MultiCons: Multiple Consensus Clustering Method Using Frequent Closed Pattern Mining

Atheer Al-Najdi, Nicolas Pasquier, Frédéric Precioso

LABORATOIRE

INFORMATIQUE, SIGNAUX ET SYSTÈMES DE SOPHIA ANTIPOLIS
UMR 7271

MultiCons: Multiple Consensus Clustering Method Using Frequent Closed Pattern Mining

Atheer Al-najdi, Nicolas Pasquier, Frédéric Precioso
EQUIPE MinD

Rapport de Recherche

Septembre-2015

Laboratoire d'Informatique, Signaux et Systèmes de Sophia-Antipolis (I3S) - UMR7271 - UNS CNRS
2000, route des Lucioles — Les Algorithmes - bât. Euclide B — 06900 Sophia Antipolis — France
http ://www.i3s.unice.fr

Membre de UNIVERSITÉ CÔTE D'AZUR

MultiCons: Multiple Consensus Clustering Method Using Frequent Closed Pattern Mining

Atheer Al-najdi, Nicolas Pasquier, Frédéric Precioso [1]

EQUIPE MinD
September-2015 - 16 pages

Abstract :
Clustering, which aim is to identify groups of similar instances in a dataset, is one of the important tasks in data mining. Many clustering algorithms were developed in the last 50 years. However, selecting an algorithm to cluster a dataset is a difficult task, especially if there is no previous knowledge on the structure of the data space. Consensus clustering methods can be used to combine multiple base clusterings into a new solution that provides better quality partitioning. In this work, we present a new consensus clustering method based on detecting underlying clustering patterns shared by base clusterings using frequent closed itemset mining. Instead of generating one consensus, this new approach generates multiple possible solutions, based on varying the number of base clusterings, and links these solutions in a tree-shaped hierarchical view that eases the selection of the most relevant clustering. This hierarchical view also provides an analysis tool of the dataset, for example, to discover strong clusters or outlier instances.

Key-words : Unsupervised learning ; Clustering ; Consensus clustering ; Ensemble clustering ; Frequent closed itemsets ; Frequent concept patterns.

1. Laboratoire I3S – Université Nice Sophia Antipolis – {alnajdi, pasquier, precioso}@i3s.unice.fr

MultiCons : Méthode de Clustering par Consensus Basée sur les Itemsets Fermés Fréquents

Résumé :
Le clustering, dont l'objectif est l'identification de groupes d'instances similaires dans un ensemble de données, est l'une des tâches les importantes de la fouille de données. De nombreux algorithmes de clustering ont été développés durant les 50 dernières années. Toutefois, choisir un algorithme pour regrouper des données par similarité est une tâche difficile, surtout si aucune connaissance préalable sur la structure de l'espace des données n'est disponible. Les méthodes de clustering par consensus peuvent être utilisées afin de combiner les résultats de plusieurs clusterings dans une nouvelle solution afin d'obtenir un partitonnement des instances de meilleure qualité. Dans ce travail, nous présentons une nouvelle méthode de clustering par consensus fondée sur la détection des motifs de regroupement sous-jacents communs aux clusterings initiaux en utilisant les itemsets fermés fréquents. De plus, au lieu de générer un unique consensus comme dans les approaches par consensus classiques, cette nouvelle approche permet de générer plusieurs solutions, en se basant sur la variation du nombre d'aggrégation des clusterings initiaux, et de les relier dans une vue hiérarchique afin de faciliter la sélection des regroupements les pus pertinents. Ce point de vue hiérarchique fournit également un outil d'analyse de l'espace des données, par exemple, afin de découvrir des sous-groupes d'instances fortements correlées ou des instances aberrantes.

Mots-clefs : Classification non-supervisée ; Clustering ; Clustering par consensus ; Ensembles clustering ; Itemsets fermés fréquents ; Motifs conceptuels fréquents.

MultiCons: Multiple Consensus Clustering Method Using Frequent Closed Patterns Mining

Atheer Al-najdi, Nicolas Pasquier, and Frédéric Precioso

Univ. Nice Sophia Antipolis, CNRS, I3S, UMR 7271, 06900 Sophia Antipolis, France
{alnajdi, pasquier, precioso}@i3s.unice.fr

Abstract. Clustering, which aim is to identify groups of similar instances in a dataset, is one of the important tasks in data mining. Many clustering algorithms were developed in the last 50 years. However, selecting an algorithm to cluster a dataset is a difficult task, especially if there is no previous knowledge on the structure of the data space. Consensus clustering methods can be used to combine multiple base clusterings into a new solution that provides better quality partitioning. In this work, we present a new consensus clustering method based on detecting underlying clustering patterns shared by base clusterings using frequent closed itemset mining. Instead of generating one consensus, this new approach generates multiple possible solutions, based on varying the number of base clusterings, and links these solutions in a tree-shaped hierarchical view that eases the selection of the most relevant clustering. This hierarchical view also provides an analysis tool of the dataset, for example, to discover strong clusters or outlier instances.

Keywords: Unsupervised learning; Clustering; Consensus clustering; Ensemble clustering; Frequent closed patterns; Bi-clustering.

1 Introduction

Clustering is the process of partitioning a dataset into groups, so that the instances in the same group are more similar to each other than to instances in any other group. This partitioning may lead to discover meaningful patterns in the dataset. Many clustering algorithms were developed in the last 50 years, and, most often, each algorithm produces different partitioning when applied to the same dataset, because they are designed to target a specific model for clustering the instances. Another factor that affects the results is the parameter settings: Most clustering algorithms require that the user specify the number of clusters targeted (usually known as parameter K), and other parameters are more specific to the clustering algorithm considered. For example, density-based methods do not require K, but instead require other parameters to define what is a dense region in the data space. Thus the question is: How to choose a clustering for a dataset from these many possibilities?

The most common solution is to use validation measure(s) to compare the results and select the one that gets the higher score [5,8]. There are two general

categories of validation measures: *Internal validation* that compares the clustering against a specific clustering model, and *external validation* that compares the clustering against true labels (class labels given on an evaluation set using domain knowledge). In both categories, we have many validation measures, and there is no one that impartially evaluates the results of any clustering algorithm [20]. In real life, the user can have similar scores for different validation measures and/or for different clustering results, while the results are different in many aspects, like in the number of clusters or in the instances grouping into clusters.

Rather than depending on validation measures, another approach is to combine the multiple clustering solutions generated by several clustering algorithms and/or settings, in order to produce a final clustering which is better than each individual algorithm can produce. This technique is called *consensus clustering*, *aggregation of clusterings* or *ensemble clustering*, and the clustering algorithms to be combined are called *base clustering algorithms*. Many consensus clustering methods have been proposed, and we discus some of them in the next section. In several articles on clustering ensemble, authors have tried to define a set of properties that endorses the use of clustering ensemble methods [7,20]:

- Robustness: The consensus must have better average performance than the single base clustering algorithms.
- Consistency: The result of the combination should be somehow, very similar to all combined single base clustering algorithm results.
- Novelty: Cluster ensembles must allow finding solutions unattainable by single base clustering algorithms.
- Stability: Results with lower sensitivity to noise and outliers.

However, validating these properties in practice is very difficult because of the unsupervised nature of the clustering ensemble process [7,18].

In this paper, we propose a new consensus clustering method named MultiCons. Instead of providing the user with a single solution, we generate multiple consensuses by varying the selection of base clusterings, then linking these multiple solutions in a hierarchical view. The user can then not only select the best solution, but also discover strong clusters in the dataset that do not change when varying the base clusterings. Hence, our proposed method is a combination of ensemble of consensus solutions with a visual data analysis tool.

The paper is organized as follows: Section 2 discusses some of the previous work in consensus clustering. Section 3 explains the proposed approach, and we demonstrate how the approach works by a synthetic example in section 4. Some experimental results are shown in section 5, and we provide conclusions in section 6.

2 Related Work

Consensus clustering refers to the problem of finding a single consensus clustering from a number of different inputs or base clusterings that have been obtained

for a given dataset [11,23]. The advantage of this technique is to have a new clustering result that is at least as good as the best clustering achieved by the base methods. Thus, it will be easier to obtain a good clustering rather than trying with different algorithms and compare them against a validation score which may favor one kind of clustering techniques over the others.

Many consensus clustering methods were developed over the past years. In Asur *et al.* [2], six predefined clustering algorithms suitable for protein-protein datasets clustering were considered as base clusterings. A cluster membership matrix[1] is then built, and a consensus clustering method is applied over this matrix (agglomerative hierarchical clustering or recursive bisection) to obtain the final consensus. All the 6 base clusterings have K clusters, so if K is high the resulting membership matrix is a sparse binary matrix and the consensus clustering of this matrix is not be efficient. Thus, PCA (Principle Components Analysis) is applied before the consensus clustering to reduce the dimensionality of the membership matrix into less, but more expressive, dimensions. Instead of PCA, Another approach considers weighting the clusters according to their reliability. These different consensus techniques were compared, and the authors conclude that the PCA based technique produced very efficient clustering and identified multiple functionalities of proteins.

Three consensus clustering methods were proposed by Strehl & Ghosh [17]. The first step for all their proposed consensus functions is to transform the given clusterings into a suitable hypergraph representation, where each cluster (column in the membership matrix) from any base clustering is considered as a hyperedge that connects several vertices (all the instances in this cluster) in a hypergraph. Based on this mapping Strehl & Ghosh propose: i) *Cluster-based Similarity Partitioning Algorithm* (CSPA) which is based on an overall similarity matrix S built from the membership matrix H by using $S = \frac{1}{r}HH^{\dagger}$, where r is the number of base clusterings. The aforementioned hypergraph is built from this similarity matrix so that each hyperdge represents the sum of similarities between a given pair of vertices (i.e. each time the two considered vertices are clustered together by any base clustering their similarity is increment by 1), then a graph-based clustering method (METIS) provides the consensus clustering; ii) *HyperGraph-Partitioning Algorithm* (HGPA) all hyperedges as well as all vertices are equally weighted, then a hypergraph partitioning algorithms (HMETIS) defines the consensus by cutting a minimal number of edges; iii) *Meta-CLustering Algorithm* (MCLA) follow the same ideas, but hyperedges weights are proportional to the similarity between vertices (instances) which is calculated using binary Jaccard measure. The resulting hypergraph is then clustered using METIS to generate K clusters. For each of these K meta-clusters, its hyperedges are collapsed into a single meta-hyperedge. Each meta-hyperedge has an association vector which contains an entry for each object describing its level of association with the corresponding meta-cluster.

With their algorithm WClustering, Li & Ding [11] proposed weighting the base clusterings to ensure removing redundant (similar) partitionings, since this

[1] See section 3.2 for a definition of cluster membership matrix.

process produces better results compared to other methods which generate the consensus from brute-force averaging of the base clusterings. Weights are automatically determined by an optimization process. Experimental results showed that more accurate clustering was achieved by the k-means algorithm when applied to the weighted consensus similarity matrix, compared to the results of CSPA and HGPA.

In Zhang & Li [23], the base clusterings are compared using pairwise similarity, and then divided into groups using K-means. On each group, one of the previously discussed consensus methods is used: PCA-based consensus algorithm [2], CSPA and HGPA from [17], and WClustering [11]. Thus, the final result is K consensuses for the user to select from.

The idea in Caruana *et al.* [3] is to generate many base clusterings, then build a similarity matrix for these different partitionings using Rand index. This similarity matrix is passed to agglomerative hierarchical clustering to build a meta clustering. The dendrogram shows how the clusterings are similar to each other, thus there is no final consensus. Instead, the user can analyze the resulting dendrogram to choose which clustering is the most relevant. To have a diversity in the base clusterings, feature weighting using Zipf distribution and PCA were used to produce different base clustering views.

For more information about consensus clustering methods, see Ghaemi *et al.* [7] and Vega-Pons & Ruiz-Shulcloper [20].

3 The Proposed Approach

Instead of using a similarity matrix to group or meta-cluster the different partitions generated by the base clustering methods, the MultiCons approach is based on the *Frequent Closed Itemsets* (FCIs) [15]. FCIs is an association rule mining approach used to discover relationships between attribute values in a dataset. FCIs are maximal sets of items (variable values) that are common to a set of objects (instances) in a data matrix. One of the advantages of FCIs is that they represent a subset of the frequent itemset lattice, yet they can be used to derive all the possible association rules. Therefore, by reducing the search space, memory consumption and execution times for generating all valid association rules are in most cases drastically reduced compared to the approach based on frequent itemsets. See [4] for an extensive survey on association rule mining.

The MultiCons approach uses FCIs to find clustering patterns among the base partitionings. That is, it finds sets of instances that are clustered similarly by maximal sets of base clusterings. The lattice of these clustering patterns, or bi-clusters[2], is then processed to build different consensuses. Thus, the user will have several final clusterings to choose from based on his/her preference for the number of clusters or their contents (does the grouping of instances into clusters produce meaningful patterns for the user). The successive steps of the proposed approach, described in algorithm 1, are explained in the following subsections.

[2] Bi-clustering is defined by Mirkin [12] as "the simultaneous clustering of both row and column sets in a data matrix".

Input : Dataset to cluster
Output: ConsTree tree of consensuses

Generate multiple base clusterings of the dataset;
Build the membership matrix $\mathcal{M}$;
Generate FCPs from $\mathcal{M}$ for *minsupport* = 0;
Sort the FCPs in ascending order of the size of their instances list;
MaxDT ← Number of base clusterings;
BiClust ← {FCPs whose FCIs contain *MaxDT* base clusters};
Assign a label to each bi-cluster in *BiClust* to build the first consensus vector and store it in a list of vectors *ConsVctrs*;
/* Build the remaining consensuses */;
for *DT = (MaxDT - 1)* **to** *1* **do**
    BiClust ← *BiClust* ∪ {FCPs whose FCIs contain *DT* base clusters};
    N ← length(*BiClust*) **// Nbr of itemsets in** *BiClust*;
    repeat
        for *i = 1* **to** *N* **do**
            $B_i \leftarrow i^{\text{th}}$ bi-cluster in *BiClust*;
            for *j = 1* **to** *N*, $j \neq i$ **do**
                $B_j \leftarrow j^{\text{th}}$ bi-cluster in *BiClust*;
                if $B_i.instances\_list \subset B_j.instances\_list$ **then**
                    Remove B_i from *BiClust*;
                    Next *i*;
                else if $B_j.instances\_list \subset B_i.instances\_list$ **then**
                    Remove B_j from *BiClust*;
                    Next *j*;
                else if $B_i.instances\_list \cap B_j.instances\_list \neq \emptyset$ **then**
                    $B_i \leftarrow B_i \cup B_j$;
                    Remove B_j from *BiClust*;
                    Next *i*;
                end
            end
        end
    until *All bi-clusters in BiClust are unique*;
    Assign a label to each bi-cluster in *BiClust* to build a consensus vector and add it to *ConsVctrs*;
end
/* Remove similar consensuses */;
ST ← Vector of '1's of length *MaxDT*;
for *i = MaxDT* **to** *2* **do**
    $V_i \leftarrow i^{\text{th}}$ consensus in *ConsVctrs*;
    for *j = (i - 1)* **to** *1* **do**
        $V_j \leftarrow j^{\text{th}}$ consensus in *ConsVctrs*;
        if *Jaccard*(V_i, V_j) *= 1* **then**
            $ST[i] \leftarrow ST[i] + 1$;
            Remove $ST[j]$;
            Remove V_j from *ConsVctrs*;
        end
    end
end
Build the tree of consensuses in *ConsVctrs*

Algorithm 1: The MultiCons approach.

3.1 Base Clusterings

The base clusterings are hard partitions of a dataset, without limitations on the number of clusters, or on the category of the clustering algorithms used. Thus, we can combine a linear partitioning-based clustering like K-means or PAM with results from hierarchical, Gaussian model, and/or density-based clustering algorithms, as long as hard partitions are retrieved from them. Hard partitions means that each instance of the dataset can belong to only one cluster. It is preferable to use different values of K for each base clustering algorithm, and to use different settings (if possible) if the same algorithm is used to generate different clusterings, to ensure the diversity in the base partitionings. See [8] and [21] for details about clustering methods.

3.2 Membership Matrix

After generating multiple clusterings of the dataset using a set of base clustering methods, a *membership matrix* $\mathcal{M}$ is built. This matrix is a binary matrix of $N \times M$ cells, where N is the number of instances in the clustered dataset, and M is the number of cluster vectors (total number of clusters generated by all base clustering algorithms) as given in definition 1.

Definition 1. *A 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 variables, each designating a cluster, 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 the instance i belongs to the cluster c.*

An example membership matrix is given in table 1. A cluster vector V $(V_1, V_2, \ldots, V_N)$ is a binary vector that represents a cluster as a column of the matrix, such that $V_i = 1$ if instance i belongs to the cluster, and $V_i = 0$ otherwise, $\forall i$, $1 \leq i \leq N$. In table 1, for instance, the first column represents the cluster vector KM C1 $= \{1, 1, 1, 0, 1, 0, 1, 1, 1, 0\}$, designated by *item* KM C1.

Definition 2. *An item of a 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 $C = \{c_1, \ldots, c_n\} \subseteq \mathcal{C}$ in $\mathcal{M}$. An itemset $C \subseteq \mathcal{C}$ is frequent in $\mathcal{M}$ iff its frequency, called support, in $\mathcal{M}$ defined as support$(C) = |\{I \in \mathcal{I} \mid \forall i \in I, \forall c \in C$, we have $(i, c) \in \mathcal{R}\}|$ is greater than or equal to the user-defined minsupport threshold.*

3.3 Generating Bi-Clusters

Having the membership matrix $\mathcal{M}$, the next step is to generate the *frequent closed patterns* (FCPs) from which bi-clusters will be generated. FCPs represent the maximum number of base clusterings that agree on grouping a set of instances. That is, any subset of these base clusterings that agree on grouping

Table 1: Example membership matrix.

Id	KM C1	KM C2	PAM C1	PAM C2	PAM C3	DB C1	DB C2
1	**1**	0	0	0	**1**	0	**1**
2	**1**	0	0	0	**1**	0	**1**
3	**1**	0	0	0	**1**	0	**1**
4	0	1	1	0	0	1	0
5	**1**	0	0	0	**1**	0	**1**
6	0	1	0	1	0	1	0
7	**1**	0	0	0	**1**	0	**1**
8	**1**	0	0	0	**1**	0	**1**
9	**1**	0	0	0	**1**	0	**1**
10	0	1	1	0	0	1	0

the same set of instances will not be considered[3]. To accomplish this, we use the FIST algorithm [13] that finds the FCPs using generalized suffix-trees. Each FCP associates an FCI (a set of base clusters identifiers) and its corresponding instance identifier list, i.e., the identifiers of dataset instances that are common to all clusters in the FCI set. Stated another way, FCPs are maximal rectangles in the membership matrix. See [22] for complexity and applicability considerations about FCIs and FCPs mining.

Definition 3. *A frequent closed pattern $P = (C, I)$ in the 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:*
i) $\forall i \in I$ *and* $\forall c \in C$*, we have* $(i, c) \in \mathcal{R}$*.*
ii) $|I| \geq$ *minsupport, i.e.,* C *is a frequent itemset.*
iii) $\nexists i' \in \mathcal{I}$ *such that* $\forall c \in C$*, we have* $(i', c) \in \mathcal{R}$*.*
iv) $\nexists c' \in \mathcal{C}$ *such that* $\forall i \in I$*, we have* $(i, c') \in \mathcal{R}$*.*

Consider the following FCP in table 1: ({KM C1, PAM C3, DB C2}, {1, 2, 3, 5, 7, 8, 9}). The itemset {KM C1, PAM C3, DB C2} is a frequent closed itemset and {1, 2, 3, 5, 7, 8, 9} is the maximal set of instances in relation with all the three items KM C1, PAM C3 and DB C2. This bi-cluster reflects the fact that the instances 1, 2, 3, 5, 7, 8 and 9 are common among cluster 1 of K-means, cluster 3 of PAM, and cluster 2 of DBScan.

The output of this step is the pool of FCPs from which subsets are selected to build a consensus.

Definition 4. *Let $\mathcal{M} = (\mathcal{I}, \mathcal{C}, \mathcal{R})$ be a membership matrix. A consensus clustering L of m bi-clusters from $\mathcal{M}$ is a set $L = \{P_1, ..., P_m\}$ with $P_{k \in [1,m]} =$*

[3] This reduces processing time by generating only clustering patterns of maximum agreement between base clusters. Using, for instance, the well-known Apriori algorithm [1] for generating frequent itemsets, and associating the corresponding instances list with each, will generate all possible patterns, which is redundant for the proposed approach.

(C_k, I_k), $C_k \subseteq \mathcal{C}$, $I_k \subseteq \mathcal{I}$ such that $\bigcup_{k=1}^{k=m} I_k = \mathcal{I}$. A consensus clustering is an unique partitioning of the dataset instances.

3.4 Generating Multiple Consensuses

The process of building multiple consensuses starts by building the first consensus, which is the FCPs whose FCIs consist of all the base clusterings. Then, to build the next consensus, we combine with the previously built bi-clusters the FCPs whose FCI contains a specific number, called *Decision Threshold* (DT), of base clusterings. The DT value represents the minimum number of base partitionings (base clustering executions) to consider for building a consensus. For the first consensus, we have a DT equals to the number of base partitionings, and we then decrease DT sequentially until DT = 1. By decrementing DT, the new consensus considers another clustering view generated by a smaller number of base clusterings.

After the first consensus $L = \{P_1, ..., P_m\}$ was built, and for each DT value, a bi-cluster $P = (C, I)$, where C is the set of base cluster identifiers, has one of the following properties that are defined according to sets of instance identifiers:

i) Uniqueness: The bi-cluster does not intersect with any other bi-cluster. That is, $\nexists P' = (C', I') \in L$ such that $I \cap I' \neq \emptyset$.
ii) Inclusion: The bi-cluster is a subset of another bi-cluster. That is, $\exists P' = (C', I') \in L$ such that $I \subseteq I'$.
iii) Intersection: The bi-cluster intersects with another bi-cluster. That is, $\exists P' = (C', I') \in L$ such that $I \cap I' \neq \emptyset$, $I \setminus I' \neq \emptyset$ and $I' \setminus I \neq \emptyset$.

To build a new consensus, intersecting bi-clusters are merged since each one represent a set of instances that are close (very similar) in the data space[4]. The *intersection* property between two bi-clusters is an indication of closeness of these bi-clusters. Any bi-cluster with *inclusion* property is removed. The merging process repeats until all the remaining bi-clusters all have the *uniqueness* property. As we use only a small set of FCPs for generating each consensus, the process of generating multiple consensuses is fast, even when the set of all FCPs is large. Also, building consensuses for lower DT values is faster than for high DT values because the previous consensus contains fewer bi-clusters.

Similar consensuses are removed, and a *stability counter* (ST) is used to count how many times each consensus is generated with different values of DT. For example, if a consensus has ST = 3, this means that this consensus is generated from 3 consecutive values of DT. The output of this step, that consists of all consensuses generated for the different DT values, constitutes the ConsTree tree of consensuses.

[4] This is the objective of clustering algorithms, yet they differ in how they define the similarity between instances.

3.5 ConsTree: A Tree of Consensuses

Each consensus clustering is then represented as a level in the ConsTree tree of consensuses.

Definition 5. *Let $\mathcal{M}$ = ($\mathcal{I}$, $\mathcal{C}$, $\mathcal{R}$) be a membership matrix. A tree of consensuses from $\mathcal{M}$ is an ordered set ($\mathcal{L}$, $\preceq$) of consensuses $\mathcal{L} = \bigcup_{DT=MaxDT}^{DT=MinDT} L^{DT}$ ordered according to inclusion relation $\subseteq$ between sets of cluster identifiers. Let's denote $L^{\alpha} = \{P_1^{\alpha}, ..., P_m^{\alpha}\}$ and $L^{\beta} = \{P_1^{\beta}, ..., P_n^{\beta}\}$ the consensuses generated for α and β DT values respectively. Let's denote $P_q^{\alpha} = (C_q^{\alpha}, I_q^{\alpha})$ the q^{th} bi-cluster in L^{α} and $P_r^{\beta} = (C_r^{\beta}, I_r^{\beta})$ the r^{th} bi-cluster in L^{β}, with $1 \leq q \leq m$ and $1 \leq r \leq n$. For α ¿ β we have $L^{\alpha} \preceq L^{\beta}$, that is $\forall P_q^{\alpha} \in L^{\alpha}$, $\exists P_r^{\beta} \in L^{\beta}$ such that $I_q^{\alpha} \subseteq I_r^{\beta}$. L^{α} is a predecessor of L^{β} in the tree of consensuses.*

After generating all the possible consensuses, a tree graphical representation (Hasse diagram of the tree of consensuses) is built to visualize these different results. Each level in the tree represent a consensus, with nodes representing its clusters. The bottom level of the tree is the first consensus. At each new level, the clusters of the lower level are merged into clusters at the higher level.

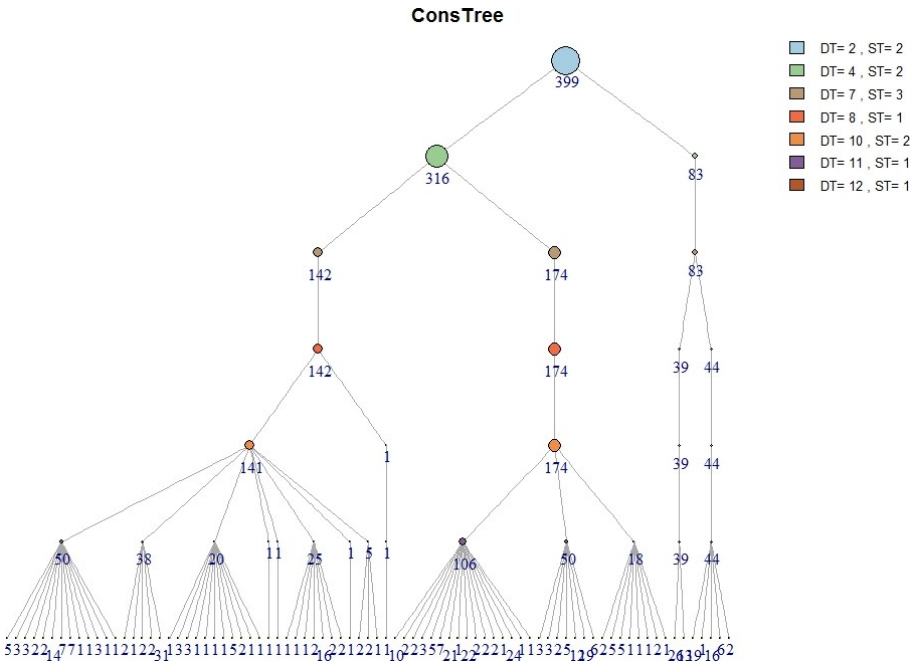


Fig. 1: Example ConsTree.

An example of the ConsTree is shown in figure 1. The tree explains the result of applying our approach on a dataset of 399 instances clustered using 12 base

clusterings. The base clusterings are selected randomly with random settings. By using ST, the tree consists of 7 levels instead of 12. Without it, we will have duplicated levels, that is, a level at DT=9 which is exactly the same clusters of DT=10. Other redundant levels that are removed because they are identical to a higher level consensus[5] are: DT=6, 5, 3, and 1. As an analysis tool, the tree shows that the clusters of sizes 44, 39, or 174 are strong clusters, so the user should consider this when he/she wants to choose a final consensus clustering. The ST value can point to the best result as the one that is more stable than others (the one at DT=7), but as the clustering task is more related to the relevance of the found patterns to user preference, a user may prefer to select the consensus at DT=8, because he/she prefers to separate the cluster of 83 instances at DT=7 into 2 clusters like in DT=8, as these 2 may reflect better patterns for him/her.

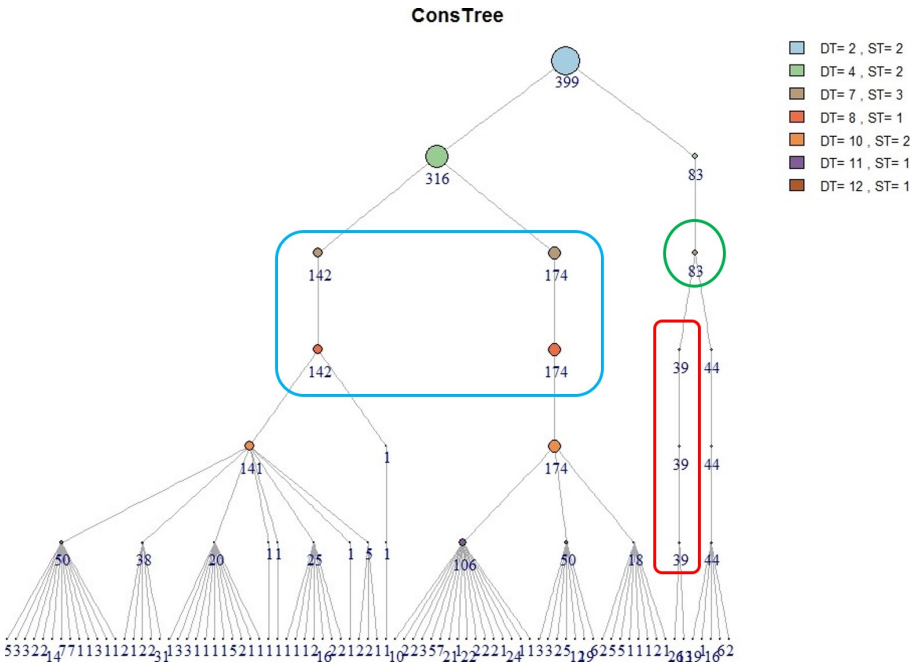


Fig. 2: Example of Analysis from ConsTree visualization.

Visualizing the tree enables the user to understand how the consensuses were built based on different combinations of base clusterings, and to discover strong partitions in the dataset: The cluster(s) that do not merge with others on a

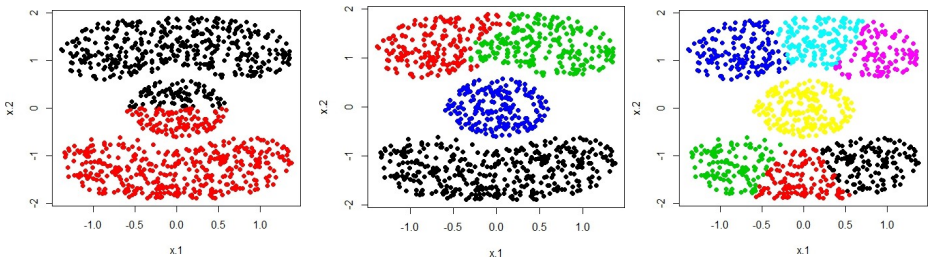
[5] A "higher level consensus" means that it is built from a higher number of base clusters.

sequence of consensuses, which reflect agreement between many base clusterings to construct it(them).

Visualizing the tree facilitates not only the analysis of the consensuses, but also the analysis of the dataset, that is, it enables the user to discover what partitions are more strong (consist of very similar instances) than others (as the ones circled in blue and red on the figure 2). It may also highlight a group of items that are far from being similar to other instances such as the column of stable cluster circled in red and the similar column beside. In the analysis of data these stable clusters provide a lot of information on the items they contain. Furthermore, the fact that these two columns merge into one cluster, circled in green, rather than merging with any of the previous stable clusters (circled in blue) provides again insight on the peculiar information their items hold. Based on that, the user may try to re-preprocess the dataset to get more relevant clustering results.

4 Synthetic Example

This example is based on the "Cassini" dataset used by Dimitriadou *et al.* [6] which consists of 3 structures in a 2D data space: The 2 external structures are banana-shaped, and the middle one is circle-shaped. The dataset consist of 1000 instances. Supposing that we do not know that the dataset contains 3 natural clusters, this dataset was clustered using 3 base clustering methods: K-means with K=2, agglomerative hierarchical clustering with average linkage and K=4, and a Gaussian model-based clustering with K=7. The graphical display of these 3 base clustering results are shown in figure 3.



(a) K-means (b) Hierarchical (c) Model-based

Fig. 3: Example of clustering Cassini dataset with 3 different base clusterings.

After building the membership matrix of 1000×13 binary cells, the FIST algorithm generates 18 FCPs describing clustering patterns common among the base clusters. The first consensus consists of 10 clusters which correspond to the instance lists of the 10 unique FCPs at DT=3. At DT=2, we have 6 FCPs added to the 10 clusters of the previous consensus. Merging intersecting bi-clusters and

removing subsets will result in only 3 clusters which are the true clustering of the dataset according to the class labels, as shown in figure 4. At DT=1, we have 2 FCPs added to the 3 clusters from previous step, and after merging intersecting bi-clusters we will end up with only 1 cluster.

Figure 5 shows the tree of consensuses. In this synthetic example, we have only 3 consensuses and, obviously, the one in the middle (orange nodes) is the best. In real tests, the tree will have many levels[6], and the user will have several consensuses to choose from, with respect to the targetted results as explained in section 3.5. Regarding execution time, since our algorithm works in an agglomerative way, with the decrease of DT (after the first consensus) fewer FCPs are processed to build a consensus, as shown in the example. That is, at DT=2, 16 bi-clusters were processed, but at DT=1, only 5 were processed by lines 9 to 32 of the MultiCons algorithm. Using this approach to build multiple consensuses from a big FCP set ensures thus a low execution time.

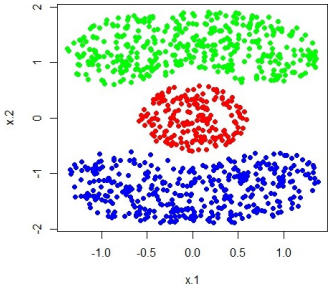


Fig. 4: MultiCons best consensus for Cassini dataset.

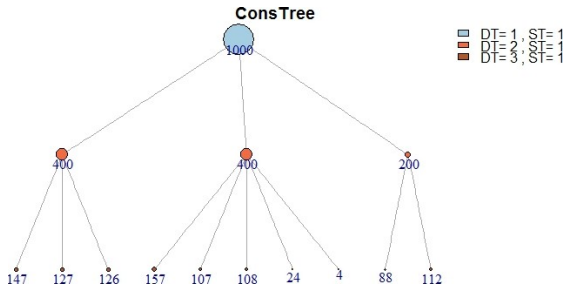


Fig. 5: ConsTree for Cassini dataset.

5 Experiments

We implemented the MultiCons algorithm using R language [16] on a DELL PRECISION M4800 with Intel(R) Core(TM) i7-4710MQ CPU @ 2.50GHz, 32 GB of RAM, and Microsoft Windows 10 Professional (64-bit) operating system. We have used the FIST algorithm implementation in Java from the website of the authors [13]. At each test, the base clusterings are generated by random selection of a set of clustering algorithms and/or parameters settings. Among the clustering algorithms used: K-means, PAM, CLARA, DBSCAN, agglomerative hierarchical clustering, AGNES, DIANA, MCLUST (Gaussian Model-Based Clustering), C-Means, FANNY, Bagged Clustering, and SOM. We compared the results of MultiCons against the consensus clustering algorithms available in R package CLUE [9], including the following consensus methods: SE, GV1, DWH, HE,

[6] The maximum depth of the tree equals the number of base clusterings used.

SM, GV3, soft/symdiff, and consensus medoid. To validate the results of our consensus method and the CLUE methods, we compared the clustering results against the true class labels of the tested dataset using several external validation measures like NMI (Normalized Mutual Information) [17], Jaccard, cRand (Corrected Rand), and FM (Fowlkes and Mallows) [5,8] available also in the CLUE package [10]. However, in real life, true class labels may not exist, and the evaluation of the results may depend on the ability to extract meaningful patterns from the generated clusters. Hence, we provide ConsTree to ease this process.

Continuing the synthetic example in section 4, table 2 shows the validation of the different base clusterings and consensus methods. MultiCons results are named Cons 1, Cons 2, and Cons 3. You can see that even when the base clusterings produce low quality results (caused by bad selection of the clustering algorithm to discover the clustering pattern in the dataset, or by bad setting of the algorithm parameters), consensus clustering can achieve better quality clustering.

Table 2: Validation of the synthetic example.

Algorithm	Jaccard	NMI	cRand	FM
k-means	0.6537341	0.6499608	0.6401733	0.8013933
Hierarchical	0.7791699	0.8904783	0.8188679	0.8827060
MClust	0.4104735	0.7397994	0.4714941	0.6406821
Cons 1	0.3593594	NaN	0.0000000	0.5994659
Cons 2	1.0000000	1.0000000	1.0000000	1.0000000
Cons 3	0.3377604	0.7011586	0.3952182	0.5811716
SE	0.7040477	0.7797105	0.7328262	0.8266073
GV1	1.0000000	1.0000000	1.0000000	1.0000000
DWH	0.5279396	0.5495195	0.5044829	0.6918356
HE	1.0000000	1.0000000	1.0000000	1.0000000
SM	1.0000000	1.0000000	1.0000000	1.0000000
GV3	1.0000000	1.0000000	1.0000000	1.0000000
soft/symdiff	0.4727269	0.2735059	0.4324315	0.6499522
medoids	0.7791699	0.8904783	0.8188679	0.8827060

We present other tests in table 3 presenting a description of the dataset, the base clusterings, and the quality of the achieved consensus results compared to the true class using Jaccard measure. We chose the Jaccard measure because it gives a moderate trade-off between the similarity to the true class and the number of generated clusters. Note that for CLUE consensus methods, we used K equal the number of the true classes of the dataset, while MultiCons does not require K and it generates multiple results with different numbers of clusters.

The datasets Zoo, Smiley, Shapes, Hyper Cube, and Wine are available in the "mlbench" R package [14]. The other datasets are from Ultsch [19] where the author of the datasets declare them as the "Fundamental Clustering Problem Suite". We can see that the MultiCons approach achieved the best results among the other consensus methods, except for Wine dataset where it achieved the second best result.

Table 3: Experimental results.

Dataset	Zoo	Smiley	Shapes	Hyper Cube	Wine	Chain link	Atom	Golf Ball	Hepta	Lsun	Target	Terta
Size	101	500	2000	800	178	1000	800	4002	212	400	770	400
# attributes	16	2	2	3	13	3	3	3	3	2	2	3
# classes	7	4	4	8	3	2	2	1	7	3	6	4
# base clusterings	9	6	8	8	6	6	6	5	7	5	8	7
K range for base clusterings	K = 7	2 to 7	2 to 9	4 to 11	2 to 9	2 to 7	2 to 7	2 to 6	4 to 10	2 to 6	3 to 10	2 to 8
Base min quality	0.41	0.40	0.46	0.41	0.32	0.24	0.46	0.17	0.36	0.47	0.33	0.50
Base max quailty	**0.87**	0.74	**1**	**1**	0.69	**1**	**1**	0.50	**1**	**1**	**1**	**1**
Best MultiCons result	0.84	**1**	**1**	**1**	0.79	**1**	**1**	**1**	**1**	**1**	**1**	**1**
# clusters in best MultiCons	5	4	4	8	9	2	2	1	7	3	6	4
SE	0.72	0.59	0.83	0.89	0.59	0.83	0.78	**1**	**1**	0.45	0.64	0.98
GV1	0.76	0.92	0.66	**1**	0.41	0.59	0.56	**1**	**1**	0.98	0.71	**1**
DWH	0.77	0.56	0.98	0.91	0.55	0.53	0.84	**1**	0.96	0.45	0.70	**1**
HE	0.77	0.78	0.84	**1**	0.58	0.64	0.60	**1**	**1**	0.45	0.67	**1**
SM	0.77	0.75	**1**	0.80	0.59	0.65	0.91	**1**	**1**	0.48	0.67	0.99
GV3	0.74	0.76	**1**	**1**	**0.87**	0.92	**1**	**1**	**1**	**1**	0.72	**1**
soft/symdiff	0.76	0.57	**1**	**1**	0.62	0.92	**1**	**1**	**1**	0.47	0.71	**1**
medoids	0.70	0.74	0.87	**1**	0.69	0.35	0.57	0.25	**1**	0.59	0.72	**1**

6 Conclusions

We presented a new multiple consensus clustering method that require no parameters, yet it can build the correct number of clusters based on finding clustering patterns common in a set of base clusterings. Other consensus methods require at least the parameter K (like in CLUE methods) to generate K clusters in the consensus. Without a prior domain knowledge, it is difficult to predict K, while our method can provide the user with a visualization of how the clusters are generated and from how many base clusterings, thus one can choose easily what are the most relevant clusters. All the generated consensuses can be attached to the original dataset, so that the data analyst can easily compare a set of different clusterings while other methods provide 1 solution and require either to modify

K or the base clusterings in order to provide a new solution, without providing any relation between the different results.

To the best of our knowledge, the proposed method is the first to use frequent closed patterns to detect similarities among base clusterings and to build a consensus from these. One of the benefits of using FCIs technique is execution time: It is not related to the size of the dataset, instead, it is related to the number of base clusterings used and whether there are many similarities or many conflicts between the base clusterings. Thus, even for large datasets, if there is a clustering structure common to most base clusterings, we will have few FCPs, while other methods based on distance matrix are limited by the size of the dataset. With our approach, even when there is a large number of FCPs, MultiCons processes them very fast since at each consensus, it requires only few FCPs to work with. Tests showed that CLUE methods GV3 and soft/symdiff require a lot of both computation and memory compared to MultiCons.

According to our tests, the proposed MultiCons method can produce very good results (the best results for all the datasets tested but one), generating any shape for clusters, by benefiting from the different shapes that the considered base clusterings provide. Thus, we recommend using different clustering algorithms for the base partitionings, with varying K values. The objective of this work is not just to find a consensus clustering, but also to make the clustering task pain-free. That is, you do not need anymore to select carefully the base clusterings or their parameters, which requires some expertise. Instead, the user only needs to build randomly varying clusterings, and the MultiCons approach will recommend several results that can be easily analyzed. Using the ConsTree result of the MultiCons approach, the user can thus detect not only the strong clusters, but also the outliers, and the borders of very close clusters for which it is difficult to decide to which cluster they should belong. Such instances will be recognized as singleton clusters or small clusters that do not merge with others on several successive levels in the ConsTree.

Our approach is currently used for small store consumer profiling and it is pretty successful to identify very relevant consumer groups and in highlighting peculiar groups not visible by any other clustering approach.

References

1. Agrawal, R., Srikant, R.: Fast algorithms for mining association rules in large databases. In: Proc. VLDB Conf. pp. 487–499 (1994)
2. Asur, S., Ucar, D., Parthasarathy, S.: An ensemble framework for clustering protein–protein interaction networks. Bioinformatics 23(13), i29–i40 (2007)
3. Caruana, R., Elhawary, M., Nguyen, N., Smith, C.: Meta clustering. In: Proc. IEEE ICDM Conf. pp. 107–118 (2006)
4. Ceglar, A., Roddick, J.F.: Association mining. ACM Computing Surveys 38(2) (2006)
5. Dalton, L., Ballarin, V., Brun, M.: Clustering algorithms: on learning, validation, performance, and applications to genomics. Current Genomics 10(6), 430 (2009)
6. Dimitriadou, E., Weingessel, A., Hornik, K.: A cluster ensembles framework. IOS Press (2003)

7. Ghaemi, R., Sulaiman, M.N., Ibrahim, H., Mustapha, N.: A survey: clustering ensembles techniques. World Academy of Science, Engineering and Technology 50, 636–645 (2009)
8. Halkidi, M., Batistakis, Y., Vazirgiannis, M.: On clustering validation techniques. Journal of Intelligent Information Systems 17(2), 107–145 (2001)
9. Hornik, K.: A CLUE for CLUster Ensembles. Journal of Statistical Software 14(12) (2005), http://www.jstatsoft.org/v14/i12/
10. Hornik, K.: CLUE: Cluster ensembles (2015), http://CRAN.R-project.org/package=clue, r package version 0.3-50.
11. Li, T., Ding, C.: Weighted consensus clustering. In: Proc. SIAM Conf. on Data Mining. pp. 798–809 (2008)
12. Mirkin, B.: Mathematical classification and clustering: From how to what and why. In: Classification, Data Analysis, and Data Highways. Springer (1998)
13. Mondal, K.C., Pasquier, N., Mukhopadhyay, A., Maulik, U., Bandhopadyay, S.: A new approach for association rule mining and bi-clustering using formal concept analysis. In: Machine Learning and Data Mining in Pattern Recognition, pp. 86–101. Springer (2012)
14. Newman, D., Hettich, S., Blake, C., Merz, C.: UCI repository of machine learning databases (1998), http://www.ics.uci.edu/~mlearn/MLRepository.html
15. Pasquier, N., Bastide, Y., Taouil, R., Lakhal, L.: Efficient mining of association rules using closed itemset lattices. Inf. Systems 24(1), 25–46 (1999)
16. R Development Core Team: R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing, Vienna, Austria (2008), http://www.R-project.org
17. Strehl, A., Ghosh, J.: Cluster ensembles – a knowledge reuse framework for combining multiple partitions. JMLR 3, 583–617 (2003)
18. Topchy, A., Law, M., Jain, A., Fred, A.: Analysis of consensus partition in cluster ensemble. In: Proc. IEEE ICDM Conf. pp. 225–232 (2004)
19. Ultsch, A.: Clustering with SOM: U*C. In: Proc. WSOM Workshop. pp. 75–82 (2005)
20. Vega-Pons, S., Ruiz-Shulcloper, J.: A survey of clustering ensemble algorithms. IJPRAI 25(03), 337–372 (2011)
21. Xu, R., Wunsch, D.: Survey of clustering algorithms. IEEE Trans. on Neural Networks 16(3), 645–678 (2005)
22. Yang, G.: The complexity of mining maximal frequent itemsets and maximal frequent patterns. In: ACM SIGKDD. pp. 344–353 (2004)
23. Zhang, Y., Li, T.: Consensus clustering + meta clustering = multiple consensus clustering. In: Proc. FLAIRS Conf. (2011)