URL: Scalable Hierarchical Algorithm

Fernando Vázquez Novoa

28 May 2021

1 Introduction

The goal of this work is not to develop a new algorithm, the idea is to choose a paper and implement the algorithm or algorithms described on the paper and compare these algorithms against other well accepted methods. The main idea is to reproduce and check the results obtained in the paper.

The option selected between the available options was the Scalable Hierarchical Algorithm. On this option the paper was the [1]. The methods explained on the paper are methods of clustering. Inside the clustering algorithms there are two main categories: hierarchical clustering and partitional clustering. The methods explained and improved on this paper are variations of the Single-Link algorithm and they use algorithms that belong to both categories.

1.1 Assumptions

The Single-Link algorithm builds a dendogram where each level represents a clustering of the dataset. The clustering is chosen from the dendogram based on some requirements. There are some different stopping conditions available for deciding where to stop building the dendogram.

For performing the clustering on this paper it is applied the Single-Link with the *h-distance stopping condition.* This condition states that the construction of the dendogram stops when all the clusters are separated by a distance of at least h.

1.2 Improvement

The proposal of this paper is to modify the classical Single-Link algorithm. The change to this algorithm is to apply first of all the leaders algorithm and then cluster the leaders returned using the Single-Link algorithm. The final step of the new algorithm is to replace the leaders by their followers.

This change is done to correct the two main disadvantages of the Single-Link method. These disadvantages are: its time complexity of $O(n^2)$ and the need to scan the dataset many times. These problems are corrected by applying the leaders-Single-Link algorithm.
This new algorithm presents a new problem, it may deviate from the original clustering obtained by applying only the Single-Link method. This deviation in the majority of cases is not problematic. For correcting the deviation is also proposed another algorithm on the same paper, an augmented version of the l-SL algorithm.

2 Background

2.1 Leaders clustering

It is a partitional clustering method. The partitions are made based on the distance between the elements. It receives a threshold distance and then it produces a set of leaders. It computes the distance of each element to all leaders and if there is a leader at least distance than the threshold distance, the example is assigned as a follower of that leader. In teh case that there are no leader nearer than the threshold distance the point constitutes a new leaders. The leaders can be seen as a representative of a cluster of the patterns/elements grouped with it.

Its time complexity is O(mn) being m the number of leaders and the space complexity is O(m). It only needs to scan the dataset once.

2.2 Single-Link clustering

This is an agglomerative hierarchical clustering method. On this paper it is used the distance based version of this method where the stopping criteria is the minimum inter-cluster distance. The distance between any two clusters is the minimum distance between all the examples on both clusters. This algorithm stops when there are no clusters with a distance between them minor to h, and returns the dendogram build until that moment.

The time and space complexity of this method is $O(n^2)$. It needs to perform several scans over the dataset.

3 Algorithms proposed

The algorithms proposed only need one parameter h. It is the threshold distance that is going to be used with the Single-Link algorithm.
In order to obtain the set of leaders that is going to be clustered it is needed to fix another threshold distance for the leaders algorithm. It is obvious that this other threshold should be smaller than h. From experimental results presented on the paper it was observed that the minimum execution time of the algorithm l-SL is obtained using h/2. Also it was observed than using h/2 or less throws results at par of the results obtained from applying Single-Link algorithm directly.

3.1 l-SL Algorithm

The pseudo-code appended on this section contains changes with respect to the pseudo-code present on the paper. The changes were done in order to make the pseudo-code more similar to the implementation followed during the realization of this work. The original pseudo-code was considered too abstract.

Algorithm 1 l-SL Algorithm

```
1: procedure L-SL (D GIVEN DATASET, H THRESHOLD DISTANCE)
2:     τ ← h/2
3:     L ← Leaders(τ, D)
4:     π_L ← SL(L, h)
5:     π_l = ∅
6:     for i in π_L do
7:         followers ← Find_Followers(i)
8:         π_l ← append(followers)
9:     Return π_l
```

First of all a set of leaders using the distance h/2 is obtained and then a Single-Link clustering is performed to this set of leaders. The final results are obtained replacing each leader by its followers.
The overall running time of this algorithm is $O(mn+m^2)$. Due to the fact

that m is always smaller than n the final time complexity is assumed to be O(mn). The space complexity of this algorithm is going to be $O(m^2)$. This algorithm is faster than the classical SL algorithm because it only applies the SL method to the set of leaders while the classical one is applied to the whole dataset.

It can be proven that for same value of h the clusters generated by SL algorithm and l-SL algorithm are the same. The condition is demonstrated by two lemmas:

- Lemma 1:

  $$\text{For } x, y \in Dataset, \text{ if SL satisfies } x \notin y_\pi \text{ then l-SL satisfies } x \notin y_\pi \tag{1}$$

  Proved by contradiction. In order to group two patterns in the same cluster with the SL algorithm it is needed to have a sequence of patterns with a distance smaller than h between them successively.
  For clustering x and y in the same cluster with the l-SL algorithm they need to be followers of leaders that are at a distance smaller than h in order to be clustered in the same cluster by the l-SL algorithm. This condition can be true to the l-SL algorithm since the leaders take patterns that are at h/2 or less distance from them. But being clustered with this condition means that with the SL algorithm the two patterns would also be clustered in the same cluster because there is a set of patterns between them on which each pattern is at a distance smaller than h with respect to the following pattern.

- Lemma 2:

  $$\text{For } x, y \in Dataset, \text{ if SL satisfies } x \in y_\pi \text{ then l-SL may not satisfy } x \in y_\pi \tag{2}$$

  The scenario where two patterns are nearer than h to eachother but they are followers of leaders that are separated by a distance greater than h can happen. This is possible due to the fact that the followers can be at a distance nearer than h/2 from their leaders. If the leaders of these two patterns are at opposite directions with respect to the patterns it can happen that the distance between the patterns plus the distance between each pattern and its leader is higher than h and the distance between the two patterns can be smaller than h. On this scenario the l-SL algorithm would not cluster the two patterns in the same cluster while the SL algorithm would do it.

The clustering obtained by the l-SL algorithm may be different than the one obtained applying only the SL algorithm, but it can be considered as a refinement of the clustering. The deviation is usually small or inexistent.

3.2 al-SL Algorithm

The pseudo-code appended on this section contains changes with respect to the pseudo-code present on the paper. The changes were done in order to make the pseudo-code more similar to the implementation followed during the realization of this work. The implementation proposed during the realization of this work avoids the realization of some steps presented on the original pseudocode. In the merging part it is not needed to apply a second time the leaders algorithm as it is proposed in the original pseudo-code.

Algorithm 2 al-SL Algorithm

procedure AL-SL (D GIVEN DATASET, H THRESHOLD DISTANCE)
$\tau \leftarrow h/2$
$L \leftarrow Leaders(\tau, D)$
$\pi_L \leftarrow SL(L, h)$
$S \leftarrow \emptyset$
for each pair of cluster $(B_i^l, B_j^l) \in \pi_L$ with Distance $(B_i^l, B_j^l) \leq$ 2h **do**
Identify l_i, l_j that $\| l_i - l_j \|$ = Distance (B_i^l, B_j^l)
$L_{Bi} \leftarrow \emptyset$
$L_{Bj} \leftarrow \emptyset$
for l_x in B_i^l **do**
if $\| l_x - l_j \| \leq 2h$ **then**
$L_{Bi} = L_{Bi} \cup \{l_X\}$
for l_y in B_j^l **do**
if $\| l_y - l_i \| \leq 2h$ **then**
$L_{Bj} = L_{Bj} \cup \{l_y\}$
$S = S \cup L_{Bi} \cup L_{Bj}$
if $S \neq \emptyset$ **then**
for each pair of cluster $(B_i^l, B_j^l) \in \pi_L$ with Distance $(B_i^l, B_j^l) \leq$ 2h **do**
for each pair of potential leaders (l_a, l_b) such that $l_a \in B_i^l$ and $l_b \in B_j^l$ **do**
Find nearest followers (x,y) such that $x \in l_a$ and $y \in l_b$
if $\| x - y \| \leq h$ **then**
Merge clusters B_i^l and B_j^l into one cluster and break
$\pi_{alSL} = \emptyset$
for i *in* π_L **do**
$followers \leftarrow Find\_Followers(i)$
$\pi_{alSL} \leftarrow append(followers)$
Return π_{alSL}

The augmented version of the l-SL algorithm is proposed in order to correct the small deviations obtained in the clustering between the l-SL algorithm and the SL algorithm. This new algorithm produces a clustering merging the pair of clusters that contain some pattern as followers of leaders (of different cluster) that are nearer than h/2.

It is not possible that two different clusters are separated by a distance smaller than h. It can also be proved that the clusters that are separated by a distance bigger than 2h are not possible to being merged, as there are no patterns inside each of the clusters that are separated by a distance smaller than h between them.
The al-SL algorithm searches for the clusters that are separated by a distance between h and 2h obtained from the clustering performed by the l-SL algorithm. This is the set of clusters that are candidates to being merged.
Then the two nearest leaders from each pair of different clusters (with an inter-cluster distance smaller than 2h) are selected (the leaders that define the inter-cluster distance). With respect to this two leaders there are selected all the leaders of the opposite cluster that are at a distance smaller than 2h. All the leaders that accomplish this condition are the candidate leaders for possible merging of the clusters. All the followers of these leaders need to be found and then the distance is computed between each pair of patterns. The merging is done if any two patterns of the different clusters are at a distance smaller than h.
By performing this merging operation the deviation from the clustering produced by the SL-algorithm is corrected and the same clustering is produced.

The time complexity of this new algorithm is O(mn), this is also the time complexity of the merging part. The merging part of the algorithm presents different time complexities but this can be assumed to be the biggest one.
The space complexity of this augmented version of the algorithm is $O(m^2)$, it also presents different space complexities in the merging part, but the highest complexity is this one.

The clustering results produced by this algorithm are independent of the scanning order of the dataset by leaders clustering method.

3.3 Estimation of h value

There are two approaches for finding the optimal h value for a dataset:

- *First approach:* Let some domain experts select the inter-cluster distance, as they may know well the distance between the natural clusters on their domains.

- *Second approach:* Random patterns are selected from a dataset and the single-link algorithm is applied to them. From the dendogram generated it can be computed the life time. The life time is the difference

of the distances between the two nearest clusters in two consecutive layers of the dendogram. Maximum life time is the one for which the life time is maximum and this can be considered as the value of h.

4 Preparing the evaluation

The evaluation of the algorithm consisted on different parts. The first part was to prepare the algorithm and select the datasets in order to make a fair evaluation between all the algorithms and their results on the different datasets.

4.1 Preparing the algorithm

Finding suitable value for h

In order to find the suitable value for the h parameter the approach used is the second one proposed. The life time is obtained by the differences of distances between the nearest clusters in two consecutive layers of the dendogram. The maximum life time is the biggest difference obtained between two consecutive layers in the clustering and will set the value of h.

In order to make easy to obtain the optimal h value following the second approach a method that computes this value is added to both algorithms, to the l-SL algorithm and to the al-SL algorithm. The pseudo-code of the algorithm proposed is the following one:

Algorithm 3 Calculate_Maximum_Life_Time

procedure CALCULATE_MAXIMUM_LIFE_TIME (X SUBSET OF THE GIVEN DATASET (THE SUBSET CONTAINS $\sqrt{len(D)}$ RANDOMLY SELECTED INSTANCES)
    $\pi_i \leftarrow SL(X)$
    $LF \leftarrow 0$
    for d_i *in* $\pi_i.distances$ **do**
        $actual\_LF \leftarrow d_i - d_{i-1}$
        if $actual\_LF \geq LF$ **then**
            $LF \leftarrow actual\_LF$
    Return MLF

In addition to the possibility of computing the value of the h parameter

in this way, it can be set manually as a parameter the distance h. This is useful if some domain expert wants to evaluate one dataset and that domain experts knows the distances of that domain. It also can be used to compare the performance and results of the algorithm for different values of the h parameter.

The parameter τ used for executing the leaders algorithm and extracting the leaders of the dataset is set automatically at the value of h/2 when initializing the class and by using the method of calculating the maximum life time. It can be changed using the method set_params of both classes (l-SL algorithm and al-SL algorithm).

Optimizing the code and execution times

Making the first experiments of the l-SL algorithm and the al-SL algorithm and comparing their execution times with the SL algorithm it was observed that the execution times of the first two algorithms were considerably higher than the execution times of the SL algorithm. Theoretically both algorithms have a smaller time execution than the SL algorithm and should end their execution faster than this algorithm.
It should be highlighted that the SL algorithm used for comparison is the AgglomerativeClustering with linkage='single' from the Sklearn library, a very optimized implementation almost impossible to improve.

For measuring the difference with real measures the execution times for a dataset of 3772 instances (Hypothyroid dataset, described after on this document) were:

- SL algorithm: about 0.5 seconds.
- l-SL algorithm: about 2.5 seconds.
- al-SL algorithm: about 55/60 seconds.

In order to correct this execution times an improvement on the code was done trying to make it more efficient. The pseudo-code included a big number of for loops and condition statements that can be reduced to comprehension lists on Python, a structure that is much faster than the previous mentioned statements.

Algorithm 4 improvement on l-SL Algorithm

procedure L-SL (D GIVEN DATASET, H THRESHOLD DISTANCE)
$\tau \leftarrow h/2$
$L \leftarrow Leaders(\tau, D)$
$\pi_L \leftarrow SL(L, h)$
$\pi_l = \emptyset$
for i *in* $range(\pi_L)$ **do**
$axis \leftarrow [j$ for j, x in $enumerate(\pi_L.labels)$ if $x == i]$
$\pi_l[axis] \leftarrow i$
Return π_l

This first algorithm gets a small improvement on the last loop. This small improvement decreases its execution time. The new execution time is smaller and is near the execution time of the SL algorithm.

Algorithm 5 improvement on al-SL Algorithm

procedure AL-SL (D GIVEN DATASET, H THRESHOLD DISTANCE)
$\tau \leftarrow h/2$
$L \leftarrow Leaders(\tau, D)$
$\pi_L \leftarrow SL(L, h)$
$S \leftarrow \emptyset$
for each pair of cluster $(B_i^l, B_j^l) \in \pi_L$ with Distance $(B_i^l, B_j^l) \leq$ 2h **do**
Identify l_i, l_j that $||\ l_i - l_j\ ||$ = Distance (B_i^l, B_j^l)
$L_{Bi} \leftarrow [leader$ for $leader$ in B_i^l if $||l_j - leader|| < 2h]$
$L_{Bj} \leftarrow [leader$ for $leader$ in B_j^l if $||l_i - leader|| < 2h]$
$S \leftarrow S \cup L_{Bi} \cup L_{Bj}$
if $S \neq \emptyset$ **then**
for each group of potential leaders $(L_{Bi}, L_{Bj}) \in S$ with possibility of being merged **do**
$followers_{LBi} \leftarrow [follower$ for $follower$ of $leader$ in $L_{Bi}]$
$followers_{LBj} \leftarrow [follower$ for $follower$ of $leader$ in $L_{Bj}]$
Find two nearest followers (x,y); $x \in followers_{LBi}, y \in followers_{LBj}$
if $||x - y|| < h$ **then**
$to\_merge \leftarrow L_{Bi}, L_{Bj}$
Perform the merges annotated in to_merge
for i *in* $range(\pi_L)$ **do**
$axis \leftarrow [j$ for j, x in $enumerate(\pi_L.labels)$ if $x == i]$
$\pi_a lSL[axis] \leftarrow i$
Return π_{alSL}

As it can be seen on this pseudo-code some conditions and loops from the algorithm 2 has been avoided and reduced to comprehension lists. For example the potential leaders to merge of each pair of clusters are evaluated all together instead pair by pair and the recover of teh followers of each leader is performed on comprehension lists instead of loops. More improvements are done with respect to the first pseudo-code.
The changes on this pseudo-code accelerates the time execution between 4/6 times in the Hypothyroid dataset, the execution times drops down from 55/60 seconds until 10/15 seconds.

The execution times, for the Hypothyroid dataset, after making the im-

provements are:

- SL algorithm: about 0.5 seconds.
- l-SL algorithm: about 0.8 seconds.
- al-SL algorithm: about 10/15 seconds.

4.2 Preprocessing

The instances of each dataset contains different attributes. That attributes can be of different types, numeric, symbolic or boolean. They also can contain missing values, or the numeric attributes can be defined with different ranges of domain.
In order to deal with the different types of attributes, the missing values and the difference between the numeric attributes a preprocessing step takes place. The aim of this preprocessing step is to paliate these differences and problems of the data in order to use a data with a higher quality and getting better and more meaningful results from the use of the algorithm.
The missing values were tackled in a different way depending on if the attribute that contains the missing value is numeric or symbolic. In the case that the missing value is on a symbolic attribute that missing value is going to be filled with the mode value of all the instances of that missing attribute. In the case the missing value occurs in a numeric attribute that value is filled with the median value of all the instances of that attribute.

4.3 Evaluation of the datasets

Before applying different clustering techniques to the datasets in order to compare the results of these techniques it would be a good practice to evaluate how good these different datasets are for the task of clustering. In order to choose meaningful datasets for the task evaluated, and also for knowing what to expect (in quality of the clustering returned) of the different clustering algorithms.
The measure selected for knowing the clustering tendency of the different datasets was the Hopkins Statistic, already explained during this course.
The Hopkins Statistic returns a value around 0.5 if the dataset is uniformly distributed (with no clustering tendency). It is computed in the following way:

1. It samples n (p_i) points from the dataset and computes their distance to their nearest neighbour ($d(p_i)$).

2. It generates n points (q_i) that are uniformly distributed in the space of the dataset and computes the distance to their nearest neighbour of the points of the dataset ($d(q_i)$).

3. The last step is to compute the following equation:

$$\frac{\sum_{i=1}^{n} d(p_i)}{\sum_{i=1}^{n} d(p_i) + \sum_{i=1}^{n} d(q_i)} \tag{3}$$

These calculations are not done by hand and it was made a function in Python in order to evaluate the Hopkins Statistic of the different datasets.

Datasets evaluated

The following ones are the clusters that were evaluated using the Hopkins statistic and the value of clutering tendency that they obtained:

- **Hypothyroid:** It is a dataset about Thyroid diseases, it contains 30 attributes, 23 symbolic or discrete attributes and 7 numeric attributes. It contains 3772 instances and they belong to four different classes. Hopkins Statistic: 0.966

- **Heart-Statlog:** Contains 13 attributes, all numeric and the data belongs to two different classes. It contains 269 instances. Hopkins Statistic: 0.702

- **Wine:** Contains 13 attributes, all numeric attributes and the data belongs to three different classes. It contains 177 instances. Hopkins Statistic: 0.724

- **Glass:** Dataset that contains 9 numeric attributes, 7 different classes of data. It contains 213 instances. Hopkins Statistic: 0.886

4.4 Selection of algorithms to compare

These algorithm as it is explained in this document combines two clustering algorithms, these algorithms are from a different type of clustering algorithms. One of them (the Single-Link algorithm) belongs to the agglomerative clustering algorithms and the other (the leader's algorithms) belongs to the partitional clustering.

For this reason it was decided to compare this algorithm with well-known methods of both families of algorithms. The algorithms selected are the following ones:

- **Single-Link:** Method already explained on the subsection 2.2 of this document. This algorithm is going to be used to validate the results obtained using the l-SL algorithm and the al-SL algorithm. Small differences between the l-SL and SL algorithm should be obtained, or no differences and no differences between the al-SL algorithm and the SL algorithm.

- **K-Means:** Partitional method that clusters the data trying to separate it in a selected number of groups with the same variance minimizing the within-cluster sum-of-squares criterion. This criteria is defined as:

$$\sum_{i=0}^{n} min(||x_i - \mu_j||^2) \quad \mu_j \in C \tag{4}$$

  Being C the centroid of each of the analyzed clusters.

- **DBSCAN:** This method clusters the data by density areas, it separates the areas with high density from areas of low density. Due to this view of the clusters the DBSCAN method can find clusters of any shape, this property is not common to clustering algorithms as K-Means or Single-Link.

- **Expectation-Maximization:** It is a method that creates the clusters iteratively. In the first step it assigns a random position to cluster centers on the data, and it assigns a probability of each point of belong to each cluster center. Then it tweaks the parameters in order to maximize the likelihood of the data given that assignments.

Different kinds of algorithms were chosen in order to compare the clustering produced by this algorithm with different kinds of clustering methods. This clustering algorithms that are going to be used to compare with respect to the l-SL algorithm and to the al-SL algorithm are not going to be developed. The version of these algorithms that is in the scikit-sklearn library of python is going to be used.

The idea is to implement the new algorithms as efficient as possible in order to try to make a fair comparison with the rest of the algorithms. It should

be taken into account that it is not assumed that an implementation as efficient as the implementations of the algorithms provided by the Sklearn library is going to be reached.

4.5 Evaluation criteria

During this course of Unsupervised and Reinforcement Learning it was explained that different criteria can be used when evaluating the clustering results.
There are three main types of criteria in order to evaluate a clustering:

- Internal Criteria: Techniques that measure the properties expected in a good clustering. For example if the groups present on the clustering are compact and if the groups are well separated between them.
- External Criteria: These techniques usually measure the similarity of the clustering with respect to a partition of the data. If there are no partition to compare the data with they can be used to compare the clusters produced by different algorithms or parameters of the same algorithm.
- Relative Criteria: They compare the results obtained employing different clusterings.

Internal Criteria

The measures selected of the type of internal criteria are:

- Calinski-Harabasz: It measures the ratio of the interclass distance with respect to the intraclass distance of the clustering.
- Davies-Bouldin criteria: It calculates the maximum interclass-intraclass distance ratio. The difference with the first one is that this one returns only the maximal one, not the sum of the different ratios.
- Silhouette index: This measure returns the ratio between the maximum class spread of the clustering and the variance present on the clustering.

This criteria were selected because as it was indicated during this course they perform well in a wide range of situations, but they have a big disadvantage for this work specifically. In the evaluation they are going to be used

to compare the K-Means results with the results produced by the algorithms implemented for this work. The K-Means will produce clusters that have a shape that is more convex than the algorithms evaluated. This measures work well in a wide range of situations but they assign higher scores for convex clusters. When doing the comparison this fact should be taken into account.

External Criteria

The measures selected of the type of external criteria are:

- Adjusted Mutual Information: It measures the Mutual Information between two clusterings, it also corrects the effect of agreement solely due to chance between clusterings.
- Adjusted Rand Statistic: It measures the similarity between two clusterings. In this case it is adjusted for the chance grouping of elements.

4.6 Results and comparison

Internal Criteria Evaluation

The first measures extracted from the algorithm were the Calinski-Harabasz, Davies-Bouldin and Silhouetted index. The returned values for the two developed algorithm and the Single-Link method for the different datasets are in the following tables:

	CH	Silhouette	DB
l-SL	422.23	0.605	0.27
al-SL	436.95	0.62	0.26
SL	436.95	0.62	0.26

Table 1: Hypothyroid results

	CH	Silhouette	DB
l-SL	8.69	0.018	0.65
al-SL	8.69	0.018	0.65
SL	8.69	0.018	0.65

Table 2: Wine results

	CH	Silhouette	DB
l-SL	16.24	0.169	0.516
al-SL	16.49	0.172	0.519
SL	16.49	0.172	0.519

Table 3: Heart-statlog results

	CH	Silhouette	DB
l-SL	16.81	-0.096	0.45
al-SL	17.08	-0.036	0.416
SL	17.08	-0.036	0.416

Table 4: Glass results

As it can be seen on the results in the tables, the al-SL algorithm and the SL algorithm return the same values for the different measures obtained from teh clustering produced by them in the same dataset. In some cases there are small differences with respect to the l-SL algorithm.
It can not be assumed already that the clustering produced by the al-SL and the SL algorithm is the same. Applying the external criteria measures of the clustering it can be proved if they produce or not the same clustering.

Before analyzing these results it should be highlighted that the SL algorithm (and by consequence these other two algorithms) tend to produce clusters with elongate shape and not very circular clusters. This shape of the clusters produced by this kind of algorithms penalize the measures of the internal criteria. For example the DB measure uses the within cluster distances, that in this case is going to be higher than using other clustering algorithms. Also the Calinski-Harabasz index is going to be affected negatively as it is calculated using the compactness of the clusters and the distance between different clusters.

Other clusterings as the K-Means for example return more circular clusters, and the DBSCAN for example returns more compact clusters.

	Hypothyroid	Wine	Heart-statlog	Glass
CH	1431.1	144.3	75.45	84.7
Silhouette	0.352	0.52	0.21	0.3
DB	1.31	1.06	1.85	1.35

Table 5: K-Means results (2 clusters, only 3 were used on the Hypothyroid dataset)

As it can be seen on the table with the measures obtained from the clustering created using K-Means, the CH score returned in all datasets using the K-Means algorithm is higher than the values obtained with the three first algorithms. The higher the value of this measure the more compact are the clusters and more separated are the cluster centers. The K-Means produces more compact cluster and more separated, this are good properties that are desired when a clustering is performed.
In all datasets excepting the Hypothyroid dataset the Silhouette score is higher than the obtained using the previous algorithms. The higher the measure the better the points of a clusters are nearer with respect to the points of the same cluster and the higher is the distance with respect to the other clusters.
The last measure is the Davies-Bouldin, the lower this measure is the better the cluster are separated. The clusters produced by the l-SL and al-SL algorithms are better separated than the ones produced by the K-Means.
Despite all these values it should be highlighted that as it was said in the section 4.5 on this specific scenario the comparison is not fair, as all the three measures assign higher values to convex clusters (the type of cluster produced by the K-Means), while the l-SL and al-SL algorithm produce clusters with chain shape.

	Hypothyroid	Wine	Heart-statlog	Glass
CH	1431.1	84.15	72.77	97.13
Silhouette	0.352	0.30	0.21	0.4
DB	1.31	1.34	1.87	1.35

Table 6: E-M results

The CH score returned in all datasets using the Expectation-Maximization algorithm is higher than the values obtained with the l-SL and al-SL algorithms. The Expectation-Maximization produces compact cluster and more separated with respect to the l-SL and al-SL algorithms.
In all datasets excepting the Hypothyroid dataset the Silhouette score is

higher than the obtained using the l-Sl and al-SL algorithms. With this measure the Expectation-Maximization returns very similar values with respect to the K-Means and also depending on the dataset it performs better or not than the l-SL and al-SL algorithms.
The last measure analyzed is the Davies-Bouldin index. The clusters produced by the l-SL and al-SL algorithms are better separated than the ones produced by the Expectation-Maximization algorithm.

	Hypothyroid	Wine	Heart-statlog	Glass
CH	495.9	2.09	16.51	7.57
Silhouette	0.70	-0.11	0.17	0.3
DB	0.29	0.81	0.52	0.4

Table 7: DBSCAN results

The DBSCAN produces clusters based on the data density. It is an approach to clustering very different from all the previous analyzed clustering algorithms (also very different from l-SL and al-SL).
The clustering produced by the DBSCAN gets more similar CH values to the l-SL and al-SL algorithms and depending on the dataset one of the algorithms makes more compact and separated clusters or other.
In general this algorithm produces clusters with higher Silhouette score (higher inter-class distance and smaller intra-class distance).
The final measure compared is the DB index, that is smaller in the clusterings returned by the l-SL and al-SL algorithm. It can be concluded that the clustering produced by the DBSCAN are worst separated than the clusters produced by the two implemented algorithms.

External criteria

The first use of the external criteria was to confirm that the al-SL algorithm and the SL algorithm return the same clustering. Both measures the Adjusted Mutual Information and the Adjusted Rand Statistic were used to compare the labels produced by both algorithms. Both measures returned 1.0 that means that the two clusters contain the same information (by the Adjusted Mutual Information) and the similarity of two clusters using the Adjusted Rand Statistic (it measures the similarity between two clusters). Using only one of the measures would have been sufficient.
Also it was concluded that the l-SL algorithm can produce the same clus-

tering than the SL algorithm depending on the dataset. Very small changes on the clustering were produced when both algorithms produce different clusterings, as it was said on the section 3.1 of this document the clustering obtained using the l-SL algorithm can be considered to be a refinement with respect to the clustering returned by the SL algorithm.

4.7 Time comparison

In addition to the quality of the clustering produced by the different algorithms the time of execution should be taken into account also, and their time complexity.
A problem for making a comparison between the execution times of the different algorithms is that the implementation offered by the sklearn package is highly optimized and it is very difficult to implement an algorithm faster than the implementations offered by this package. It should be highlighted that the algorithms analyzed in this document (the l-SL algorithm and the al-SL algorithm) are much simpler with respect to the time complexity. Both algorithms have a time complexity of O(mn) being m the number of leaders, and n the number of examples to cluster, being m considerably smaller than n. While SL, K-Means and DBSCAN have a time complexity of $O(n^2)$ and the time complexity of the Expectation-Maximization algorithm has a time complexity of $O(nkd^3)$ being n the number of data points, k the number of gaussian components and d the problem dimension.
In small datasets as 3 of the datasets analyzed on this work the effect of the time complexity does not affect so much the execution times and a very efficient implementation of the algorithm returns better execution times:

Algorithm	Time (s)
l-SL	0.025
al-SL	0.39
SL	0.0037
K-Means	0.091
E-M	0.18
DBSCAN	0.064

Table 8: Glass execution times

Algorithms like the DBSCAN and the K-Means have a time execution similar to the one obtained using l-SL and lower than the one returned by the al-SL. The faster algorithm is the SL (it is theoretically slower than the l-SL

and al-SL algorithms). The execution times obtained for the Hypothyroid dataset (it is a bigger dataset, with 3772 instances).

Algorithm	Time (s)
l-SL	0.76
al-SL	12.66
SL	0.39
K-Means	3.14
E-M	2.25
DBSCAN	1.55

Table 9: Hypothyroid execution times

As it can be seen the increasing of times from the small dataset to the bigger dataset is higher in all algorithms that the increasing of time execution produced in the l-SL algorithm (excepting the E-M algorithm, it does not depend as much as the other algorithm on the training data, the gaussian components affect too and the parameters of the data). The change in execution times from the smaller dataset to the bigger one shows that despite the bad implementation of the l-SL algorithm in bigger datasets it is going to perform better than this group of well recognized clustering algorithms, at least in execution time.

The al-SL increasing of time execution from the small dataset to this bigger dataset is only higher (in proportion) than the increasing produced by the l-SL algorithm, the al-SL always will produce worst execution times than the l-SL algorithm, and also higher than the one produced by the E-M algorithm. The same as it happened with the l-SL happens with the al-SL, despite having worst execution times due to the inefficient implementation, with a higher enough dataset this algorithm is going to cluster it in less time than these well recognized clustering algorithms.

4.8 Evaluation equivalent to the evaluation done in the paper

After the evaluation proposed by the student, a evaluation performing the same experiments that were performed on the paper was done in order to get the same conclusions that were taken from the paper. On the paper the comparison of the clusterings obtained for different datasets was done only applying the Rand Index measure to the clustering produced by the l-SL

and the al-SL algorithms for comparing their clusterings with the clustering produced by the SL algorithm. Additionally an evaluation of the times expent by the algorithms was done.
It should be highlighted than in this case the comparison of the algorithms is not fair, the l-SL and the al-SL algorithm were implemented by the student while the SL algorithm is taken from the well known sklearn library. This library contains algorithms highly optimized so the version of the SL algorithm used fo rtaking the times is much more optimized than the other two algorithms.
The datasets used on this evaluation were created using the function make_blobs present on the kemlg library provided by the teacher of this subject. There were two kind of experiments done, fixing a value for h as it was done in the paper, and the second kind of experiments was done computing the h value as it was explained on the paper.

Algorithm	n=1000	n=2000	n=4000	n=8000	n=16000
l-SL	0.132	0.584	0.574	3.64	10.91
al-SL	4.134	19.345	14.459	26.85	51.67
SL	0.006	0.052	0.105	0.405	1.683
RI l-SL	0.992	0.998	0.999	0.999	0.999
RI al-SL	1.0	1.0	1.0	1.0	1.0

Table 10: Created datasets times (s) and Rand Index (with h=5)

Algorithm	n=1000	n=2000	n=4000	n=8000	n=16000
l-SL	0.128	0.264	0.538	1.13	6.988
al-SL	0.806	2.516	1.45	8.23	17.659
SL	0.006	0.024	0.101	0.399	1.57
RI l-SL	1.0	0.998	1.0	1.0	1.0
RI al-SL	1.0	1.0	1.0	1.0	1.0

Table 11: Created datasets times (s) and Rand Index (with h=6)

The dataset from the last two tables was created with the function make_blobs setting the value of cluster_std to 2.5, the n_features to 5 and the centers to 6.

Algorithm	n=100	n=1000	n=5000
l-SL	0.026	0.355	2.32
al-SL	0.381	37.00	181.62
SL	0.0009	0.014	0.169
RI l-SL	0.994	0.990	0.997
RI al-SL	1.0	1.0	1.0

Table 12: Created datasets times (s) and Rand Index with h computed using the data

These experiments show that the al-SL produces the same clusters that are produced by the SL algorithm and that the differences between the cluster produced by the l-SL algorithm and the SL algorithm are very small and they can be considered as a refinement. The RI returns always 1.0 that means the clusters are the same for the comparison of the al-SL clusters and the SL clusters. For the comparison of the clusters created by the l-SL algorithm and the SL algorithm the values returned by the RI are always higher than 0.9 and sometimes 1.0 meaning that clusters are equal or very similar.

With respect to the time it should be highlighted again that the execution time is not a fair comparison due to the fact of the very optimized implementation of the SL algorithm used. On this cases the execution time of the SL algorithm is smaller than the execution times obtained with the other two algorithms in all the experiments.

Despite the fact that the execution time obtained is worst it can be analyzed the different growings of the time and the time complexity the different increasings in time show. In Table 11 the SL increases the execution time from 0.006 seconds with a dataset of 1000 until the 1.57 seconds with a dataset of 16000 instances. The increasing on the SL times represent an increasing of 262 times, while the al-SL algorithm goes from 0.806 seconds up to 17.659 seconds that is a growing of 22 times and the l-SL time grows 55 times. On table 10 similar changes on the execution times are found. This means that despite having worst times, the complexity of the algorithms is the same as it was in the paper and the l-SL algorithm and al-SL algorithm have less computational complexity and for very large datasets this two algorithms will perform better.

With the table 12 it can be seen that the executions times are very sensitive to the h value used, with 16000 instances the al-SL took 17 seconds with h=6 but with the h computed using some data of the dataset this time increased up (with 5000 instances) to 181 seconds, this could be because the

h computed was much smaller than 6 and this have a huge impact on the executions times of the l-SL and al-SL algorithms. Despite this problem the l-SL algorithm also has a better growing in time from 100 instances to 5000 instances in the Table 12 than the SL algorithm.

5 Conclusion

As it was explained during this document an algorithm that produces the same clustering, or a refinement of this clustering, than the SL algorithm but with a smaller time complexity was developed.
Despite the inefficient implementation provided (inefficient in comparison with the implementations of the algorithms that are in the sklearn package) it was seen that the time complexity of the implementations grows as expected.
Also the properties of the clusterings produced by these two algorithms were analyzed. They produce long clusters, like "chains". These clusterings are less compact and have the cluster centers less separated than the "convex" clusterings while they have more well separated clusters. With respect to the DBSCAN ("density" clustering) the clusterings produced have the clusters more well separated and also the clusters are more compact and with more distance between the cluster centers.

During the development of this work the student had to read and summarize a scientific paper improving his research skills and getting deep knowledge of a family of clustering algorithms. He also went into deeper and had to provide an implementation of the algorithms present on the paper, after developing the algorithms an evaluation of their properties were performed. The student has learned to evaluate different properties of the clusterings using measures of internal criteria. Also the student learned how to compare different clusterings using external criteria.
Finally the student got a better understanding of how different clustering algorithms work and how are the clustering returned by these algorithms.

References

[1] Bidyut Kr. Patra, Sukumar Nandi, P. Viswanath, *A distance based clustering method for arbitrary shaped clusters in large datasets,* 2011.