Using Frequent Closed Pattern Mining to Solve a Consensus Clustering Problem

Atheer Al-najdi Nicolas Pasquier Frédéric Precioso
Univ. Nice Sophia Antipolis, CNRS, I3S, UMR 7271, 06900 Sophia Antipolis, France
E-mail: {alnajdi, pasquier, precioso}@i3s.unice.fr

Abstract

Clustering is the process of partitioning a dataset into groups based on the similarity between the instances. Many clustering algorithms were proposed, but none of them proved to provide good quality partition in all situations. Consensus clustering aims to enhance the clustering process by combining different partitions obtained from different algorithms to yield a better quality consensus solution. In this work, we propose a new consensus method that uses a pattern mining technique in order to reduce the search space from instance-based into pattern-based space. Instead of finding one solution, our method generates multiple consensus candidates based on varying the number of base clusterings considered. The different solutions are then linked and presented as a tree that gives more insight about the similarities between the instances and the different partitions in the ensemble.

keywords Unsupervised learning; Clustering; Consensus clustering; Ensemble clustering; Frequent closed itemsets.

1. Introduction

Clustering is one of the important tasks in data mining. It can discover patterns in a dataset by classifying the instances into groups based on their similarities. However, different algorithms were proposed in the last decades to enhance this unsupervised learning process. Unfortunately, none of these algorithms proved to provide "good" clustering solution in all situations. Therefore, a new branch of research emerged in the last years that focuses on combining different clusterings into one consensus solution, known as *Consensus Clustering* or *Clusterings Aggregation*. The idea behind it is that we can benefit from the advantages of each algorithm used in building the initial clusterings ensemble, to produce a new solution that is more stable and achieves better quality final result.

Different approaches were used to generate the consensus solution. However, until very recently, all these methods search for the optimal consensus in the whole search space, making some of them inapplicable for large datasets. Thus, new research focused on finding the consensus partition in a pruned space. Wu *et. al.* [22] worked on a reduced space of "data fragments" instead of the instances space. Vega-Pons and Avesani [20] defined two functions to prune the search space, namely the unanimity and the majority prune functions. In this work, we use the *Frequent Closed Itemsets* (**FCI**) [14] technique from the domain of pattern mining and association rules discovery, in order to find a different pruning of the search space. FCI, a technique designed to discover patterns in very large datasets, can be used to transform the search space from instance-based into pattern-based space. Each pattern defines agreement between a set of base clusters on grouping a set of instances. Therefore, we can even partition this patterns space into subspaces based on the number of base clusterings that define the patterns. On each subspace, we find a consensus solution by clustering the patterns based on their similarity. Thus, our approach involves generating multiple consensuses, then recommend the one the most similar to the ensemble. All these solutions are linked and presented in a tree of consensus clusters that enables the analyst to discover which clusters are more stable than others, pointing out strong intra-cluster similarity between the instances.

This paper is organized as follows: The next section provides a brief summary of the main categories of consensus clustering methods. The proposed approach is explained in Sect. 3. Description of the performed tests and the achieved results is presented in Sect. 4. We conclude in Sect. 5.

2. Related Work

Different methods were proposed to find a consensus partition from an ensemble of clusterings. They can be categorized based on the underlying approach used:

- **Graph partitioning methods:** By representing the relation between the instances and the base clusters they belong to as a graph, a consensus solution is obtained using a graph partitioning algorithm. For example: Cluster-based Similarity Partitioning Algorithm (CSPA), HyperGraph Partitioning Algorithm

DOI reference number: 10.18293/SEKE2016-117

(HGPA), Meta CLustering Algorithm (MCLA) (Strehl and Ghosh [17]), and Hybrid Bipartite Graph Formulation (HBGF) (Fern and Brodley [5]).

- **Voting methods:** These methods try first to find a unified labeling among the base clusterings, then apply a voting approach to generate the consensus solution. In the Plurality Voting method (Dudoit and Fridlyand[4], Fischer and Buhmann[6]), the relabeling problem is solved using the Hungarian algorithm, then the final label of an instance is the one the mostly assigned to it in the relabeled ensemble.
- **Co-association based methods:** A co-association matrix can be used to define the similarity between the instances in terms of how many times each pair belongs to the same cluster in the ensemble. Then, a clustering algorithm can be applied to find the consensus partition. For example, Fred and Jain [7] used the single-linkage hierarchical clustering algorithm on the matrix to generate a consensus solution.
- **Finite mixture methods:** The labels of the base clusterings are considered as random variables drawn from a probability distribution. The consensus partition here is the solution of a maximum likelihood estimation problem, using for example the EM algorithm as done by Topchy *et. al.* [18].

More details in the surveys by Ghaemi *et al.* [8], Sarumathi *et al.* [16], and Vega-Pons & Ruiz-Shulcloper [21].

3. The Proposed Pattern-Based Approach

To discover the relationships between the instances and the base clusters in the ensemble, we use the frequent closed itemset technique from pattern mining discipline. FCI finds first the sets of instances that are clustered together by all base clusterings, similar to the data fragments in [22] or the unanimity function in [20]. Instead of using the algorithms proposed in [22], or the majority function in [20] that provides further pruning to the search space, FCI finds also the sets of instances that are clustered together by different combinations of base clusterings. Therefor, FCI does not just transform the search space from instance-based into patterns-based, but it also enables us to divide this pruned space into subspaces based on the number of base clusterings used to define the patterns, and search for a consensus solution in each subspace. The proposed approach is explained in detail in the following subsections.

3.1. Clusterings Ensemble

The first step in any consensus clustering method is to build an ensemble of different partitions for the dataset. However, some consensus methods may impose restrictions on the ensemble generation process. For example, voting methods require that all the partitions in the ensemble have the same number of clusters [8, 21]. For our approach, there are no limitations, as long as the base clusterings define hard partitions, that is, an instance belongs to only one cluster in each base clustering.

3.2. Cluster Membership Matrix

As in [1], we use a cluster membership matrix $\mathcal{M}$ to record the relationships between the instances and the base clusters as a binary relation. $\mathcal{M}$ consists of n rows and m columns, where n is the number of instances, and m is the total number of clusters of all base clusterings.

Definition 1 *A cluster membership matrix $\mathcal{M}$ is a triplet $(\mathcal{I}, \mathcal{C}, \mathcal{R})$ where $\mathcal{I}$ is a finite set of instances represented as rows, $\mathcal{C}$ is a finite set of clusters represented as columns, and $\mathcal{R}$ is a binary relation defining relationships between rows and columns: $\mathcal{R} \subseteq \mathcal{I} \times \mathcal{C}$. Every couple $(i, c) \in \mathcal{R}$, where $i \in \mathcal{I}$ and $c \in \mathcal{C}$, means that instance i belongs to cluster c. This binary relation is represented in the matrix by 1 at $\mathcal{M}_{ic}$, and 0 if there is no relationship.*

Consider as an example a dataset of eight instances $\mathcal{D} = \{1, 2, 3, 4, 5, 6, 7, 8\}$. Suppose that we partitioned it with 4 clusterings as follows: $P1$= $\{\{1, 2, 3\}$, $\{4, 5, 6, 7, 8\}\}$, $P2$= $\{\{1, 2\}$, $\{\{3, 4, 5\}$, $\{6, 7, 8\}\}$, $P3$= $\{\{5, 6, 7, 8\}$, $\{1, 2, 3, 4\}\}$, and $P4$= $\{\{1, 2, 3\}$, $\{\{4, 5\}$, $\{6, 7, 8\}\}$. The resulting cluster membership matrix is shown in table 1. Each column P^i_j defines cluster j in partition i as a binary vector where values '1' identify the instances that belong to the cluster.

Table 1: Example cluster membership matrix.

Instance ID	P^1_1	P^1_2	P^2_1	P^2_2	P^2_3	P^3_1	P^3_2	P^4_1	P^4_2	P^4_3
1	1	0	1	0	0	0	1	1	0	0
2	1	0	1	0	0	0	1	1	0	0
3	1	0	0	1	0	0	1	1	0	0
4	0	1	0	1	0	0	1	0	1	0
5	0	1	0	1	0	1	0	0	1	0
6	0	1	0	0	1	1	0	0	0	1
7	0	1	0	0	1	1	0	0	0	1
8	0	1	0	0	1	1	0	0	0	1

3.3. Frequent Closed Clustering Patterns

$\mathcal{M}$ can be viewed as a pattern mining problem, where each column (cluster) represent an item as defined below.

Definition 2 *An item of a cluster membership matrix $\mathcal{M} = (\mathcal{I}, \mathcal{C}, \mathcal{R})$ is a cluster identifier $c \in \mathcal{C}$, and an itemset is a non-empty finite set of items $C = \{c_1, ..., c_k\} \subseteq \mathcal{C}$ in $\mathcal{M}$.*

The frequency of C is defined as $\mathcal{F}(C) = |\{I \in \mathcal{I} \mid \forall i \in I, \forall c \in C$, *we have* $(i,c) \in \mathcal{R}\}|$.

Applying FCI technique on $\mathcal{M}$ produces *Frequent Closed Pattern* (**FCP**)s, that is, a set of instance identifiers that share a binary pattern of cluster memberships (itemset)[1]. Table 2 shows the FCPs extracted from table 1[2]. The closed patterns represent maximal rectangles of '1's in the membership matrix.

Definition 3 *A pattern* $\rho = (C, I)$ *in the cluster membership matrix* $\mathcal{M} = (\mathcal{I}, \mathcal{C}, \mathcal{R})$ *is a pair of sets* $C \subset \mathcal{C}$ *and* $I \subset \mathcal{I}$ *such that* $\forall i \in I$ *and* $\forall c \in C$, *we have* $(i,c) \in \mathcal{R}$. *If* $C \subset C'$, *then* ρ *is a frequent closed pattern iff* $\mathcal{F}(C) \neq \mathcal{F}(C')$.

The closed property ensures not generating redundant patterns for our approach, that is, those with identical instance sets. This reduces greatly memory consumption and execution time compared to generating all the possible frequent patterns[3].

Table 2: Frequent closed patterns extracted from table 1.

FCP ID	Itemset (FCIs)	Instance ID set
1	$\{P_2^1, P_2^2, P_2^3, P_2^4\}$	{4}
2	$\{P_2^1, P_2^2, P_1^3, P_2^4\}$	{5}
3	$\{P_1^1, P_2^2, P_2^3, P_1^4\}$	{3}
4	$\{P_2^2, P_2^3\}$	{3,4}
5	$\{P_2^1, P_2^2, P_2^4\}$	{4,5}
6	$\{P_1^1, P_1^2, P_2^3, P_1^4\}$	{1,2}
7	$\{P_2^2\}$	{3,4,5}
8	$\{P_1^1, P_2^3, P_1^4\}$	{1,2,3}
9	$\{P_2^1, P_3^2, P_1^3, P_3^4\}$	{6,7,8}
10	$\{P_2^3\}$	{1,2,3,4}
11	$\{P_2^1, P_1^3\}$	{5,6,7,8}
12	$\{P_2^1\}$	{4,5,6,7,8}

3.4. Multiple Consensuses

After generating the FCPs that summarize the relation between the instances and sets of clustering decisions, we will work on this patterns space to find consensus solutions. First, we partition this patterns space into subspaces based on the number of base clusterings that define the pattern. We use a *Decision Threshold* (**DT**) to identify each subspace, as it defines the size of the itemset in each pattern.

[1]In association rule mining, the *support* of an itemset is the percentage of instances that have the itemset. Only the itemsets with *support* > *minsupport* threshold are considered. However, in our approach, we use *minsupport* = 0 to consider all the possible closed patterns.

[2]For small datasets, we can have more patterns than the number of instances. However, in large datasets, the number of patterns will be much smaller compared to dataset size, since many instances will share the same pattern.

[3]See [2] for more details about association rules mining techniques.

The first subspace consists of the patterns having itemsets of size max DT (the number of base clusterings used in the ensemble). The instance identifier sets of the patterns in this initial subspace represent the clusters ("data fragments") of the first consensus. Next, we sequentially decrement DT toward 1, and in each subspace, a consensus solution is built from the instance identifier sets of the patterns in that subspace, plus the clusters of the previous consensus.

Definition 4 *Let* $\alpha = Max(DT)$. *The first consensus is* $\mathbb{P}^\alpha = \{\pi_1^\alpha, \pi_2^\alpha, ..., \pi_m^\alpha\}$, *where* π_k^α *is an instance set of a FCP built from* α *base clusterings. Let* $\beta < \alpha$ *and* $\mathbb{S}^\beta = \mathbb{I}^\beta \cup \mathbb{P}^{\beta+1}$ *is the pool of instance sets at* $\beta = DT$, *where* $\mathbb{I}^\beta$ *is the instance sets of the FCPs built from* β *base clusterings, and* $\mathbb{P}^{\beta+1}$ *is the instance sets (clusters) of the previous consensus. A new consensus* $\mathbb{P}^\beta$ *is the result of applying a consensus function* $\mathcal{Y}$ *on* $\mathbb{S}^\beta$, *that is,* $\mathbb{P}^\beta = \mathcal{Y}(\mathbb{S}^\beta) = \{\pi_1^\beta, \pi_2^\beta, ..., \pi_n^\beta\}$ *such that* $\pi_i^\beta \cap \pi_j^\beta = \emptyset, \forall (i,j) \in \{1,...,n\}$, $i \neq j$, *and* $\bigcup_{i=1}^{i=n} \pi_i^\beta = \mathcal{I}$.

In each subspace, an instance identifier set $I \subseteq \mathcal{I}$ has one of the three following properties:

i) Uniqueness: It does not intersect with any other set $I' \subseteq \mathcal{I}$, that is, $I \cap I' = \emptyset$.

ii) Inclusion: It is a subset of another set $I' \subseteq \mathcal{I}$, that is, $I \subseteq I'$.

iii) Intersection: It intersects with another set $I' \subseteq \mathcal{I}$, that is, $I \cap I' \neq \emptyset$, $I \setminus I' \neq \emptyset$ and $I' \setminus I \neq \emptyset$.

The objective of our consensus function is to build disjoint clusters from the instance sets, that is, all the sets have uniqueness property. For the first consensus, all the instance sets are unique. However, for the following consensuses, the sets of instance identifiers can have any of the above properties, because when we consider fewer base clusterings, instances can belong to several patterns. The instance sets with inclusion property are usually the clusters of the previous consensus, as they will become subsets of new grouping of the instances defined by fewer number of base clusters. Thus, they are removed to consider the new decisions. What remains are the sets with intersection property. To make unique clusters from them, we need to either merge or split intersecting sets based on their similarity. Jaccard index [11] is a well known measure of the similarity between two sets X and Y:

$J(X,Y) = \frac{|X \cap Y|}{|X \cup Y|} = \frac{|X \cap Y|}{|X|+|Y|-|X \cap Y|}$

For example, let us take the 3 cases of sets intersection shown in Fig. 1 and calculate the Jaccard score for each case:

$J(A,B) = \frac{3}{27}$, $J(B,C) = \frac{7}{23}$, $J(D,E) = \frac{7}{23}$

But the same score is given to cases 2 and 3, despite that in case 2, most of set B is part of set C. Thus, instead of using Jaccard, we define a new measure for deciding to merge or split sets based on the size of intersection to the size of each set:

$$Avg\_I(X|Y) = (\frac{|X \cap Y|}{|X|} + \frac{|X \cap Y|}{|Y|}) \times 0.5$$

Going back to the sets in Fig. 1:

$$Avg\_I(A|B) = (\frac{3}{20} + \frac{3}{10}) \times 0.5 = 0.225$$

$$Avg\_I(C|B) = (\frac{7}{20} + \frac{7}{10}) \times 0.5 = 0.525$$

$$Avg\_I(D|E) = (\frac{7}{16} + \frac{7}{14}) \times 0.5 = 0.469$$

Thus, our new measure gave the highest score to case 2.

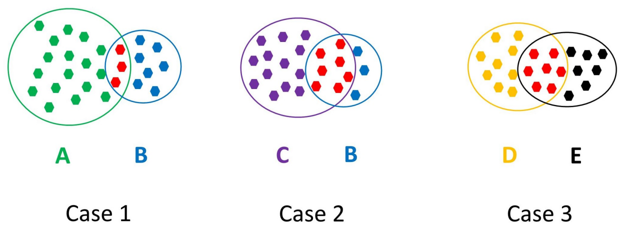

Figure 1: Examples of sets intersection

Based on the new measure, our consensus function will decide to merge/split intersecting sets using a *Merging Threshold* (**MT**). That is, if the score given to 2 intersecting sets is more than or equals MT, then the 2 sets are merged using a union operation. Else, they are split into 2 disjoint sets, by removing the shared instances from the largest set, and keeping them in the smaller set as it represent a more coherent cluster. To enhance the merge/split process, the consensus function searches, for each set, which of the other sets produces the highest score by the average intersection ratio measure, before comparing with MT. This process repeats until having all the remaining sets as unique. After generating all the possible consensuses, the one the most similar to the ensemble is recommended. The similarity is calculated using Jaccard index. Algorithms 1 and 2 present in detail the proposed multiple consensuses generation method.

Continuing the running example and the FCPs in table 2: We start with DT=4. The first consensus consists of the instance sets of FCPs 1, 2, 3, 6, and 9 as they are all unique. The next subspace is DT=3, which consists of FCPs 5 and 8, plus the clusters of the consensus at DT=4. Therefore, we have the following sets: {4}, {5}, {3}, {1,2}, {6,7,8}, {4,5}, and {1,2,3}. The first 4 sets are subsets of the last 2, thus they are removed. The consensus clusters at DT=3 are the remaining sets. At DT=2, we have: {6,7,8}, {4,5}, {1,2,3}, {3,4}, and {5,6,7,8}. The first set is removed, while the second intersects with 2 other sets, thus, we will use average intersection ratio:

$Avg\_I(\{4,5\},\{3,4\}) = 0.5.$

$Avg\_I(\{4,5\},\{5,6,7,8\}) = 0.375.$

The higher score is for $Avg\_I(\{4,5\},\{3,4\})$. If

```
Input  : Dataset to cluster, merging threshold MT
Output : ConsTree tree of consensuses, list of consensus
         clustering vectors
1  Generate clusterings ensemble of the dataset;
2  Build the cluster membership matrix $\mathcal{M}$;
3  Generate FCPs from $\mathcal{M}$ for minsupport = 0;
4  Sort the FCPs in ascending order according to the size of the
     instance sets;
5  MaxDT $\leftarrow$ Number of base clusterings;
6  BiClust $\leftarrow$ {instance sets of FCPs built from MaxDT base
   clusters};
7  Assign a label to each set in BiClust to build the first
     consensus vector and store it in a list of vectors ConsVctrs;
8  for DT = (MaxDT - 1) to 1 do
9      BiClust $\leftarrow$ BiClust $\cup$ {instance sets of FCPs built from
       DT base clusters};
10     N $\leftarrow$ $|BiClust|$ ;
11     Call the consensus function (Algo. 2);
12     Assign a label to each set in BiClust to build a
         consensus vector and add it to ConsVctrs;
13 end
14 Find stable consensuses in ConsVctrs and remove extra
     duplicates;
15 For each remaining consensus, calculate its average
     similarity to the ensemble using Jaccard index;
16 Build a tree from the consensuses in ConsVctrs, with a
     recommended solution as the one that has the highest
     average similarity to the ensemble;
```

Algorithm 1: Generate Multiple Consensuses

MT=0.4[4], then the 2 sets are merged to form {3,4,5}. Now we have $Avg\_I(\{1,2,3\},\{3,4,5\}) = 0.33 <$ MT, then the sets are split into {1,2,3} and {4,5}. The last is $Avg\_I(\{4,5\},\{5,6,7,8\}) = 0.375 <$ MT, then split into {4,5} and {6,7,8}. The final consensus at DT=2 is {1,2,3}, {4,5}, and {6,7,8} which is identical to the consensus at DT=3, thus it is removed because it becomes redundant solution. A *stability counter* (**ST**) is used to reflect that a consensus solution is generated multiple times from different DT spaces. Therefore, the consensus at DT=3 will have ST=2. The same process is performed for DT=1, resulting in grouping all the instances in 1 cluster.

3.5. ConsTree

The final step is presenting all the generated consensuses in a tree structure that explains how the instances regroup at each DT subspace. Each level in the ConsTree depicts the final consensus clusters of a specific DT subspace. The levels are sorted according to DT, where the first consensus is assigned to the bottom level of the tree. In each tree level, the node's label and size reflect the cluster size. The ConsTree of the running example is shown in Fig.2.

[4]Default value based on extensive experimental tests.

```
repeat
    for $i$ = 1 to $N$ do
        $B_i \leftarrow i^{th}$ set in *BiClust*;
        $BestIntrsc \leftarrow 0$;
        $Index \leftarrow 0$;
        for $j$ = 1 to $N$, $j \neq i$ do
            $B_j \leftarrow j^{th}$ set in *BiClust*;
            $IntrscSz \leftarrow |B_i \cap B_j|$;
            if $IntrscSz = 0$ then
                Next j ;
            else if $IntrscSz = |B_i|$ then
                /* $B_i \subset B_j$ */ ;
                Remove $B_i$ from *BiClust*;
                Next $i$;
            else if $IntrscSz = |B_j|$ then
                /* $B_j \subset B_i$ */ ;
                Remove $B_j$ from *BiClust*;
                Next $j$;
            else
                $IntrscRatio \leftarrow (\frac{IntrscSz}{|B_i|} + \frac{IntrscSz}{|B_j|}) \times 0.5$;
                if $IntrscRatio > BestIntrsc$ then
                    $BestIntrsc \leftarrow IntrscRatio$;
                    $Index \leftarrow j$;
        end
        if $BestIntrsc > 0$ then
            $j \leftarrow Index$;
            $B_j \leftarrow j^{th}$ set in *BiClust*;
            if $BestIntrsc \geq MT$ then
                /* merge */ ;
                $B_j \leftarrow B_i \cup B_j$;
                Remove $B_i$ from *BiClust*;
            else
                /* split */ ;
                if $|B_i| \leq |B_j|$ then
                    $B_j \leftarrow B_j \setminus B_i$ ;
                else
                    $B_i \leftarrow B_i \setminus B_j$ ;
    end
until *All sets in BiClust are unique*;
```

Algorithm 2: Consensus function

Definition 5 *A tree of consensuses is an ordered set ($\mathcal{P}$, $\preceq$) of consensuses $\mathcal{P} = \bigcup_{DT=MaxDT}^{DT=MinDT} \mathbb{P}^{DT}$ ordered in descending order of DT values. Let's denote $\mathbb{P}^{\alpha} = \{\pi_1^{\alpha}, ..., \pi_m^{\alpha}\}$ and $\mathbb{P}^{\beta} = \{\pi_1^{\beta}, ..., \pi_n^{\beta}\}$ the consensuses generated for α and β DT values respectively. Let's denote π_q^{α} the q^{th} cluster in $\mathbb{P}^{\alpha}$ and π_r^{β} the r^{th} cluster in $\mathbb{P}^{\beta}$, with $1 \leq q \leq m$ and $1 \leq r \leq n$. For $\alpha > \beta$ we have $\mathbb{P}^{\alpha} \preceq \mathbb{P}^{\beta}$, that is $\forall \pi_q^{\alpha} \in \mathbb{P}^{\alpha}$, $\exists \pi_r^{\beta} \in \mathbb{P}^{\beta}$ such that $\pi_q^{\alpha} \cap \pi_r^{\beta} \neq \emptyset$. $\mathbb{P}^{\alpha}$ is a predecessor of $\mathbb{P}^{\beta}$ in the tree of consensuses.*

Another example of a ConsTree is shown in Fig. 3. By reading the tree from the bottom level to the root, we can see that all the clusters are linked to only 1 cluster at the next level, except 2 clusters at DT=8 that are linked to 2 clusters at DT=7, meaning that theirs instances are regrouped differently at DT=7. This shifting of the instances may happen for many of the clusters in other examples (especially for real datasets), making the tree difficult to visualize and analyze. Therefore, we developed a tree refinement process, on which we remove the instances that shift. Figure 4 shows the refined version of the tree in Fig. 3. The *Removed Instances* (RI) at the bottom tells how many instances are removed. Tree refinement do not alter the original consensuses, it just simplifies the visualization. What we can understand from the trees is not just how the instances regroup on different view points of a set of base clusterings, but also what clusters (or consensuses) are stable. Stable clusters, those that do not change over several consecutive tree levels, suggest strong intra-cluster similarity. Even the merging of the clusters at a higher tree level explains that the merged clusters are very close in the data space compared with others. The recommended solution, the one circled in a red line, is the consensus that has the highest average Jaccard similarity to the ensemble. However, the most stable consensus can also be considered, as it usually identify a well separated clusters structure in the data space. In this example, it is the consensus of DT=5, with stability ST=4.

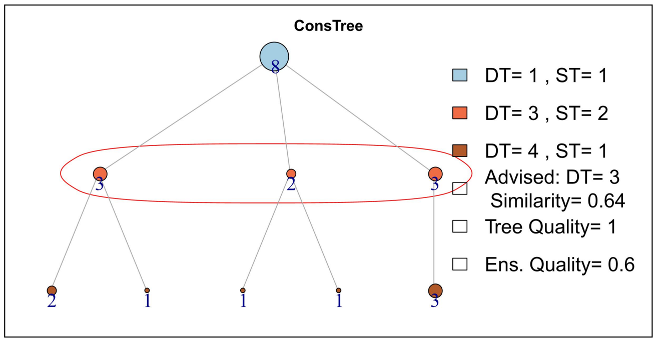

Figure 2: The ConsTree of the running example.

4. Experiments

The proposed method was implemented using R language [15] on a DELL PRECISION M4800 with Intel® Core^{TM} i7-4710MQ @ 2.50GHz, 32 GB of RAM, and Microsoft Windows 10 Professional (64-bit) operating system. Function *apriori* in *arules* R package [9] was used to discover the FCPs, by setting the *target* parameter to "closed frequent itemsets" and *support* = 1 / Datasize[5]. By considering the clusters of the different consensuses as nodes in a graph, that are linked by edges based

[5]A faster algorithm for generating the FCPs called FIST is proposed by [13], and an implementation of it in Java is available on the website of the authors.

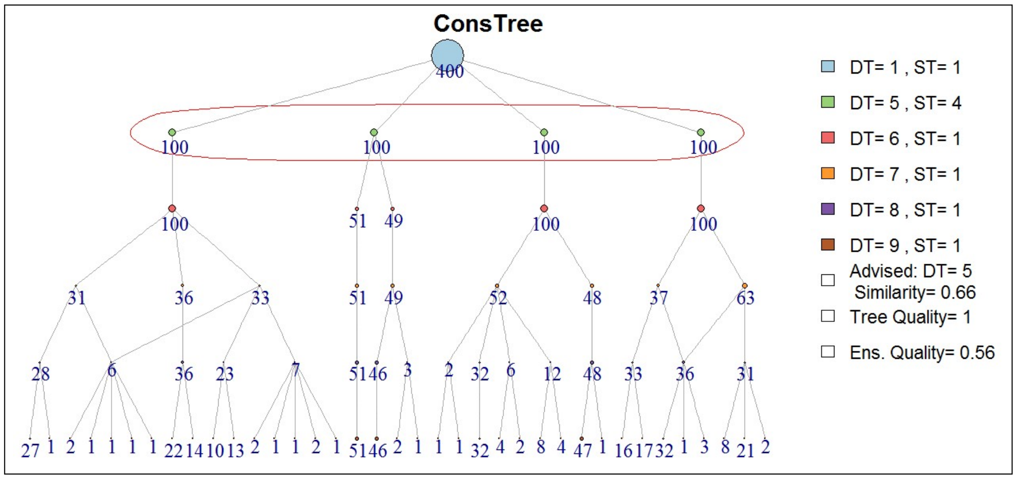

Figure 3: Example of a ConsTree.

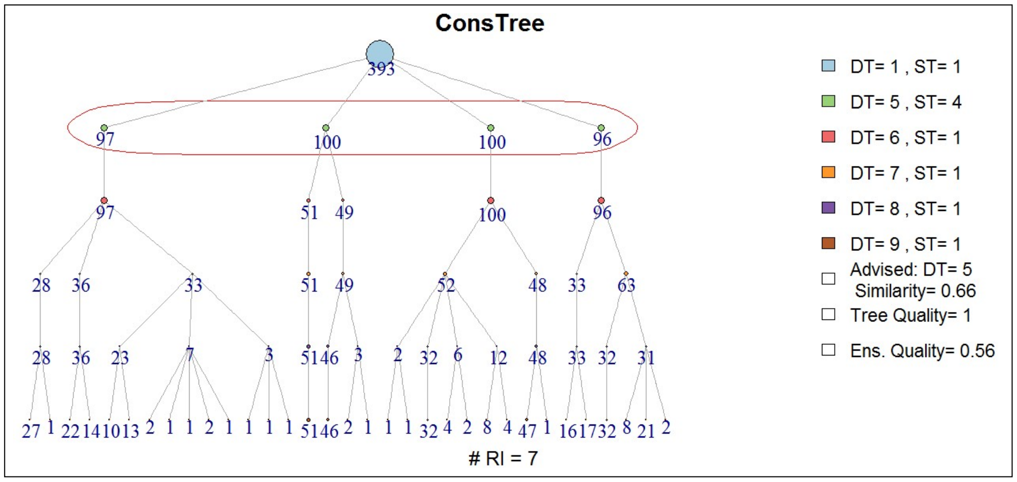

Figure 4: The result of refining the tree in Fig. 3.

on the shared instances between them, the ConsTree can be drawn using the *plot* function in the *igraph* R package [3]. If a node at a certain tree level is connected to several nodes at the next level (reading the tree from the bottom), then the refinement process keeps only one edge of the node that has the maximum shared instances with the next node.

We used the following real datasets from the UCI repository [12] for testing: Magic Gamma, Zoo, E.Coli, Iris, Breast Cancer, Congress Votes and Wine. Wingnut, Terta and EngyTime are synthetic datasets from [19]. Table 3 shows the properties of each dataset (size, number of attributes, number of actual classes), how many clusterings used in the ensemble, that each partitioned the dataset into a number of clusters within the k range. Among the clustering algorithms used to build the ensemble (all available in R): K-means, PAM, agglomerative hierarchical clustering, AGNES, DIANA, MCLUST (Gaussian Model-Based Clustering), C-Means, FANNY, Bagged Clustering, and SOM. In-ensemble similarity is a measure of the similarity between the base clusterings, calculated using Jaccard index. As all the datasets have true class labels, we used them to validate the quality of the base partitions (ensemble min and max similarity to the true class calculated also using Jaccard). In addition, we used the true classes to compare the performance of our proposed method (the "recommended consensus" with MT=0.4) against voting-based consensus methods available in R package CLUE [10], which include the following: SE, GV1, DWH, HE, GV3, SM, soft/symdiff, and consensus medoid. Note that the CLUE methods require specifying the number of clusters in the consensus solution, which is unnecessary for our method. Thus, we present also how many discovered clusters in our recommended consensus, while we used the true number of classes for CLUE methods.

The Magic Gamma dataset test is not about the quality of the results, but more about the applicability of our method on large datasets. Note that CLUE methods GV3 and soft/symdiff failed to work on this dataset as they required more than 32 GB of memory. For Magic Gamma, Breast Cancer and EngyTime tests, we considered that we have domain knowledge on how many clusters there should be in the dataset (like differentiating between positive and negative instances). Thus, all the base clusterings generate the same number of clusters, which is the best scenario for voting-based consensus methods.

In table 4, we present the execution time (in seconds) of the consensus methods used. For our method, we separated between the time required to discover the FCPs, and the time required to generate all the consensuses and find the recommended one. We can see that the total time of our method is acceptable, although not the fastest compared to CLUE, but much faster than GV3, SM, and Soft/symdiff. In fact, the execution time of our method is related to the size of the ensemble, and the in-ensemble similarity, as both will determine the number of generated FCPs.

5. Conclusions

We presented a new consensus clustering method, using the frequent closed pattern mining technique in order to transform the relation between the instances and the partition ensemble into clustering patterns. By dividing this pattern space into subspaces, we were able to generate multiple consensuses from different combinations of base clusterings. In each subspace, we tried to re-cluster the instances based on the information provided in the patterns, rather than finding a “median” partition for the ensemble. The consecutive processing of the different clustering views enables us to discover the number of hidden clusters in the dataset (without the need to specify this explicitly), and to build a ConsTree to visualize the relationships between the instances.

The proposed method achieved good results in terms of quality and the number of discovered clusters, with acceptable execution time considering that it generates multiple solutions. As the FCI technique was designed to efficiently discover patterns in very large datasets, our method can be applied on large datasets, as the patterns space will be a high pruning for the actual instances space. For example, for the Magic Gamma dataset (19020 instances), the patterns space contains 156 patterns only. The number of discovered patterns depends on the size of the ensemble, and how much agreement exists between the base clustering decisions.

Using the ConsTree, the analysts are not limited to one final solution, but rather they can choose another one based on their observation and preferences. For example, they may prefer to choose a solution where a certain cluster is divided into two, as this may reflect a more meaningful grouping for them.

References

[1] Sitaram Asur, Duygu Ucar, and Srinivasan Parthasarathy. An ensemble framework for clustering protein–protein interaction networks. *Bioinformatics*, 23(13):i29–i40, 2007.

[2] Aaron Ceglar and John F. Roddick. Association mining. *ACM Computing Surveys*, 38(2), 2006.

[3] Gabor Csardi and Tamas Nepusz. The igraph software package for complex network research. *InterJournal*, Complex Systems:1695, 2006.

[4] Sandrine Dudoit and Jane Fridlyand. Bagging to improve the accuracy of a clustering procedure. *Bioinformatics*, 19(9):1090–1099, 2003.

[5] Xiaoli Zhang Fern and Carla E Brodley. Solving cluster ensemble problems by bipartite graph partitioning. In *Proceedings of the twenty-first international conference on Machine learning*, page 36. ACM, 2004.

[6] Bernd Fischer and Joachim M Buhmann. Bagging for path-based clustering. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 25(11):1411–1415, 2003.

[7] Ana LN Fred and Anil K Jain. Data clustering using evidence accumulation. In *Pattern Recognition, 2002. Proceedings. 16th International Conference on*, volume 4, pages 276–280. IEEE, 2002.

[8] Reza Ghaemi, Md Nasir Sulaiman, Hamidah Ibrahim, and Norwati Mustapha. A survey: Clustering ensembles techniques. *WASET*, 50:636–645, 2009.

[9] Michael Hahsler, Bettina Gruen, and Kurt Hornik. arules – A computational environment for mining association rules and frequent item sets. *Journal of Statistical Software*, 14(15):1–25, 2005.

[10] Kurt Hornik. A CLUE for CLUster Ensembles. *Journal of Statistical Software*, 14(12), 2005.

[11] Paul Jaccard. The distribution of the flora in the alpine zone.1. *New Phytologist*, 11(2):37–50, 1912.

[12] M. Lichman. UCI machine learning repository, 2013.

[13] Kartick Chandra Mondal, Nicolas Pasquier, Anirban Mukhopadhyay, Ujjwal Maulik, and Sanghamitra Bandhopadyay. A new approach for association rule mining and bi-clustering using formal concept analysis. In *Machine Learning and Data Mining in Pattern Recognition*, pages 86–101. Springer, 2012.

Table 3: Tests validation.

Dataset	Magic Gamma	Zoo	E.Coli	Iris	Breast Cancer	Congress Votes	Wine	Wingnut	Terta	EngyTime
Dataset Size	19020	101	336	150	699	435	178	1016	400	4096
# of attributes	10	16	7	4	9	16	13	2	3	2
# of true classes	2	7	8	3	2	2	3	2	4	2
Ensemble size	6	10	9	14	8	8	9	10	9	8
K range	[2]	[3,12]	[4,12]	[2,6]	[2]	[3,6]	[2,6]	[2,6]	[2,7]	[2]
In-ensemble similariy	0.70	0.52	0.53	0.54	0.81	0.66	0.5	0.55	0.59	0.82
Ensemble Min.	0.36	0.27	0.27	0.29	0.36	0.63	0.43	0.40	0.29	0.46
Ensemble Max.	0.46	0.94	0.65	0.88	0.89	0.60	0.89	0.99	1.00	0.88
Our method	0.44	0.82	**0.67**	0.60	**0.87**	0.60	0.82	0.94	**1.00**	**0.84**
# of clusters in our method	2	9	5	3	2	2	3	2	4	2
SE	0.42	0.76	0.44	0.64	0.86	0.61	0.47	0.88	**1.00**	**0.84**
GV1	0.42	0.78	0.44	**0.65**	0.86	0.61	0.59	0.88	**1.00**	**0.84**
DWH	0.42	0.76	0.51	0.59	0.86	0.61	0.53	0.93	**1.00**	0.83
HE	0.42	0.74	0.47	0.64	0.85	0.60	0.87	0.88	**1.00**	0.83
GV3	NA	0.76	0.46	0.59	0.86	0.61	**0.89**	0.92	**1.00**	**0.84**
SM	0.41	0.82	0.48	0.63	0.85	0.60	0.50	0.88	0.93	0.83
soft/symdiff	NA	0.83	0.38	0.63	0.85	**0.63**	0.67	0.92	**1.00**	0.83
Medoids	**0.46**	**0.84**	0.49	0.55	0.86	0.47	0.54	**0.99**	0.89	0.83

Table 4: Execution time of the consensus methods (in seconds).

Dataset	Magic Gamma	Zoo	E.Coli	Iris	Breast Cancer	Congress Votes	Wine	Wingnut	Terta	EngyTime
Patterns	1.088	0.086	0.196	0.445	0.079	0.141	0.128	0.255	0.117	0.398
Our method	1.299	0.303	2.529	0.751	0.070	1.171	0.534	1.616	0.638	0.555
SE	0.084	0.012	0.027	0.013	0.013	0.011	0.008	0.017	0.016	0.037
GV1	0.113	0.067	0.484	0.050	0.017	0.013	0.103	0.025	0.052	0.035
DWH	0.064	0.007	0.008	0.008	0.008	0.007	0.005	0.012	0.007	0.020
HE	0.144	0.010	0.018	0.013	0.010	0.008	0.010	0.024	0.011	0.030
GV3	NA	0.715	6.306	0.816	13.274	4.276	0.953	25.663	6.388	651.835
SM	33.681	0.783	5.894	0.758	1.399	1.033	0.809	3.296	1.926	10.884
soft/symdiff	NA	5.479	39.051	10.921	138.933	57.454	8.247	377.998	49.040	5639.546
Medoids	0.292	0.039	0.037	0.76	0.036	0.026	0.026	0.127	0.036	0.121

[14] Nicolas Pasquier, Yves Bastide, Rafik Taouil, and Lotfi Lakhal. Efficient mining of association rules using closed itemset lattices. *Inf. Systems*, 24(1):25–46, 1999.

[15] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2015.

[16] S Sarumathi, N Shanthi, and M Sharmila. A comparative analysis of different categorical data clustering ensemble methods in data mining. *IJCA*, 81(4):46–55, 2013.

[17] Alexander Strehl and Joydeep Ghosh. Cluster ensembles – a knowledge reuse framework for combining multiple partitions. *JMLR*, 3:583–617, 2003.

[18] Alexander P Topchy, Anil K Jain, and William F Punch. A mixture model for clustering ensembles. In *SDM*, pages 379–390. SIAM, 2004.

[19] Alfred Ultsch. Clustering with SOM: U*C. In *Proc. WSOM Workshop*, pages 75–82, 2005.

[20] Sandro Vega-Pons and Paolo Avesani. On pruning the search space for clustering ensemble problems. *Neurocomputing*, 150:481–489, 2015.

[21] Sandro Vega-Pons and José Ruiz-Shulcloper. A survey of clustering ensemble algorithms. *IJPRAI*, 25(03):337–372, 2011.

[22] Ou Wu, Weiming Hu, Stephen J Maybank, Mingliang Zhu, and Bing Li. Efficient clustering aggregation based on data fragments. *IEEE Trans Syst Man Cybern B Cybern.*, 42(3):913–926, 2012.