precisely because they ignore the structure of their data. So, this constraint is intractable in the general case.

In fact, several authors recommend to use a hierarchical clustering before starting the k-means algorithm, since it does not require any a priori knowledge and provides a good representation of the data. But hierarchical clustering generally requires a great amount of memory, since it increases with the square of the number of elements in the database. So, treating 10,000 elements requires 200 Mo of RAM, while 800 Mo are needed to deal with 20,000 elements. Such a memory cost may explain why hierarchical clustering is rarely used in industrial applications.

On the other hand, clustering techniques consider that the given database is completely representative of the problem, whereas in fact, this is rarely the case when dealing with complex problems. In other words, from a practical point of view, a complex problem is often an incremental one. Consequently, it would be very useful to be able to enrich a database to take into account patterns which were not available at the time of the constitution of the database. The problem is then to update one's clustering.

Another advantage of incremental approaches is to consider the original database as a small growing one. In this scope, it might be possible to reduce the memory requirements of the hierarchical clustering by building a first numerical taxonomy (i.e. the tree resulting of a hierarchical clustering) using the available memory and updating it when new elements are taken into account. The success of this strategy depends on the capability to update a numerical taxonomy while keeping the largest possible part of it. This is the objective of the algorithm that is described in this paper.

In the next section, the main characteristics of the hierarchical clustering are recalled. Sections 3 and 4 describe respectively the way to determine the anchoring point of a new element in a numerical taxonomy and how to update this one, starting from the anchoring point. Eventually, section 5 presents experimental results over real databases.

2. A brief recall on the hierarchical clustering

A hierarchical clustering method is a procedure to represent data as a nested sequence of partitions. An

example of the corresponding graphical representation, called a dendrogram, is shown on Figure 1. It is important to note that the height of a node is proportional to the distance between groups it links. Consequently, the shape of a dendrogram gives information on the number of clusters in a data-set. Thus, cutting a dendrogram horizontally engineers a clustering (5 clusters appear in the example of Figure 1). Numerous methods have been proposed to determine the best cutting point, to automatically find the number of clusters [Mil88]. Although these methods are often used, it has been shown that a multi-point cutting leads to better results for real data-sets [14][15].

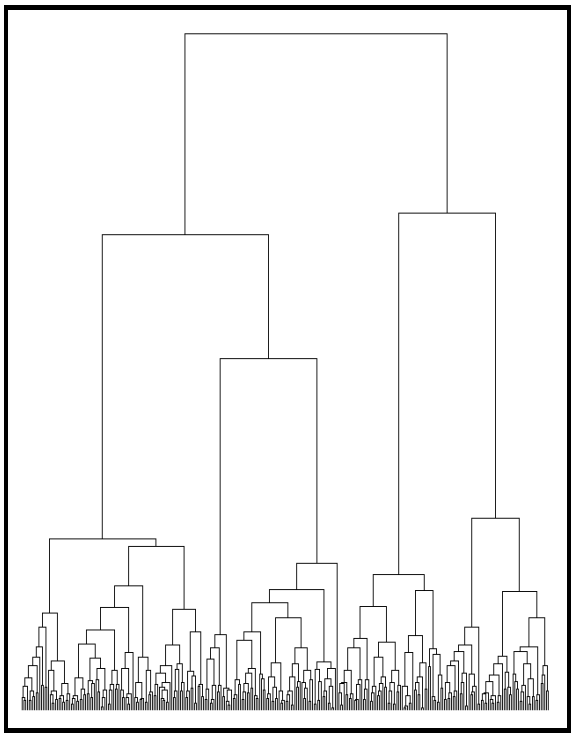

Figure 1 : Example of dendrogram

Hierarchical clustering algorithms generally work by sequential groupings of clusters. A numerical taxonomy can be built using the following algorithm, where initial points are considered as individual clusters.

Compute the euclidian distance between every pair of points;
Merge into a single cluster the two closest points;
While (There are more than one group) **Do**
    Compute the distance between the new cluster and every existing one;
    Merge into a single cluster the two closest ones;
EndWhile

It can be noticed that the euclidian distance can be replaced by any other dissimilarity measure, that is to say a metric d such that $d(x,x) = 0$, $d(x,y) = d(y,x)$ and $d(x,y) \geq 0$ (x,y). Moreover, an additional metric has to be introduced to measure the distance between two groups. Commonly used metrics are : the minimum or maximum euclidian distance (called single and complete link) or the average distance between the groups. This choice has a great influence on the representation capabilities of the taxonomy. The algorithm described in this article makes no assumption on the used metric, so that the user choice is free. However, a well-suited metric, which will be detailed further, enables one to obtain better performance. On a simplification purpose, we will consider in section 3 and 4 that the average link is used.

3. Determining the place of the new element in the hierarchy

When a new element has to be added in a taxonomy, one has firstly to determine its place in the tree. Several methods have been proposed to achieve this [8]. They are generally based on the estimation of a boundary layer between the two sub-trees (i.e. groups) encounted at each node of the taxonomy. Each boundary layer is used to place the new element (NE) like a decision tree would do. However, to our knowledge, none of these methods is endowed of a stopping criterion, allowing for instance to place a new element at the root of the taxonomy if it is required.

It is of course impossible to consider that NE will be the brother of its nearest neighbour. Indeed, the minimum average distance between NE and a group can be reached for a group which does not contain NE's nearest neighbour. So, the research has to begin from the root of the taxonomy. This implies the definition of a descent procedure in the tree (including a stopping condition) so that the process ends up on top of the right sub-tree, which will be called R.

The implemented principle uses the notion of Region Of Influence for a group G. Let $D(G_1, G_2)$ be the distance between two groups G_1 and G_2. Then, as shown on Figure 2, a point X belongs to the ROI of G_1 if $D(X, G_1) \leq D(G_1, G_2)$.

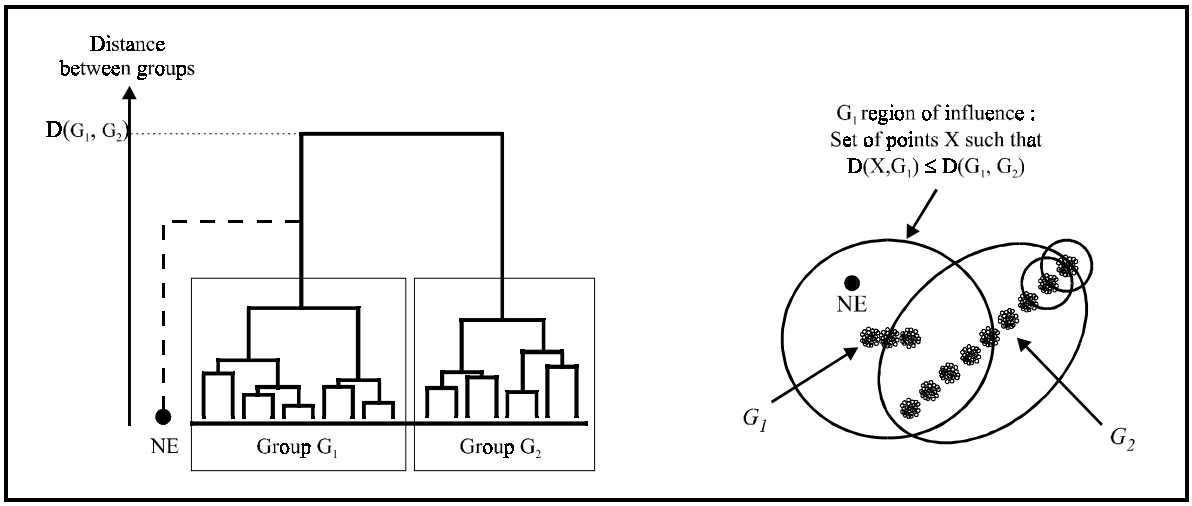

Figure 2 : Finding the place of a new element in a numerical taxonomy

Let Ω be the data set used to build a first taxonomy T. Since NE is not present, R (the search sub-tree) is aggregated with another cluster, H. When building a hierarchy for Ω+{NE}, R is aggregated with NE when D(R, {NE}) is the smallest distance between two clusters. Consequently, it can be said that D(R, {NE}) ≤ D(R, H). In other words, NE belongs to the ROI of R.

During the building process, it is known that the creation of a cluster (composed of two groups G_i and G_j) before the merging of R and NE implies : $D(G_i, G_j) < D(R, \{NE\})$. This is particularly true for every sub-tree of R. Consequently, NE does not belong to any ROI of the sub-trees of R.

Moreover, as D(R, {NE}) is the smallest inter-cluster distance when R and NE are aggregated, it can be said that among the groups of T verifying the two first properties, the searched one is the closest from NE. It can be shown that these three properties are necessary and sufficient to define (and find) R when dealing with binary hierarchies, since in this case only one minimum exists [15].

It can also be noticed that this algorithm can be used to perform a supervised classification or to detect exceptions.

4. Updating the numerical taxonomy

The second phase of the algorithm consists of updating the taxonomy nodes, starting from the previously determined point. The principle is to cut a sub-tree only when it is needed, and to update node levels in all other cases. The problem is to determine the conditions to be verified to cut a sub-tree. The basic algorithm presented in the following paragraphs proposes criteria to solve this problem. A first optimisation is then proposed.

4.1. The basic algorithm

When a new element is added in a taxonomy, there is theoretically no guaranty that a part of it will be kept [13]. However, practically, experiments over 20,000 synthetic databases (without any data structure in the representation space, in order to favour important changes after the addition of the new element) have shown that the whole recontruction never occured for databases of more than 50 elements.

Two ways may be envisaged to update a numerical taxonomy. On the one hand, one can try to determine the exact changes that will occur in the taxonomy due to the presence of a new element. On the other hand, one can also determine what part of the taxonomy will never be changed after the addition of a new element. Since the first solution seems to be particularly difficult, the second strategy has been retained.

Thus, when the place of the new element (NE) is determined, an ascendant update of the hierarchy (starting from the anchoring point) is performed. The objective is to determine the groups inside the current sub-tree which cannot be changed by the integration of the new element. In fact, depending on the considered scale, NE insertion may engineer two different effects on the data structure. At a given level (i.e. on the top of a sub-tree), the new element can contribute to bring two sub-trees nearer, or on the contrary tend to move them away from each other.

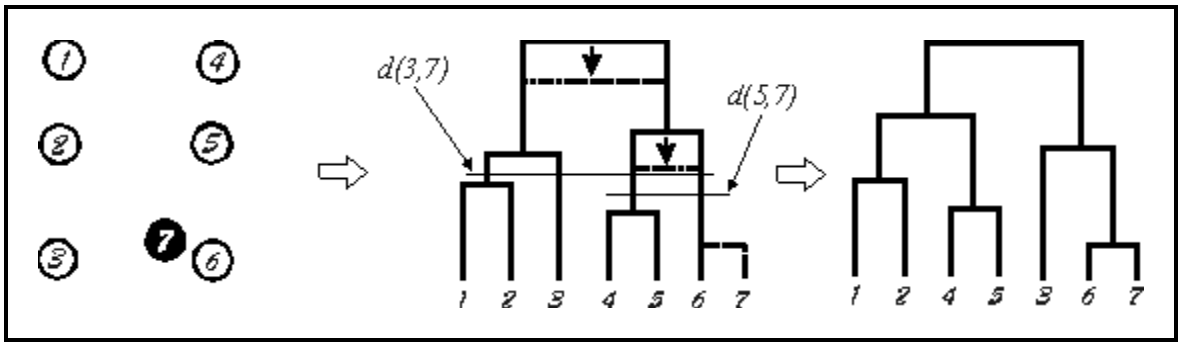

Figure 3 : The new element brings two sub-trees nearer

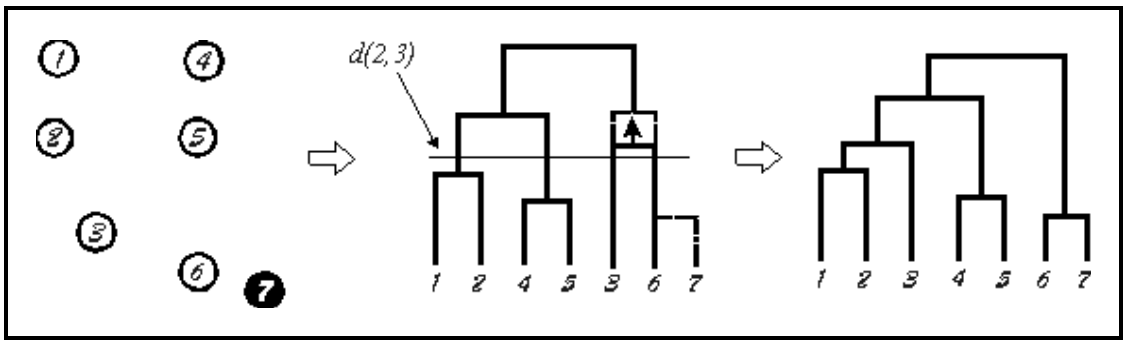

Figure 4 : The new element moves two sub-trees away

The first case is illustrated on figure 3. In a first time, element 7 contributes to bring element 6 and group {4,5} nearer. Group {4,5} had been made because d(4,5) was inferior to d(4,6) and d(6,5). Element 7 cannot call this into question because it is not close enough from element 5. On the contrary, it calls into question the merging of 3 and group {1,2}. Indeed, the only parts of the taxonomy that are necessarily kept are sub-trees whose threshold is inferior to d(3,7), which represents the smallest distance between {1,2,3} and {4,5,6}. More generally, it can be proved [15] that when NE contributes to bring two sub-parts A and B nearer, the smallest tree containing them has to be cut at threshold C_t (for "Critical Threshold") which is set to the minimum euclidian distance between A and B (considering NE in A). This cutting is followed by a classical taxonomy building.

The second case is illustrated by figure 4. Elements 3 and 6 are moved away because of element 7. Consequently, no regrouping between 3 and 6 or 7 can occur. But if 3 is moved away from {6,7}, it can at the same time be moved closer from another group, particularly from group {1,2}. So, such a merging has to be allowed during the reconstruction of the taxonomy in cutting the tree at its critical threshold. More generally, it has been demonstrated [15] that when two sub-trees A and B are moved away from each other, a cutting in their grandfather at its critical threshold (C_t) has to be performed. Nevertheless, it can be noticed that this will not occur if the average distance between A and B does not exceed C_t. Indeed, in this case, there is no risk to have another grouping than A and B one : their average distance is only updated without any structure change.

It can be observed that this algorithm is partly based on the non-incremental taxonomy building. Consequently, our algorithm can take advantage of every kind of optimisation in the literature [3][6][4].

4.2. A major optimisation

The previous algorithm has been tested over a great number of synthetic and real databases, showing that it led to the good tree, but with an important computational cost. Indeed, in many cases, it was necessary to cut two consecutive trees although the first threshold was higher than the second one. Consequently, a lot of calculations were made several times. The following optimisation proposes a way to avoid this.

1. Compute the cutting thresholds, considering that a cut will be needed at each node during the ascendant phase, even for increasing average distances.

2. Starting from the anchorage node, select cutting thresholds such that any following cutting threshold is strictly superior.

3. Run the ascendant phase, cutting only at the selected thresholds.

It can be noticed that we have to consider that a cut will occur even for increasing node indices. This is because it is impossible to know whether the critical threshold for a given sub-tree will be reached or not after the update of the sub-trees that it contains. This potential loss is nevertheless negligible in comparison to the gains engineered by the proposed optimisation. This version has thus been used in the following experiments.

5. Experimental results

The aim of these experiments is to measure gains provided by the incremental algorithm in comparison to the classical one (given in section 2). In order to favour good performance, an original inter-cluster measure has been developped. Indeed, the single link is the most favourable measure for the incremental algorithm, because the height of a node equals its critical threshold. Consequently, a cutting would often be replaced by a simple re-evaluation of the node height. Unfortunatly, data representation provided by the single link suffers from the « chaining effect » which tends to group all the patterns in a unique cluster. A metric presenting a compromise between the incremental algorithm performance and the representation capabilities of the numerical taxonomy has therefore been designed.

This metric has been implemented using the Lance-Williams' formula. This one is an efficient way to compute the distance between a new cluster (consisting of groups r and s) and an existing one (k). The metric used in these experiments is computed by the expression below.

$$D[(k),(r,s)] = \frac{1}{2} D[(k),(r)] + \frac{1}{2} D[(k),(s)] - 0{,}4960 \left| D[(k),(r)] - D[(k),(s)] \right|$$

Numeric terms have been determined by a dichotomic procedure for a real database (handwritten digit recognition) such that the supervised analysis of the clusters reveals a good match between groups of data in the representation space and the reality.

Tests have also been carried out for a handwritten digit recognition problem (the NIST database), but for different fractions of the data-set. The employed feature vector was constituted by the 85 (1+4+16+64) grey levels of a resolution pyramid (see Figure 5 below) [2].

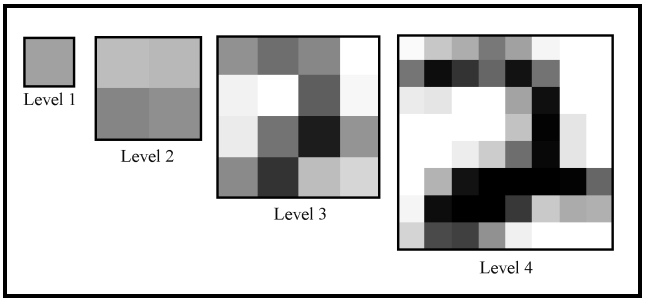

Figure 5 : Representation of a 2 using a 4 level resolution pyramid

Savings of memory and number of computed distances appear in Figure 6 below. This curves represent the averages over 50 different experiments.

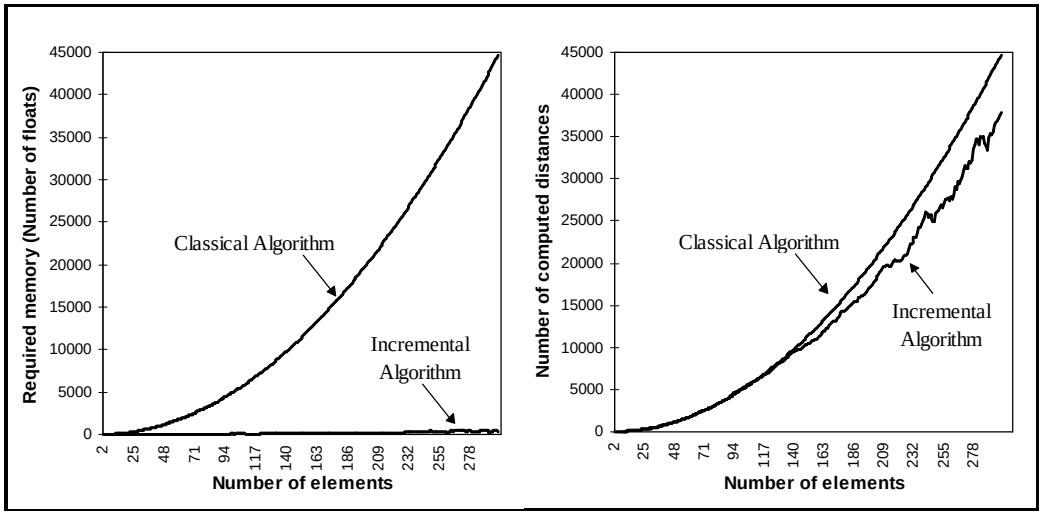

Figure 6 : Memory and distance savings for a real database

A precise analysis of these curves show that for a given memory size, the memory savings offered by the incremental algorithm allow to handle databases comprising 7 times more elements than the classical building method. On the other hand, computational requirements are very important and have to be reduced. In spite of this, since the incremental algorithm can transform memory requirements into computational ones, it can be said that it allows to surpass the material limitations of any computer.

6. Conclusion

This article describes an original algorithm to update a numerical taxonomy after addition of new patterns in a database. Tests have shown that using this algorithm allow to progressively perform a hierarchical clustering of big sets of data, which can then contain seven times more elements than using the classical algorithm. In spite of the computational cost of the method, it can be said that the incremental algorithm described in this paper is a first stage towards the use of hierarchical clustering in industrial applications, thus offering an interesting alternative to partitional clustering techniques. Further works should lead to significant reduction of the computational cost of the method by integrating already available optimisation algorithms and adding more efficient rules of building.

Bibliography

[1] Arabie P, L.J. Hubert, G. De Soete, (eds.): "Clustering and Classification", World Scientific Publ., River Edge, NJ, 1996.

[2] Ballard D.H., Brown C.M., "Computer Vision", Prentice Hall, 1982.

[3] Bruynooghe M., "Recent Results in Hierarchical Clustering. I - The Reducible Neighborhoods Clustering Algorithm", IJPRAI, Vol. 7, N° 3, pp. 541-571, 1993.

[4] De Rham C, "La classification hiérarchique ascendante selon la méthode des voisins réciproques", *Cahiers de l'Analyse de Données*, Vol. 5, n°2, 1980.

[5] Forgy E., "Cluster analysis of multivariate data : efficiency versus interpretability of classifications", Biometrics, Vol. 21, 768, 1965.

[6] Hattori K., Torii Y., "Effective algorithms for the nearest neighbor method in the clustering problem", *Pattern Recognition*, Vol. 26, n°5, pp. 741-746, 1993.

[7] Jain A.K., Dubes R.C., *Algorithms for clustering data*, Prentice Hall, 1988.

[8] Jambu M., "Exploration informatique et statistique des données", Dunod, Paris, 1989.

[9] J.D. Jobson, "Applied Multivariate Data Analysis, Volume II : Categorical and Multivariate Methods", Springer-Verlag, New-York, 1992.

[10] Kohonen T., "Self-organisation and associative memory", Springer-Verlag, 1989.

[11] MacQueen J.B., "Some methods for classification and analysis of multivariate observations". Proceedings of the 5th Berkeley Symposium on Math. Statistics and Probability, Vol. 1, pp. 281-297, 1967.

[12] Milligan G.W., M.C. Cooper, "An examination of procedures for determining the number of clusters in a data set", *Psychometrika*, 50, n° 2, 159-179, 1985.

[13] S.Régnier, "Stabilité d'un opérateur de classification", Mathématiques et Sciences Humaines, N°82, pp. 75-84, 1983.

[14] Ribert A., Ennaji A., Lecourtier Y., "Hiérarchies indicées et catégorisation multi-échelle", 6ièmes Rencontres de la Société Francophone de Classification, pp. 187-190, Montpellier, Septembre 1998.

[15] Ribert A., "Structuration évolutive de données : application à la construction de classifieurs distribués", Ph.D Thesis, Rouen, France, October, 1998.

[16] Venkateswarlu N.B., Raju P.S.V.S.K., "Fast isodata clustering algorithms", *Pattern Recognition*, Vol. 25, n° 3, pp. 335-342, 1992.