Aggregation based on intervals as similarity measure for hierarchical clustering

Noelia Rico[1], Pedro Huidobro[2], Irene Díaz[1] and Susana Montes[2]

[1]Dept. of Computer Science, University of Oviedo, Spain
[2]Dept. of Statistics and Operational Research, University of Oviedo, Spain

Abstract

Hierarchical clustering algorithms create groups of objects in a data set based on their similarity. This similarity is commonly measured by a distance function, which makes the resulting groups dependent on the distance function used and also the technique employed to merge the clusters. In this work, we propose to transform the objects to a representation where each variable is defined by an interval. Using this representation, we define a new method that measures the similarity of the objects variable by variable based on the overlapping of their intervals, instead of using a distance function. The results obtained for each variable are later passed to an aggregation function, which allows comparing the similarity between the pairs of clusters. An example of algorithm is proposed in this work using a binary function to check the overlapping and aggregating its results with the average function. Finally, the method is compared with other approaches by means of two examples.

Keywords

hierarchical clustering, interval data, aggregation function

1. Introduction

Clustering methods define how to create groups of objects (also known as clusters) in a data set. The aim is that the groups contain the most similar objects and, at the same time, they are as different as possible from the objects in other groups [1]. Hierarchical clustering is an aglomerative algorithm that starts assuming that each object belongs to an individual cluster. Then, it performs pairwise comparisons between all the clusters of the data set in order to measure their similarity. This comparison allows us to determine the two most similar ones, which are grouped together into a single cluster. The pairwise comparison between clusters is repeated sequentially until all the objects are merged into one single cluster [2]. The result of the hierarchical clustering algorithm is commonly represented as a tree graph of the groups called *dendrogram*. Usually, the number of clusters to be obtained is determined once the complete dendrogram is built. Therefore, to define the objects in each cluster, the tree is cut at the level corresponding to the desired number of clusters.

WILF'21: The 13th International Workshop on Fuzzy Logic and Applications
noeliarico@uniovi.es (N. Rico); huidobropedro@uniovi.es (P. Huidobro); sirene@uniovi.es (I. Díaz); montes@uniovi.es (S. Montes)
https://www.noeliarico.dev (N. Rico)
0000-0002-6169-4523 (N. Rico); 0000-0002-0170-0426 (P. Huidobro); 0000-0002-3024-6605 (I. Díaz); 0000-0002-4701-2207 (S. Montes)

In the process of creating the groups, classical approaches treat the clusters as points and measure their similarity by using point-based distances such as the Euclidean distance. In this work, we propose to transform the data in order to represent the clusters as intervals. In the literature, some methods based on intervals can be found. For instance, Galdino and Maciel [3] consider some arithmetic operations between intervals in order to define the Euclidean distance as an interval. On the other hand, Ramos-Guajardo [4] proposes some methods based on the Jaccard index and on statistical hypothesis tests. However, with this interval representation, we propose the use of a function that measures the degree of coincidence between two objects in each variable and an aggregation function in order to determine the clusters that should be merged together in each level of the tree. These two key points allow modeling uncertainty by allowing intervals to represent not only the objects but also their neighborhood, which could be very useful, for example, in case the data set contains noisy data.

This paper is organized as follows. Section 2 gives an overview of the clustering problem and the transformation we apply for an interval-based approach. Section 3 details the two key points introduced into the hierarchical clustering algorithm for creating the clusters using intervals and also presents the proposed algorithm. A comparison with other approaches is presented in Section 4 by means of two examples. Conclusions of this work can be found in the final section.

2. Clustering using intervals

Let us consider a data set $\mathbb{X}$ where each object $x_i \in \mathbb{X}$ is defined by n variables such that $x_i = \{x_i^1, ..., x_i^n\}$ with $x_i^k \in \mathbb{R}$, $1 \leq k \leq n$. The aim of any clustering algorithm is to group these objects into m different clusters $\mathcal{C}$ such that each object belongs uniquely to one cluster and $\mathcal{C}_1 \cup \cdots \cup \mathcal{C}_m = \mathbb{X}$. As the number of clusters is usually unknown beforehand, hierarchical clustering algorithms use a binary tree structure of n levels that groups the objects in $\mathbb{X}$ from the bottom of the tree to the top. Therefore, the algorithm defines for each level ℓ of the tree, with $1 \leq \ell \leq m$, a set of clusters $\mathbb{C}_\ell = \{{}^\ell\mathcal{C}_1, \ldots, {}^\ell\mathcal{C}_\ell\}$. Following this, the root of the generated tree $\mathbb{C}_1$ (which is the last level reached by the algorithm) contains one single cluster with all the elements in $\mathbb{X}$. On the other hand, the bottom level (where the algorithm starts) contains the leaves of the tree, this is, a set of clusters that has as many clusters as objects in the data set, since each of the clusters contains a single object of $\mathbb{X}$. As the tree is built from the bottom (i.e. ℓ equal to the number of objects in $\mathbb{X}$) to the top (i.e. $\ell = 1$), in each iteration the clusters in level ℓ are almost the same than the clusters in level $\ell + 1$, varying only in the two clusters of $\ell + 1$ that are substituted by one in ℓ, corresponding to the cluster generated after merging two most similar clusters in $\ell + 1$ into one single cluster in ℓ. The process is repeated iteratively until the root of the tree is reached. After the tree is obtained, the clusters for any number m of different clusters can be determined by cutting the tree at the corresponding level.

Taking into account that the clustering algorithm seeks for creating groups of similar objects, it is necessary to define how the similarity between the objects is measured. This is usually done by means of a distance function and many variations have been proposed in the literature [5, 6, 7], although the Euclidean distance is usually the one used in practice. There are three different cases for which the measure must be defined:

- Distance between two objects $\delta(x_i, x_j) \ : \ x_i, x_j \in \mathbb{X}$.

- Distance between an object and a cluster $\delta(x_i, {}^{\ell}\mathcal{C}_j) \ : \ x_i \in \mathbb{X}\,,\ {}^{\ell}\mathcal{C}_j \in \mathbb{C}_{\ell}$.
- Distance between two clusters $\delta({}^{\ell}\mathcal{C}_i, {}^{\ell}\mathcal{C}_j) \ : \ {}^{\ell}\mathcal{C}_i, {}^{\ell}\mathcal{C}_j \in \mathbb{C}_{\ell}$.

However, as hierarchical clustering algorithms consider that each object belongs to its own cluster in the first iteration (i.e. the bottom of the tree), all three types of distances could be generalized as $\delta({}^{\ell}\mathcal{C}_i, {}^{\ell}\mathcal{C}_j)$. How the distance between two clusters is computed is usually referred to as *linkage method*. Depending on the linkage method chosen, the algorithm will result in different sets of clusters [8]. Many linkage methods have been defined in the literature. The most popular ones are the listed below accompanied by the explanation on how they define the similarity between two clusters:

- **single**: the distance between the closest points of the clusters.
- **complete**: the distance between the furthest points of the clusters.
- **average**: the distance between all the points of the clusters and then compute the average.
- **centroid**: calculates the centroid of each cluster (i.e. for each variable the average of all the points that belong to the cluster) and computes the distance between the two clusters using their centroids.

To sum up, this means that each linkage method represents the cluster by choosing a different object of the cluster. Then, using this object as a representation of the cluster, the distance is computed pairwisely and the closest ones are merged together.

In this work, we propose to identify the clusters using intervals instead of one single point. For defining a measure of similarity, we keep in mind the original methods and how the distance between two objects is a particular case of the distance between two clusters. Then, we propose a new hierarchical clustering method based on intervals that substitutes the linkage method by an interval-based function and an aggregation function. By using this method, the distance involving clusters of intervals can be computed by components applying a function f to each variable and then using an aggregation function A in order to obtain a single value that allows comparing the clusters pairwisely. This comparison is necessary in order to establish which two of the clusters are the most similar and consequently should be merged together into a single cluster.

In order to identify the objects of the data set with an interval-based representation, each object $x_i \in \mathbb{X}$ is turned into an object $y_i \in \mathbb{Y}$ such that each component x_i^k of the object is transformed into a degenerated interval $y_i^k = [\underline{x_i^k}, \overline{x_i^k}]$, where $\underline{x_i^k} = \overline{x_i^k} = x_i^k$. Notice that this is a particular case of a cluster, therefore any cluster can be defined as ${}^{\ell}\mathcal{C}_i = \{{}^{\ell}\mathcal{C}_i^1, \ldots, {}^{\ell}\mathcal{C}_i^n\}$ where ${}^{\ell}\mathcal{C}_i^k = [\underline{{}^{\ell}\mathcal{C}_i^k}, \overline{{}^{\ell}\mathcal{C}_i^k}]$ with $1 \leq k \leq n$. The data set $\mathbb{Y}$ with the transformed representation of all the objects in $\mathbb{X}$ is given as input to the clustering algorithm.

Whenever two clusters are merged together, and therefore contain more than one object of the data set, the new cluster is created taking from the two clusters merged the lowest and the greatest endpoints for the interval that represents each variable. Formally, given two clusters ${}^{\ell}\mathcal{C}_i$ and ${}^{\ell}\mathcal{C}_j$, the resulting cluster ${}^{\ell-1}\mathcal{C}_h = \{{}^{\ell-1}\mathcal{C}_h^1, \ldots, {}^{\ell-1}\mathcal{C}_h^n\}$ is built such that ${}^{\ell-1}\mathcal{C}_h^k = [\min\{\underline{{}^{\ell}\mathcal{C}_i^k}, \underline{{}^{\ell}\mathcal{C}_j^k}\}, \max\{\overline{{}^{\ell}\mathcal{C}_i^k}, \overline{{}^{\ell}\mathcal{C}_j^k}\}], 1 \leq k \leq n$.

3. Proposed algorithm

The first step in order to obtain an algorithm based on this interval approach is to define the function f used to measure the similarity between two intervals $[a,b],[c,d] \subseteq \mathbb{R}$. Although this function could be defined in multiple forms, here we consider a binary basic approach so the following function f is used to determine whether there is overlapping between the two intervals:

$$f([a,b],[c,d]) = \begin{cases} 1 & \text{if } [a,b] \cap [c,d] \neq \emptyset \\ 0 & \text{otherwise.} \end{cases}$$

The algorithm starts by applying the function $f({}^{\ell}\mathcal{C}_i^k, {}^{\ell}\mathcal{C}_j^k)$ to each component of the clusters ${}^{\ell}\mathcal{C}_i, {}^{\ell}\mathcal{C}_j \in \mathbb{C}_\ell$ such that ℓ is equal to the number of objects in $\mathbb{X}$ and $1 \leq k \leq n$. By doing this, it is possible to obtain the vector $\vec{s}_{\ell_{\mathcal{C}_i}, \ell_{\mathcal{C}_j}} = (s^1_{\ell_{\mathcal{C}_i}, \ell_{\mathcal{C}_j}}, \ldots, s^n_{\ell_{\mathcal{C}_i}, \ell_{\mathcal{C}_j}})$. This vector gives an intuition about what we call the *degree of coincidence* between the two objects.

After $\vec{s}_{\ell_{\mathcal{C}_i}, \ell_{\mathcal{C}_j}}$ has been calculated for all pairs of objects, these must be compared. To obtain a single value that eases this pairwise comparison, the elements of $\vec{s}_{\ell_{\mathcal{C}_i}, \ell_{\mathcal{C}_j}}$ are aggregated by means of an aggregation function $A(\vec{s}_{\ell_{\mathcal{C}_i}, \ell_{\mathcal{C}_j}})$. For this work, as a first approach, we define A simply as the average of the elements in $\vec{s}$, such that:

$$A(\vec{s}_{\ell_{\mathcal{C}_i}, \ell_{\mathcal{C}_j}}) = \frac{1}{n} \sum_{k=1}^{n} s^k_{\ell_{\mathcal{C}_i}, \ell_{\mathcal{C}_j}} .$$

The two clusters ${}^{\ell}\mathcal{C}_i, {}^{\ell}\mathcal{C}_j$ with the greatest value of $A(\vec{s}_{\ell_{\mathcal{C}_i}, \ell_{\mathcal{C}_j}})$ are merged together into a cluster.

Notice that, using this basic function, in case that none of the elements of the objects are equal, the overlapping between the initial vectors will be 0 so none of the objects could be grouped together. To solve this, the neighborhood of the interval in each component will be considered taking into account the range of each variable in the original data set. Formally, we can define the vector $\vec{w}$ of n elements, which associates each value w_k with the k-th variable of the data set using the range of the variable in the initial data set $\mathbb{X}$. Prior to the execution of the clustering the algorithm, the percentage of the variable that will be considered for this update is set using p, which determines the amount that the interval will be extended in case that there is no overlapping, such that $w_k = (\max_{x_i \in \mathbb{X}}(x_i^k) - \min_{x_i \in \mathbb{X}}(x_i^k)) \times p$.

Using the coefficient w_k, when there is no overlapping between any of the intervals, the interval ${}^{\ell}\mathcal{C}_i^k$ is transformed into:

$${}^{\ell}\mathcal{C}_i^k = [\underline{{}^{\ell}\mathcal{C}_i^k} - w_k, \overline{{}^{\ell}\mathcal{C}_i^k} + w_k].$$

The resulting algorithm obtained from the considerations explained is shown in Algorithm 3.

4. Comparison with classical approaches

In order to illustrate the method, an example for a toy data set is presented below.

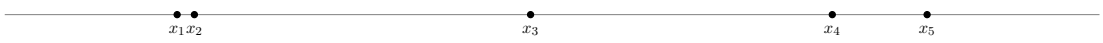
Algorithm 1 Proposed methodology for hierarchical clustering

1: Transform $\mathbb{X}$ into $\mathbb{Y}$ ▷ Define the interval-based objects
2: Compute the vector $\vec{w}$ using $\mathbb{X}$
3: $\ell = n$ ▷ Initially each object has its own cluster, the algorithm starts with the leaves
4: **while** $\ell > 0$ **do** ▷ Build the tree from bottom to top
5: k = 0
6: **for all** ${}^{\ell}\mathcal{C}_i, {}^{\ell}\mathcal{C}_j \in \mathbb{C}_\ell$ **do**
7: **for** each variable k defined with $[a, b]$ in ${}^{\ell}\mathcal{C}_i$ and $[c, d]$ in ${}^{\ell}\mathcal{C}_j$ **do**
8: $s^k_{{}^{\ell}\mathcal{C}_i, {}^{\ell}\mathcal{C}_j,} = f([a, b], [c, d])$ ▷ Compute the similarity according to the variable
9: **end for**
10: **end for**
11: Compute the global similarity $A(\vec{s}_{{}^{\ell}\mathcal{C}_i, {}^{\ell}\mathcal{C}_j})$.
12: **while** $A(\vec{s}_{{}^{\ell}\mathcal{C}_i, {}^{\ell}\mathcal{C}_j}) = 0\,, \forall {}^{\ell}\mathcal{C}_i, {}^{\ell}\mathcal{C}_j \in \mathbb{C}_\ell$ **do** ▷ If none of the clusters overlap
13: Update the intervals using $\vec{w}_i$
14: Increase the value of k
15: Repeat lines 6 to 11
16: **end while**
17: Determine the two closest clusters and merge them into one
18: Define the set $\mathbb{C}_\ell$
19: **end while**

Example 4.1. *Let us consider the one-dimensional data set of five objects* $\mathbb{X} = \{5, 7, 46, 81, 92\}$.

In order to apply the proposed method, first of all, it is necessary to transform the data set $\mathbb{X}$ *into an interval data set* $\mathbb{Y}$ *by using degenerated intervals. Therefore, the transformed data set* $\mathbb{Y} = \{[5, 5], [7, 7], [46, 46], [81, 81], [92, 92]\}$ *is obtained. Also, it is necessary to compute the weight that will be used for updating the intervals when there is no overlapping. In this case, as there is only one variable, only one value must be computed. We define* $p = 0.01$ *so, taking into account the range* $[5, 92]$ *of the variable, the weight* w_1 *can be computed such that* $w_1 = (92 - 5) \times 0.01 = 0.87$. *This value will be used whenever there is no overlapping between the objects, so the intervals are augmented in their extremes until some overlap is found. The first step is to compare the objects pairwisely by applying the function* f *to compare the intervals associated with each variable. The results are shown below:*

	$y_2 = [7, 7]$	$y_3 = [46, 46]$	$y_4 = [81, 81]$	$y_5 = [92, 92]$
$y_1 = [5, 5]$	0	0	0	0
$y_2 = [7, 7]$		0	0	0
$y_3 = [46, 46]$			0	0
$y_4 = [81, 81]$				0



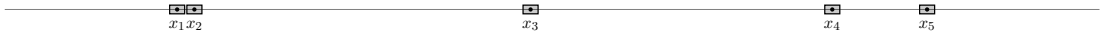
Notice that these objects could also had been represented as:

$$\mathbb{C}_5 = \{{}^5\mathcal{C}_1 = \{[5, 5]\}, {}^5\mathcal{C}_2 = \{[7, 7]\}, {}^5\mathcal{C}_3 = \{[46, 46]\}, {}^5\mathcal{C}_4 = \{[81, 81]\}, {}^5\mathcal{C}_5 = \{[92, 92]\}\},$$

as this is the set containing the leaves at the bottom of the tree, which corresponds to level $\ell = 5$ for this data set of five objects. Henceforth, this notation will be used in order to be consistent in all the iterations of the example.

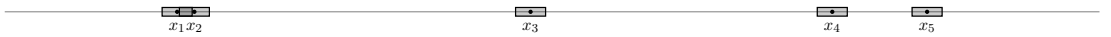
As there are no overlapping between any of the objects, the extremes of the intervals must be updated using w_1. The new intervals obtained after subtracting w_1 to the left extreme of the interval and adding w_1 to the right extreme of the interval are shown below:

	$^5\mathcal{C}_2 = [6.13, 7.87]$	$^5\mathcal{C}_3 = [45.13, 46.87]$	$^5\mathcal{C}_4 = [80.13, 81.87]$	$^5\mathcal{C}_5 = [91.13, 92.87]$
$^5\mathcal{C}_1 = [4.13, 5.87]$	0	0	0	0
$^5\mathcal{C}_2 = [6.13, 7.87]$		0	0	0
$^5\mathcal{C}_3 = [45.13, 46.87]$			0	0
$^5\mathcal{C}_4 = [80.13, 81.87]$				0



As shown in the previous figure, there is still no overlapping so more increase is needed. The weight w_1 is applied again such that the following table is obtained:

	$^5\mathcal{C}_2 = [5.26, 8.74]$	$^5\mathcal{C}_3 = [44.26, 47.74]$	$^5\mathcal{C}_4 = [79.26, 82.74]$	$^5\mathcal{C}_5 = [90.26, 93.74]$
$^5\mathcal{C}_1^1 = [3.26, 6.74]$	1	0	0	0
$^5\mathcal{C}_2 = [5.26, 8.74]$		0	0	0
$^5\mathcal{C}_3 = [44.26, 47.74]$			0	0
$^5\mathcal{C}_4 = [79.26, 82.74]$				0



After this update, there is an overlapping between the clusters $^5\mathcal{C}_1$ and $^5\mathcal{C}_2$. Therefore, these two clusters are merged together in the next iteration, creating in the upper level of the tree $\ell = 4$ the cluster $^4\mathcal{C}_1$. The interval takes the lowest value of the left limit and the greatest value of the right limit of the two clusters that are merged, therefore, $^4\mathcal{C}_1 = \{[3.26, 8.74]\}$.

The representation of the clusters in the next iteration is given by the set of intervals $\mathbb{C}_4$, which contains all the previous intervals and the new cluster created. This set can be defined as:

$$\mathbb{C}_4 = \{^4\mathcal{C}_1 = \{[3.26, 8.74]\}, ^4\mathcal{C}_2 = \{[44.26, 47.74]\}, ^4\mathcal{C}_3 = \{[79.26, 82.74]\}, ^4\mathcal{C}_4 = \{[90.26, 93.74]\}\}.$$

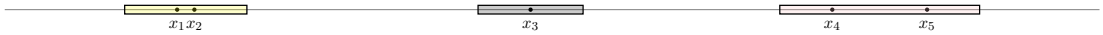
After the creation of this new cluster, it is again necessary to update the intervals to increase their extremes until they overlap and a new cluster can be created. As shown in the comparison below, in the first iteration there is no overlap:

	$^4\mathcal{C}_2 = [44.26, 47.74]$	$^4\mathcal{C}_3 = [79.26, 82.74]$	$^4\mathcal{C}_4 = [90.26, 93.74]$
$^4\mathcal{C}_1 = [3.26, 8.74]$	0	0	0
$^4\mathcal{C}_3 = [44.26, 47.74]$		0	0
$^4\mathcal{C}_4 = [79.26, 82.74]$			0

Therefore, w_1 must be iteratively applied and after five times, the following clusters are obtained:

	${}^4\mathcal{C}_2 = [39.91, 59.02]$	${}^4\mathcal{C}_3 = [74.91, 87.09]$	${}^4\mathcal{C}_4 = [85.91, 98.09]$
${}^4\mathcal{C}_1 = [-1.09, 13.09]$	0	0	0
${}^4\mathcal{C}_2 = [39.91, 59.02]$		0	0
${}^4\mathcal{C}_3 = [74.91, 87.09]$			1

According to the values obtained at this point, it is necessary to merge the clusters ${}^4\mathcal{C}_3$ and ${}^4\mathcal{C}_4$, which contains the objects x_4 and x_5 respectively, obtaining the following clusters for the next iteration, must be merged, which corresponds with the following situation in $\ell = 3$:



Following this procedure and increasing the range of the intervals iteratively using w_1, the following overlapping is obtained:

Meaning that these are merged together making the two following clusters in $\ell = 2$:

The last iteration will conclude the method by merging the two remaining clusters together in the root of the tree representation, with one single cluster that contains all the objects in the data set.

On the other hand, for creating the groups using the classical linkage methods, it is necessary to perform a pairwise comparison of the objects in $\mathbb{X}$ to compute their distance. The results obtained from this comparison are shown below:

	$x_2 = 7$	$x_3 = 46$	$x_4 = 81$	$x_5 = 92$
$x_1 = 5$	2	41	76	87
$x_2 = 7$		39	74	85
$x_3 = 46$			35	46
$x_4 = 81$				11

Using this, when the single linkage method is employed, the clusters obtained are the same as the ones with our method (left dendrogram). However, when the complete, average or centroid linkage methods are considered the results are different (right dendrogram).

The one-dimensional version of the algorithm yields the same results that the single linkage method using Euclidean distance. Intuitively, it is natural that in one dimension the results

are the same than using this binary function as, the closer the objects are, the sooner they will overlap. However, the single method is only equivalent to our method in the case of objects with one dimension. A counterexample showing that this is not the case when the number of variables increases is shown below.

Example 4.2. *Here we present a counterexample using a data set where the objects are defined with two variables: age and height. This shows that our method is different from single linkage using Euclidean distance in the case when the number of the variables is greater than one. Let us consider the following data set (left) and the corresponding pairwise distance between the objects of the data set (right):*

	Age	*Height*
x_1	*10*	*140*
x_2	*11*	*150*
x_3	*10*	*170*

	x_2	x_3
x_1	10.04988	30
x_2		20.02498

According to the Euclidean distance, the most similar objects in the bottom level of the tree (i.e. $\ell = 3$) are x_1 and x_2. This means that these two objects are merged into one single cluster ${}^2C_1$ in the upper level $\ell = 2$. Then, the third object x_3 will be merged with this cluster in an upper level of the tree, creating the single cluster in the root ${}^1C_1$ that contains all the objects. On the other hand, with our method, we obtain a different result. First of all, let us transform the data set into a data set $\mathbb{Y}$ of degenerated intervals (shown below on the table). The individual results of each pairwise comparison of the objects after applying the function f to each variable are shown in the tables below. Then, the aggregation function is applied to each pairwise comparison, such that the table Average is obtained. The objects y_1 and y_3 are the most similar according to the first approach so these would be merged together first.

	Age	*Height*
y_1	*[10,10]*	*[140,140]*
y_2	*[11,11]*	*[150,150]*
y_3	*[10,10]*	*[170,170]*

Age

	y_2	y_3
y_1	0	1
y_2		0

Height

	y_2	y_3
y_1	0	0
y_2		0

Average

	y_2	y_3
y_1	0	1/2
y_2		0

So using this method, on the contrary to what happened using the single linkage method with the Euclidean distance, the objects y_1 and y_3, (which represent the original x_1 and x_3) form ${}^2C_1$. This makes the clusters in $\ell = 2$ be different than the ones obtained before.

As illustrated in the previous examples, classical methods based on distances require comparing all the pairs of objects in the cluster. On the contrary, with the proposed method, this matrix is reduced in each step which benefits the implementation of the algorithm as well as its suitability to be applied in practice to large problems. Furthermore, using the basic function f that we are considering in this work, the Boolean matrix used is more efficient and less memory space is needed to store the comparison.

5. Conclusion and future work

In this work, a new hierarchical clustering method based on grouping intervals has been proposed. This method provides an approach that uses a coincidence function as well as an aggregation function, which means that can be extended by changing these parts in order to explore different properties of the created groups. Future work will include the exploration of different aggregation functions in order to create the groups.

Acknowledgments

This research has been partially supported by Spanish MINECO projects TIN2017-87600-P (Noelia Rico and Irene Díz) and PGC2018-098623-B-I00 (Pedro Huidobro and Susana Montes). Pedro Huidobro and Noelia Rico are alspp supported by the Severo Ochoa predoctoral grant program by the Principality of Asturias (PA-20-PF-BP19-169 and PA-20-PF-BP19-167, respectively).

References

[1] L. Rokach, O. Maimon, Clustering methods, in: Data mining and knowledge discovery handbook, Springer, 2005, pp. 321–352.

[2] S. C. Johnson, Hierarchical clustering schemes, Psychometrika 32 (1967) 241–254.

[3] S. Galdino, P. Maciel, Hierarchical cluster analysis of interval-valued data using width of range euclidean distance, in: 2019 IEEE Latin American Conference on Computational Intelligence (LA-CCI), 2019, pp. 1–6. doi:10.1109/LA-CCI47412.2019.9036754.

[4] A. B. Ramos-Guajardo, A hierarchical clustering method for random intervals based on a similarity measure, Computational Statistics (2021) 1–33.

[5] C. C. Chang, P. W. Lu, J. Y. Hsiao, A hybrid method for estimating the euclidean distance between two vectors, in: First International Symposium on Cyber Worlds, 2002. Proceedings., 2002, pp. 183 – 190. doi:10.1109/CW.2002.1180878.

[6] R. Lustig, Angle-average for the powers of the distance between two separated vectors, Molecular Physics 65 (1988) 175 – 179. doi:10.1080/00268978800100931.

[7] I. Olkin, F. Pukelsheim, The distance between two random vectors with given dispersion matrices, Linear Algebra and its Applications 48 (1982) 257 – 263. doi:https://doi.org/10.1016/0024-3795(82)90112-4.

[8] F. Murtagh, A survey of recent advances in hierarchical clustering algorithms, The computer journal 26 (1983) 354–359.