[DRAFT v2] Burj Khalifa Clustering Method: The Solid Approach

Dr. Ivan Reznikov

Abstract

This paper covers a novel clustering technique named the "Burj Khalifa" method, that utilizes the concept of cluster solidity and transforms dendrogram structures into multi-way tree structures. The method, despite requiring minimum-to-none parameter tuning and no information regarding the number of clusters, provided top results on 201 benchmark and real-world clustering datasets. The performance was evaluated using Rand Index, Mutual Information, V-measure, and Fowlkes-Mallows Index metrics. The results demonstrate that the Burj Khalifa method consistently achieves the highest median values for these metrics compared to 11 other clustering techniques. Moreover, the method includes built-in outlier detection and multi-label hierarchical clustering capabilities, making it a comprehensive solution for various clustering tasks.

Introduction

Clustering is a category of unsupervised learning, which involves partitioning unlabeled data into groups with similar data instances assigned to the same cluster while dissimilar data instances are assigned to different clusters. Some of the most used and described approaches include:

- centroid-based (K-means [1], K-medoids [2])
- connectivity-based (Hierarchical Clustering [3]),
- density-based (DBSCAN [4], OPTICS [5], HDBSCAN [6])
- distribution-based (Gaussian Mixture Model [7])
- compression-based (Spectral Clustering [8], BIRCH [9])
- graph-based (Affinity Propagation [10])

Clustering methods have found extensive application in diverse fields, including marketing, biology, finance, image and video processing, and social network analysis. Clustering has been used in marketing for market basket analysis [11], targeting and positioning [12, 13], and customer segmentation [14, 15]. Clustering has been used in biology for biological sequences [16] and molecular biology [17]. Clustering has also been used in finance for fraud prevention and portfolio management [18, 19, 20]. Clustering has been utilized in image and video processing for object recognition [21], motion analysis [22], and image segmentation [23, 24]. Clustering has been used in social network analysis for influencer identification, connection prediction, and community detection [25, 26, 27]. Clustering techniques are valuable tools in many disciplines because they can draw out significant patterns from unstructured and high-dimensional data.

Different clustering techniques have their strengths and weaknesses, making them suitable for specific scenarios [28]. The choice of clustering method depends on the specific application and dataset characteristics. Table 1 summarizes the merits and drawbacks of different clustering methods.

Table 1. Merits and Drawbacks of Different Clustering Methods

Clustering Method	Merits	Drawbacks
Centroid-based	Easy to implement, computationally efficient, and suitable for large datasets.	Sensitive to the number of clusters chosen and the initial seed selected, it may converge to a suboptimal solution and be unsuitable for irregularly shaped or overlapping clusters. Highly sensitive to outliers.
Connectivity-based	Flexible, produces a hierarchy of clusters, and robust to noise and outliers.	Computationally intensive, not scalable to large datasets, and affected by choice of linkage criteria
Density-based	Handles irregularly shaped clusters and varying densities and can automatically determine the number of clusters by most frequent tree distance.	Requires careful tuning of hyperparameters, sensitive to choice of distance metric, and not suitable for varying densities.
Distribution-based	Models underlying data distribution, handles overlapping clusters and provides probabilistic cluster assignments.	Sensitive to the number of mixture components, may suffer from local optima, and may not perform well for high-dimensional datasets or data-insufficient clusters. Data points in clusters should follow Gaussian distribution.
Compression-based	Handles large datasets and datasets with varying densities	Sensitive to the choice of hyperparameters, computationally intensive, and not suitable for complex structures
Graph-based	Handles complex relationships, not limited to fixed cluster shapes, and detects dense and sparse clusters	Computationally expensive, sensitive to the choice of similarity metric, and requires careful tuning of hyperparameters

As can be observed from Table 1, clustering approaches have several drawbacks while being frequently employed for unsupervised machine learning applications. It can be difficult to precisely estimate the number of clusters in real-world datasets, a prerequisite for many clustering techniques. Some clustering algorithms produce distinct primary points because they are sensitive to the initial seed choice. Additionally, outliers in the data impact the clustering outcomes, producing biased or erroneous cluster allocations. Additionally, because clustering approaches presuppose that clusters are well-defined and non-overlapping, they might not be appropriate for datasets with irregular forms or overlapping clusters. Table 2 summarizes these and other disadvantages of various clustering approaches.

Table 2. Practical disadvantages of different clustering methods

Disadvantage	Centroid	Connectivity	Density	Graph	Distribution	Compression
Algorithm configuration						
Requires prior knowledge of the number of clusters	True	Partly True	False	False	True	False
Sensitive to initial seed selection	True	False	False	False	True	False
Requires careful tuning of hyperparameters	False	False	True	True	True	True
Sensitive to choice of similarity/ distance metric or linkage criteria	False	True	True	True	False	False
Data						
Not suitable for irregular shapes or overlapping clusters	True	False	False	False	True	False
May not perform well for high-dimensional datasets	False	False	True	False	True	True
Not suitable for datasets with complex structures	False	False	False	True	False	True
Optimization						
May suffer from local optima	True	False	False	False	True	False
Computationally intensive	False	True	False	True	False	True

A hierarchical structure of the clusters is provided by hierarchical clustering, which can be used to locate and understand subgroups of the data. The dendrogram created by hierarchical clustering can assist in illustrating the data's structure and the connections between various clusters (Figure 1). One may determine the main clusters, the subclusters, and their connections to one another by looking at the dendrogram. The supplementary information can aid in understanding the underlying trends in the data. It might offer perceptions of the nature of the data that other clustering techniques cannot. Furthermore, the hierarchical structure allows for the performance of cluster analysis at various granularity levels, allowing for flexibility in the interpretation of the findings.

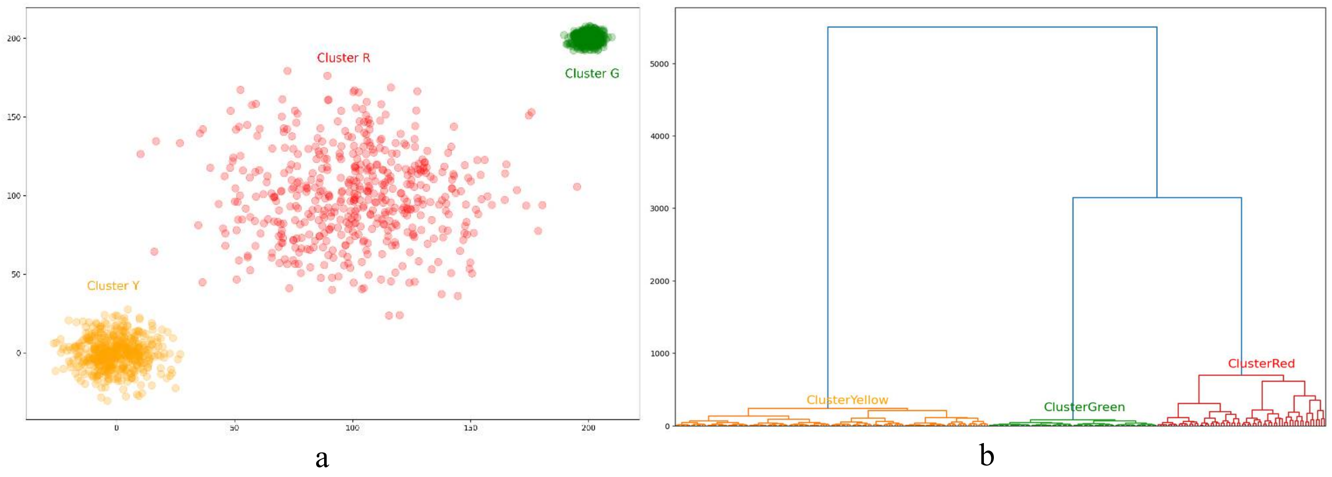


Figure 1. Generated clusters with 500 datapoints and different standard deviations (a) and its dendrogram representation (b)

Another advantage of hierarchical clustering is that it can be used with different distance metrics and linkage methods, allowing for customization based on the specific characteristics of the data. The linkage method can influence the shape and structure of the resulting clusters, and different distance metrics may be appropriate for different types of data.

Method

A density threshold in density-based clustering techniques like DBSCAN determines the size and structure of the clusters. Clusters with a high point density are referred to as "*dense*" clusters. Clusters with a higher density are more compact and may include smaller subclusters.

A dendrogram illustrates the order in which data points are combined into clusters according to their similarity or distance in hierarchical clustering. Each branching point stands in for a cluster on the dendrogram, and the branch length shows the separation between groups.

Looking at the height of the branching point that corresponds to the cluster of interest is one method for calculating the density of a cluster from a dendrogram. The distance between the clusters that are being merged determines how tall the branching point is. The density of the resulting cluster increases with decreasing distance. As a result, clusters created by joining data points at lower dendrogram heights are denser than clusters created at higher dendrogram heights.

Utilizing the cophenetic correlation coefficient is another method for calculating the density of a cluster from a dendrogram. The pairwise distances calculated from the dendrogram and the original pairwise distances between the data points are compared using the cophenetic correlation coefficient. The dendrogram provides a good representation of the initial pairwise distances when the cophenetic correlation coefficient is higher. This suggests that the clusters produced are more likely to be dense. HDBSCAN extends the DBSCAN algorithm by converting it into a hierarchical clustering algorithm. Later, a technique is employed to extract a flat clustering based on the stability of clusters.

Despite having a tree cut technique widely used for hierarchical clustering [29] there are other studies [30, 31, 32, 33, 34, 35, 36] that implemented automatic number of cluster detection using referent distribution generated under the null hypothesis of no clusters in the data, nearest neighbor concepts, Spearman distance, Calinski-Harabasz index, Fukuyama-Sugeno index [37], Gaussian mixtures and reachability plots.

This study introduces the term "*solidity*" to describe the hierarchical clustering technique. The physical quality of a substance that allows it to hold its shape and resist deformation under an applied force is known as solidity in physics and chemistry. Strong intermolecular interactions that maintain the particles of solids tightly packed in a stable arrangement give them relatively stiff and incompressible properties. Solidity is a measure of the structural stability of the cluster in the context of hierarchical clustering rather than a direct indicator of the density of the clusters. Put another way, a cluster with a high solidity is more stable and less likely to change its shape or fragment into smaller subclusters when perturbed.

As it is seen in Figure 1, the density of the cluster can be found from the height of the cluster dendrogram. The green cluster (ClusterGreen) has the highest density and the lowest dendrogram height, whereas the red cluster (ClusterRed) has the lowest density and the highest height. What makes each cluster stable or isolated is a significant height to the following data point. Further, such linkage height will be referred to as *h'*, while the standard cluster height will be shown as *h*. The explanation of the terms is illustrated in Figure 2.

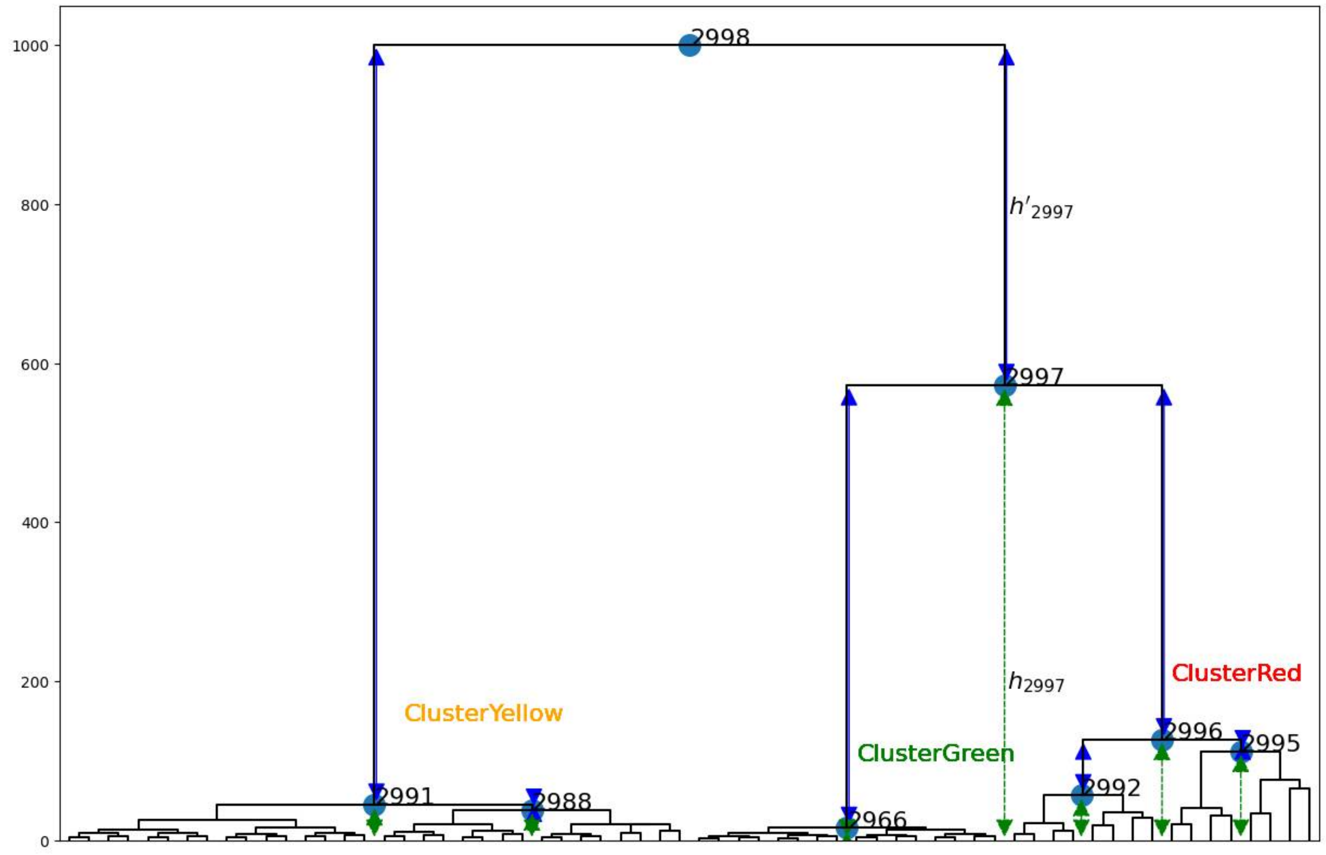


Figure 2. Modified dendrogram of Figure 1b, displaying the indexes of clusters, cluster and linkage heights.

The difference between *solidity* and *density* of clusters is shown in the formula below, where the solidity of cluster$_i$ is calculated as following:

$$solidity_i \sim \frac{N_i}{h_i^2} * h_i^{'} \sim \frac{density_i * h_i^{'}}{h_i}$$

,where N_i is the number of datapoints,
h_i – cluster height
h_i' – linkage height

For clusters 1-4 in Figure 2 the solidity calculations are performed in Table 3.

Table 3. Number of datapoints, cluster and linkage height, and calculated solidity for clusters.

Cluster	**Number of datapoints (N)**	**Cluster height (h)**	**Linkage height (h')**	**Solidity**
2966	500	15.1	557.15	1221.06
2988	306	37.92	6.46	1.37
2991	500	44.38	955.62	242.54

Cluster	Number of datapoints (N)	Cluster height (h)	Linkage height (h')	Solidity
2992	124	56.16	70.39	2.77
2995	376	111.57	14.98	0.45
2996	500	126.54	445.71	13.92
2997	1000	572.26	427.74	1.31
2998	1500	1000	(-1000)	-1.5

The definition of a "*solid cluster*" may offer benefits to the standard hierarchical clustering method. For example, we can calculate the solidity for all clusters and sub-clusters in Figure 1 and conclude that there are three separate clusters (2966, 2991, and 2996). By looking at Table 3 and Figure 2, one can identify Cluster 2966 as the most "solid" due to its highest solidity. The split of Cluster 2997 will be discussed after proposing the method algorithm.

Another important term to be introduced is "*corks*" or "*cork clusters*." If we develop an algorithm to identify "*solid clusters*," there are probably situations where some of the leaves will face multiple, single, or none "solid clusters" on their path to the root. Suppose the situation with multiple solid clusters can be resolved by comparing solidity or similar techniques. In that case, if there are no solid clusters for some leaves, that might mean one of the two options: the leaf is an outlier or no part of a solid cluster.

Dealing with outliers is spoken about later in the discussion session. If a leaf is not a single solid cluster, we will use "cork clusters." Their function is to cover the missing leaves and prevent their points from being left without a cluster assignment. The concept of "cork" clusters adds flexibility to the proposed clustering process, allowing for the inclusion of all data points and preserving the most "solid" clusters.

The overall algorithm of the proposed method can be found in Algorithm 1.

Algorithm 1.

```
Require: The dataset X to be clustered, the linkage method name.
    1. Calculate the linkage dataframe using the linkage function and the specified linkage method.
    2. Normalize the "distance" column of the linkage dataframe.
    3. Calculate the formal density of each node as the ratio of its points count to its normalized "distance".
    4. Convert the linkage dataframe to a tree structure.
    5. For each node in the tree structure, set/calculate "solidity", "chain_solidity", "self_cover", "chain_cover", and
       "children_count" attributes. The "chain" attributes are calculated as sum of children attributes. The "cover"
       attribute displays if the node is "covered" by children leaves in terms of solid clusters.
    6. Find the tree nodes to be nominated as "solid clusters" based on solidity factors:
        ○ check if nodes "chain_solidity" is higher than self "solidity"
        ○ recursively check if children's "solidity" is higher than parents
        ○ check if grandchildren on depth=n have higher "solidity" than current node or parents
    1. Find the missing "cork" nodes.
    2. Traverse the resulted tree with "cover" attributes, to define the cluster split.
    3. Finalize the end clusters list as "solid" aggregates and "corks".
    4. Assign each leaf it's ancestor from the final cluster list.
Ensure:  A well-trained clustering network N with estimated clusters K
```

Based on the algorithm above, let's take a simplified look (depth=1) how the split of Cluster 2998 on Figure 2 works:

- Current node is root (2998). Solidity of node is -1.5. Children of node are [2991, 2997]. Their solidity are [242.54, 1.31] respectively. As both children's solidity is higher than parents we split the cluster. Current solid list is empty.
- Current node is 2997. Solidity of node is 1.31. Children of node are [2966, 2996]. Their solidity are [1221.06, 13.92] respectively. As both children's solidity is higher than parents we split the cluster. Current solid list is empty.
- Current node is 2991. Solidity of node is 242.54. Children of node are [2982, 2988]. Their solidity are [6.24, 1.37] respectively. As both children's solidity is lower than parents and the depth-1 == 0 condition is fulfilled, we stop traversing the current cluster. Current solid list is [2991].
- Current node is 2966. Solidity of node is 1221.06. Children of node are [2945, 2962]. Their solidity are [15.8, 3.23] respectively. As both children's solidity is lower than parents and the depth-1 == 0 condition is fulfilled, we stop traversing the current cluster. Current solid list is [2991, 2966].
- Current node is 2996. Solidity of node is 13.92. Children of node are [2992, 2995]. Their solidity are [2.77, 0.45] respectively. As both children's solidity is lower than parents and the depth-1 == 0 condition is fulfilled, we stop traversing the current cluster. Current solid list is [2991, 2966, 2996].
- The tree is traversed, all leaves can be assigned to one of the clusters in the solid list.

In the example below, there was no situation where cork clusters were required. Otherwise, their positions are calculated to maximize the solidity of the final solid list.

As mentioned above, hierarchical clustering is a method for grouping similar data points into clusters based on distance. There are different methods for calculating the distance between the newly formed cluster and each point. Some commonly used linkage methods are:

- Single linkage: This method is computationally efficient and helps identify compact clusters with arbitrary shapes. It is particularly effective when clusters are sparse or have a large spread. Single linkage tends to produce long, branching clusters helpful in identifying outliers or hierarchical relationships in data.
- Ward linkage: This method produces well-separated, spherical clusters of approximately equal sizes. It is considered an excellent all-purpose method for most clustering problems, and it works well when the variances of the clusters are similar. However, it can be sensitive to outliers and computationally intensive for large datasets.

Other popular linkage methods include complete, average, centroid, and median linkages, each with its strengths and weaknesses. Ultimately, the choice of linkage method will depend on the specific characteristics of the dataset and the research question being addressed.

Previously an approach to evaluate which clustering linkage to choose by measuring the correlation between the distance matrix and the cophenetic distance was tested [38, 39]. Despite the method showing positive results, there are several practical drawbacks:

- limited to pairwise distance – the cophenetic distance is only concerned with pairwise distances between points rather than the overall structure of the cluster, meaning that it may not be a good measure of the overall quality of the clustering.
- sensitivity to noise – the cophenetic distance is sensitive to outliers or noise in the data and may be overly influenced by outliers rather than the underlying structure of the data.
- limitation to Euclidean distance – the cophenetic distance assumes that the underlying distance metric is Euclidean and may not be appropriate for data that is better represented by other distance metrics, such as Manhattan or Mahalanobis distance.

In this study, instead of using the cophenetic distance, the aggregated value of solidities calculated for multiple clusters is used as a decision parameter. In this case, for Algorithm 2 – an extension of Algorithm 1, we need only the data to be provided. By default, in Algorithm 2, only single and ward methods are specified, although an extension is possible.

Algorithm 2.

```
Require: The dataset X to be clustered. Optionally linkage function name or list of names can be defined, along with
optional parameters for Algorithm1.
    1.  Run Algorithm 1 for different linkage methods
    2.  Calculate the aggregation metric for clusters defined. By default mean value is selected.
    3.  Select the best performance linkage method and serve the calculated clusters
Ensure:  A well-trained clustering network N with estimated clusters K
```

Results

The clustering results are calculated for 201 datasets with ground labels, making it possible to use several extrinsic measures to evaluate the clustering method. The list of the datasets and their links is provided in the Appendix. Some standard preprocessing took part for some datasets, such as excluding rows with missing values, dummy encoding for categorical values, and columns with unique values removed (id columns).

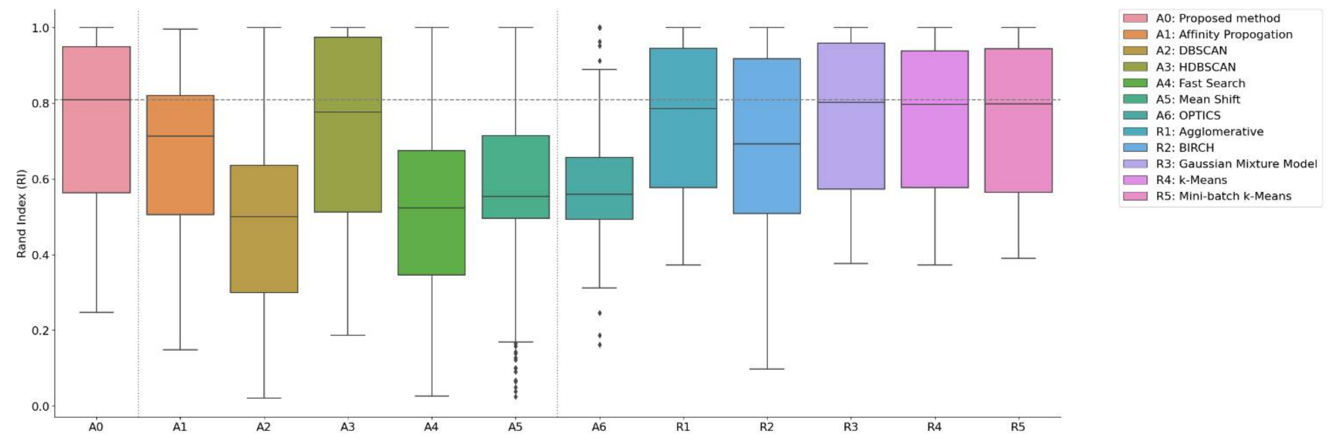
Affinity Propagation, DBSCAN, Mean Shift, OPTICS, Fast Search [40], and the proposed method, further labeled as class A (*agnostic*) methods, were run with default parameters in their Python implementations. In contrast, the class R (*required*) methods, including Agglomerative, BIRCH, Gaussian Mixture Models, k-Means, and Mini-batch k-Means, required the number of clusters parameter to be passed. For R-class methods, the true number of clusters was passed as the only parameter during experimentation. Random seed, if any method required so, was set to 1. For Agglomerative method tree-cut method was used to determine clusters.

For all 201 dataset Rand Index, Mutual Information, V-measure, and Fowlkes-Mallows Index metrics were calculated for all 11 methods under study. The aggregation method to compare the performance of different algorithms was chosen as median to avoid outlier performance on some of the sets. This practice is well-known and provides stable results, especially on a high number of datasets [41].

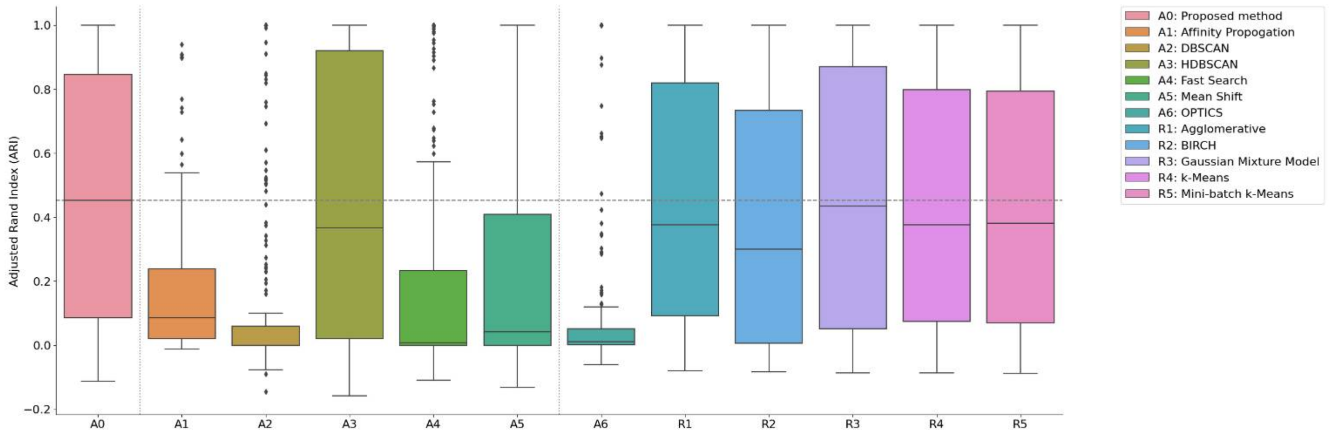
Rand Index

The Rand Index (RI) is a metric that measures the similarity between cluster assignments by comparing pairs [34]. A higher score indicates a higher degree of similarity. However, the RI does not consider the possibility of a random cluster assignment, which can lead to many false positives by chance.

The Adjusted Rand Index (ARI) corrects for chance by including a normalization term [42, 43]. The ARI is suitable for cases where the cluster structure is unknown and can be applied to all clustering algorithms.



a



b

Figure 3. RI: Rand Index (a) and ARI: Adjusted Rand Index (b) metrics for 201 studied datasets.

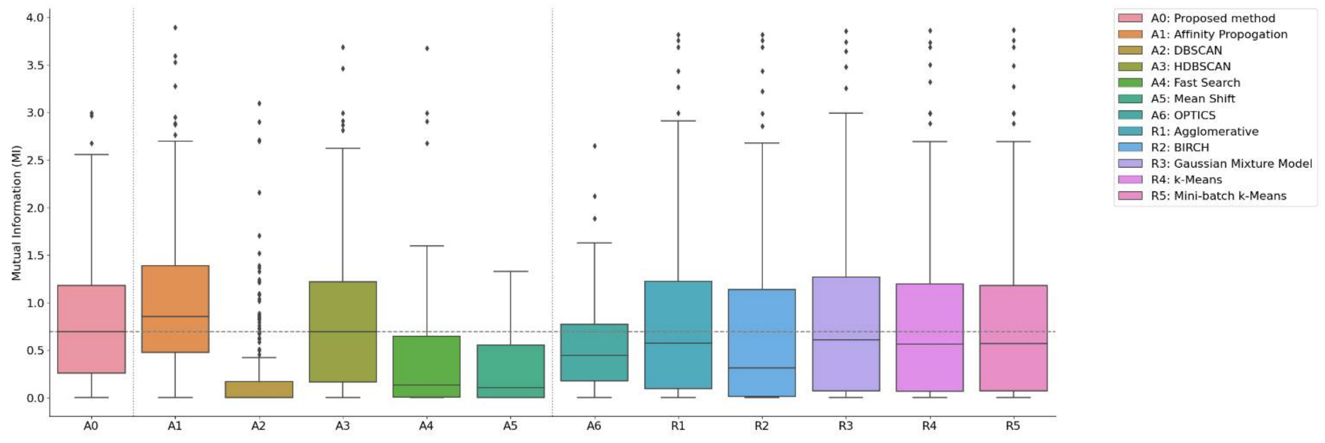
Both RI and ARI are commonly used to compare the interpretability of results.

Mutual Information

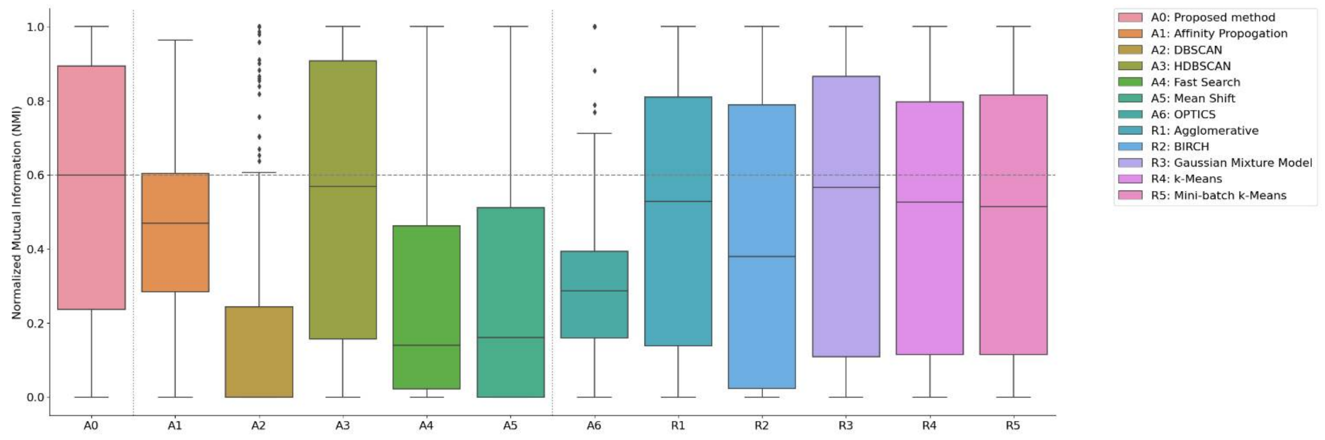
Mutual Information (MI) measures the agreement between cluster assignments based on joint and marginal probabilities [44, 45]. A higher score in MI indicates higher similarity between clusters.

As with the Rand Index, chance agreement between the true and predicted labels is possible, which can inflate the MI score. For this reason, MI has two variations: Normalized Mutual Information (NMI) and Adjusted Mutual Information (AMI), with AMI, adjusting for chance [46].

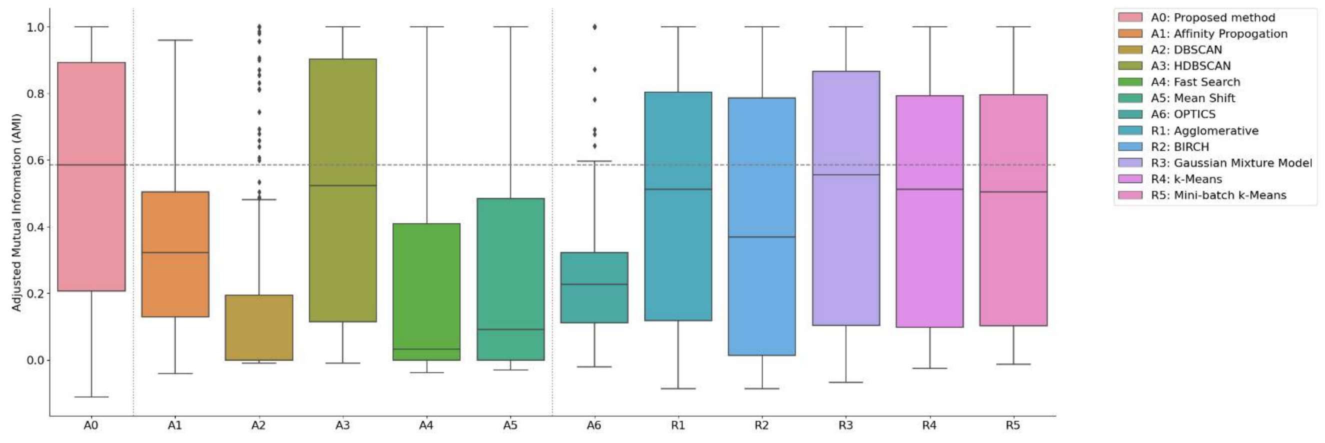
NMI normalizes the MI score by dividing it by the average entropy of the true labels and predicted labels, which considers the overall distribution of labels. AMI further adjusts the NMI score by subtracting the expected MI value that can be achieved by chance alone, which accounts for the possibility of random agreement.



a



b



c

Figure 4. MI: Mutual Information (a), NMI: Normalized Mutual Information (b) and AMI: Adjusted Mutual Information (c) metrics for 201 studied datasets.

V-measure

V-measure is a clustering evaluation metric that measures the correctness of cluster assignments using conditional entropy analysis [47]. It includes two metrics, homogeneity and completeness, which measure the correctness of cluster assignments. V-measure is the harmonic mean of homogeneity and completeness measures, similar to how the F-score is a harmonic mean of precision and recall. V-measure does not adjust for chance, meaning that random labeling would not yield zero scores, especially if the number of clusters is large. The median V-measure for the studied 201 datasets is the highest for the suggested method, which is shown in Figure 5 and Table 4.

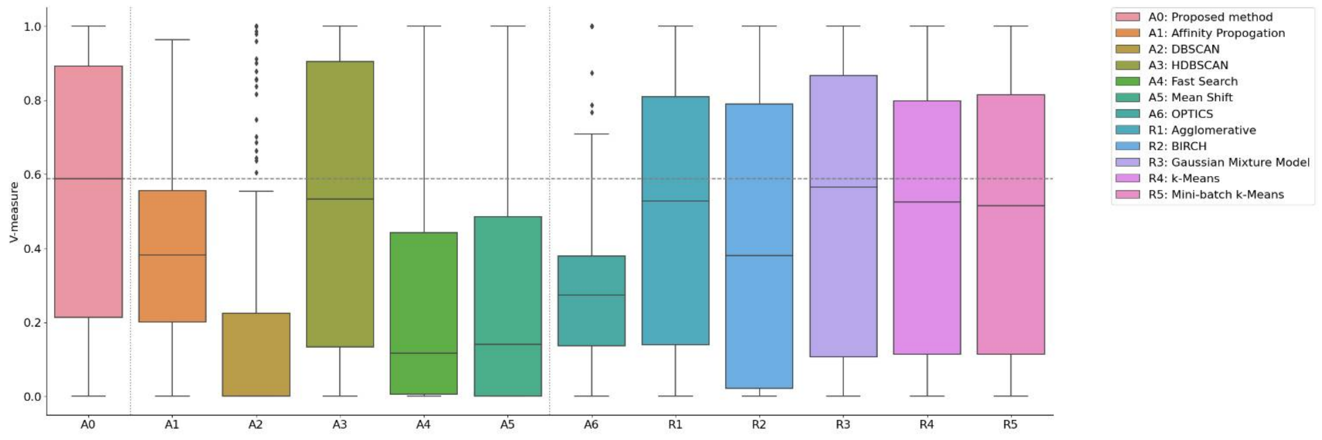


Figure 5. V-measure metric for 201 studied datasets.

Fowlkes-Mallows Index

The Fowlkes-Mallows Index (FMI) measures the correctness of the cluster assignments using pairwise precision and recall and is the geometric mean of these two measures [48]. It calculates the scores by using True Positive, False Positive, and False Negative. Unlike Rand Index and Mutual Information, FMI does not consider True Negative, which makes it unaffected by chance, and there is no need for chance adjustments. The median FMI for the studied 201 datasets is the highest for the suggested method (same score as Gaussian Mixture Models), which is shown in Figure 6 and Table 4.

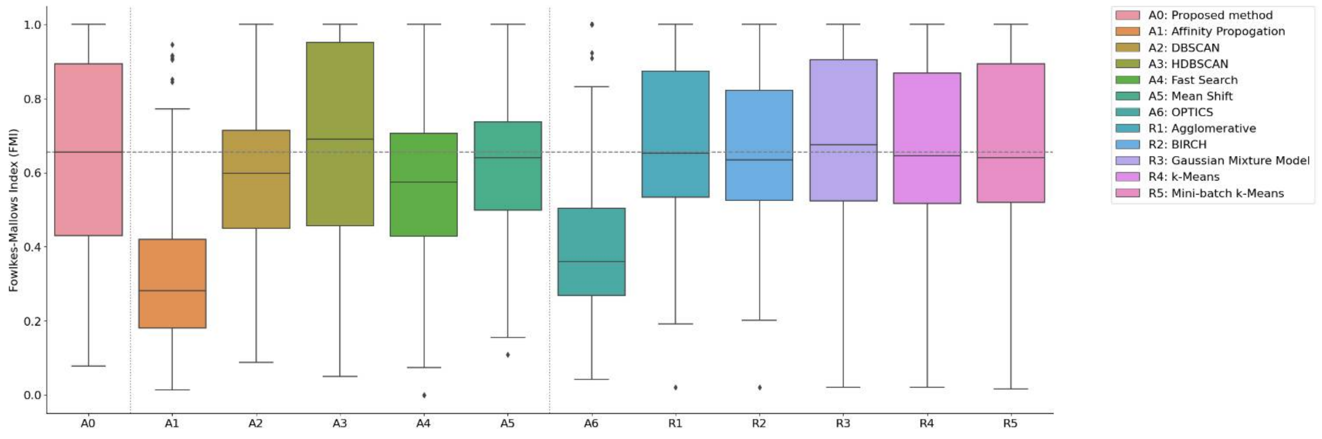


Figure 6. Fowlkes-Mallows Index (FMI) metric for 201 studied datasets.

Discussion

All the results for the studied datasets for all metrics and clustering techniques are aggregated in Table 4. As can be seen from Table 4 the proposed method performs as a TOP-3 method in 8 out of 9 metrics described above. It makes sense to focus on the metrics, that do not allow chance or shifts towards specific result (NMI, AMI, ARI, FMI and V-measure) to be judged by. In this case, the proposed method performs as a TOP-3 method in 4 out of 5 metrics (Table 4).

Table 4. Aggregation table of the medians of all metrics for all clustering techniques. Class A algorithms are agnostic to the number of clusters, while class R algorithms require their prior knowledge.

	Name	1.1	1.2	1.3	2.1	2.2	3	4.1	4.2	4.3
A0	Proposed method	0.693	**0.600**	**0.586**	**0.810**	**0.453**	0.655	0.654	0.568	**0.588**
A1	Affinity Propogation	**0.852**	0.468	0.323	0.713	0.086	0.281	**0.903**	0.260	0.382
A2	DBSCAN	0.000	0.000	0.000	0.499	0.000	0.599	0.000	**1.000**	0.000
A3	HDBSCAN	0.692	0.569	0.522	0.776	0.366	**0.691**	0.631	0.622	0.533
A4	Fast Search	0.130	0.139	0.031	0.524	0.007	0.575	0.133	0.351	0.117
A5	Mean Shift	0.106	0.160	0.092	0.553	0.042	0.641	0.113	0.802	0.141
A6	OPTICS	0.446	0.287	0.227	0.560	0.010	0.360	0.417	0.240	0.274
R1	Agglomerative	0.573	0.529	0.513	0.786	0.376	0.653	0.537	0.526	0.527
R2	BIRCH	0.315	0.380	0.368	0.692	0.299	0.635	0.345	0.581	0.380
R3	Gaussian Mixture Model	0.608	0.566	0.556	0.803	0.434	0.675	0.576	0.577	0.565
R4	k-Means	0.563	0.527	0.513	0.796	0.377	0.645	0.520	0.525	0.525
R5	Mini-batch k-Means	0.567	0.514	0.503	0.798	0.380	0.640	0.519	0.506	0.514

1.1 - Mutual Information (MI)	2.1 - Rand Index (RI)	4.1 - V-measure: Homogeneity
1.2 - Normalized Mutual Information (NMI)	2.2 - Adjusted Rand Index (ARI)	4.2 - V-measure: Completeness
1.3 - Adjusted Mutual Information (AMI)	3 - Fowlkes-Mallows Index (FMI)	4.3 - V-measure

Interpretability is vital in clustering for understanding and explaining results. Clustering is often used as an exploratory data analysis technique to identify patterns, group similar objects, or discover hidden structures within a dataset. Interpretable clustering results can provide meaningful insights and explanations about the underlying structure of the data, allowing researchers or analysts to understand and explain the results to stakeholders or decision-makers. As can be seen from Figure 3 and Table 4, the value of the median for both RI and ARI metrics, which are responsible for interpretability, is the highest for the proposed method.

Mutual Information is essential for evaluating clustering performance, as it provides a quantitative assessment of the similarity or dissimilarity between the clustering results and the true underlying clustering structure of the data. By accounting for chance agreement, NMI and AMI provide a more accurate and reliable evaluation, especially when dealing with imbalanced datasets or when the number of clusters is not known in advance (relevant for A-class methods). This makes NMI and AMI preferred choices for evaluating clustering algorithms compared to MI, as they provide a fairer and more robust assessment of clustering results. As seen from Figure 4 and Table 4, the median value for both NMI and AMI metrics is the highest for the proposed method. The pure MI metric is the highest for the Affinity Propagation method, with the proposed method being the second best. A high MI value with significantly lower NMI and AMI could indicate a high level of chance agreement between the true labels and predicted labels in the clustering results. As is seen from Figure 4, the proposed method is not affected by such a drop in metric results, meaning that the results are statistically valuable.

V-measure reflects the goal of clustering: to group similar data points together while separating different groups. It combines homogeneity (Table 4 – 4.1) and completeness (Table 4 – 4.2) into a single measure, making it preferable for algorithm performance comparison rather than comparing homogeneity and completeness separately. As one can see from the table, V-measure is the highest for the proposed method. It is also important to notice that the V-measure is not very different from the homogeneity and completeness, similar to R-class methods.

V-measure combines homogeneity (precision) and completeness (recall) using a harmonic mean. In contrast, the Fowlkes-Mallows Index calculates the geometric mean of pairwise precision and recall, utilizing True Positive, False Positive, and False Negative counts. As is seen from Table 4, the proposed method is one of the three best algorithms under study.

As one can notice from Figures 3-6 and Table 4, the proposed method is significantly better than other A-class methods and comparatively better than R-class methods. This is mostly gained by using solidity for detecting the number of clusters and automatic linkage selection. As for R-class methods, one first calculates the number of clusters (Knee, Elbow, Silhouette) and uses these in the clustering algorithm. For such operations, the number of clusters calculation is only sometimes accurate, and it is common to lose information why such a number of clusters was chosen in the clustering process. For example, for each number of clusters, one must calculate a clustering criterion or an evaluation metric that measures the quality of the clustering result. One of the popular choices of such a metric is a within-cluster sum of squares (WCSS) for K-Means. Later, the selected metric is used in a Gaussian Mixture Models method, leading to information loss as the number of clusters and the clusters are calculated differently.

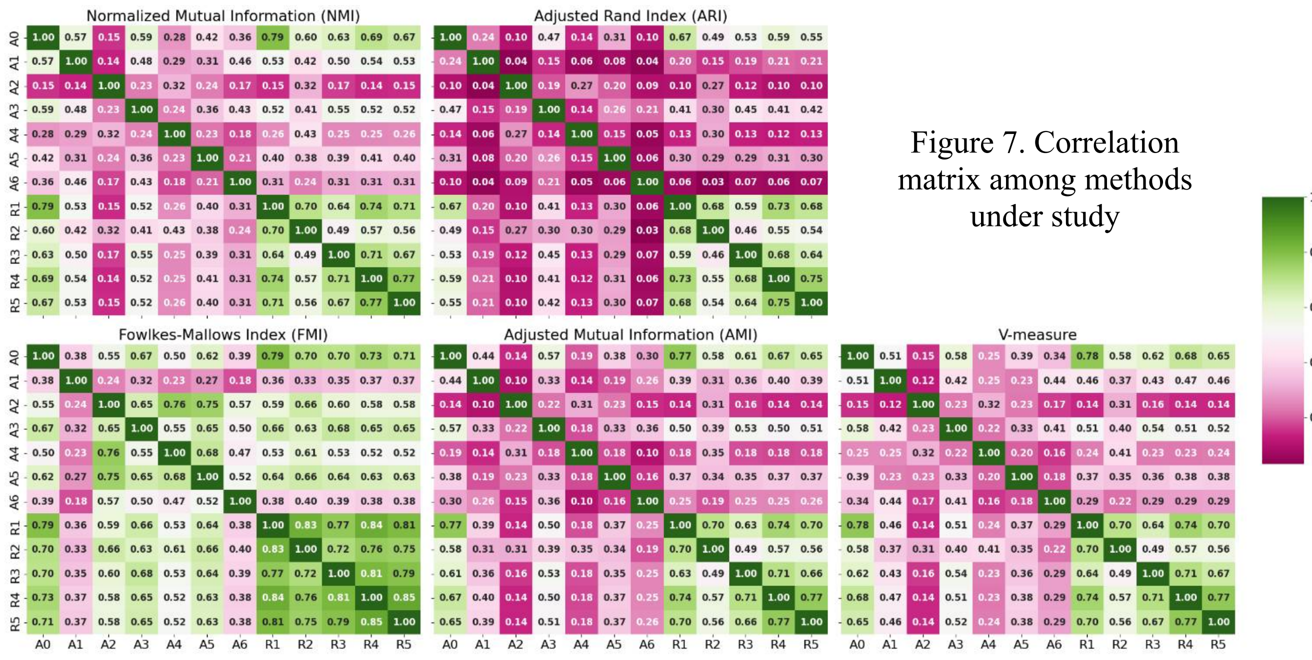


Figure 7. Correlation matrix among methods under study

As mentioned before, NMI, AMI, ARI, FMI and V-measure are metrics used in this study to determine the performance of clustering models. These metrics can also be used to compare the predicted labels of different methods and assess their alignment. By creating a correlation matrix (as shown in Figure 7) among the methods under study, we can easily visualize their similarities and differences. The arithmetic average was used as the aggregation function, as all measurements are relative.

Upon analyzing the correlation matrix, it can be observed that the Proposed method (A0) correlates the most with the Agglomerative method (R1). This finding is expected as both methods are based on distance metrics. Interestingly, all the R-methods are closely aligned, with the Proposed method (A0) being the most correlated with them among all A-methods. However, it is worth noting that despite delivering similar high performance, the Proposed method (A0), HDBSCAN (A3), and Gaussian Mixture Model (R3) are not closely correlated, indicating that their performance may vary based on the dataset structure and potential use case.

Another advantage of a built-in understanding of the number of clusters is the ability to avoid significant mistakes while detecting a large number of clusters. In order to illustrate the problem, we will use the benchmark dataset "a-set3" [49], which contains 50 clusters (Figure 8).

Figure 8 shows that at least 1 cluster was completely missed for all random states. This is easy to understand, as the K-Means centroids are initially allocated randomly, which causes local minimums. As a result, several clusters are incorrectly split among others, while some data points are organized in a cluster that is not a true one.

The proposed method may not get the correct number of clusters, but the clusters generated will not have the disadvantage of being incorrectly allocated. Moreover, one can set the number of clusters and get their allocations due to the tree structure.

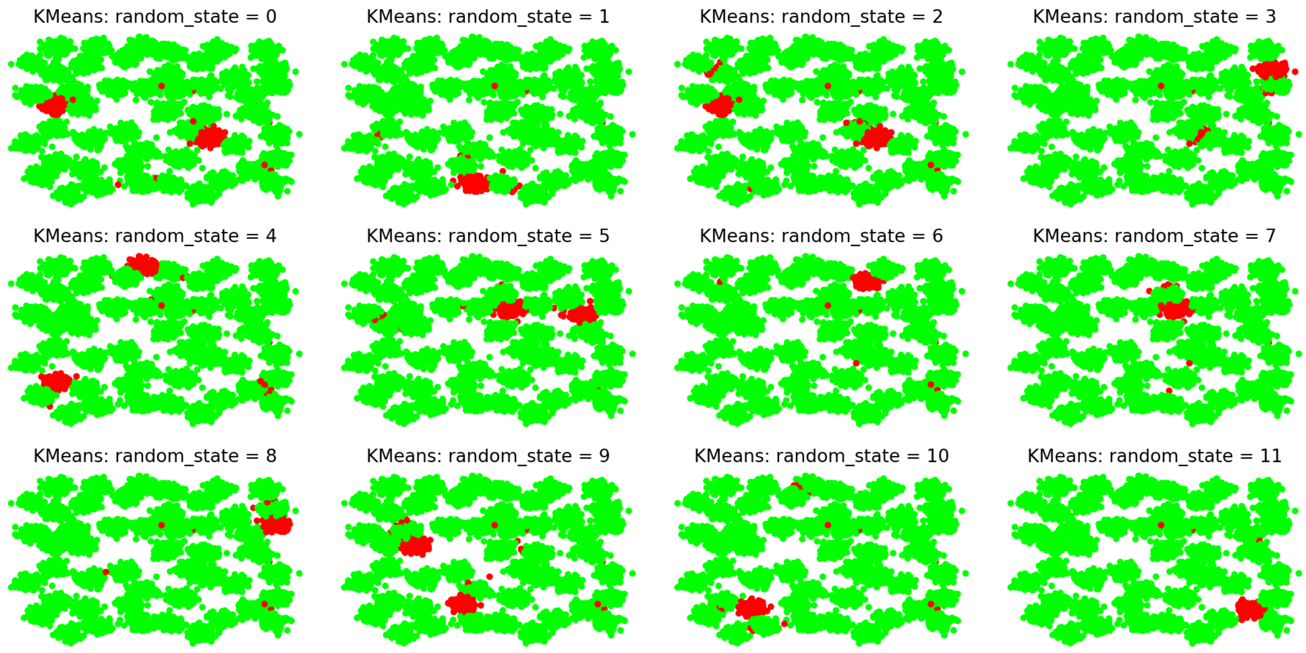


Figure 8. Correctly (green) and incorrectly (red) assigned data points by K-Means algorithm by different randoms states for the a-set3 dataset with 50 clusters. (replace 11 with BK)

As mentioned above, all the listed methods were used with their default values. Working on a specific clustering dataset, the default values can be overwritten to increase the model score. As the proposed method is a distance method, another valuable improvement might be scaling the clustering dataset by standardizing or normalizing the data.

Outlier detection

Outlier detection is an important aspect of clustering, a widely used machine learning technique for grouping similar data points together. Clustering algorithms aim to identify patterns or structures within datasets by grouping data points that are similar to each other based on specific similarity measures. However, outliers, which are data points that deviate significantly from the typical patterns in the data, can significantly impact the clustering results [50].

Detecting outliers in clustering involves identifying data points that do not conform to the clusters or exhibit unusual behavior. These outliers can distort the clustering results by affecting the clusters' centroids, density, or boundaries. Outliers can also introduce noise and reduce the accuracy of the clustering algorithms. Therefore, detecting and handling outliers is crucial for obtaining reliable and meaningful clustering results [51].

There are various approaches for outlier detection in clustering. One approach is to use distance-based methods, where the distance of a data point to its nearest neighbors or centroid is calculated, and data points with distances above a certain threshold are considered outliers [52, 53]. Another approach is to use statistical methods, such as the z-score or the interquartile range, to identify data points that deviate significantly from the mean or median of the data [54, 55]. Machine learning techniques, such as one-class support vector machines or isolation forests, can also be used for outlier detection in clustering [56]. These techniques can help identify and handle outliers effectively, improving the accuracy and reliability of clustering results [57].

As stated above, there were previous works on implementing outlier detection in hierarchical clustering [58]. The proposed method in this paper inherits the benefits of distance-based dendrograms, one can expect the outlier detection to be built within the method. For demonstration purposes, three outlier data points were added to the initial ClusterYellow, shown previously in Figure 1a, at specific coordinates (Figure 9).

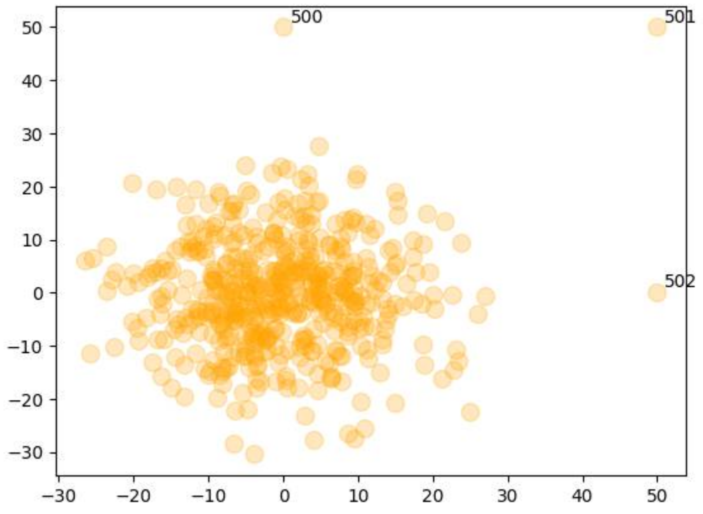


Figure 9. Modified ClusterYellow with added outliers (500, 501, 502)

A dendrogram starts with each data point represented as an individual leaf node at the bottom of the diagram. The data points are then grouped or clustered based on their similarity or distance measures, and the clusters are merged or linked together as the dendrogram ascends. If an outlier is present in the dataset, it would be represented as a single solid cluster that does not merge or link with any other clusters in the dendrogram on low levels. The outlier would have its distinct branch or sub-tree, separate from the main tree structure, indicating that it is significantly dissimilar from the other data points and does not belong to any of the clusters formed by the hierarchical clustering algorithm.

In this way, an outlier in a dendrogram would be visually depicted as a distinct and separate cluster, appearing as a single solid branch or sub-tree that does not merge with any other clusters in the hierarchical structure, highlighting its unique and dissimilar nature from the rest of the data points, as it is shown on Figure 10.

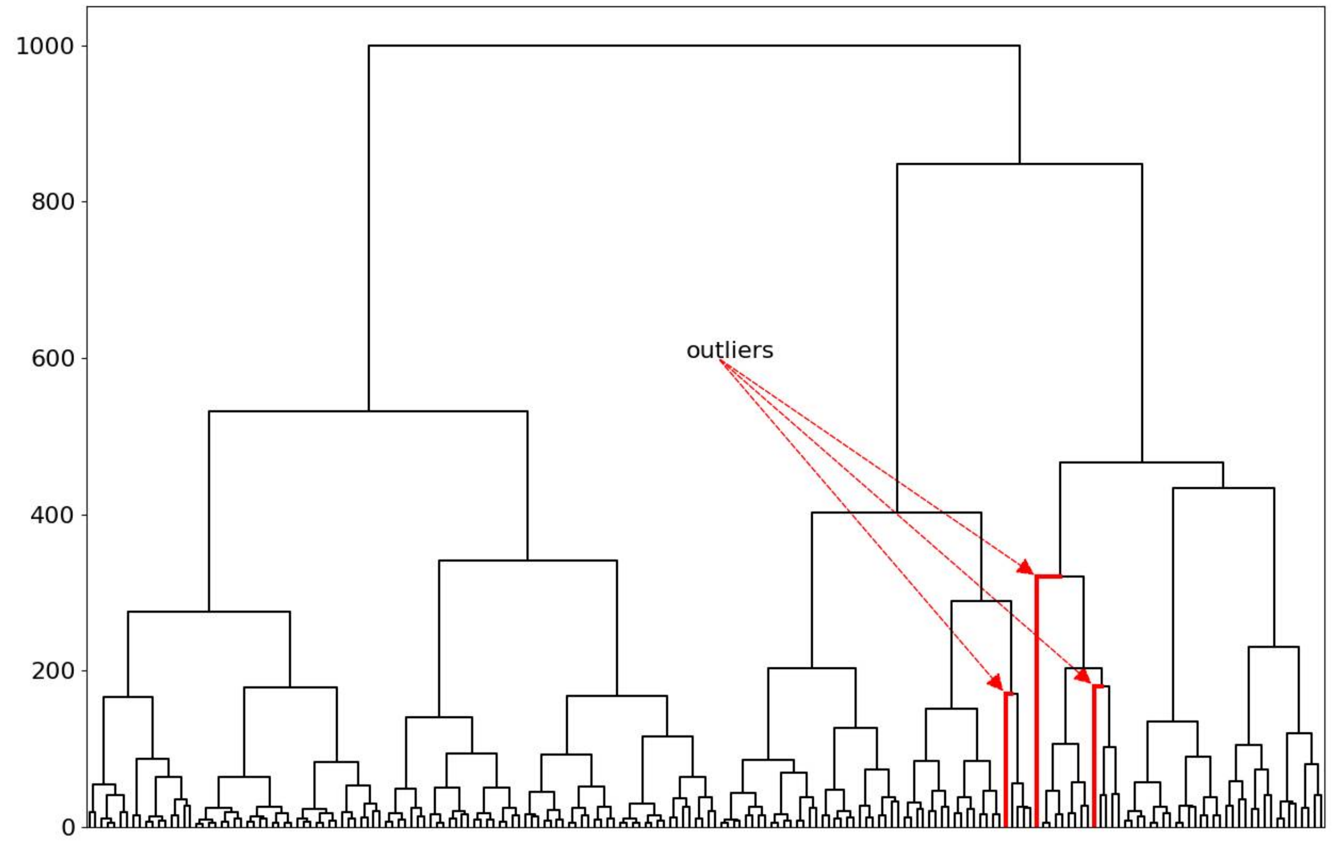


Figure 10. Dendrogram for ClusterYellow + outliers, highlighted as solid clusters with single datapoint.

As we can calculate the solidity of all clusters, based on the formula above, potential outliers will have much higher solidity than other single-leaf nodes due to a much higher h_i' value. In Table 5, sorted single nodes are given for the outlier-modified ClusterYellow.

One can easily spot the margin between the artificially added outliers (500, 501, 502) and ClusterYellow, most distant from the center data points. There needs to be more than the method itself to provide the full outlier detection procedure, but it significantly simplifies their detection process.

Table 5. Cluster height, number of datapoints, linkage height and solidity, for all leaves in the outlier-modified ClusterYellow.

Cluster	Number of datapoints (N)	Cluster height (h)	Linkage height (h')	Solidity
501	1	1	319.24	193.58
500	1	1	179.61	108.91
502	1	1	169.78	102.95
367	1	1	54.42	33
471	1	1	32.62	19.78

The logic is similar to the one described above for small aggregations of several data points. For demonstration purposes, two additional data points (503, 504) were added to a previously added outlier data point (502). The obtained cluster of 3 outlier data points was found to be the most solid among all clusters, consisting of more than one datapoint. The results are shown below in Figure 11.

One could apply specific rules using relative or absolute thresholds to identify outliers. By converting the N-dimensional data into a one-dimensional array, various other outlier detection techniques become available for use alongside these rules.

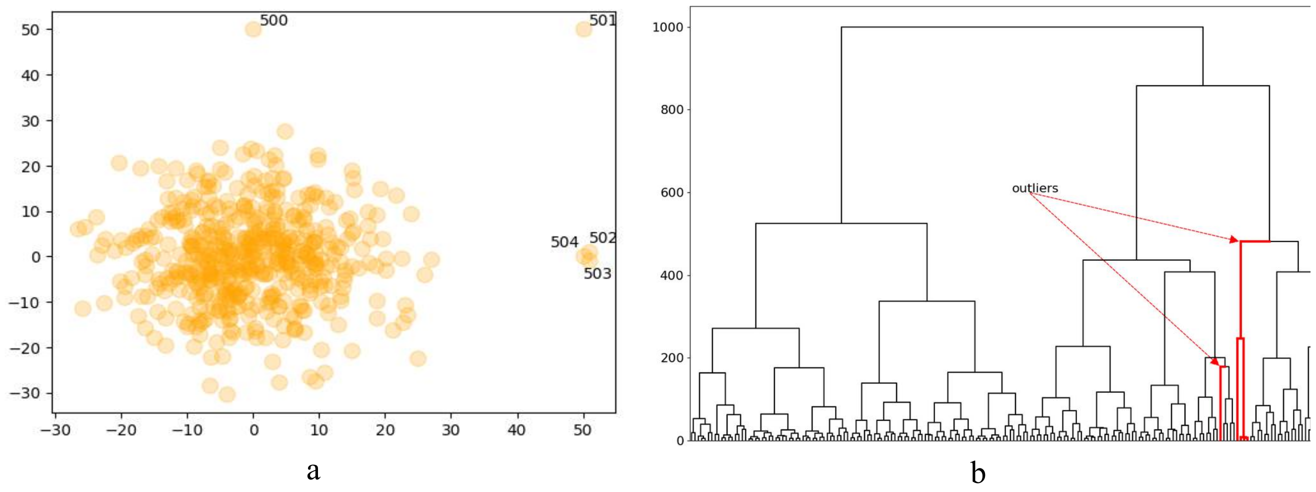


Figure 11. ClusterYellow modified with 5 outlier datapoints (a) and it's dendrogram representation (b)

Using solidity instead of tree cut

Cutting the hierarchical tree at a certain height is a common approach to determining clusters in hierarchical clustering. The height at which the tree is cut corresponds to the desired level of granularity for the clustering result. When the tree is cut at a certain height, the data points are partitioned into non-overlapping clusters. The clusters are formed by grouping data points that belong to the same branch or subtree of the hierarchical tree [29].

Despite all the advantages, cutting a dendrogram tree at a certain height can result in suboptimal cluster boundaries, as the clusters are formed based on the branch or subtree structure of the tree rather than optimizing the within-cluster similarity or dissimilarity.

The difference between using tree cut and solidities can be illustrated using the benchmark dataset "aggregation" [59] (Figure 12, 13).

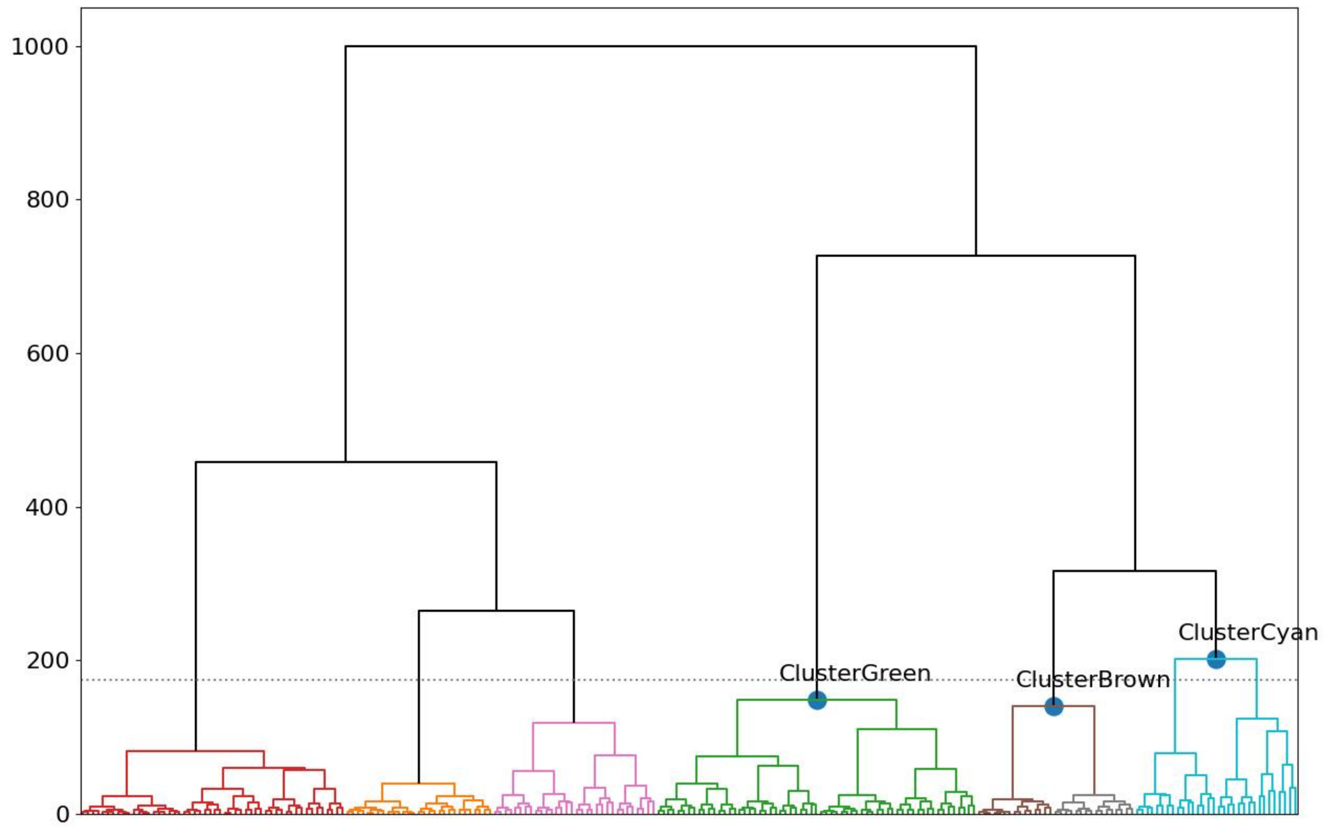


Figure 12. Dendrogram of the "aggregation" dataset using Ward linkage. 6 clusters are highlighted with different colors. True number of clusters for the "aggregation" dataset is 7.

The pure Agglomerative method (Python implementation) requires knowing the number of clusters. For the dataset under discussion, the number of true clusters is 7. According to the above logic, in this case, ClusterCyan will be split, as it has the highest cluster height (Table 6). This leads to misclustering, displayed in Figure 13(d).

As we know the ground truth, ClusterBrown (Figure 11) is the cluster to be divided. This can be visually seen in Table 6 and Figure 13(b). Alternatively, one can use calculated solidities instead of cutting lines, as shown in Table 6.

Table 6. Height, self-solidity and children solidity for ClusterCyan, ClusterGreen and ClusterBrown (Figure 12).

Cluster Name	**Cluster height (h)**	**Solidity**	**Children solidity**	
ClusterCyan	**201.90**	0.768	1.811	0.892
ClusterGreen	148.47	**4.458**	1.120	0.268
ClusterBrown	139.86	0.614	**11.776**	**6.670**

As one can notice, the next cluster in line to be cut would have also not been ClusterBrown (Figure 12). Instead, the dendrogram suggests cutting ClusterGreen, which is illogical from a solidity point of view, as the parent cluster is much more solid than the children. However, based on the solidity of ClusterBrown and its' children (Table 6), it becomes evident that ClusterBrown is the cluster to be separated.

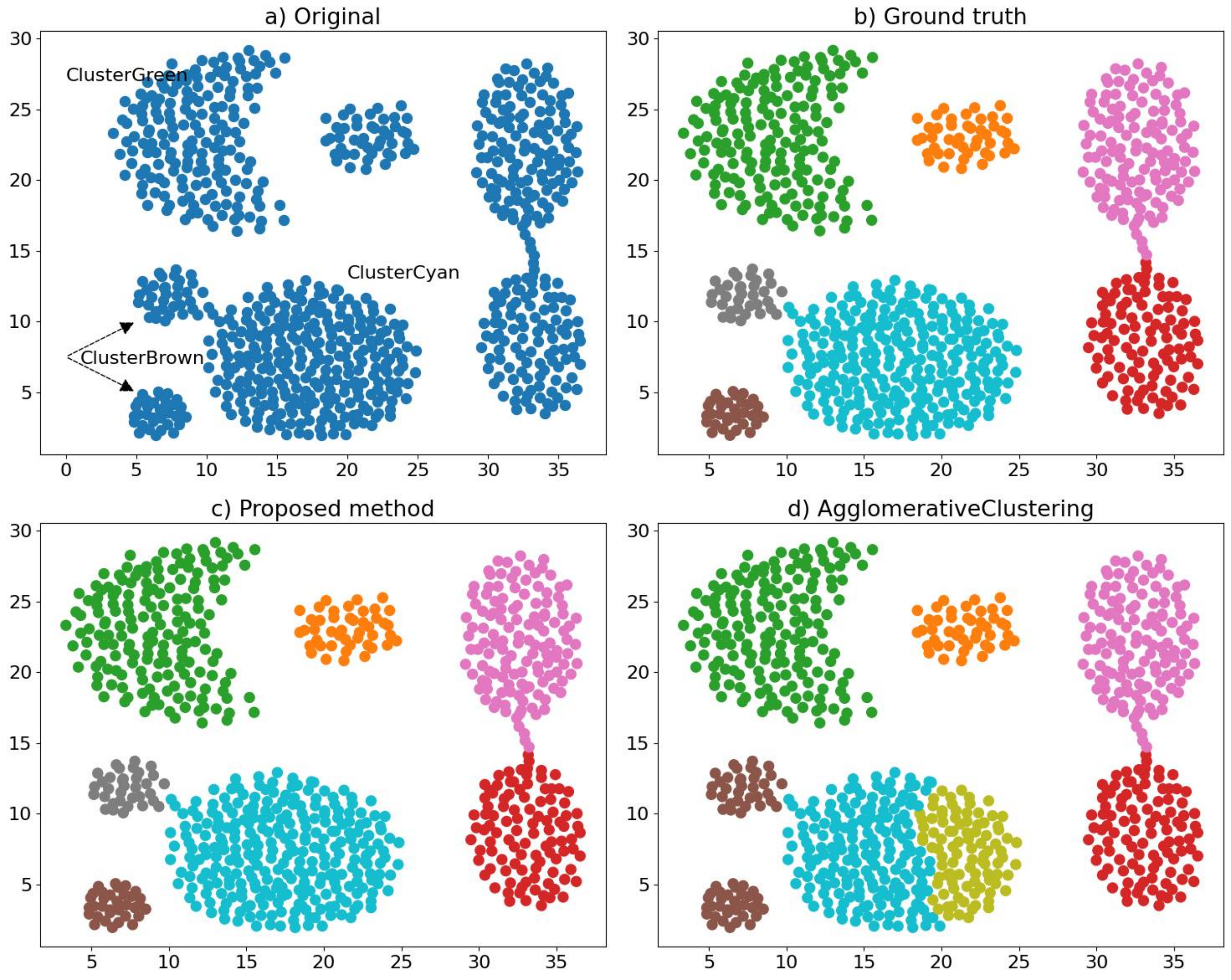


Figure 13. Visualization of the "aggregation" dataset (a), ground truth (b), proposed method labels (c) and AgglomerativeClustering labels (d)

Using separation based on solidity allows for preserving tree information and avoiding whole clusters being misclassified. This is especially seen for imbalanced datasets.

Multi-label clustering

Multi-label clustering is a machine learning or data mining technique that involves grouping or clustering data points into multiple categories or labels. Each data point can belong to one or more labels simultaneously. One the types of multi-label clustering is hierarchical multi-label clustering [60].

Hierarchical multi-label clustering can be used in tasks where data points have multiple attributes or labels organized in a hierarchical structure. The goal is to group similar data points based on both the hierarchical structure and the multi-label attributes. Hierarchical multi-label clustering consists of two main steps:

1. building the hierarchical clustering tree
2. assigning labels or categories to the clusters

Step 2 in hierarchical multi-label clustering has been a problem for a long time, mainly due to hierarchical structure complexity [61]. The hierarchical structure can be complex and multi-faceted, with multiple levels, branches, and overlapping clusters. Determining the appropriate level of granularity for label assignment can be challenging, as it may vary depending on the specific task or domain. Introducing the "solidity" attribute for clusters allows us to define the data's abstraction level.

Some of the implementation examples may leverage hierarchical structure of the data:

- Image or description classification for different animal species, where labels could include mammals, birds, reptiles, and others with some hierarchical relationship between them.
- Product categorization in e-commerce, where labels could include electronics, home appliances, and fashion, with some categories being sub-categories of others.
- Traffic sign recognition in autonomous driving, where labels could include regulatory signs, warning signs, and informational signs, may have some dependency in terms of their meanings and functions.
- Movie, news, articles genre or topic classification, where labels could include action, comedy, romance, drama or others, with some genres being sub-genres.
- Detailed sentiment analysis in customer reviews, where labels could include positive, negative, neutral sentiments, and sub-labels for specific emotions such as joy, anger, and sadness.
- Weather prediction, where labels could include different weather conditions such as sunny, rainy, and snowy. Some labels are sub-labels or have temporal dependencies.

Multiple machine learning approaches transform data into vector space representations. This includes creating different sorts of embeddings: word embeddings for natural language processing, image embeddings for computer vision, graph embeddings for working with graph-structured data, and others. All vector space approaches, particularly for classification tasks, can leverage the proposed clustering methods, especially as they already have the distance matrices calculated.

Usage of the proposed method for data generalization, semi-automatic labeling, filling missing values, and other spheres seems promising but is outside of the discussion of this paper.

Conclusions

In this study, we introduced the Burj Khalifa method, a novel clustering technique that offers simplicity, high performance, and versatility. Our approach leverages the concept of cluster solidity and transforms dendrogram structures into tree structures with multiple children and varying lengths. Demonstrated through extensive experimentation on 201 datasets, the Burj Khalifa method consistently outperformes other clustering techniques regarding median values for popular evaluation metrics. Great practical importance, the method proposed doesn’t not require extensive parameter tuning, especially knowing the number of clusters in advance.

Additionally, the method includes built-in outlier detection and multi-label hierarchical clustering, making it a promising solution for diverse real-world applications. The Burj Khalifa method has significant potential for advancing clustering research and finding practical applications in various data analysis and machine learning domains as a plug-n-play method.

Acknowledgment

The Burj Khalifa, the world's tallest skyscraper and a landmark of Dubai, United Arab Emirates, is distinguished by its recognizable ladder-view construction style. The Burj Khalifa's ladder-view design consists of a succession of terraces or setbacks that, when viewed from the outside, resemble the branches of a tree or a dendrogram and gradually get smaller as the structure rises.

The inspiration for the method was taken from the view of the building: instead of having the width changed gradually, the terraces are located at specific, uneven from different sides and levels, skipping some of them. Such construction is closely associated with the proposed method, where a dendrogram is modified to a tree structure on solid levels.

References

[1] "Hartigan, John A., and Manchek A. Wong. "Algorithm AS 136: A k-means clustering algorithm." Journal of the royal statistical society. series c (applied statistics) 28.1 (1979): 100-108.".

[2] "Park, Hae-Sang, and Chi-Hyuck Jun. "A simple and fast algorithm for K-medoids clustering." Expert systems with applications 36.2 (2009): 3336-3341.".

[3] "Day, William HE, and Herbert Edelsbrunner. "Efficient algorithms for agglomerative hierarchical clustering methods." Journal of classification 1.1 (1984): 7-24.".

[4] "Ester, Martin, et al. "A density-based algorithm for discovering clusters in large spatial databases with noise." kdd. Vol. 96. No. 34. 1996.".

[5] "Ankerst, Mihael, et al. "OPTICS: Ordering points to identify the clustering structure." ACM Sigmod record 28.2 (1999): 49-60.".

[6] "Campello, R. J., Moulavi, D., & Sander, J. (2013). Density-based clustering based on hierarchical density estimates. In Advances in Knowledge Discovery and Data Mining: 17th Pacific-Asia Conference, PAKDD 2013, Gold Coast, Australia, April 14-17, 2013, Pr".

[7] "A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. Journal of the Royal Statistical Society B, 39:1-38, 1977.".

[8] "A. Ng, M. Jordan and Y. Weiss, On Spectral Clustering: Analysis and an Algorithm. In Proceedings of Neural Information Processing Systems (NIPS 2001), 2001.".

[9] "Zhang, Tian, Raghu Ramakrishnan, and Miron Livny. "BIRCH: an efficient data clustering method for very large databases." ACM sigmod record 25.2 (1996): 103-114.".

[10] "Frey, Brendan J., and Delbert Dueck. "Mixture modeling by affinity propagation." Advances in neural information processing systems 18 (2005).".

[11] "Annie, Loraine Charlet MC, and Ashok D. Kumar. "Market basket analysis for a supermarket based on frequent itemset mining." International Journal of Computer Science Issues (IJCSI) 9.5 (2012): 257.".

[12] "Paruchuri, Harish. "Market segmentation, targeting, and positioning using machine learning." Asian Journal of Applied Science and Engineering 8 (2019): 7-14.".

[13] "Andaleeb, Syed Saad. "Market segmentation, targeting, and positioning." Strategic Marketing

Management in Asia. Emerald Group Publishing Limited, 2016. 179-207.".

[14] "Wu, Jing, and Zheng Lin. "Research on customer segmentation model by clustering." Proceedings of the 7th international conference on Electronic commerce. 2005.".

[15] "Kansal, Tushar, et al. "Customer segmentation using K-means clustering." 2018 international conference on computational techniques, electronics and mechanical systems (CTEMS). IEEE, 2018.".

[16] "Huang, Ying, et al. "CD-HIT Suite: a web server for clustering and comparing biological sequences." Bioinformatics 26.5 (2010): 680-682.".

[17] "Nugent, Rebecca, and Marina Meila. "An overview of clustering applied to molecular biology." Statistical methods in molecular biology (2010): 369-404.".

[18] "Nanda, S. R., Biswajit Mahanty, and M. K. Tiwari. "Clustering Indian stock market data for portfolio management." Expert Systems with Applications 37.12 (2010): 8793-8798.".

[19] "Cai, Fan, Nhien-An Le-Khac, and Tahar Kechadi. "Clustering approaches for financial data analysis: a survey." arXiv preprint arXiv:1609.08520 (2016).".

[20] "Dimmock, Stephen G., and William C. Gerken. "Predicting fraud by investment managers." Journal of Financial Economics 105.1 (2012): 153-173.".

[21] "Saxena, Amit, et al. "A review of clustering techniques and developments." Neurocomputing 267 (2017): 664-681.".

[22] "Yuan, Guan, et al. "A review of moving object trajectory clustering algorithms." Artificial Intelligence Review 47 (2017): 123-144.".

[23] "Singh, Vijai, and Ak K. Misra. "Detection of plant leaf diseases using image segmentation and soft computing techniques." Information processing in Agriculture 4.1 (2017): 41-49.".

[24] "Shen, Jianbing, et al. "Real-time superpixel segmentation by DBSCAN clustering algorithm." IEEE transactions on image processing 25.12 (2016): 5933-5942.".

[25] "Wang, Peng, et al. "Link prediction in social networks: the state-of-the-art." arXiv preprint arXiv:1411.5118 (2014).".

[26] "Papadopoulos, Symeon, et al. "Community detection in social media: Performance and application considerations." Data mining and knowledge discovery 24 (2012): 515-554.".

[27] "Ahmed, Wasim, et al. "COVID-19 and the 5G conspiracy theory: social network analysis of Twitter data." Journal of medical internet research 22.5 (2020): e19458.".

[28] "Xu, Dongkuan, and Yingjie Tian. "A comprehensive survey of clustering algorithms." Annals of Data Science 2 (2015): 165-193.".

[29] "Langfelder, Peter, Bin Zhang, and Steve Horvath. "Defining clusters from a hierarchical cluster tree: the Dynamic Tree Cut package for R." Bioinformatics 24.5 (2008): 719-720.".

[30] "Sebastiani, P., & Perls, T. T. (2016). Detection of significant groups in hierarchical clustering by resampling. Frontiers in genetics, 7, 144.".

[31] "Almeida, J. A. S., Barbosa, L. M. S., Pais, A. A. C. C., & Formosinho, S. J. (2007). Improving hierarchical cluster analysis: A new method with outlier detection and automatic clustering. Chemometrics and Intelligent Laboratory Systems, 87(2), 208-217.".

[32] "R. Abe, S. Miyamoto, Y. Endo and Y. Hamasuna, "Hierarchical clustering algorithms with automatic estimation of the number of clusters," 2017 Joint 17th World Congress of International Fuzzy Systems Association and 9th International Conference on Soft Comp".

[33] "Treeratpituk, P., & Callan, J. (2006, May). Automatically labeling hierarchical clusters. In Proceedings of the 2006 international conference on Digital government research (pp. 167-176).".

[34] "Karna A., Gibert K. Automatic identification of the number of clusters in hierarchical clustering //Neural Computing and Applications. – 2022. – T. 34. – №. 1. – C. 119-134.".

[35] "Yu, M., Hillebrand, A., Tewarie, P., Meier, J., van Dijk, B., Van Mieghem, P., & Stam, C. J. (2015). Hierarchical clustering in minimum spanning trees. Chaos: An Interdisciplinary Journal of Nonlinear Science, 25(2), 023107.".

[36] "Sander, J., Qin, X., Lu, Z., Niu, N., & Kovarsky, A. (2003). Automatic extraction of clusters from hierarchical clustering representations. In Advances in Knowledge Discovery and Data Mining: 7th Pacific-Asia Conference, PAKDD 2003, Seoul, Korea, April 30".

[37] "Y. Fukuyama and M. Sugeno, A new method for choosing the number of clusters for the fuzzy c-means method, Proc. 5th Fuzzy System Symposium, pp. 247-250, July 1989".

[38] "Saraçli, Sinan, Nurhan Doğan, and İsmet Doğan. "Comparison of hierarchical cluster analysis methods by cophenetic correlation." Journal of inequalities and Applications 2013.1 (2013): 1-8.".

[39] "Carvalho, Priscilla Ramos, Casimiro Sepúlveda Munita, and André Luiz Lapolli. "Validity studies among hierarchical methods of cluster analysis using cophenetic correlation coefficient." Brazilian Journal of Radiation Sciences 7.2A (2019).".

[40] "Rodriguez, Alex, and Alessandro Laio. "Clustering by fast search and find of density peaks." science 344.6191 (2014): 1492-1496.".

[41] "Morris, Tim P., Ian R. White, and Michael J. Crowther. "Using simulation studies to evaluate statistical methods." Statistics in medicine 38.11 (2019): 2074-2102.".

[42] "Yeung, Ka Yee, and Walter L. Ruzzo. "Details of the adjusted rand index and clustering algorithms, supplement to the paper an empirical study on principal component analysis for clustering gene expression data." Bioinformatics 17.9 (2001): 763-774.".

[43] "Santos, Jorge M., and Mark Embrechts. "On the use of the adjusted rand index as a metric for evaluating supervised classification." Artificial Neural Networks–ICANN 2009: 19th International Conference, Limassol, Cyprus, September 14-17, 2009, Proceedings,".

[44] "Romano, Simone, et al. "Standardized mutual information for clustering comparisons: one step further in adjustment for chance." International conference on machine learning. PMLR, 2014.".

[45] "Kraskov, Alexander, et al. "Hierarchical clustering using mutual information." Europhysics Letters 70.2 (2005): 278.".

[46] "Romano, Simone, et al. "Adjusting for chance clustering comparison measures." The Journal of Machine Learning Research 17.1 (2016): 4635-4666.".

[47] "Rosenberg, Andrew, and Julia Hirschberg. "V-measure: A conditional entropy-based external cluster evaluation measure." Proceedings of the 2007 joint conference on empirical methods in natural language processing and computational natural language learning".

[48] "Fowlkes, Edward B., and Colin L. Mallows. "A method for comparing two hierarchical clusterings." Journal of the American statistical association 78.383 (1983): 553-569.".

[49] "I. Kärkkäinen and P. Fränti, "Dynamic local search algorithm for the clustering problem", Research Report A-2002-6".

[50] "Wang, Hongzhi, Mohamed Jaward Bah, and Mohamed Hammad. "Progress in outlier detection techniques: A survey." Ieee Access 7 (2019): 107964-108000.".

[51] "Rousseeuw, Peter J., and Annick M. Leroy. Robust regression and outlier detection. John wiley & sons, 2005.".

[52] "Knorr, Edwin M., Raymond T. Ng, and Vladimir Tucakov. "Distance-based outliers: algorithms and applications." The VLDB Journal 8.3 (2000): 237-253.".

[53] "Angiulli, Fabrizio, Stefano Basta, and Clara Pizzuti. "Distance-based detection and prediction of outliers." IEEE transactions on knowledge and data engineering 18.2 (2005): 145-160.".

[54] "Rousseeuw, Peter J., and Mia Hubert. "Robust statistics for outlier detection." Wiley interdisciplinary reviews: Data mining and knowledge discovery 1.1 (2011): 73-79.".

[55] "Ben-Gal, Irad. "Outlier detection." Data mining and knowledge discovery handbook (2005): 131-146.".

[56] "Singh, Karanjit, and Shuchita Upadhyaya. "Outlier detection: applications and techniques." International Journal of Computer Science Issues (IJCSI) 9.1 (2012): 307.".

[57] "Zhao, Yue, Zain Nasrullah, and Zheng Li. "Pyod: A python toolbox for scalable outlier detection." arXiv preprint arXiv:1901.01588 (2019).".

[58] "Campello, R. J., Moulavi, D., Zimek, A., & Sander, J. (2015). Hierarchical density estimates for data clustering, visualization, and outlier detection. ACM Transactions on Knowledge Discovery from Data (TKDD), 10(1), 1-51.".

[59] "Gionis A., Mannila H., Tsaparas P. Clustering aggregation //Acm transactions on knowledge discovery from data (tkdd). – 2007. – T. 1. – №. 1. – C. 4-es.".

[60] "Zhang L., Shah S. K., Kakadiaris I. A. Hierarchical multi-label classification using fully associative ensemble learning //Pattern Recognition. – 2017. – T. 70. – C. 89-103.".

[61] "Levatić J., Kocev D., Džeroski S. The importance of the label hierarchy in hierarchical multi-label classification //Journal of Intelligent Information Systems. – 2015. – T. 45. – C. 247-271.".

[62] "McInnes, Leland, John Healy, and Steve Astels. "HDBSCAN: Hierarchical density based clustering." J. Open Source Softw. 2.11 (2017): 205.".

[63] "Krieger, Abba M., and Paul E. Green. "A generalized Rand-index method for consensus clustering of separate partitions of the same data base." Journal of classification 16.1 (1999): 63-89.".

[64] "Hodge, Victoria, and Jim Austin. "A survey of outlier detection methodologies." Artificial intelligence review 22 (2004): 85-126.".

Appendix

All the illustration, tables, datasets and code can be found at the GitHub repository: https://github.com/IvanReznikov/bk_clustering

The python package of the implementation of the proposed method can be found: https://pypi.org/project/bk-clustering/