Partitional vs Hierarchical Clustering Using a Minimum Grammar Complexity Approach

Ana L.N. Fred and José M. N. Leitão

Instituto de Telecomuncações / Instituto Superior Técnico, Lisboa, Portugal
IST-Torre Norte, Av. Rovisco Pais, 1049-001, Lisboa, Portugal
afred@lx.it.pt

Abstract. This paper addresses the problem of structural clustering of string patterns. Adopting the grammar formalism for representing both individual sequences and sets of patterns, a partitional clustering algorithm is proposed. The performance of the new algorithm, taking as reference the corresponding hierarchical version, is analyzed in terms of computational complexity and data partitioning results. The new algorithm introduces great improvements in terms of computational efficiency, as demonstrated by theoretical analysis. Unlike the hierarchical approach, clustering results are dependent on the order of patterns' presentation, which may lead to performance degradation. This effect, however, is overcome by adopting a resampling technique. Empirical evaluation of the methods is performed through application examples, by matching clusters between pairs of partitions and determining an index of clusters agreement.

1 Introduction

A diversity of clustering procedures can be found in the literature [11]. From the methodological point of view, algorithms can be divided in two major classes: partitional methods and hierarchical methods. Partitional structure organizes patterns into a small number of clusters. It usually assumes the *a priori* specification of the number of clusters to partition the data or the definition of cluster validity criteria. Hierarchical clustering consists of a sequence of nested data partitions in a hierarchical structure. A particular partition is obtained by cutting the hierarchical structure at some level.

Concerning structural patterns represented as sequences of symbols, clustering algorithms are extensions of these methods by adopting adequate string similarity measures [7]. Viewing similarity computation as a matching process [8,1,2,15,13,3,16], references [8,12] present sentence-to-sentence clustering procedures based on the comparison of a candidate string with sentences in previously formed clusters (clustering based on a nearest-neighbor rule) or with cluster center strings (cluster center technique), respectively. String editing operations are there used in the transformation of strings to perform the matching. Following the string matching paradigm while modeling clusters' structure using grammars, error-correcting parsing and grammatical inference are combined in a clustering algorithm described in [8,9]. Basically it implements a nearest-neighbor rule, where sentences are compared, not directly with the patterns included in previously formed clusters, but with the best matching elements in the languages generated by the grammars inferred from the clusters' data. Also using grammars for modeling clusters' structure, a distinct approach, based on the concept of minimum description, is proposed in [4]. Structural resemblance between patterns is assumed to reflect common rules of composition; a normalized reduction in grammar complexity obtained by associating patterns gives the measure of similarity underlying the hierarchical clustering algorithm there proposed. In [5] the search of common subpatterns by means of Solomonoff's coding [17,6] forms the basis of a clustering algorithm that defines similarity between patterns as a ratio of decrease in code length.

This paper focuses on clustering procedures capturing structural resemblance in the form of rules of composition between primitives, as opposed to string matching techniques. The grammar formalism is adopted to describe these rules and simultaneously provide a model for cluster representation. To this purpose, a new clustering algorithm, of the partitional type, is proposed and compared with the hierarchical method described in [4]. Also, within the scope of empirical comparison of partitions produced by the different algorithms, a global partitioning agreement index is proposed.

Section 2 presents the grammar-based similarity measure that forms the core of the clustering algorithms, emphasizing the distinction between structural resemblance and string matching. The new clustering algorithm is described in section 3; theoretical algorithmic complexity evaluations and comparative performance analysis are addressed in section 4. A measure of partitions agreement is proposed for empirical assessment of the methods. Application examples are presented in section 5.

2 Structural Similarity Measure

The concept of resemblance between strings has typically been viewed from two perspectives [7]: (1)- similarity as matching, based on string editing operations; (2)- structural resemblance, based on the similarity of their composition rules and primitives.

The similarity measure described next, which forms the basis of the clustering algorithms described in section 3, falls into the second category. According to this approach, patterns' structure is modeled by syntactic rules, automatically inferred from the data [10,14]. Grammar complexity gives a measure of the compactness of this representation:

$$C(G) = \sum_{i=1}^{r} \sum_{j=1}^{l_i} C(\alpha_{ij}) \tag{1}$$

where $C(G)$ is the complexity of grammar G, α_{ij} represents the right side of the jth production for the ith non-terminal symbol of the grammar, and

$$C(\alpha) = (n+1)log(n+1) - \sum_{i=1}^{m} k_i log k_i \tag{2}$$

with k_i being the number of times that the symbol a_i appears in α, and n is the length of the grammatical sentence α. Structural resemblance is captured by shared rules of composition, which lead to a reduction of the global description (grammar inferred from the patterns ensemble) when compared to the descriptions of the patterns considered individually. Similarity is then defined as the ratio of decrease in grammar complexity (RDGC) [4], as follows:

$$RDGC(s_1, s_2) = \frac{C(G_{s_1}) + C(G_{s_2}) - C(G_{s_1,s_2})}{min\{C(G_{s_1}), C(G_{s_2})\}} \tag{3}$$

where s_1, s_2 are strings and $C(G_{s_i})$ denotes grammar complexity. Figure 1 outlines the $RDGC$ similarity computation procedure.

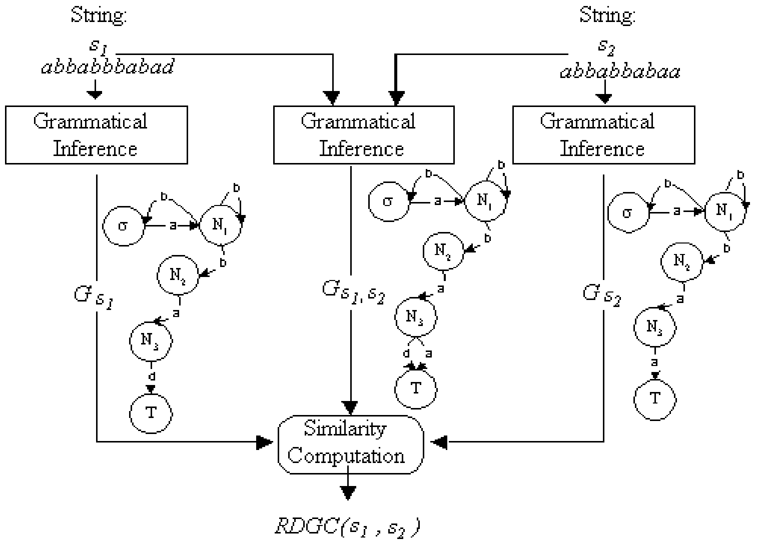


Fig. 1. Computing the similarity between strings under the minimum grammatical complexity approach. Separate grammars are inferred from the individual strings and from the strings ensemble; similarity is defined as the normalized reduction in grammatical complexity obtained by joining the patterns. The choice of the grammatical inference algorithm influences the similarity index obtained. For simplicity of illustration, graphs representing regular grammars are depicted next to the grammatical inference blocks.

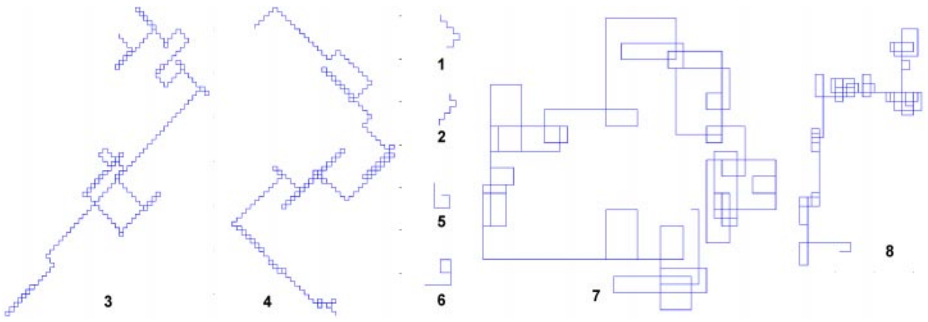


Fig. 2. Graphical representation of strings of the form $(2^*6^*)^*$ (patterns 1 to 4) and $(0^*6)^*$ (patterns 5 to 8). The graphical interpretations of the symbols are as follows: 0 – maintain the line direction; 2 – turn right; 6 – turn left. The string's lengths for the several patterns are: (1)-9; (2)-9; (3)-389; (4)-409 (5)-18; (6)-9; (7)-487; (8)-411.

The emphasis of the RDGC on structural similarity rather than on string alignment is put in evidence in the example depicted on figure 2, which represents a graphical description of instances, with various lengths, of strings of the form $(2^*6^*)^*$ or $(0^*6)^*$, with the symbol $*$ indicating an arbitrary repetition of the element on the left (parenthesis are used for delimiting elements with more than one character).

	1	2	3	4	5	6	7	8
1	1	1	1	0.707	0	0	0	0
2	1	1	1	0.707	0	0	0	0
3	1	1	1	0.757	0	0	0	0
4	0.707	0.707	0.757	1	0	0	0	0
5	0	0	0	0	1	0.913	0.928	0.928
6	0	0	0	0	0.913	1	1	1
7	0	0	0	0	0.928	1	1	1
8	0	0	0	0	0.928	1	1	1

Table 1. Similarity matrix for the string patterns in figure 1.

Table 2 shows the similarity matrix between the string patterns, computed using Crespi-Reghizzi's algorithm [10] for grammatical inference, without imposing *a priori* information other then left-to-right precedence of the characters. As shown, the similarity measure provides complete separation of the two structures (zero valued blocks on the matrix). Furthermore, the similarity index is independent of the strings length, yielding maximum similarity to most of the patterns exhibiting the same structure. The non-unit values regarding the similarity between string 4 and the first three strings (or between string 5 and strings 6 to 8) reflect the sensitivity and asymmetry of the inference algorithm to initial and terminal string values with the *a priori* information used.

3 Minimum Grammar Complexity Clustering

The underlying idea for the clustering algorithms described next, is that, if sequences exhibit a similar structure, then their joint description will be more compact than the combination of the descriptions of the individually considered elements. Using the grammar formalism to model the string patterns, similarity of structures and primitives leads to shared sets of grammar productions, and hence to a reduction in the global grammar complexity. Taking the grammar complexity, as defined in expressions 1 and 2, as a measure of description compactness, and the associated similarity between string patterns, defined by expression 3, the later is extended to sets of sequences, providing a similarity measure between clusters:

$$RDGC(C_1, C_2) = \frac{C(G_{C_1}) + C(G_{C_2}) - C(G_{C_1,C_2})}{min\left\{C(G_{C_1}), C(G_{C_2})\right\}} \tag{4}$$

with G_{C_i} representing the grammar inferred from the data in cluster C_i.

Section 3.1 presents a hierarchical clustering algorithm based on this similarity concept, proposed in [4]. The new algorithm, a sentence to sentence clustering procedure, is presented in section 3.2. These algorithms, besides the data partitioning, provide a model for cluster representation.

3.1 Hierarchical Clustering

Input: A set of strings $S = \{s_1, s_2, \ldots, s_n\}$ and a threshold th.

Output: A partition of S into m clusters $C_1, C_2, \ldots, C_m$ and their grammatical representations $G_{C_1}, G_{C_2}, \ldots, G_{C_m}$

Steps:

1. Assign s_i to C_i, $i = 1, \ldots, n$ and infer a grammar, G_{C_i}, for each cluster. Let $m = n$.
2. Among the m^2 possible associations of two clusters, compute

$$sim = max\left\{RDGC(C_i, C_j)\right\}, i, j = 1 \ldots m, i \neq j$$

If $sim > th$, then associate these clusters, set their grammatical description as the grammar inferred from the joint set of data G_{C_i,C_j} and decrease m by one; otherwise stop, returning the clusters found.

3.2 Sentence to Sentence Clustering Procedure

Input: A set of strings $S = \{s_1, s_2, \ldots, s_n\}$ and a threshold th.

Output: A partition of S into m clusters $C_1, C_2, \ldots, C_m$ and their grammatical representations $G_{C_1}, G_{C_2}, \ldots, G_{C_m}$.

Steps:

```
1. Assign $s_1$, the first string in $S$, to $C_1$, and infer the grammar $G_{C_1}$. Let $m = 1$.
2. For each remaining element $s_i \in S$ do:
   – Infer a grammar for $s_i$ and compute the similarity $RDGC(s_i, C_k), k = 1, \ldots m$. Let $sim$ be the highest value found and $C_j$ the matching cluster.
   – If $sim > th$, include $s_i$ in $C_j$ and update the cluster's grammatical description; otherwise, form a new cluster with $s_i$ and set $m = m + 1$.
3. Return the clusters found and their grammatical descriptions.
```

4 Comparison of the Algorithms

The hierarchical and partitional algorithms described previously are based on a common similarity measure and cluster representation paradigm: syntactic model. They therefore provide compact models for cluster description that, while capturing data structure, also provide a mechanism for the recognition of new data by means of parsing algorithms, and generative capability.

The demarcation of the two approaches deals with computational efficiency, optimality of the solutions and structure evidencing aspects.

4.1 Computational Complexity

Let n be the total number of samples. The computational analysis takes grammatical inference as elementary operation, as this is the most expensive processing performed. It should be emphasized that the adopted grammatical inference method (Crespi-Reghizzi's method) has linear time complexity on the length of the patterns, $O(l)$. By adequate memorization of grammar structure information (profiles), merging of clusters involves simple calculations on these structures, without requiring recalculations with the samples.

Step 1 of the hierarchical algorithm (see section 3.1) involves n distinct grammar inferences, one per sample. The first iteration of step 2 performs $\frac{(n-1)^2}{2}$ inferences, filing a $n \times n$ similarity matrix (this matrix is upper triangular with unitary diagonal). The remaining iterations of step 2 undertake comparisons on this decreasing order matrix (each association of patterns, or clusters merging, dictate a reduction by one in the matrix dimension and its corresponding actualization, involving the computation of the similarity of the merged cluster with all the others – $m - 1$ computations, m being the current dimension of the similarity matrix). In the overall, the hierarchical algorithm has $O(n^2)$ time and space complexities.

The analysis of the steps of the partitional algorithm (section 3.2) shows that n inferences are needed (one per sample) and the computation of the similarity with existing clusters and actualization of clusters' grammars involves at the most m grammar merging operations. As a result, the algorithm has $O(mn)$ time complexity and $O(m)$ space complexity. This represents a significant reduction in computational complexity in comparison with the hierarchical version as m, the total number of clusters is usually small in relation to the number of samples.

4.2 Optimality of the Solutions Found

Since the hierarchical data structuring, produced by the hierarchical algorithm, is based on the pre-computation of a similarity matrix between all sample pairs, solutions found are only dependent of the value of th, a design parameter meaning the minimum similarity of patterns within a cluster. The partitional algorithm, however, may produce solutions dependent of the order of presentation of patterns, in which case over-fragmentation of the data may occur. This situation is more likely to arise when clusters are not well separated, and highly dissimilar patterns, belonging to the same cluster, are on the top of the presentation list, the similarity of which being smaller than the value of the threshold th. This dependency on the order of patterns' presentation may be overcome by a combined resampling and consistent clusters gathering algorithm, not described here due to space restrictions.

Empirical evaluation of the algorithms undertakes the comparison of the partitions produced, which, in general, will include differing numbers of clusters and unlike clusters organization and ordering. In order to evaluate the consistency of two data partitions or to compare the results of two clustering algorithms taking as reference an ideal partitioning, it is necessary to determine the correspondence between clusters in both partitioning. In other words, one needs to determine the best matching associations of clusters and an index of clusters agreement.

In the following, we define pc_idx, the *partitions consistency index*, as the fraction of shared samples in matching clusters in two data partitions, over the total number of samples:

$$pc\_idx = \frac{1}{n} \sum_{i=1}^{min\{nc_1, nc_2\}} n\_shared_i$$

where nc_1, nc_2 are the number of clusters in the first and second partitions, respectively, and n_shared_i is the number of samples shared by the ith matching clusters. An algorithm for the computation of matching clusters and corresponding consistency index is described elsewhere.

5 Application Examples

The first example consists of the patterns presented in section 2. Figure 3 illustrates the failure of string matching techniques in identifying the structure of these patterns (dendrogram on the left); by applying the hierarchical algorithm of section 3.1 (dendrogram on the right) perfect separation is obtained for threshold values, th, smaller than 0.7. With the same range of values for th, the partitional algorithm, described in section 3.2, consistently produces the same two classes, not being dependent on the order of pattern presentation.

The other example, depicted in figure 4, concerns the clustering of 84 contour images of two types of hardware tools, using string descriptions. The dendrogram, produced by the hierarchical algorithm, shows the variability of similarity

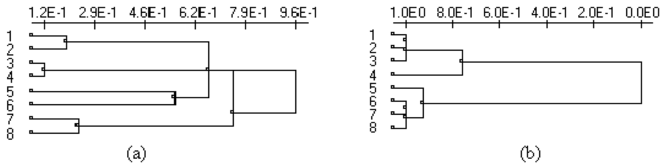
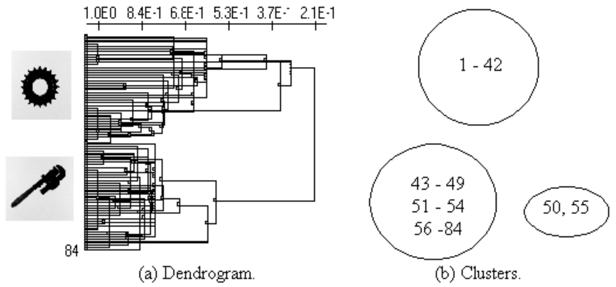

1.2E-1 2.9E-1 4.6E-1 6.2E-1 7.9E-1 9.6E-1
1.0E0 8.0E-1 6.0E-1 4.0E-1 2.0E-1 0.0E0
1 2 3 4 5 6 7 8
(a)
(b)

Fig. 3. Dendrograms produced by hierarchical clustering of patterns in figure 2, using: (a) - dissimilarity based on string editing operations; (b) the RDGC similarity .



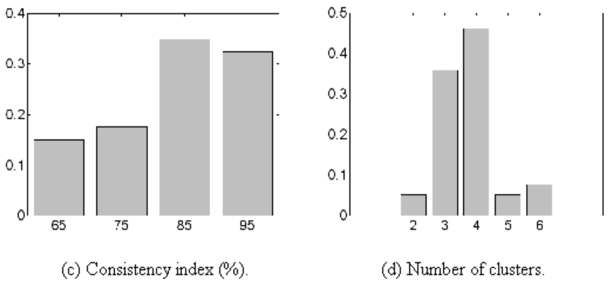


Fig. 4. Clustering of contour images represented by an 8-directional differential chain code. (a)- Dendrogram produced by the hierarchical algorithm; samples 1-42 and 43-84 are of the type represented aside. (b)- Consistent clusters found after 10 resampling experiments, with the partitional-type algorithm. (c)- Histogram of the *consistency index* (in percentage – see section 4.2) between the ideal partitioning and the partitions found with the proposed algorithm, over 40 resampling experiments of the data (random ordering). (d)- Corresponding histogram of the number of clusters found.

values within the cluster of the first tool; class separation is obtained by choosing th in a narrow interval: $].21;.3[$. By applying the sentence-to-sentence algorithm to this data, with $th = .22$, clustering results are dependent on the order of pattern presentation, as shown by the histograms in figure 4 (c) and (d). In these results, wheel-type contours were usually grouped in a single cluster, while the other object class was often fragmented into several, variable composition clusters (most of the experiments led to the partitioning of the data into 3 or 4 clusters – see plot (d)). Consistency index values lower than 1 are also the result of overfragmentation of the data, rather than incorrect pattern associations. However, this overfragmentation, dependent on the order of presentation of patterns, can be overcome by applying a technique of data resampling followed by clustering and determination of consistent clusters, not described here due to space limitations. The results obtained using this technique are depicted in figure 4 (b), showing that only two patterns (number 50 and 55, in the initial data ordering) did not join their natural group, forming an additional cluster.

6 Conclusions

This paper presented a new clustering algorithm for string patterns, of the partitional type, based on a minimum grammar complexity criterion. The ability of the underlying similarity measure to capture structural resemblance, as opposed to string matching, was emphasized, total independence on the string's lengths being achieved. A theoretical analysis of the new algorithm revealed lower computational complexity ($O(n_c n)$ and $O(n_c)$ time and space complexities, respectively, with n being the number of patterns and n_c the number of clusters found) when compared with the hierarchical version of the algorithm presented in [4] ($O(n^2)$ time and space complexities). As a drawback, the partitioning produced by the new algorithm is dependent of the order of pattern presentation, the relevance of this effect being problem dependent and subject to empirical evaluation. To this purpose, an index of clusters agreement in data partitions was proposed to assess the performance of clustering algorithms on practical grounds.

The dependency on the order of presentation of the patterns of the new algorithm can be overcome by a combined resampling/consistent clusters finding technique, as illustrated by an application example, with not significant increase in the computational burden. Therefore, the proposed algorithm constitutes a feasible clustering strategy, able to handle much larger data sets than hierarchical techniques.

References

1. H. Bunke. String matching for structural pattern recognition. In H. Bunke and A. Sanfeliu, editors, *Syntactic and Structural Pattern Recognition, Theory and Applications*, pages 119–144. World Scientific, 1990.
2. H. Bunke. Recent advances in string matching. In H. Bunke, editor, *Advances in Structural and Syntactic Pattern Recognition*, pages 107–116. World Scientific, 1992.
3. G. Cortelazzo, D. Deretta, G. A. Mian, and P. Zamperoni. Normalized weighted levensthein distance and triangle inequality in the context of similarity discrimination of bilevel images. *Pattern Recognition Letters*, 17:431–436, 1996.
4. A. L. Fred. Clustering of sequences using a minimum grammar complexity criterion. In *Grammatical Inference: Learning Syntax from Sentence*, pages 107–116. Springer-Verlag, 1996.
5. A. L. Fred and J. Leitão. A minimum code length technique for clustering of syntactic patterns. In *Proc. Of the 13th IAPR Int'l Conference on Pattern Recognition*, Vienna, August 1996.

6. A. L. Fred and J. Leitão. Solomonoff coding as a means of introducing prior information in syntactic pattern recognition. In *Proc. Of the 12th IAPR Int'l Conference on Pattern Recognition*, pages 14–18, 1994.
7. A. L. Fred and J. Leitão. A comparative study of string dissimilarity measures in structural clustering. In S. Singh, editor, *International Conference on Advances in Pattern Recognition*, pages 385–384. Springer, 1998.
8. K. S. Fu. Syntactic pattern recognition. In *Handbook of Pattern Recognition and Image Processing*, pages 85–117. Academic Press, 1986.
9. K. S. Fu and S. Y. Lu. A clustering procedure for syntactic patterns. *IEEE Trans. Systems Man Cybernetics*, 7(7):537–541, 1977.
10. K. S. Fu and S. Y. Lu. Grammatical inference: Introduction and survey -part i and ii. *IEEE Trans. Pattern Anal. and Machine Intelligence*, 8(5):343–359, 1986.
11. A. K. Jain and R. C. Dubes. *Algorithms for Clustering Data.* Prentice Hall, 1988.
12. S. Y. Lu and K. S. Fu. A sentence-to-sentence clustering procedure for pattern analysis. *IEEE Trans. Systems Man Cybernetics*, 8(5):381–389, 1978.
13. A. Marzal and E. Vidal. Computation of normalized edit distance and applications. *IEEE Trans. Pattern Anal. and Machine Intelligence*, 2(15):926–932, 1993.
14. L. Miclet. Grammatical inference. In H. Bunke and A. Sanfeliu, editors, *Syntactic and Structural Pattern Recognition - Theory and Applications*, pages 237–290. Scientific Publishing, 1990.
15. B. J. Oomen and R. S. K. Loke. Pattern recognition of strings containing traditional and generalized transposition errors. In *Int. Conf. on Systems, Men and Cybernetics*, pages 1154–1159, 1995.
16. E. S. Ristad and P. N. Yianilos. Learning string-edit distance. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 20(5):522–531, May 1998.
17. R. J. Solomonoff. A formal theory of inductive inference (part i and ii). *Information and Control*, 7:1–22,224–254, 1964.