



A closed patterns-based approach to the consensus clustering problem

Atheer Al-Najdi

► To cite this version:

| Atheer Al-Najdi. A closed patterns-based approach to the consensus clustering problem. Other [cs.OH]. Université Côte d'Azur, 2016. English. NNT : 2016AZUR4111 . tel-01478626

HAL Id: tel-01478626

<https://theses.hal.science/tel-01478626>

Submitted on 28 Feb 2017

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

École doctorale STIC
Unité de recherche : I3S (UMR 7271 UNS/CNRS)

Thèse de doctorat

Présentée en vue de l'obtention du
grade de docteur en Informatique
de
L'UNIVERSITE COTE D'AZUR

par
Atheer AL-NAJDI

A Closed Patterns-based Approach to the Consensus Clustering Problem

Dirigée par Frédéric Precioso
et co-encadrée par Nicolas Pasquier

Soutenue le 30/11/2016
Devant le jury composé de :

Andrea	Tettamanzi	Professeur, Université Cote d'Azur	Examineur
Arnaud	Revel	Professeur, Université de La Rochelle	Examineur
Frédéric	Precioso	Professeur, Université Cote d'Azur	Directeur de thèse
Karell	Bertet	Maître de conférence, Université de La Rochelle	Rapporteur
Nicolas	Pasquier	Maître de conférence, Université Cote d'Azur	Co-encadrant
Philippe	Fournier-Viger	Professeur, Harbin Institute of Technology (Chine)	Rapporteur
Sanghamitra	Bandyopadhyay	Indian Statistical Institute (Inde)	Examineur

Abstract

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. 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, a new consensus clustering method, called MultiCons, is proposed. It uses the *frequent closed itemset* mining technique in order to discover the similarities between the different base clustering solutions. The identified similarities are presented in a form of *clustering patterns*, that each defines the agreement between a set of base clusters in grouping a set of instances. By dividing these patterns into groups based on the number of base clusters that define the pattern, MultiCons generates a consensus solution from each group, resulting in having multiple consensus candidates. These different solutions are presented in a tree-like structure, called ConsTree, that facilitates understanding the process of building the multiple consensus, and also the relationships between the data instances and their structuring in the data space.

Five consensus functions are proposed in this work in order to build a consensus solution from the clustering patterns. Approach 1 is to just merge any intersecting clustering patterns. Approach 2 can either merge or split intersecting patterns based on a proposed measure, called *intersection ratio*. Approach 3 differs from the previous approaches by searching for the best similar pattern before making a merge/split decision, and, in addition, it uses the *average intersection ratio*. While approach 3 works sequentially on the clustering patterns, approach 4 uses a similarity matrix of intersection ratios to search for the best merge/split. Approach 5 is a simple graph partitioning process to build clusters of clustering patterns. These five approaches are tested with many benchmark datasets to compare their performance on different clustering problems.

Keywords: Clustering; Unsupervised learning; Consensus clustering; Clusterings ensemble; Frequent closed itemsets.

Résumé

Le *clustering* est le processus de partitionnement d'un ensemble de données en groupes, de sorte que les instances du même groupe sont plus semblables les unes aux autres qu'avec celles de tout autre groupe. De nombreux algorithmes de clustering ont été proposés, mais aucun d'entre eux ne s'avère fournir une partition des données pertinente dans toutes les situations. Le *clustering par consensus* vise à améliorer le processus de regroupement en combinant différentes partitions obtenues à partir de divers algorithmes afin d'obtenir une solution de consensus de meilleure qualité. Dans ce travail, une nouvelle méthode de clustering par consensus, appelée MultiCons, est proposée. Cette méthode utilise la technique d'extraction des itemsets fréquents fermés dans le but de découvrir les similitudes entre les différentes solutions de clustering dits de base. Les similitudes identifiées sont représentées sous une forme de motifs de clustering, chacun définissant un accord entre un ensemble de clusters de bases sur le regroupement d'un ensemble d'instances. En traitant ces motifs par groupes, en fonction du nombre de clusters de base qui définissent le motif, la méthode MultiCons génère une solution de consensus pour chaque groupe, générant par conséquent plusieurs consensus candidats. Ces différentes solutions sont ensuite représentées dans une structure arborescente appelée arbre de consensus, ou *ConsTree*. Cette représentation graphique facilite la compréhension du processus de construction des multiples consensus, ainsi que les relations entre les instances et les structures d'instances dans l'espace de données.

Cinq approches de clustering par consensus, permettant de construire une solution de consensus à partir des motifs de clustering, sont proposées dans ce travail. La première approche fusionne simplement successivement tous les motifs de clustering qui se recoupent. La seconde approche va soit fusionner, soit diviser les motifs qui se recoupent selon le résultat d'une nouvelle mesure appelée *ratio d'intersection*. La troisième approche diffère des approches précédentes en recherchant, pour chaque motif, le motif le plus similaire parmi ceux qui se recoupent avant de faire une fusion ou division ; de plus, cette approche utilise la mesure du *ratio moyen d'intersection* afin de décider de fusionner ou diviser les motifs. Alors que la troisième approche traite les motifs de clustering séquentiellement, la quatrième approche utilise une matrice de similarité des ratios d'intersection pour rechercher la meilleure fusion ou division. La cinquième approche se base sur un processus de partitionnement de graphe afin de créer des regroupements de motifs de clustering. Les expérimentations qui ont menées avec ces cinq approches concernent de nombreux ensembles de données utilisés usuellement pour les comparaisons de performances d'approches traitant divers problèmes de clustering.

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

Acknowledgments

I offer my sincere gratitude to my supervisors, Frédéric PRECIOSO and Nicolas PASQUIER, for supporting me throughout this research and providing me invaluable feedbacks, comments and suggestions. Their strong belief in the potential of this research and their continuous encouragement has been a tremendous influence towards the completion of this work.

I would like to express my appreciation to the reviewers, for their very helpful comments and suggestions. Also, I would like to express my gratefulness to the members of the jury for agreeing to participate to the presentation.

Many thanks to my colleagues for their help, and for the interesting discussions we made: Yoann BERTRAND, Romaric PIGHETTI, Stéphanie LOPEZ, Katy BLANC, Ameni BOUAZIZ, Mélanie DUCOFFE, and Lucas MALLEUS.

Finally, my thanks and deepest gratitude, I address to my dear family, who supported and encouraged me during my years of study.

Contents

1	Introduction	1
1.1	Overview	1
1.2	Motivation	2
1.3	Contribution	4
1.4	Thesis Structure	4
2	Background	5
2.1	Distance Measures	6
2.2	Clustering Algorithms	6
2.2.1	Partitioning-Based	7
2.2.2	Hierarchical Clustering	9
2.2.3	Fuzzy Clustering	12
2.2.4	Density-Based	12
2.2.5	Distribution-Based	13
2.2.6	Graph-Based	13
2.2.7	Grid-Based	15
2.2.8	Spectral Clustering	17
2.3	Clustering Validation Techniques	17
2.3.1	Internal Validation	18
2.3.2	External Validation	21
2.4	Consensus Clustering	24
2.4.1	Graph-Based	24
2.4.2	Relabeling and Voting-Based	27
2.4.3	Co-association Matrix Based	29
2.4.4	Distance-Based	30
2.4.5	Fragment-Based	31
2.4.6	Other Methods	32
3	The Proposed Method	35
3.1	Frequent Pattern Mining	35
3.2	MultiCons	38
3.2.1	The Base Clusterings	39
3.2.2	The Binary Membership Matrix	40
3.2.3	Identifying Clustering Patterns	41
3.2.4	Generating Multiple Consensuses	42
3.2.5	The ConsTree	47
3.3	Example	49

4	Enhancements	55
4.1	A Measure of Similarity Between Instance Sets	57
4.2	MultiCons Approach 2	58
4.3	MultiCons Approach 3	60
4.4	MultiCons Approach 4	62
4.5	MultiCons Approach 5	64
4.6	Enhancing the ConsTree	64
4.7	Default values for MT	66
4.8	Example	66
5	Experiments	75
5.1	Synthetic Datasets	77
5.2	Real Datasets	85
5.3	Summary	90
6	Conclusions and Future Work	93
	Bibliography	95
A	Traduction Française	105
A.1	Aperçu	105
A.2	Motivation	106
A.3	Contribution	108
A.4	Structure de la Thèse	109
A.5	Résumé du Chapitre 2 : Contexte	109
A.6	Résumé du Chapitre 3 : Méthode proposée	113
A.7	Résumé du Chapitre 4 : Améliorations	116
A.8	Résumé du Chapitre 5 : Expérimentations	118
A.9	Conclusions et travaux futur	119

List of Figures

1.1	The KDD process as explained in [Fayyad 1996].	2
1.2	Example dataset where the instances form 2 clusters.	3
2.1	Simple illustration of the difference between single and complete linkage clustering [Aggarwal 2013].	11
3.1	Itemset lattice of $\mathcal{D}$, from [Pasquier 1999].	37
3.2	Closed Itemset lattice of $\mathcal{D}$, from [Pasquier 1999].	38
3.3	Process diagram of MultiCons.	39
3.4	Illustration of the relationship between the pattern space and the original data space.	44
3.5	The ConsTree of the running example.	47
3.6	Example of analysis from ConsTree visualization.	50
3.7	The cassini dataset with 8 different base clusterings.	51
3.8	The ConsTree for the cassini dataset.	52
3.9	Another ConsTree for the cassini dataset.	52
4.1	Illustrative example of the relations between instance sets as nodes in a graph.	57
4.2	Examples of sets intersection.	57
4.3	The difference in results between approaches 1 and 2 for the example in table 4.1.	60
4.4	Illustrative example of approach 4.	62
4.5	The idea behind approach 5.	64
4.6	The result of tree refinement.	68
4.7	An example dataset of 5 Gaussian distributions in a 2D features space.	69
4.8	Different clusterings for the dataset in figure 4.7.	70
4.9	The ConsTree of approach 1 for the ensemble in figure 4.8.	71
4.10	The ConsTree of approach 2 for the ensemble in figure 4.8.	71
4.11	The ConsTree of approach 3 for the ensemble in figure 4.8.	72
4.12	The ConsTree of approach 4 for the ensemble in figure 4.8.	72
4.13	The ConsTree of approach 5 for the ensemble in figure 4.8.	73
4.14	The refined ConsTree for the results in figure 4.10.	73
4.15	The recommended consensus solutions of approaches 2, 3, and 4 for the dataset in figure 4.7.	74
5.1	Hepta dataset [Ultsch 2005b].	78
5.2	Lsun dataset [Ultsch 2005b].	79
5.3	Tetra dataset [Ultsch 2005b].	79
5.4	Chainlink dataset [Ultsch 2005b].	80

5.5	Atom dataset [Ultsch 2005b].	80
5.6	EngyTime dataset [Ultsch 2005b].	82
5.7	Target dataset [Ultsch 2005b].	82
5.8	The ConsTree of approach 1 for the Target dataset.	83
5.9	The ConsTree of approach 4 for the Target dataset. We can recognize the stable clusters that represent the noise and the middle core cluster.	83
5.10	TowDiamonds dataset [Ultsch 2005b].	84
5.11	Wingnut dataset [Ultsch 2005b].	84
5.12	The ConsTree of approach 2 for the Iris dataset using the ensemble of the Iris1 test.	88
A.1	Processus d'extraction de connaissances à partir des données [Fayyad 1996].	106
A.2	Exemple d'ensemble de données bi-dimensionnel.	107

List of Tables

3.1	Example transactional dataset from [Pasquier 1999].	36
3.2	The binary matrix representation for the transactional dataset in table 3.1.	36
3.3	An example membership matrix.	41
3.4	Clustering patterns extracted from table 3.3.	42
4.1	Example membership matrix.	56
4.2	Clustering patterns extracted from table 4.1.	56
4.3	Comparison of the quality of the clustering results.	69
5.1	Tests validation table 1.	81
5.2	Execution time table 1 (in seconds).	81
5.3	Tests validation table 2.	85
5.4	Execution time table 2 (in seconds).	86
5.5	Real-world benchmark datasets.	86
5.6	Tests validation table 3.	88
5.7	Execution time table 3 (in seconds).	89
5.8	Tests validation table 4.	90
5.9	Execution time table 4 (in seconds).	91

List of Algorithms

1	K-Means	8
2	PAM	9
3	A general agglomerative clustering	10
4	DBCSAN	13
5	EM for Gaussian distributions	14
6	STING	16
7	Spectral clustering	17
8	In-ensemble similarity	40
9	MultiCons	45
10	Consensus clustering approach 1	46
11	ConsTree	49
12	Consensus clustering approach 2	59
13	Consensus clustering approach 3	61
14	Consensus clustering approach 4	63
15	Consensus clustering approach 5	65
16	Refined ConsTree	67

Introduction

Contents

1.1 Overview	1
1.2 Motivation	2
1.3 Contribution	4
1.4 Thesis Structure	4

1.1 Overview

Data Mining (**DM**) is “*the application of specific algorithms for extracting patterns from data*” [Fayyad 1996]. DM represents the core step of a more general process called the Knowledge Discovery in Databases (**KDD**). KDD involves 5 main steps, as shown in figure A.1: *i*) Selection: of the relevant data for the mining task; *ii*) Pre-processing: of the selected data to be appropriate for deriving correct patterns, like cleaning outlier and missing values; *iii*) Transformation: of the preprocessed data to be ready for the mining task, like standardizing data attribute values; *iv*) Data mining: applying a mining algorithm based on the required discovery task; and finally *v*) Interpretation and/or Evaluation: of the discovered patterns to validate and extract the new knowledge. However, the term data mining is widely used to describe the KDD process instead of being part of it.

The dataset¹ consists of many instances (objects), each defined by a set of attributes. We can represent the dataset as a table, where each row is an instance, and each column is an attribute. The major data mining tasks that can be performed over the dataset to discover new knowledge are:

- **Classification:** The dataset contains an attribute, called the target or the response variable, that categories the instances into different classes. The objective of the classification task is to learn a model (tree, rules, function, ...) to be used to assign a class label to new unlabeled instances. Classification is considered as a supervised learning approach, because it uses domain knowledge (the response variable) to learn how to classify other data.
- **Regression:** Similar to classification, but the response variable of the dataset is a numerical value. Thus, the objective here is to predict a numerical response value for new data.

¹The term *dataset* is widely used in data mining instead of *database*.

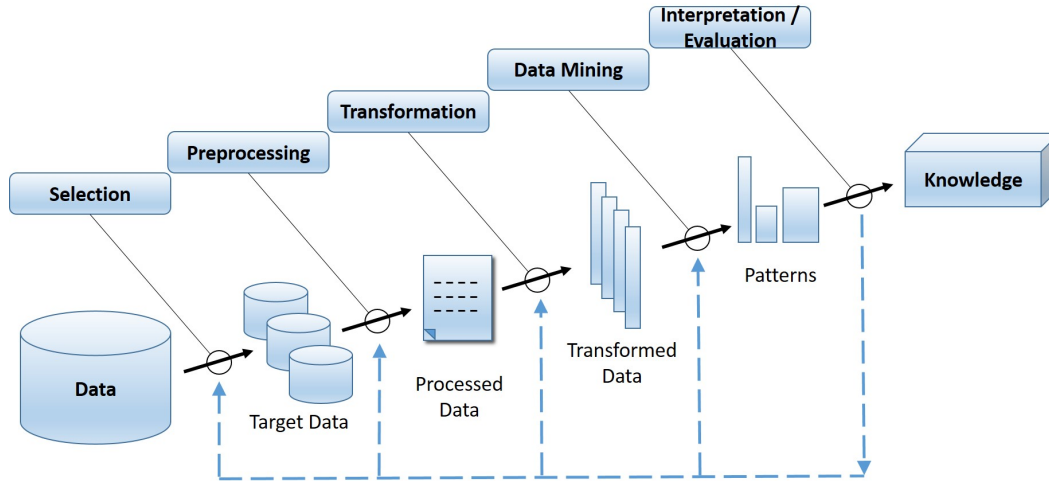


Figure 1.1 – The KDD process as explained in [Fayyad 1996].

- **Clustering:** Unlike classification and regression, clustering is considered as an unsupervised learning approach. The dataset does not have a response variable, and the objective is to group the instances into clusters based on their similarity.
- **Association Rules:** The dataset here is different: It is a transactional dataset, where each instance (called transaction or object) have a set of attribute values called items. The transactions does not necessarily have values for all attributes. The objective is to find relationships between the items in very large datasets. The main application of association rules is in market basket analysis, to discover buying habits of customers. For example, if customers buy cereals and sugar, then they may buy also milk. The result of association rules mining is a set of rules that define such relationships. Like clustering, this data mining task is also unsupervised.

1.2 Motivation

This thesis focuses on the clustering task. As unsupervised process, clustering is very challenging, because we do not have any knowledge on how the instances assemble or how gathering the instances together in the data space. That is, how many natural groups are there, and what are the shapes of these groups. For example, consider the dataset in figure A.2. We can see that the instances (represented by dots) form two dense regions in the 2D data space. However, in real life datasets, we have many dimensions (attributes), making the visualization of the data instances to identify clusters not an option.

Many clustering algorithms were developed in the last 50 years, and, most often, each algorithm produces different partitioning when applied to the same dataset,

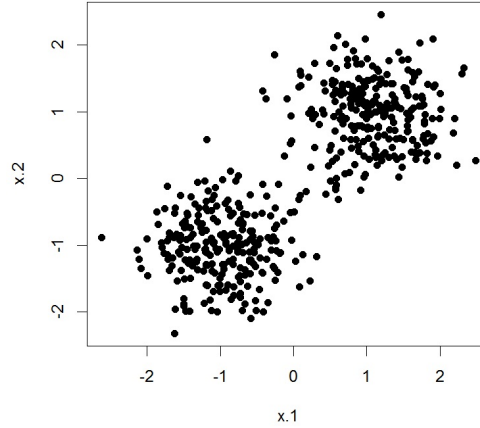


Figure 1.2 – Example dataset where the instances form 2 clusters.

because they are designed to target a specific model (compact clusters, non-convex clusters, etc). 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 parameter 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 [Dalton 2009, Halkidi 2001a]. 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 [Vega-Pons 2011]. In real life, the user can have similar scores for different validation measures and/or for different clustering results, whereas the results are different in many aspects, like in the number of clusters or in the instance 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 discuss some of them in the next chapter. However, some con-

sensus clustering approaches impose limitations on the ensemble, such as all the base clusterings must produce the same number of clusters. Other approaches require consensus function with high storage or time complexities. In this work, we present a new category of consensus clustering methods, that is, a pattern-based consensus generation using the frequent closed itemset (**FCI**) technique from the frequent pattern mining and association rules discovery domain. The advantages of the proposed approach are outlined in the next section.

1.3 Contribution

- No constraints are imposed on the selection of the base clustering algorithms and/or their settings.
- There is no need to specify K for the consensus solution, as the proposed method is able to discover automatically the internal structure of the hidden clusters.
- Using the FCI, the search for the consensus solution is performed on a pattern-based space instead of the data instances space. The pattern space provide high pruning of the instances space, especially for large datasets.
- The process of building the consensus clustering is explained to the data analyst by a Hasse diagram called ConsTree. The ConsTree provides significant demonstration of the relationships between the instances, making understanding the internal structure of the natural clusters possible even with multi-dimensional datasets.
- The proposed method generates multiple consensus candidates, then recommend one of them to the analysts. Visualizing the ConsTree gives the analysts the ability to consider another consensus solution instead of the proposed one, based on their preferences and observations.

1.4 Thesis Structure

The next chapter presents general overview about clustering algorithms, validation techniques, and the categories of the consensus methods. Chapter 3 explains the frequent pattern mining, and the use of the FCI to build the proposed consensus clustering method. Chapter 4 presents other approaches to enhance the proposed method and better deal with the similarities among clustering patterns. Experiments on synthetic and real benchmark datasets are explained in chapter 5. Conclusions and ideas for future work are summarized in chapter 6.

Background

Contents

2.1	Distance Measures	6
2.2	Clustering Algorithms	6
2.2.1	Partitioning-Based	7
2.2.2	Hierarchical Clustering	9
2.2.3	Fuzzy Clustering	12
2.2.4	Density-Based	12
2.2.5	Distribution-Based	13
2.2.6	Graph-Based	13
2.2.7	Grid-Based	15
2.2.8	Spectral Clustering	17
2.3	Clustering Validation Techniques	17
2.3.1	Internal Validation	18
2.3.2	External Validation	21
2.4	Consensus Clustering	24
2.4.1	Graph-Based	24
2.4.2	Relabeling and Voting-Based	27
2.4.3	Co-association Matrix Based	29
2.4.4	Distance-Based	30
2.4.5	Fragment-Based	31
2.4.6	Other Methods	32

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. The three main purposes of clustering are [Jain 2010]: *i*) Data analysis: To gain more insight about the structure of the data, generate hypotheses, or detect anomalies; *ii*) Natural classification: To identify the degree of similarity between data instances; and *iii*) Summarization: of the dataset by cluster prototypes.

Many clustering algorithms were developed during the last 50 years. Each algorithm has a specific model to classify the instances into clusters. Consequently, there is no single algorithm that can provide the best partitioning of the instances for all datasets. Section 2.2 provides an overview of the widely known clustering algorithms. But before going to clustering details, it is necessary to understand how the similarity between instances is measured.

2.1 Distance Measures

To measure the distance (dissimilarity) between two instances X and Y of d numerical attributes, Minkowski distance measure can be used [Gan 2007]:

$$dist(X, Y) = \left(\sum_{j=1}^d |x_j - y_j|^r \right)^{\frac{1}{r}}, \quad r \geq 1$$

When $r = 2$, the Minkowski formula presents the Euclidean distance, which is the widely used measure for numerical data. Manhattan distance (or city-block distance) is achieved when $r = 1$. Note that it is necessary to normalize the attributes before applying Minkowski distance, to prevent the largest-scale attribute from dominating the result [Gan 2007]. The Z-score can be used for this purpose:

$$x'_j = \frac{x_j - \mu_j}{\sigma_j}$$

where μ_j is the mean of all values of attribute j , and σ_j is the standard deviation.

Sometimes, the dataset may have mixed attribute types, thus, a more general formula is the Gower distance coefficient [Gan 2007]:

$$dist_{gower}(X, Y) = \left(\frac{1}{\sum_{j=1}^d W(x_j, y_j)} \sum_{j=1}^d W(x_j, y_j) dist^2(x_j, y_j) \right)^{\frac{1}{2}}$$

where $W(x_j, y_j) = 0$ if one of the instances X or Y has a missing value in attribute j , otherwise $W(x_j, y_j) = 1$. $dist(x_j, y_j)$ is defined according to attribute type as follows [Han 2011, Gan 2007]:

For ordinal and continuous attributes:

$$dist(x_j, y_j) = \frac{|x_j - y_j|}{R_j}, \text{ where } R_j \text{ is the range of the } j\text{th attribute.}$$

For binary and nominal: $dist(x_j, y_j) = 0$ if $x_j = y_j$, otherwise, $dist(x_j, y_j) = 1$.

Other distance measures exist for specific clustering application, for example, cosine similarity which is usually used for document or text clustering. Chapter 6 in [Gan 2007] provide more details on the similarity and distance measures.

2.2 Clustering Algorithms

Clustering algorithms can be categorized according to their underlying model:

- **Partitioning-based:** Also known as centroid-based, as the instances are grouped according to their distance to optimally-found cluster centroids.
- **Hierarchical:** The instances are grouped in a hierarchy of clusters.

- **Fuzzy:** Each instance belongs to all clusters but with different membership degrees.
- **Density-based:** A cluster is a high density region in the data space.
- **Distribution-based:** The instances are grouped into clusters based on fitting a distribution model.
- **Graph-based:** The instances are considered as nodes in a graph, connected by edges based on their similarities. Clusters are formed from partitioning this graph.
- **Grid-based:** The data space is divided into cells of instances, and clusters are formed by grouping the cells.
- **Spectral:** The data space is transformed into eigenvectors space to better recognize the clusters.

2.2.1 Partitioning-Based

The idea behind partitioning-based clustering algorithms is to group the instances in K clusters, each represented by a centroid prototype. It starts by selecting K instances as initial centroids of clusters. Then, the algorithm iterates to update this selection by calculating the distance between the data instances and the centroids, until reaching a convergence criterion. An important parameter for these algorithms is obviously K , which is the number of required clusters. We briefly present representative algorithms of this strategy in the following paragraphs:

K-Means:

K-means [MacQueen 1967, Hartigan 1979] can be considered the widely used clustering algorithm for its simplicity and efficiency. It starts by choosing K instances as initial centroids of K clusters. Then, each other instance is assigned to the closest centroid to which it has the minimum squared distance. After forming the initial clusters, new centroids are calculated as the means (the average) of the instances in each cluster. This process repeats to find the best centroids that minimize the Sum of Squared Error (SSE). The algorithm stops when the centroids stabilize or after a certain number of iterations.

Lets consider a dataset $D = \{x_1, x_2, \dots, x_n\}$ that consist of N instances. The objective of K-means is to partition D into a set of K clusters $C = \{c_1, c_2, \dots, c_k\}$ such that the SSE is minimized [Jain 2010, Aggarwal 2013]:

$$SSE = \sum_{k=1}^K \sum_{x_i \in c_k} ||x_i - \mu_k||^2 \text{ where } \mu_k \text{ is the mean of cluster } c_k.$$

Algorithm 1 describes the process [Han 2011].

The main advantages of K-means are its low time complexity: $O(KNT)$ [Xu 2015], where T is the number of iterations, and its low space complexity: $O(N + K)$

Algorithm 1: K-Means**Input** : D (Dataset); K (number of required clusters)**Output**: Classification of the instances in D into K clusters1 Choose K instances from D as initial centroids2 **repeat**

3 (Re)assign each instance into the cluster with the closest centroid

4 Calculate the mean of each cluster and use it to update the previous centroid

5 **until** *no change in centroids*

[Xu 2005]. However, using the arithmetic mean to define cluster centers, K-means is sensitive to outliers. Other disadvantages include the effect of the selection of the initial centroids and K on the resulted clustering, and the convergence into a local optimum solution [Wu 2008, Xu 2015]. Different variations of K-means exist (check chapter 4 in [Aggarwal 2013]), however, the basic algorithm is still the widely used for its efficiency with large datasets.

PAM:

Partitioning Around Medoids (PAM) [Kaufman 1990] is similar to K-means, but rather than using the means as representatives of cluster centers, PAM uses actual instances, making it less sensitive to outliers. After initializing as in K-Means, PAM then iterates to replace the K centroids with one of the remaining $N - K$ instances. If a replacement minimizes the objective function (the sum of absolute errors SAE instead of SSE), the algorithm swaps the centroid with the new candidate, and performs another iteration. If not, after going through all the K medoids, it stops resulting in convergence into a local optimum [Aggarwal 2013, Han 2011, Miller 2001]. PAM uses a dissimilarity matrix of the actual dataset to enhance the impact of the swap operations on execution time. However, because of its high time complexity, $O(K^3 N^2)$ [Xu 2015], PAM is suitable only for small datasets. Algorithm 2 [Han 2011] explains the process.

CLARA:

To overcome the problem of high execution time in PAM, Clustering LARge Application (CLARA) [Kaufman 1990] algorithm is proposed. CLARA uses random sampling to divide the dataset into S samples, and performs PAM within each sample. The best sample-based clustering is then considered the final clustering for the whole dataset. The efficiency of CLARA depends on sample size. However, a good clustering based on a sample does not necessarily represent the best solution for the whole dataset [Halkidi 2001a]. The time complexity of CLARA is $O(KS^2 + K(N - K))$ [Xu 2015].

Algorithm 2: PAM

Input : D (Dataset); K (number of required clusters)
Output: Classification of the instances in D into K clusters

- 1 Choose K instances from D as initial representatives of cluster medoids
- 2 **repeat**
- 3 Assign each remaining instance into the cluster with the closest medoid
- 4 Randomly choose a non-representative instance (not in the set of medoids) x_r
- 5 Compute the cost (SAE) of swapping x_r with one of the medoids m_j
- 6 if the new cost is minimized compared to the previous cost, swap m_j by x_r
- 7 **until** *no change in medoids*

CLARANS:

Clustering Large Applications based upon RANdomized Search (CLARANS) [Ng 1994] algorithm performs a random selection and replacement of a medoid by a non-medoid instance at each iteration. The replacement happens when it enhances the SAE. This random search process is repeated l times, and the final medoids at the l th iteration are returned as local optimum. CLARANS repeats this process m times and returns the best local optimum [Han 2011].

2.2.2 Hierarchical Clustering

There are two scenarios for hierarchical clustering: Bottom-up and top-down. In the bottom-up approach, also called agglomerative, the clustering process starts by considering each instance as a cluster, then merges the two closest clusters at each iteration until combining all the instances in one cluster. On the other hand, a top-down process, also called a divisive process, starts by considering the entire dataset as one cluster, then at each iteration, it splits a cluster into two, until each instance becomes a stand alone cluster. In both techniques, the partitioning of the instances into clusters in each iteration can be viewed as a level in a binary tree, called a dendrogram. The multiple levels of this dendrogram illustrates the hierarchical clustering process. The final clustering is then determined by choosing the tree level that has the required number of clusters defined by the end-user. Thus, K is not essential for running the algorithm as in partitioning methods, but to decide where to “cut” the tree. The main disadvantages of hierarchical methods are their (at least) quadratic time and storage complexities, and their sensitivity to outliers [Xu 2005]. Some of the algorithms that fall within this category:

Single Linkage:

Single linkage [Sneath 1957] is an agglomerative method. At each iteration, two clusters are merged based on the distance between their two closest instances (the nearest neighbors). The algorithm uses a dissimilarity matrix for maintaining the distance information and updating it based on the merges. The resulted clusters

are non elliptical elongated shaped, which is an advantage compared to partitioning-based methods that produce spherical shaped clusters [Aggarwal 2013]. However, the algorithm is sensitive to outliers, which may result in merging two unrelated clusters because of the existence of some noisy instances between them. Algorithm 3 describes a general agglomerative clustering process [Aggarwal 2015]. The distance between two clusters C and C' in a single linkage approach is calculated as [Gan 2007]:

$D(C, C') = \min_{x \in C, y \in C'} d(x, y)$, where $d(., .)$ is the distance measure used in creating the distance matrix, for example the Euclidean distance.

Algorithm 3: A general agglomerative clustering	
Input	D (Dataset)
Output:	Dendrogram (a hierarchy of clusters)
1	Build a distance matrix M of $N \times N$ cells
2	repeat
3	Pick the closest clusters i and j in M
4	Merge them, that is, $C_k = C_i \cup C_j$
5	Delete the rows/columns i and j from M and add row and column k for the new cluster
6	update the entries of row and column k using the distance function between 2 clusters
7	until <i>until termination criterion</i>

Complete Linkage:

The idea in complete linkage [Sørensen 1948] is the opposite to single linkage. That is, two clusters are merged based on the distance between their furthest instances (farthest neighbors). As this method estimates the diameter of a new cluster by the distance of its furthest instances, and its focus on minimizing the diameter, the resulted clusters are compact, similar sized and spherical shaped. Thus, this method is unable to discover natural clusters of different sizes (large and small clusters) [Aggarwal 2015]. To perform the complete linkage using algorithm 3, the distance between two clusters is calculated as follows [Gan 2007]:

$$D(C, C') = \max_{x \in C, y \in C'} d(x, y)$$

Figure 2.1 from [Aggarwal 2013] explains the difference between the resulted clusters using single and complete linkage methods.

Average Linkage:

This agglomerative method is also known as UPGMA (Unweighted Pair Group

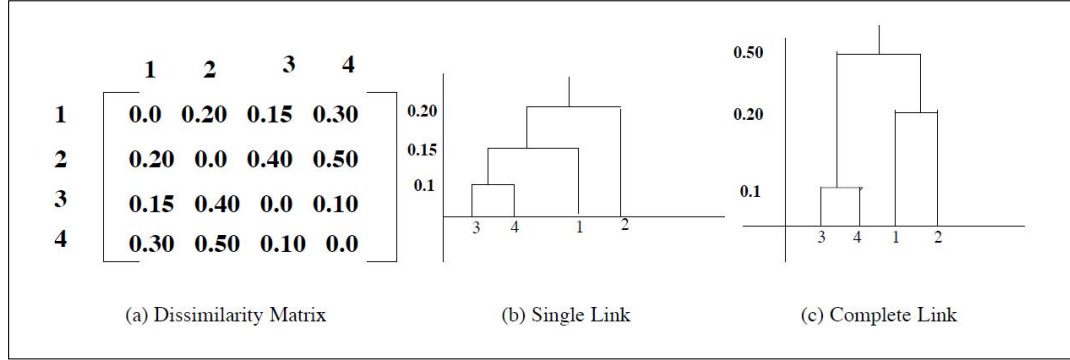


Figure 2.1 – Simple illustration of the difference between single and complete linkage clustering [Aggarwal 2013].

Method with Arithmetic Mean) [Sokal 1958]. The distance between two clusters is calculated as the average (arithmetic mean) of the distance between all possible pairs of instances from each cluster [Gan 2007, Aggarwal 2013]. Resulted clusters are spherical shaped. The distance between two clusters is [Gan 2007]:

$$D(C, C') = \frac{1}{|C||C'|} \sum_{x \in C, y \in C'} d(x, y)$$

Centroid Linkage:

Also known as UPGMC (Unweighted Pair Group Method using Centroids) [Sokal 1958], is another agglomerative method where the distance between two clusters is calculated as [Gan 2007]:

$$D(C, C') = \frac{1}{|C||C'|} \sum_{x \in C, y \in C'} d(x, y) - \frac{1}{2|C|^2} \sum_{x, y \in C} d(x, y) - \frac{1}{2|C'|^2} \sum_{x, y \in C'} d(x, y)$$

DIANA:

DIANA or DIvisive ANAlysis [Kaufman 1990] is a divisive hierarchical clustering algorithm based on the approach proposed in [Macnaughton-Smith 1964]. At each iteration, it tries to split the biggest diameter cluster into two, until reaching each instance as a singleton cluster. To estimate the diameter, DIANA uses the largest dissimilarity between two instances in a cluster [Gan 2007]:

$$diam(C) = \max_{x, y \in C} d(x, y)$$

Resulted clusters are also spherical shaped, and the algorithm is sensitive to outliers.

2.2.3 Fuzzy Clustering

Fuzzy clustering can be considered as special type of partitioning clustering: Instead of assigning each instance to one cluster during the clustering process, a fuzzy algorithm considers each instance as a member in all clusters but with different membership value in the range $[0,1]$. Fuzzy C-Means or FCM [James 1981] is the widely used fuzzy clustering algorithm. Like K-Means, it tries to minimizing the SSE, but it multiplies the squared difference of each instance to the mean by a membership coefficient w that is updated at each iteration:

$$SSE = \sum_{k=1}^K \sum_{x_i \in c_k} w_{xik}^\beta \|x_i - \mu_k\|^2$$
, where $\beta \geq 1$ controls the influence of the membership degree.

The time complexity of FCM is low, $O(N)$ [Xu 2015], which is an advantage to deal with large dataset. However, FCM results are affected by outliers and the initial selection of cluster centers [Xu 2005].

2.2.4 Density-Based

In density-based clustering, a cluster is defined as a dense region in the data space, separated from other clusters by regions which density is sparse. Under this assumption, clusters can take arbitrary shape, instead of the spherical shape that is created by most of the above methods. DBSCAN (Density Based Spatial Clustering of Applications with Noise) [Ester 1996] is the main clustering algorithm for this category. It defines three types of instances in the dataset [Aggarwal 2015] using two parameters: MinPts (minimum number of nearest neighbor instances to form a cluster) and Eps (maximum distance between nearest neighbors) as follows: *i*) Core point: If the instance has MinPts number of neighbors within a radius Eps; *ii*) Border point: If it has less than MinPts neighbors within the radius Eps, but at least one of them is a core point; *iii*) Noise point: otherwise. Next, a connectivity graph is created, where each node correspond to a core point. An edge exist between nodes if and only if they are within a distance Eps from each other. Connected groups of nodes are identified, and a border point is assigned to the group with which it is best connected. This way, clusters are formed, and noise points are considered as outliers [Aggarwal 2015]. The time complexity of DBSCAN is $O(N \log N)$. However, the resulted clustering is very sensitive to its parameters [Xu 2015]. Algorithm 4 summarizes the clustering process [Aggarwal 2015].

Another algorithm that is designed to work with high dimensional data is DEN-CLUE (DENsity based CLUstEring) [Hinneburg 1998]. It uses a Gaussian kernel function to find *density attractor* instances, those that result on a local maximum for the kernel function. To identify significant density attractors, only those with kernel value $\geq \xi$ (noise threshold parameter) are considered [Han 2011]. An instance is called a *density attracted* to a density attractor if they are connected by a path of instances whose distance to each other is within σ value (distance threshold

parameter). Then, an arbitrary shaped cluster is formed from connected attractors (through paths of instances with density $\geq \xi$), plus all the attracted points to the attractors [Aggarwal 2013]. All other instances, that are neither attractors nor attracted, are considered as noise. DENCLUE, like DBSCAN, is efficient with large datasets, as its time complexity is $O(N \log N)$ [Xu 2015]. DENCLUE clusters will be identical to clusters generated by DBSCAN if $\sigma = Eps$ and $\xi = MinPts$ [Aggarwal 2013].

Algorithm 4: DBCSAN

Input : D (Dataset), Eps , $MinPts$

Output: Instances at each connected group as a cluster

- 1 Using Eps and $MinPts$, determine core, border, and noise instances
- 2 Create graph in which core points are connected if they are within a distance Eps from each other
- 3 Find the connected components in the graph
- 4 Assign border points to the component with which it is best connected

2.2.5 Distribution-Based

Also known as model-based clustering. The instances are viewed as generated from a mixture of K probability distributions, each represent a cluster. Hence, we need not only to determine K , but also the parameters of the distribution models that best fit the data [Gan 2007]. In the case of Gaussian distributions, the parameters are the means, covariances, and the prior generative probabilities [Aggarwal 2015]. The EM (Expectation-Maximization) algorithm [Dempster 1977] is the popular method under this category. It starts by initial set of parameters, then it iterates performing two steps: *i*) The E-step, in which it calculates the posterior probability for each instance as being part of each distribution; *ii*) and the M-step, where the maximum likelihood approach is used to determine the values of all the parameters (for the next iteration) that best fits the current assignment of instances to distributions (clusters). Algorithm 5 presents the EM method for the case of Gaussian distributions [Aggarwal 2013].

EM is popular for its simple implementation and robustness to outliers. However, the major disadvantage of EM, as with other clustering algorithms, is its sensitivity to initial parameter selection. In some situations, it converges slowly into a local optimum [Aggarwal 2013, Aggarwal 2015].

2.2.6 Graph-Based

In this approach, the similarity between data instances is represented as a graph, on which the nodes correspond to instances, and the edges correspond to the similarity between them. Chameleon [Karypis 1999b] is one of the well known algorithms that incorporates the graph idea, despite that sometimes it is categorized as a hierar-

Algorithm 5: EM for Gaussian distributions**Input** : D (Dataset); K (number of required clusters)**Output:** Classification of the instances in D into K clusters

- 1 Initialize the mean μ_k^0 , covariance Σ_k^0 , and prior probabilities π_k^0 for the K mixture components (distributions)

- 2 **E-step:** Calculate the responsibilities $\gamma(z_{nk})$ (the posterior probabilities) using:

$$\gamma(z_{nk}) = \frac{\pi_k \mathcal{N}(x_n | \mu_k, \Sigma_k)}{\sum_{j=1}^K \pi_j \mathcal{N}(x_n | \mu_j, \Sigma_j)}$$

where $\mathcal{N}(x | \mu, \Sigma)$ is the Gaussian distribution model defined for a d dimensional vector x by:

$$\mathcal{N}(x | \mu, \Sigma) = \frac{1}{(2\pi)^{d/2}} \frac{1}{|\Sigma|^{1/2}} \exp\left\{-\frac{1}{2}(x - \mu)^T \Sigma^{-1}(x - \mu)\right\}$$

- 3 **M-step:** Update the parameters using:

$$\mu_k = \frac{\sum_{n=1}^N \gamma(z_{nk}) x_n}{\sum_{n=1}^N \gamma(z_{nk})} \quad \Sigma_k = \frac{\sum_{n=1}^N \gamma(z_{nk}) (x_n - \mu_k)(x_n - \mu_k)^T}{\sum_{n=1}^N \gamma(z_{nk})}$$

$$\pi_k = \frac{\sum_{n=1}^N \gamma(z_{nk})}{N}$$

And compute the maximized log-likelihood with the updated parameters:

$$P(x_n | \pi, \mu, \Sigma) = \sum_{k=1}^K \pi_k \mathcal{N}(x_n | \mu_k, \Sigma_k)$$

- 4 Repeat steps 2 and 3 until convergence condition is met

chical clustering method. Using a distance matrix, Chameleon constructs a sparse graph on which an edge exists between two vertices if they are within the k -nearest neighbors to each other. As edge weights correspond to the similarity between the instances, a minimum cuts approach is used to partition the sparse graph into many small compact sub-clusters. The idea is to remove (cut) the edges such that the sum of weights of the removed edges is minimized. This process of graph partitioning continues until reaching a MinSize parameter for cluster size [Karypis 1999b]. The next step is to merge the sub-clusters in an agglomerative process based on their similarity. The similarity between two sub-clusters C_i and C_j is measured using

their Relative Interconnectivity (RI) and Relative Closeness (RC):

$$RI(C_i, C_j) = \frac{|EC|_{\{C_i, C_j\}}}{\frac{1}{2}(|EC_{C_i}| + |EC_{C_j}|)}$$

where $EC_{\{C_i, C_j\}}$ is the edge-cut of the cluster that previously contained C_i and C_j , whereas EC_{C_i} (EC_{C_j}) is the minimum sum of the cut edges to bisect C_i (C_j) into two clusters [Han 2011, Karypis 1999b].

$$RC(C_i, C_j) = \frac{\bar{S}_{EC_{\{C_i, C_j\}}}}{\frac{|C_i|}{|C_i| + |C_j|} \bar{S}_{EC_{C_i}} + \frac{|C_j|}{|C_i| + |C_j|} \bar{S}_{EC_{C_j}}}$$

where $\bar{S}_{EC_{\{C_i, C_j\}}}$ is the average weight of the edges that connect C_i with C_j , and $\bar{S}_{EC_{C_i}}$ ($\bar{S}_{EC_{C_j}}$) is the average weight of the edges that belong to the min-cut bisector of C_i (C_j) [Han 2011, Karypis 1999b].

Two clusters C_i and C_j are merged if they maximize $RI(C_i, C_j) * RC(C_i, C_j)^\alpha$, where α is a user specified parameter that controls the importance of clusters closeness on the merging decision [Karypis 1999b].

By using RI and RC , Chameleon apply a dynamic model to assess the similarity between clusters, rather than using a static linkage technique like in other hierarchical methods. Hence, this algorithm is capable of recognizing complex cluster structures. It is also robust against outliers. On the other hand, its time complexity can reach $O(N^2)$ in the worst case [Aggarwal 2013, Han 2011].

2.2.7 Grid-Based

The idea here is to map instances into cells in a grid, where each cell summarizes several instances. STING (STatistical INformation Grid-based) [Wang 1997] clustering method divides the features space into rectangular cells, stacked in a hierarchy of grids with different resolutions. Let the root of the hierarchy is level 1 that consist of 1 cell, then its children are in level 2, and so on. In STING, the number of children for a cell in a level is fixed to 4 (except for the leaves). Each child correspond to 25% of its parent cell, and a parent is simply the union of its children. A cell in this hierarchy is defined by a number of parameters [Aggarwal 2013]:

n : number of instances that belong to the cell.

m : mean of each dimension.

std : standard deviation of each dimension.

min : minimum value of each dimension.

max : maximum value of each dimension.

$dist$: the distribution of instances in the cell (normal, uniform, exponential, or none). This parameter either fixed by the user, or obtained using a hypothesis test such as χ^2 .

These parameter values are easily calculated for the parent cells from its children. After building the hierarchy of cells with their statistical information, STING is able

to answer SQL-like queries from the user to impose some restrictions on the returned results. For example, “*Select the maximal regions that have at least 100 houses per unit area and at least 70% of the house prices are above \$400K*” [Wang 1997]. Algorithm 6 describes how STING processes queries [Aggarwal 2013, Gan 2007].

The major advantage of STING is its efficiency: it passes over the dataset once to compute the statistical parameters, thus it has $O(N)$ time complexity for building the cells. Query processing also has a low time complexity: $O(g)$, where g is the number of cells in the leaves level, which is usually much smaller than N . In addition, the grid structure enables parallel processing and incremental updating. However, the quality of STING results depends on the granularity of the leaves level [Han 2011].

Algorithm 6: STING

```

1 Construct the grid hierarchy and calculate cell parameters
2 Determine a level from which to start finding an answer for the query
3 For each cell in this level, calculate the confidence range of the probability
  that this cell is relevant to the query
4 Label the cells as relevant or not-relevant
5 if the current level is not the leaves level then
6   | Move down to the next level in the hierarchy. Go to step 3 and check only
   | the children of the relevant cells
7 else if the query is met then
8   | Find the regions of the relevant cells. Return the regions that meet the
   | requirements of the query
9 else
10  | Retrieve the instances that fall within the relevant cells. Do further
   | processing and return the results that meet the requirements of the query
11 end
```

Another grid-based method is CLIQUE (CLustering In QUEst) [Agrawal 1998] which is designed to find clusters in high-dimensional datasets. It also divides the features space into rectangular cells (or units) using a threshold ξ . Then, using another density threshold τ , it identifies dense units, those that have a number of instances assigned to them $\geq \tau$. A cluster of arbitrary shape is formed from connected dense units [Aggarwal 2013, Han 2011]. This algorithm is known as a subspace clustering algorithm, as it finds clusters in dense subspaces.

An interesting grid-based clustering algorithm is WaveCluster [Sheikholeslami 1998]. It assigns the instances into cells formed from binning the features spaces, then apply a wavelet transformation on the cells. The idea is that clusters can be better recognized in the transformed space. WaveCluster has many advantages: low time complexity $O(N)$ for low dimensional data, insensitive to outliers, and the ability to find arbitrary-shaped clusters [Aggarwal 2013]. However, WaveCluster is not suitable for high dimensional data, as its time complexity may become exponential [Berkhin 2006].

2.2.8 Spectral Clustering

Instead of clustering dataset instances according to their similarities based on the feature space, spectral clustering methods try to find similarities on a smaller space of eigenvectors. The process involves three main steps [Aggarwal 2013]: *i)* Construct a graph to identify the similarity between dataset instances using an affinity matrix; *ii)* Transform these similarities into another space using the eigenvectors of the graph Laplacian; *iii)* Finally, use a partitioning clustering algorithm like K-Means to find clusters in the space of eigenvectors. Ng *et al.* proposed in [Ng 2002] an algorithm, shown in algorithm 7, to perform normalized spectral clustering.¹ One of the advantages of spectral clustering is its ability to recognize arbitrary shaped clusters [Aggarwal 2013]. In addition, it helps in reducing the dimensionality of the data while preserving hidden cluster structure. However, computing the eigenvectors on a large matrix is costly [Han 2011], making it one of the slowest clustering techniques. Therefore, it is not suitable for large datasets. Another disadvantage is the sensitivity of the results to the scaling parameter [Xu 2015].

Algorithm 7: Spectral clustering

Input : D (Dataset), K (number of required clusters)

Output: Classification of the instances in D into K clusters

1 Build the affinity matrix A defined as:

$$A_{ij} = \exp\left(\frac{-||x_i - x_j||^2}{2\sigma^2}\right), \forall i \neq j, A_{ii} = 0, \text{ where } \sigma^2 \text{ is a scaling parameter}$$

2 Let G be a diagonal matrix whose $[i, i]$ element is the sum of i^{th} row in A , and construct matrix L such that: $L = G^{-1/2}AG^{-1/2}$

3 Find the K largest eigenvectors $f_1, f_2, \dots, f_k$ of the graph Laplacian L and form the matrix F of size $N \times K$ by stacking the eigenvectors in columns

4 Build matrix Y to re-normalize the rows of F : $Y_{ij} = \frac{F_{ij}}{\sqrt{\sum_j F_{ij}^2}}$

5 Cluster Y using K-Means

6 Assign instance x_i to cluster k if and only if row i in Y belongs to cluster k

2.3 Clustering Validation Techniques

It is obvious that the results of all clustering algorithms discussed in the previous section are affected by their parameter settings. Changing these parameter settings will produce different grouping of the instances in clusters, and different number of clusters. So, for a single clustering algorithm, how to know that changing the parameters enhanced or reduced the quality of the resulting clusters? How to know the correct number of the hidden clusters in the dataset? In addition, as we have

¹More spectral clustering techniques are discussed in chapter 8 of [Aggarwal 2013].

different categories of clustering algorithms, how to choose a specific algorithm if we do not know the internal structure of the hidden clusters? Thus, we need also to compare the results produced from different algorithms to identify which one is better than the others. To answer these questions, several clustering validation measures were proposed. These measures can be categorized into two main categories based on the validation technique performed, as presented in the two following subsections.

2.3.1 Internal Validation

Internal validation methods analyze the properties of the resulting clusters, like their separation and compactness [Dalton 2009]. Below are some of the validation measures that fall within this category:

Dunn index:

Dunn index [Dunn 1974] compares the separation between clusters (expressed by the minimum distance between them) to the compactness of the clusters (expressed by the maximum diameter of a cluster) [Aggarwal 2013, Cichosz 2014, Rendón 2011]. It is calculated as follows [Legány 2006]:

$$D = \min_{i=1\dots K} \left\{ \min_{j=i+1\dots K} \left(\frac{d(c_i, c_j)}{\max_{k=1\dots K} \text{diam}(c_k)} \right) \right\}$$

where:

$$d(c_i, c_j) = \min_{x \in c_i, y \in c_j} \{d(x, y)\}$$

$$\text{diam}(c_k) = \max_{x, y \in c_k} \{d(x, y)\}$$

This measure is useful to find the best number of clusters (K), as a higher index value means better clustering [Halkidi 2001a]. However, it is very sensitive to outliers (as maximum cluster diameter can be large in noisy data) [Aggarwal 2013, Legány 2006].

Davies-Bouldin (DB):

The DB index [Davies 1979] is based on measuring the similarity between two clusters in terms of their dispersion, relative to the distance between them. The dispersion S of a cluster C can be defined as the average distance of all its instances to cluster center [Gan 2007]:

$$S = \left(\frac{1}{|C|} \sum_{x \in C} d^p(x, \mu) \right)^{\frac{1}{p}}, \text{ where } p > 1 \text{ and } \mu \text{ is cluster center}$$

Satisfying that: $S \geq 0$ and $S = 0$ if and only if $x = y, \forall x, y \in C$

The similarity R_{ij} between two clusters C_i and C_j must satisfy the following conditions [Gan 2007]:

$$\begin{aligned} R_{ij} &\geq 0; \\ R_{ij} &= R_{ji}; \\ R_{ij} &= 0 \text{ if and only if } S_i = S_j; \\ \text{If } S_j = S_k \text{ and } D_{ij} < D_{ik}, &\text{ then } R_{ij} > R_{ik}; \\ \text{If } S_j > S_k \text{ and } D_{ij} = D_{ik}, &\text{ then } R_{ij} > R_{ik}; \end{aligned}$$

Where S_i, S_j, S_k are the dispersion measures of clusters C_i, C_j, C_k , respectively, and D_{ij} is the distance between clusters C_i and C_j , which can be defined as the distance between their centers. Then, R_{ij} is calculated as:

$$R_{ij} = \frac{S_i + S_j}{D_{ij}}, \text{ and the DB index is defined as:}$$

$$DB = \frac{1}{K} \sum_{i=1}^K R_i, \text{ where } R_i = \max_{i \neq j} R_{ij}$$

As DB represents the average similarity between clusters, minimum DB value reflect a better clustering, especially when comparing different clusterings resulted from applying different K values [Halkidi 2001a]. Unlike Dunn, DB is less sensitive to noise [Aggarwal 2013].

SD index:

SD index [Halkidi 2000] focuses on the average scattering of clusters and the total separation between them. The average scattering of all clusters is defined based on calculating the variance of the clusters ($\sigma(C_i)$) to the variance of the dataset ($\sigma(D)$), as shown below [Halkidi 2001a, Legány 2006]:

$$Scat = \frac{1}{K} \sum_{k=1}^K \frac{||\sigma(C_i)||}{||\sigma(D)||}$$

Low value of $Scat$ means that the clusters are compact [Legány 2006]. The total separation between clusters is defined based on the distances between their centers (μ) as shown below [Halkidi 2001a, Legány 2006]:

$$Dis = \frac{\max_{i,j=1 \dots K} (||\mu_i - \mu_j||)}{\min_{i,j=1 \dots K} (||\mu_i - \mu_j||)} \sum_{k=1}^K \left(\sum_{z=1}^K ||\mu_k - \mu_z|| \right)^{-1}$$

Now, the SD index is:

$$SD = \alpha Scat + Dis$$

where α is a weighting factor equal to Dis in the case of maximum number of clusters [Halkidi 2001a, Legány 2006].

Low SD value reflect a better clustering in terms of compact and well-separated clusters [Legány 2006] when comparing different clusterings of different K values.

S_Dbw index:

The S_Dbw index [Halkidi 2001b] is similar to SD index in validating a clustering result based on the compactness and the separation of clusters. But, instead of calculating the distance between clusters, S_Dbw uses the density between clusters defined as [Gan 2007]:

$$Dens_bw = \frac{1}{K(K-1)} \sum_{i=1}^K \left(\sum_{j=1, j \neq i}^K \frac{density(C_i \cup C_j)}{\max\{density(C_i), density(C_j)\}} \right)$$

where the density of a cluster is defined as:

$$density(C) = \sum_{i=1}^{|C|} f(x_i, \mu)$$

where μ is cluster center, and the function $f(x, u)$ is defined as:

$$f(x, u) = \begin{cases} 0 & \text{if } d(x, u) > Std \\ 1 & \text{otherwise} \end{cases}$$

where Std is the average standard deviation of clusters:

$$Std = \frac{1}{K} \sqrt{\sum_{i=1}^K ||\sigma(C_i)||}$$

To calculate the density of $C_i \cup C_j$, the center is considered the middle point in the line between the centers of the two clusters. Now, the S_Dbw measure is defined as:

$$S_Dbw = Scat + Dens_bw$$

where $Scat$ is estimated similarly as in SD index. Like SD, lower S_Dbw value indicates better clustering. The advantage of S_Dbw over SD is its ability to identify non well-separated clusters [Aggarwal 2013].

Silhouette index:

The silhouette index [Rousseeuw 1987] tries to identify, for each instance, if it belongs to the correct cluster. Let $a(x)$ be the average distance between instance x to all instances within its cluster C_i , which is calculated as follows [Han 2011]:

$$a(x) = \frac{\sum_{y \in C_i, y \neq x} d(x, y)}{|C_i| - 1}$$

And let $b(x)$ be the minimum average distance between $x \in C_i$ and all other clusters to which x does not belong [Han 2011]:

$$b(x) = \min_{C_j: 1 \leq j \leq K, j \neq i} \left\{ \frac{\sum_{y \in C_j} d(x, y)}{|C_j|} \right\}$$

Then the silhouette of x is defined as:

$$S(x) = \frac{b(x) - a(x)}{\max\{a(x), b(x)\}}$$

The value of S is within the range $[-1, 1]$, where a value close to -1 means that x belongs to the wrong cluster, whereas a value close to 1 means that it belongs to the correct cluster. If $S = 0$, then this means that it is not clear where x should belong [Han 2011, Kaufman 1990]. From $S(x)$, we can calculate the average silhouette for each cluster, and hence the average silhouette for the whole clustering result [Dalton 2009].

Other internal measures include: The cophenetic correlation coefficient (CPCC) [Sokal 1962] to assess the quality of a dendrogram generated from a hierarchical clustering method. The Calinski-Harabasz index (CH) [Calinski 1974] that uses the average between- and within-cluster sum of squares to estimate the quality of clustering. The Xie-Beni index (XB) [Xie 1991] which is useful for determining the optimal number of clusters K in fuzzy clustering results. All these measures have different performance in recognizing the best clustering considering different clustering problems, like the existence of noise, variable densities, and non well-separated clusters (very close clusters). Chapter 23 in [Aggarwal 2013] presents an extensive survey of the performance of the internal validation measures regarding these issues. One drawback of using these measures is that they overrate the algorithm that uses the same clustering model. For example, a distance-based measure assigns high score to K-means because it naturally optimizes this distance. In addition, high scores do not necessarily reflect good results in terms of effective information retrieval application [Delen 2014, Manning 2008].

2.3.2 External Validation

External measures evaluate clustering results by comparing it against class labels predefined by domain experts [Delen 2014]. The name ‘external’ is from the fact that these classes are not part of the data used in the clustering process. Below are some of the widely used external measures:

Rand index:

Rand index [Rand 1971] is based on counting the pairs of instances where the clus-

tering (C) and the true class labels (C') agree or disagree [Meilă 2007]. Let's define:

N_{11} : The number of pairs that are in the same cluster in C and in the same class in C'

N_{00} : The number of pairs that are in different clusters in C and in different classes in C'

N_{10} : The number of pairs that are in the same cluster in C but in different classes in C'

N_{01} : The number of pairs that are in different clusters in C but in the same class in C'

$$M = N_{11} + N_{00} + N_{10} + N_{01} = \frac{N(N-1)}{2}$$

Then, the Rand index is defined as:

$$R(C, C') = \frac{N_{11} + N_{00}}{M}$$

R takes a value in the range (0,1] [Aggarwal 2013], where 1 indicates exact similarity between the clustering result and the ground truth class labels.

Jaccard index:

Jaccard index [Jaccard 1912] is also based on the same idea of counting pairs. However, the formula is different:

$$J(C, C') = \frac{N_{11}}{N_{11} + N_{10} + N_{01}}$$

J takes a value in the range [0,1].

Fowlkes & Mallows index:

Another counting-pairs measure is Fowlkes & Mallows [Fowlkes 1983], defined as:

$$FM(C, C') = \sqrt{\frac{N_{11}}{N_{11} + N_{10}} \frac{N_{11}}{N_{11} + N_{01}}}$$

FM also takes a value in the range [0,1].

Purity:

This is the simplest external validation measure, which counts the number of correct classes in each cluster. The purity of clustering C with respect to class labels C' is defined as [Manning 2008]:

$$Purity(C, C') = \frac{1}{N} \sum_{k=1}^K \max_{k'=1 \dots K'} |C_k \cap C'_{k'}|$$

Purity takes a value in the range (0,1] [Aggarwal 2013]. However, it assigns a high score if C consist of many small clusters that correspond to one class. Thus,

we can not use it to justify the quality of a clustering in terms of the number of generated clusters [Manning 2008].

Mutual Information:

This measure is based on information theory. It measures how much information shared between the clustering C and the class labels C' , as defined below [Meilă 2007]:

$$MI(C, C') = \sum_{k=1}^K \sum_{k'=1}^{K'} P(C_k, C'_{k'}) \log \frac{P(C_k, C'_{k'})}{P(C_k)P(C'_{k'})}$$

where $P(X) = \frac{|X|}{N}$, and $P(X, Y) = \frac{|X \cap Y|}{N}$

$MI \geq 0$, but is not bounded by a constant value, which makes it difficult to interpret. However, we can use the Normalized MI [Strehl 2003] to limit the value of the index to the range $[0,1]$:

Let the entropy of a discrete random variable Z that can take K values be defined as:

$$H(Z) = - \sum_{k=1}^K P(Z_k) \log P(Z_k), \text{ then:}$$

$$NMI(C, C') = \frac{MI(C, C')}{\sqrt{H(C)H(C')}}}$$

Variation of Information:

Variation of Information [Meilă 2007] is another measure based on the shared information between two partitions. It can be defined using the entropy and the mutual information as:

$$VI(C, C') = H(C) + H(C') - 2MI(C, C')$$

which takes a value in the range $[0, 2 \log \max(K, K')]$ [Aggarwal 2013]. This is the only measure among the presented above where a lower score is better, as $VI(C, C') = 0$ if and only if $C = C'$ [Meilă 2007]. To normalize it to have a value in the range $[0,1]$, we can use the Adjusted VI [Aggarwal 2013, Meilă 2007]:

$$AVI(C, C') = \frac{VI(C, C')}{2 \log \max(K, K')}$$

The problem with external validation measures is that they are applicable only if domain knowledge about class labels exist, which is generally not the case in real life datasets. Moreover, using such measures, usually applied to synthetic datasets, may not be sound for real-world datasets, because the classes may contain internal

structure and the present attributes may not allow the separation of corresponding clusters, or the classes may contain anomalies [Färber 2010].

2.4 Consensus Clustering

It is clear that each validation measure, whether internal or external, uses a different approach to justify the quality of a clustering result. In practice, the best suitable validation measure(s) remain unknown [Aggarwal 2013]. Therefore, another technique to obtain good quality clusters is to combine different clustering solutions (called *base clusterings*) and build a consensus partition, that can be better than what each single base clustering could achieve. Such process is called *consensus clustering*, *ensemble clustering*, or *aggregation of clusterings*. It involves 2 steps: first, building an ensemble of partitions (i.e. the combination of all the partitions provided by the base clustering algorithms), then applying a consensus function. In several articles on clustering ensemble, authors have tried to define a set of properties that endorses the use of clustering ensemble methods [Ghaemi 2009, Vega-Pons 2011]:

- 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 [Ghaemi 2009, Topchy 2004]. The consensus clustering problem can be defined as follows [Vega-Pons 2011]:

Let's consider a d -dimensional dataset D , that consists of N instances, $D = \{x_1, x_2, \dots, x_N\}$. Let $\mathbb{P} = \{P_1, P_2, \dots, P_M\}$ be the set of M partitions acquired from applying different clustering scenarios on D . Each partition P_i divides D into k_i clusters, that is, $P_i = \{C_1^i, C_2^i, \dots, C_{k_i}^i\}$. C_j^i identifies cluster j in partition i . Let $\mathbb{P}_D$ denotes the set of all possible partitions of D . We have $\mathbb{P} \subset \mathbb{P}_D$. The goal of consensus clustering is to find a partition $P^* \in \mathbb{P}_D$ that better represents the partitions in $\mathbb{P}$.

To solve this problem, different consensus clustering methods have been proposed. These methods can be categorized based on the underlying approach used as defined in the following subsections.

2.4.1 Graph-Based

The consensus problem is formulated as a graph (or hypergraph) partitioning problem. Strehl and Ghosh [Strehl 2003] defined the consensus partition P^* as the one

that shares the most information with the ensemble $\mathbb{P}$. To measure the shared information, the NMI index (previously discussed in section 2.3.2) is used. Thus, the consensus solution can be defined as [Vega-Pons 2011]:

$$P^* = \arg \max_{P \in \mathbb{P}_D} \sum_{i=1}^M NMI(P, P_i)$$

which is computationally intractable. Therefore, they presented three heuristic methods: CSPA, HGPA, and MCLA. These methods use a binary membership indicator matrix ($\mathcal{M}$) to represent the relationship between the instances and the clusters in the ensemble: $\mathcal{M}$ consists of N rows corresponding to instances, and N_C columns corresponding to clusters from the partition ensemble.² An entry $[x, y]$ in $\mathcal{M}$ is set to 1 if instance x belongs to cluster y , otherwise, it is set to 0.

In Cluster-based Similarity Partitioning Algorithm (CSPA), each base clustering is considered as a $N \times N$ co-association matrix, where an entry $[i, j]$ is set to 1 if instances i and j belong to the same cluster, otherwise $[i, j] = 0$. An overall similarity matrix S is built by entry-wise averaging of all the co-association matrices of the base clusterings [Vega-Pons 2011]. It is possible also to calculate S from $\mathcal{M}$ using: $S = \frac{1}{M} \mathcal{M} \mathcal{M}^T$. S can be viewed as an adjacency matrix for a graph, such that each instance represents a node, and each entry $[i, j]$ in S represents a weight of an edge that connects node i with node j . Now, the consensus solution can be found by simply partitioning this graph using a partitioning algorithm. The METIS algorithm [Karypis 1998] is used in CSPA.

In HyperGraph-Partitioning Algorithm (HGPA), $\mathcal{M}$ can be viewed as a hypergraph representation, where each cluster (column in $\mathcal{M}$) is a hyperedge that connects several nodes (instances). Hence, the task here is to partition the hypergraph by cutting the minimal number of hyperedges, considering that they have the same weight [Vega-Pons 2011]. The HMETIS algorithm [Karypis 1999a] is used to accomplish this task.

The idea in Meta-CLustering Algorithm (MCLA) is to cluster the clusters (the columns in $\mathcal{M}$). This is done by creating a co-association matrix between the base clusters, where the entry $[i, j]$ is set according to the similarity between clusters i and j . The similarity between clusters is calculated using the binary Jaccard index, which measures the proportion of instances that belong to both clusters i and j [Vega-Pons 2011]. This way, the problem can be viewed as in CSPA, that is, the co-association matrix between clusters (whereas in CSPA it was between instances) represents an adjacency matrix of graph. In this graph, nodes represent clusters, that are linked by weighted edges according to their similarity. Like in CSPA, the METIS algorithm is used to partition this graph. The resulting clusters are called meta-clusters, and the instances are assigned to them according to the maximum number of times an instance is found in a meta-cluster [Vega-Pons 2011].

All the above three methods require the parameter K (the number of required

² $N_C = \sum_{i=1}^M k_i$, where k_i is the number of clusters in base clustering i

consensus clusters) to partition the graphs into K “comparable sized” clusters (which is already a constraint [Aggarwal 2013]). The authors state that the time complexities of the three methods are as follows: CSPA is the slowest with complexity $O(N^2KM)$; HGPA is the fastest with complexity $O(NKM)$; and MCLA has a good time complexity of $O(NK^2M^2)$. The authors conclude from various tests that MCLA achieve the best quality consensus clustering compared to the other two methods.

Another graph-based consensus algorithm is the Hybrid Bipartite Graph Formulation (HBGF) [Fern 2004]. Here, both instances and clusters represent nodes in a bipartite graph. No edge exists between two instance nodes or two cluster nodes. Instead, an edge exists between an instance node and a cluster node if the former belongs to the latter. To do this, the graph adjacency matrix W is built from the membership matrix $\mathcal{M}$ using:

$$W = \begin{bmatrix} 0 & \mathcal{M}^T \\ \mathcal{M} & 0 \end{bmatrix}$$

This graph can be partitioned by either the spectral clustering algorithm proposed by [Ng 2002] (previously discussed in section 2.2.8), or by using the METIS algorithm.

Punera and Ghosh [Punera 2008] modified the consensus methods CSPA, MCLA, and HBGF to work with an ensemble of fuzzy partitions. A fuzzy partition ensemble $\mathbb{P} = \{P_1, P_2, \dots, P_M\}$ consists of M fuzzy clusterings. A clustering $P_i = \{m_1^i, m_2^i, \dots, m_{K_i}^i\}$ consists of K_i membership functions, where $m_j^i(x)$ defines the degree of membership of instance x to cluster j in partition i . Thus, we have $\sum_{i=1}^M K_i$ membership functions in the ensemble. By considering each instance as a vector in the $\sum_{i=1}^M K_i$ dimensional space, the Euclidean distance between 2 instances a and b in this space is [Vega-Pons 2011]:

$$d_{a,b} = \sqrt{\sum_{i=1}^M \sum_{j=1}^{K_i} (m_j^i(a) - m_j^i(b))^2}$$

This distance is converted into a similarity measure between a and b using: $SC_{a,b} = e^{-d_{a,b}^2}$. SC represents a pairwise similarity matrix for all instances in the dataset. As in CSPA, METIS is used to generate the final consensus solution. This method is called sCSPA [Punera 2008]. Unlike MCLA that uses the binary Jaccard index, sMCLA [Punera 2008] uses SC to measure the similarity between 2 clusters from 2 partitions, by calculating the difference in the memberships of all instances to these 2 clusters. sHBGF [Punera 2008] uses the membership values m_j^i as the weight of the edge that connects an instance into a cluster in the bipartite graph. Like in HBGF, the weight between 2 instances or 2 clusters is set to 0.

Domeniconi and Al-Razgan [Domeniconi 2009] focused on the curse of dimensionality problem that most clustering algorithms suffer. The Locally Adaptive Clustering (LAC) algorithm [Domeniconi 2007] can discover clusters in subspaces by assigning weights to the attributes. It requires two parameters: K (as in other clustering methods), and h that controls the relative differences between feature weights.

However, Domeniconi and Al-Razgan state that setting h is difficult, and the resulted clustering of LAC highly affected by this parameter. Thus, they proposed three consensus clustering methods over an ensemble of LAC partitions produced by variable values of h . The Weighted Similarity Partitioning Algorithm (WSPA) [Domeniconi 2009] constructs a pairwise similarity matrix for each clustering in the ensemble. The similarity between two instances x_i and x_j is measured by the cosine similarity between the probability vectors of the two instances. A probability vector p_i associated with instance x_i is calculated from the weighted distance of x_i to each cluster center (weights and clusters are the results of LAC). An overall similarity matrix S is the average of the m similarity matrices constructed from running LAC m times. The problem now can be mapped into a graph partitioning problem. A graph of n nodes (n is the number of instances in the dataset) can be built from S , where the edge weights are defined by the entries of S . Then, the METIS algorithm [Karypis 1998] is used to find the consensus partition.

The second method proposed by Domeniconi and Al-Razgan is the Weighted Bipartite Partitioning Algorithm (WBPA) [Domeniconi 2009]. This is similar to HBGF [Fern 2004] method described above, but instead of the binary membership matrix, a probability matrix is used. Having m partitions in the ensemble, each instance has m probability vectors that define its membership degree to each cluster in the ensemble.³ An overall probability matrix A of n rows and $K \times m$ columns is built from the probability vectors associated to each instance. A bipartite graph that connects only instances to clusters is created from the probability matrix as follows:

$$E = \begin{bmatrix} 0 & A^T \\ A & 0 \end{bmatrix}$$

METIS [Karypis 1998] is also used to find the consensus partition.

The Weighted Subspace Bipartite Partitioning Algorithm (WSBPA) is the third method proposed by Domeniconi and Al-Razgan. It extends WBPA by associating attribute weight vectors to the resulted consensus clusters. To explain: Applying METIS on the bipartite graph yields K groups. Each of them consists of nodes of two types: Instance and base cluster node. In WBPA, only instance nodes are used to define the consensus clusters, whereas nothing is done with the base cluster nodes. In WSBPA, the base cluster nodes are used to define a weight vector for each consensus cluster. A consensus cluster weight represent the average of the weights of the base clusters included in it. This weight reflects the influence of the attributes in each consensus cluster. In all the three methods proposed by Domeniconi and Al-Razgan, K is set to the actual number of clusters for the tested datasets for both LAC and the consensus clustering methods.

2.4.2 Relabeling and Voting-Based

The base clusterings are regarded as voters, where each one votes that an instance belongs to a particular cluster. However, as each cluster label in a clustering is a

³The probability vectors are calculated as in WSPA.

symbol that is unrelated to the other clusterings in the ensemble, the main task here is to solve the label correspondence problem throughout the ensemble. For example, clustering A may partition a dataset into 2 clusters, the first is labeled as 1, and the other as 2. Whereas another clustering B may perform the same partitioning of the instances into 2 clusters, but may label the first cluster in A as 2, and the other as 1, depending on its label assignment process or its initialization.

Dudoit and Fridlyand [Dudoit 2003] and also Fischer and Buhmann [Fischer 2003] presented the Bagged Clustering approach, on which bootstrap samples are drawn from the dataset. A clustering algorithm is applied on each sample (PAM is used for illustration in [Dudoit 2003]), and all the results are combined to build the clustering ensemble. Cluster labels are permuted to find the maximum agreement between the assignments (using the Hungarian algorithm, with time complexity of $O(K^3)$ [Fischer 2003]). Then, a plurality voting procedure is performed to find the final consensus cluster labels for each instance.

Dimitriadou *et al.* [Dimitriadou 2002] proposed a simple heuristic method to find the consensus solution that minimizes the difference to the ensemble. It reads the ensemble sequentially and updates the votes as follows:

- Let P_1 and P_2 be the first two clusterings in the ensemble.
- For each cluster C_j^2 in P_2 , find a cluster C_i^1 in P_1 that shares the maximum percentage of instances with C_j^2 , and unify cluster labels.
- After relabeling the clusters in P_2 so that C_j^2 corresponds to $C_j^1, \forall j$, the instances are assigned to a new common partition $\bar{P}$: If an instance x has been assigned to both clusters C_j^1 and C_j^2 , then it will be assigned to the common cluster $\bar{C}_j$. If x has been assigned to C_i^1 and to a different label cluster in P_2 , say C_j^2 , then it will be assigned to the common clusters $\bar{C}_i$ and $\bar{C}_j$ with weight 0.5 for each.

The process is then repeated between $\bar{P}$ and the remaining partitions in the ensemble, resulting in a final fuzzy matrix of assignments of each instance to clusters. This matrix can be accepted as a final fuzzy consensus result, or a hard consensus partition can be obtained by assigning the instances to the clusters to which it has the maximum membership. The latter process is equivalent to a simple majority voting after relabeling [Hornik 2005b].

The above idea of finding a consensus that has the minimum distance to the ensemble was previously modeled by Gordon and Vichi [Gordon 2001]. Each base partition is represented as a membership matrix m of $N \times K_m$ entries, where $m[i, j]$ denotes the degree of membership of instance i in cluster j . For hard partitions, the membership degree is either 0 or 1, whereas in soft (fuzzy) base partitions, it can take any real value between 0 and 1. Starting with an initial guess of the consensus (initial membership matrix), the objective of the first model in [Gordon 2001] is to find a consensus membership matrix that minimizes the sum of the squared distance to the membership matrices of all partitions in the ensemble. However, it

is necessary here to solve the label correspondence problem (using the Hungarian algorithm) to measure the difference between corresponding clusters in the partitions. Gordon and Vichi focused in the first model on solving the consensus problem for fuzzy base partitions to generate a fuzzy consensus. Note that the calculated distance takes into account only cluster labels that are available in all partitions; that is, case that K is different in each partition, then only matching clusters are considered [Hornik 2007]. The third model in [Gordon 2001] uses co-membership matrices (formed from mm') to address the same consensus. Thus, it is not necessary to solve the label correspondence.

The main restriction in voting-based consensus methods is that all the clusterings, and hence the final consensus, have the same number of clusters K . This is essential for solving the label correspondence problem with certain accuracy [Vega-Pons 2011].

2.4.3 Co-association Matrix Based

A co-association matrix⁴ can be used as an intermediate representation of the clustering ensemble, which can be passed into a clustering algorithm to obtain the consensus partition [Vega-Pons 2011]. This matrix is calculated as follows:

$$CA[i, j] = \frac{1}{M} \sum_{r=1}^M \delta(P_r(x_i), P_r(x_j))$$

where $1 \leq i, j \leq N$, $P_r(x)$ is the cluster label associated with instance x in the base partition P_r , and $\delta(x, y) = 1$ if $x = y$, and 0 otherwise. The CA matrix can be viewed as a similarity measure between the instances, taking into account the clustering ensemble.

In the Evidence Accumulation (EA) algorithm proposed by Fred and Jain [Fred 2002], a single linkage hierarchical clustering algorithm (see section 2.2.2) is run over the co-association matrix. To decide the number of consensus clusters from the dendrogram, a K-cluster lifetime technique is used: K-cluster lifetime is the range of threshold values (the maximum - the minimum threshold values) on the dendrogram that leads to the identification of k clusters [Fred 2002]. Then, the highest lifetime is used to obtain the final consensus partition. Other linkage methods can be used as variations of this method [Vega-Pons 2011].

An enhanced association matrix is proposed by Wang *et al.* [Wang 2009], with their algorithm Probability Accumulation (PA). The idea is to add more information about the similarity between instances within a cluster, not just a 0/1 similarity. The basic assumption is that, the longer the distance between two instances in the data space, the weaker their correlation. Hence, the proposed matrix is calculated as follows:

⁴The idea of co-association matrix is already discussed with CSPA in section 2.4.1. Thus, [Aggarwal 2013] lists CSPA under this category.

$$CA[i, j] = \begin{cases} 1 & i = j \\ 0 & i \neq j \text{ and } P_r(x_i) \neq P_r(x_j) \\ \frac{1}{1 + \sqrt[d]{|C_k^r|}} & i \neq j \text{ and } P_r(x_i) = P_r(x_j) = k \end{cases}$$

where C_k^r is the cluster that contains both x_i and x_j in the base clustering r , and d is the dimensionality of the dataset. The overall association matrix is calculated as the mean of all association matrices for all base clusterings. Then, the same process used in EA [Fred 2002], which is described above, is used.

2.4.4 Distance-Based

In distance-based approaches, the consensus partition is defined as the one that has the minimum Mirkin distance to the ensemble [Vega-Pons 2011]:

$$P^* = \arg \min_{P \in \mathbb{P}_D} \sum_{i=1}^M d_M(P, P_i)$$

where the Mirkin distance $d_M(P_a, P_b)$ is defined as the number of pairs of instances that are in the same cluster in one of the clusterings P_a or P_b , and in different clusters in the other partition (i.e. $d_M(P_a, P_b) = N_{01} + N_{10}$, where N_{01} and N_{10} are explained in section 2.3.2). P^* is considered the *median* partition of the ensemble. As finding this median is an NP-complete problem [Vega-Pons 2011], several heuristic solutions were proposed:

Filkov and Skiena [Filkov 2004] proposed three methods: In the Best Of K method (BOK), the consensus is simply one of the partitions in the ensemble that has the minimum distance to the other partitions. The Simulated Annealing One-element Move (SAOM) is a local search method. It starts with an initial partition (either chosen randomly, or as a result of BOK). Then, it iterates, moving one random instance from its cluster into another cluster or an empty cluster, to enhance the sum of distances. The Best One-element Move (BOM) algorithm follows the same idea of SAOM, but instead of a random selection, the best instance to enhance the sum of distances is chosen at each iteration. Tests showed that SAOM achieved a better quality consensus compared to other methods.

Gionis *et al.* [Gionis 2007] presented several other heuristic methods. In the Balls algorithm, a pairwise distance matrix is calculated. The distance between two instances is defined as the average number of base clusterings on which the two instances do not belong to the same cluster. The entries of the matrix are considered as weights of edges that connect the instances in a graph. The instances are sorted in increasing order of the total weight of all the edges of each instance. Then, the algorithm choses, at each iteration, the first instance u (according to the sorting) that is not yet clustered. The set of instances B that are of distance at most 0.5 from u are identified. The average distance $d(u, B)$ between B instances and u is calculated. If $d(u, B) \leq \alpha$, then a cluster is formed from $B \cup \{u\}$. Otherwise, u becomes a singleton cluster. The time complexity of this algorithm is $O(MN^2)$ for

generating the matrix, and $O(N^2)$ for running it [Gionis 2007].

Another algorithm from [Gionis 2007] is the Agglomerative algorithm. As any agglomerative hierarchical clustering (see section 2.2.2), it starts by considering each instance as a cluster. Next, it iterates merging the closest two clusters if their average distance is less than 0.5. The algorithm stops when there are no more clusters to merge according to this condition. The time complexity for running this algorithm is $O(N^2 \log N)$, in addition to the time required for generating the distance matrix [Gionis 2007].

The Farthest algorithm [Gionis 2007] is a divisive hierarchical clustering method. First, all the instances are combined in one cluster. Then, the two farthest instances are chosen as centers of two clusters. All the remaining instances are assigned to one or the other of the two clusters based on their distance to the centroid. After these two initial steps, the algorithm repeats the following process. It chooses an instance that is the farthest to the current centroids. This instance becomes the center of a singleton cluster. The remaining instances are assigned to clusters based on minimizing a cost. The cost of the new solution is calculated. If it is lower than the previous solution cost, the algorithm continues. Else, it stops, presenting the previous solution.

The LocalSearch algorithm [Gionis 2007] starts by an initial clustering of the instances (either randomly generated or from running one of the above methods). Then, it goes through all the instances, calculating the cost of moving an instance from its cluster to another one or generating a singleton cluster from it. An instance is moved to the cluster to which it has the smallest moving cost. This process is repeated until there is no other move that can enhance the cost.

2.4.5 Fragment-Based

Rather than searching for a consensus solution in the full data space, recently, Wu *et al.* [Wu 2012] presented the notion of *Data Fragments* to prune the search space. A data fragment (considering the clustering ensemble) is a set of instances that is not divided by any of the base clusterings, and meanwhile not contained in any other set that is also not divided by the base partitions. Wu *et al.* proved that clustering aggregation is possible using fragments instead of instances, resulting in enhanced performance as the number of fragments is much lower than the number of instances. Three consensus algorithms were proposed: F-Agglomerative, F-Farthest, and F-LocalSearch. These algorithms are modifications of the algorithms presented in [Gionis 2007] (discussed in section 2.4.4), by updating the distance calculation to consider data fragments. After finding the consensus partition, the fragments are replaced by the instances they represent to obtain the consensus solution for the instances.

Chung and Dai [Chung 2014] proposed the F-CARS algorithm, that uses the fragments approach to generate a consensus. The algorithm is similar to local search algorithm. It starts by an initial partition. Next, it iterates on moving fragments to clusters (in the current partition or an empty cluster) based on a similarity threshold

that is updated at each iteration. The algorithm stops when the current partition is not changed from a previous iteration.

The data fragments represent the set of instances that are grouped together in all base clusters. From this view, Vega-Pons and Avesani [Vega-Pons 2015] proved that a further pruning of the search space is possible, to obtain a consensus that represents the “median” of the ensemble. The new fragments (the majority fragments) represent the set of instances that are grouped together in 50% (or more) of the base clusters (with merging intersecting fragments). Vega-Pons and Avesani used the Simulated Annealing One-element Move (SAOM) algorithm [Filkov 2004] (discussed in section 2.4.4) to compare the quality of the resulted consensus, when generating it from the full data space, from the fragments space, and from the majority fragments space. They concluded from tests that working on the majority fragments not just enhanced the performance of the algorithm, but also achieved a higher quality consensus. The only drawback is that in some situations, we may get only one majority fragment, meaning a consensus solution of 1 cluster, which is unacceptable. Thus, they recommend trying other percentage of agreement between the base clusters.

2.4.6 Other Methods

A Genetic Algorithm (GA), as a heuristic search method, can be used to find a consensus solution. Yoon *et al.* [Yoon 2006] proposed the Heterogeneous Clustering Ensemble (HCE) method. Considering the ensemble as the population of GA, HCE starts by choosing two clusterings as parents (those with the highest overlap). The crossover operation is performed between the two parents by selecting a cluster C_i^1 from the first parent P_1 that has the highest overlap (sets intersection) with the clusters of the second parent P_2 . C_i^1 is then used to replace a cluster C_j^2 in P_2 (the most similar cluster in P_2 to C_i^1). If there are instances in C_i^1 that already belong to other clusters in P_2 , then they are removed. The result of this swap operation is the first offspring. The same process is performed for the second parent P_2 against P_1 to generate the second offspring. The two offsprings replace their parents in the population, and the algorithm repeats the process until convergence condition is met.

The idea proposed by Caruana *et al.* [Caruana 2006] is to generate many base clusterings, then build a similarity matrix for these different partitions using Rand index. This similarity matrix is passed to agglomerative hierarchical clustering to build a meta clustering. The dendrogram shows how the partitions 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 (Principle Components Analysis) were used to produce different base clustering views.

Asur *et al.* [Asur 2007] used six predefined clustering algorithms suitable for protein-protein datasets clustering as base clusterings. A cluster membership ma-

trix 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, and if K is high the resulting binary membership matrix is sparse and the consensus clustering of this matrix is ineffective. Thus, PCA is applied before the consensus clustering to reduce the dimensionality of the membership matrix into less, but more expressive, dimensions. These different consensus techniques were compared, and the authors concluded that the PCA based technique produced very efficient clustering and identified multiple functionalities of proteins.

With their algorithm *WClustering*, Li and Ding [Li 2008] proposed weighting the base clusterings to ensure removing redundant (similar) partitions, since this process produces better results compared to other methods that 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.

Zhang and Li [Zhang 2011] divided the base clusterings into groups based on their similarity using K-means. On each group, one of the consensus methods is used: PCA-based consensus algorithm [Asur 2007], CSPA and HGPA from [Strehl 2003], and WClustering [Li 2008]. Thus, the final result is K consensuses for the user to select from.

The discussed consensus methods present the major categories of the proposed solutions for the problem. More details and other methods can be found in: Chapter 22 in [Aggarwal 2013], [Ghaemi 2009], [Sarumathi 2013], and [Vega-Pons 2011]. Generally, the following summarize some of the constraints in the presented methods:

- K (the number of required clusters in the consensus solution) is mandatory in graph-based and voting-based consensus methods, or any other methods that apply a clustering algorithm to generate a consensus solution. An exact value of K is difficult to predict.
- Some may impose restrictions on the ensemble generation process, like in voting-based methods where the label correspondence problem is better solved when all partitions in the ensemble consist of the same number of clusters.
- Using averaged co-association matrix or distance matrix makes the consensus method inapplicable for large datasets, as these matrices demand high storage space.
- Using a clustering algorithm to generate a consensus solution imposes two restrictions: the time complexity of the clustering algorithm, and the shape of clusters generated from it.

In the following chapter, a new consensus clustering method is proposed that bypasses all the above restrictions.

The Proposed Method

Contents

3.1	Frequent Pattern Mining	35
3.2	MultiCons	38
3.2.1	The Base Clusterings	39
3.2.2	The Binary Membership Matrix	40
3.2.3	Identifying Clustering Patterns	41
3.2.4	Generating Multiple Consensuses	42
3.2.5	The ConsTree	47
3.3	Example	49

In this chapter, a new consensus clustering method called MultiCons (Multiple Consensuses) is proposed. This is the first method to transform the consensus clustering problem into a pattern mining problem. That is, by using the binary membership matrix (section 2.4.1) to represent the ensemble, a pattern mining technique known as *Frequent Closed Itemset* (FCI) [Pasquier 1999] is applied over the binary matrix to identify the similarities between the base clusterings. The result is a lattice of *clustering patterns*, each defines the agreement between a set of base clusterings on grouping a set of instances. Considering this lattice as the search space for the consensus problem, we can further divide it into subspaces based on the number of base clusterings that participate in the patterns. This yields in generating multiple consensus solutions by clustering the patterns in each subspace.

An additional and important component of MultiCons is the *tree of consensuses*, or simply ConsTree. It depicts at each of its levels a consensus solution, where the nodes represent its clusters. It also shows the relation between the clusters of sequential consensuses, which gives more understanding on the relationships between the instances in the data space.

Before explaining the proposed method, it is necessary to understand what is frequent pattern mining, and in particular, the FCI technique which is used in MultiCons.

3.1 Frequent Pattern Mining

The widely used data mining application to describe frequent patterns and hence association rules discovery is the market basket analysis. Suppose that we have a

dataset $\mathcal{T}$. Each row in $\mathcal{T}$ defines a customer transaction, that consist of a set of items bought by the customer, for example, milk, bread, cereals, The objective of market basket analysis is to find which sets of items are frequently bought together by customers. The identification of these buying habits can be used to define *rules* that tells, for example, 70% of customers who buy cereals and sugar also buy milk. Such rules, in the form of set of items (cereals and sugar) lead to a set of items (milk), are called Association Rules. They can be used to enhance marketing plan, advertising strategies, or design new catalogs [Han 2011].

For consensus clustering, the identification of similarities among clusterings in the ensemble is the core part of the consensus generation process. If we consider each cluster in the ensemble as an item, then we can use frequent pattern mining to identify the similarity among a set of clusters in terms of their common set of instances. In the following, association rules generation is not discussed, as only the frequent patterns are necessary for MultiCons.

A transactional dataset $\mathcal{T}$ can be represented as a binary dataset $\mathcal{D}$ that consists of N rows and N_I columns. Each row in $\mathcal{D}$ denotes an object, and each column denotes an item from the set of all items in $\mathcal{T}$ (N_I items). An example transactional dataset $\mathcal{T}$ is shown in table 3.1 from [Pasquier 1999]. It consists of 5 objects identified by TID numbers, and a total of 5 items. Its binary matrix representation, that is $\mathcal{D}$, is shown in table 3.2.

Table 3.1 – Example transactional dataset from [Pasquier 1999].

TID	Items
1	A C D
2	B C E
3	A B C E
4	B E
5	A B C E

Table 3.2 – The binary matrix representation for the transactional dataset in table 3.1.

TID	A	B	C	D	E
1	1	0	1	1	0
2	0	1	1	0	1
3	1	1	1	0	1
4	0	1	0	0	1
5	1	1	1	0	1

Frequent pattern mining is explained by the following definitions [Mondal 2012, Pasquier 1999]:

Definition 1 (Dataset) A dataset $\mathcal{D}$ is a triplet $(\mathcal{O}, \mathcal{L}, \mathcal{R})$, where $\mathcal{O}$ is a finite set of objects (represented as rows), $\mathcal{L}$ is a finite set of items (represented as columns), and $\mathcal{R}$ is a binary relation defining relationships between objects and items: $\mathcal{R} \subseteq \mathcal{O} \times \mathcal{L}$. Every couple $(o, i) \in \mathcal{R}$, where $o \in \mathcal{O}$ and $i \in \mathcal{L}$, denotes that item i belongs to object o .

Definition 2 (Itemset) An itemset L is a non-empty finite set of items, $L \subseteq \mathcal{L}$. L is a k -itemset if it consists of k items.

The closed itemset lattice for the example dataset $\mathcal{D}$ (in table 3.2) is shown in figure 3.2. Considering a minimum support threshold $s = 0.4$, we have only 5 frequent closed itemsets. Using a suitable mining algorithm, all the possible association rules that can be generated from the frequent itemsets can efficiently be generated from the frequent closed itemsets. However, the pattern mining algorithms that generate the frequent itemsets or the frequent closed itemsets are not discussed here, as such discussion is outside the scope of this work. Interested reader is advised to check [Aggarwal 2014], chapters 4 and 5 in [Aggarwal 2015], [Ceglar 2006], chapters 6 and 7 in [Han 2011], [Mondal 2012], and [Pasquier 1999].

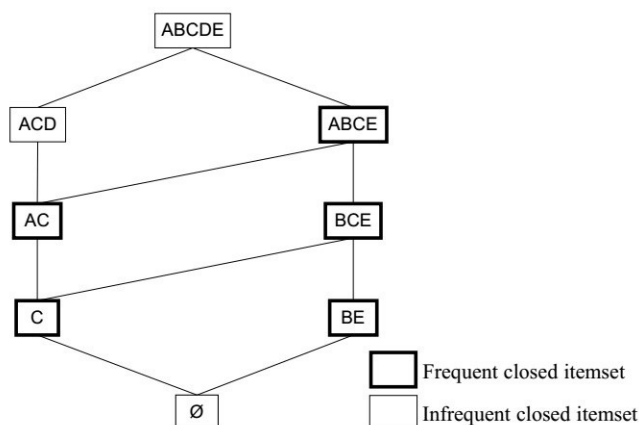


Figure 3.2 – Closed Itemset lattice of $\mathcal{D}$, from [Pasquier 1999].

3.2 MultiCons

MultiCons is the first consensus clustering method to use frequent closed patterns to identify the similarities among the base clusterings. Having multiple clustering results for a dataset, a binary membership matrix (section 2.4.1) is used to represent the ensemble. This matrix is similar to the binary matrix used in frequent pattern mining to represent a transactional dataset. That is, each instance is considered as an object that belongs to certain clusters in the ensemble. As the items here are clusters, using the FCI identifies the sets of instances that are grouped together by sets of base clusterings. This can be viewed as a similarity among a set of base clusters in terms of the shared instances between them. This is obviously different from other consensus methods that consider the similarities among the full set of base clusterings only.

The following subsections explain the details of the proposed consensus method, which is summarized in figure 3.3.

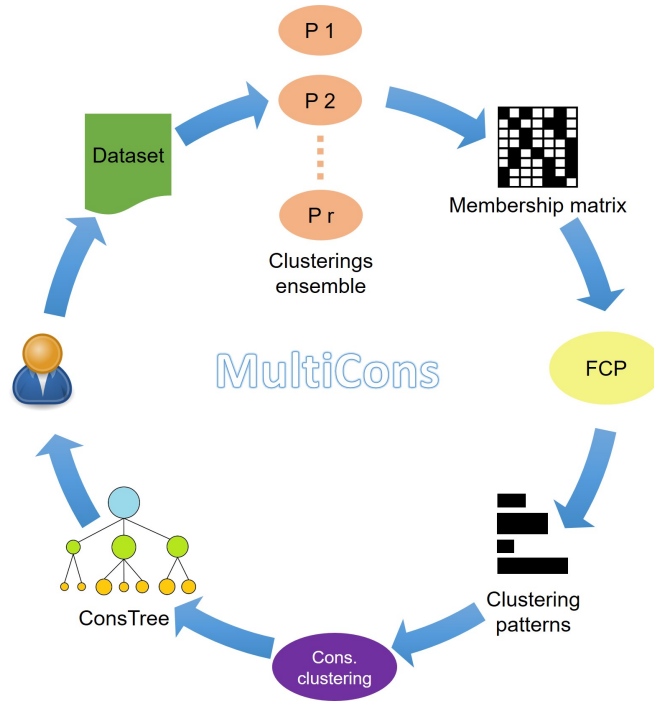


Figure 3.3 – Process diagram of MultiCons.

3.2.1 The 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 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 partitions.

In-ensemble similarity is proposed to calculate the average similarity among the clusterings in the ensemble. Each base clustering is compared with the others using Jaccard index (section 2.3.2), though, it is possible to use other measures for this purpose. The average is then calculated for each clustering, and finally for the whole ensemble. In-ensemble similarity can help, for example, in distinguishing which ensemble is better if we generate several ensembles. If the hidden structure of the dataset instances can be recognized by most base clusterings, then the value of this measure will normally be high. On the other hand, a low value indicates that few agreements exist between the base clusterings in partitioning the dataset, which may lead eventually into a low quality consensus. Remember that it is more important to ensure the diversity in the ensemble, as a high value of this measure

can also be achieved from many similar low quality base clusterings.

In-ensemble similarity can be calculated using the following formula, and as explained in algorithm 8:

$$\text{In_Ensemble_Similarity}(\mathbb{P}) = \frac{1}{r} \sum_{i=1}^r \frac{1}{r-1} \sum_{j=1, j \neq i}^r \text{Jaccard}(P_i, P_j)$$

where $P_i, P_j \in \mathbb{P}$, and $\mathbb{P}$ is an ensemble of r base partitions.

Algorithm 8: In-ensemble similarity

Input : An ensemble of base clusterings
Output: *InEnsSim* (In-ensemble similarity score)

- 1 *EnsSim* $\leftarrow$ matrix of $r \times r$ cells of 0, where r is the # of base clusterings
- 2 *AvgSim* $\leftarrow$ vector of '0's of length r
- 3 **for** $i = 1$ **to** $r - 1$ **do**
- 4 $C_i \leftarrow i^{\text{th}}$ clustering in the ensemble
- 5 **for** $j = i + 1$ **to** r **do**
- 6 $C_j \leftarrow j^{\text{th}}$ clustering in the ensemble
- 7 $\text{EnsSim}[i, j] \leftarrow \text{EnsSim}[j, i] \leftarrow \text{Jaccard}(C_i, C_j)$
- 8 **end**
- 9 $\text{AvgSim}[i] \leftarrow \frac{\text{sum}(\text{EnsSim}[i, \])}{r - 1}$
- 10 **end**
- 11 $\text{AvgSim}[r] \leftarrow \frac{\text{sum}(\text{EnsSim}[r, \])}{r - 1}$
- 12 *InEnsSim* $\leftarrow \text{mean}(\text{AvgSim})$

3.2.2 The Binary Membership Matrix

This matrix is already explained in section 2.4.1. However, it is defined here to comply with the pattern mining terminology:

Definition 7 (Membership matrix) A binary 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.

The following example will be used to help understanding each step in Multi-Cons:

Consider a dataset of nine instances $\mathcal{D} = \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$ partitioned using five base clusterings into the five following partitions: $P1 = \{\{1, 2, 3\}, \{4, 5, 6, 7, 8, 9\}\}$, $P2 = \{\{1, 2, 3\}, \{4, 5, 6, 7, 8, 9\}\}$, $P3 = \{\{1, 2, 3, 4, 5\}, \{6, 7\}, \{8, 9\}\}$, $P4 = \{\{4, 5, 6, 7\}, \{1, 2, 3, 8, 9\}\}$, $P5 = \{\{1, 2, 3, 4, 5, 6, 7\}, \{8, 9\}\}$.

$\{1, 2, 3\}$, $\{8, 9\}$, and $P5 = \{\{4, 5, 6, 7\}, \{1, 2, 3\}, \{8, 9\}\}$. Table 3.3 shows the resulting binary membership matrix consisting of 9 rows (instances) and 13 columns (total number of clusters in the ensemble). Each column P_j^i represents cluster j in partition i as a binary vector where values ‘1’ identify the instances that belong to the cluster.

Table 3.3 – An example membership matrix.

Instance ID	P_1^1	P_2^1	P_1^2	P_2^2	P_1^3	P_2^3	P_3^3	P_1^4	P_2^4	P_3^4	P_1^5	P_2^5	P_3^5
1	1	0	1	0	1	0	0	0	1	0	0	1	0
2	1	0	1	0	1	0	0	0	1	0	0	1	0
3	1	0	1	0	1	0	0	0	1	0	0	1	0
4	0	1	0	1	1	0	0	1	0	0	1	0	0
5	0	1	0	1	1	0	0	1	0	0	1	0	0
6	0	1	0	1	0	1	0	1	0	0	1	0	0
7	0	1	0	1	0	1	0	1	0	0	1	0	0
8	0	1	0	1	0	0	1	0	0	1	0	0	1
9	0	1	0	1	0	0	1	0	0	1	0	0	1

3.2.3 Identifying Clustering Patterns

$\mathcal{M}$ can be viewed as a pattern mining problem. Using the FCI technique (section 3.1), we can identify *clustering patterns*, or more precisely, *Frequent Closed Patterns* **FCP**. Note that for MultiCons, the minimum support threshold is set to 0. This means that all the possible closed patterns are considered. It ensures not eliminating some instances in the consensus solution.

Definition 8 (Clustering Pattern) A clustering pattern $\rho = (C, I)$ 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 $\forall i \in I$ and $\forall c \in C$, we have $(i, c) \in \mathcal{R}$, and C is a frequent closed itemset for a minimum support threshold = 0.

Table 3.4 shows the clustering patterns extracted for the running example dataset (table 3.3). Using the FCI technique prevents generating patterns of identical instance set. This is essential because the proposed method focuses on how the instances are grouped together, whereas cluster labels are unnecessary because they are unrelated. Hence, FCI eliminates many redundant patterns compared to generating all frequent patterns. For example, generating all frequent patterns will result in having two additional patterns for the 7th FCP in table 3.4. The additional patterns are the following:

$$\begin{aligned} &\{\{P_2^1\}, \{4, 5, 6, 7, 8, 9\}\} \\ &\{\{P_2^2\}, \{4, 5, 6, 7, 8, 9\}\} \end{aligned}$$

In terms of clustering decisions, these two patterns are redundant as they suggest the same instance set of FCP 7. On the other hand, the itemset of FCP 7 is more

Table 3.4 – Clustering patterns extracted from table 3.3.

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

important because it defines the maximum number of base clusterings that agreed on grouping the instances $\{4, 5, 6, 7, 8, 9\}$ together. If we generate all the frequent patterns for the binary matrix in table 3.3, we get 105 frequent patterns (with minimum support = 0), whereas we got only 7 using the FCI technique. Thus, we eliminated 98 redundant clustering decisions.

3.2.4 Generating Multiple Consensuses

Each clustering pattern defines agreement among a set of base clusters (itemset) on grouping a set of instances together. A *Decision Threshold* (DT) is used to categorize the patterns based on the size of the itemset. Using DT , the pattern space is further divided into subspaces, and this ensures the efficiency of processing even if we have a large number of clustering patterns. In each subspace, a consensus solution is built by clustering the instance sets. The consensus generated from each subspace ensures a minimum agreement among base clusterings (DT agreements) to form a consensus cluster.

Compared to the fragment-based consensus methods discussed in section 2.4.5, the instance sets of the patterns in the subspace of maximum DT value (which is the number of base clusterings in the ensemble) are exactly the data fragments defined in [Wu 2012]. These data fragments or sub-clusters are considered here as consensus clusters for an initial consensus solution. Rather than applying clustering algorithms to cluster the data fragments (as explained in section 2.4.5), the FCPs provide solutions to cluster the fragments as defined by a smaller agreement among the base clusterings. For example, FCP 5 in table 3.4 tells that 4/5 base clustering algorithms suggest grouping the instance sets of FCPs 1 and 2 together. This represents clustering of data fragments defined by agreement among 4 base clusterings.

After the initial (first) consensus of data fragments, where DT is at its maximum value, the following consensus candidates are built in an agglomerative process. At each iteration, DT is decremented, until reaching 1. At iteration n , a consensus solution is built from the instance sets of patterns whose itemset is of size n , plus the clusters of the previous consensus (at iteration $n + 1$).

Definition 9 (Generate Multiple Consensuses) Let $\alpha = \text{Max}(DT)$. The first consensus is $\Pi^\alpha = \{\pi_1^\alpha, \pi_2^\alpha, \dots, \pi_{k_\alpha}^\alpha\}$, where π_i^α is an instance set of a FCP built from α base clusterings. Let $1 \leq \beta < \alpha$ and $\mathbb{S}^\beta = \mathbb{I}^\beta \cup \Pi^{\beta+1}$ 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 $\Pi^{\beta+1}$ is the instance sets (clusters) of the previous consensus. A new consensus Π^β is the result of applying a consensus function $\mathcal{Y}$ on $\mathbb{S}^\beta$, that is, $\Pi^\beta = \mathcal{Y}(\mathbb{S}^\beta) = \{\pi_1^\beta, \pi_2^\beta, \dots, \pi_{k_\beta}^\beta\}$ such that $\pi_i^\beta \cap \pi_j^\beta = \emptyset$, $\forall (i, j) \in \{1, \dots, k_\beta\}$, $i \neq j$, and $\bigcup_{i=1}^{i=k_\beta} \pi_i^\beta = \mathcal{I}$.

At iteration n , an instance 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 the consensus function is to generate non overlapping clusters from the available instance sets at each iteration. To do this, the following approach is proposed (further approaches will be discussed in the next chapter):

MultiCons approach 1:

- An instance set with uniqueness property means that the grouping of the instances based on agreement between DT base clusterings didn't change when considering $DT - 1$ base clusterings. Thus, this is already a strong clustering decision, and the instance set becomes a consensus cluster at the current iteration $n = DT - 1$.
- Instance sets with inclusion property are removed, to consider new clustering decision made by $DT - 1$ base clusterings.
- Intersecting instance sets are grouped together to form a new cluster, as the existence of shared instances between them is an indication of the closeness of these sets in the data space (other possibilities are discussed in the next chapter).

The above processing is repeated until reaching all the sets as unique sets, that is, do not overlap. After generating all the consensus candidates, repeated ones are removed. Jaccard index is used to indicate if two consensus candidates are identical (thus, the one with lower DT is removed). A *STability* (ST) counter is used to indicate how many times a consensus is generated by various values of DT . Note that we call all the generated solutions as consensuses, because they are generated from agreements between base clusterings. However, it is obvious that the first consensus (the data fragments), and the last one that groups all the instances in one cluster (if generated), are usually not acceptable solutions.

Which of the remaining consensus candidates is the final clustering solution? The high stability of a consensus candidate may reflect its strength. But, as this may not always occur, the possible answer to this question is to choose the consensus that is the most similar to the ensemble. To estimate this, each generated candidate is compared using Jaccard index against each clustering in the ensemble. The one that scores the highest average similarity is recommended as the final solution.

Figure 3.4 illustrates the relationship between the pattern space (the FCP lattice) and what each subspace (level in the lattice) gives as clustering information about the instances in the original data space. Consider the upper subspace is the one with the highest DT value (the one with the 6 colored rectangles), that corresponds to generating data fragments in the original data space (the 6 colored convex clusters). Going to the next subspace with $DT - 1$, only 2 new patterns are added, that define a new clustering for some of the instances. Performing the presented consensus clustering process of the patterns at $DT - 1$, the consensus solution is shown on the right. Continuing deeper in the lattice presents other clustering information from lower number of agreements between the base clusterings.

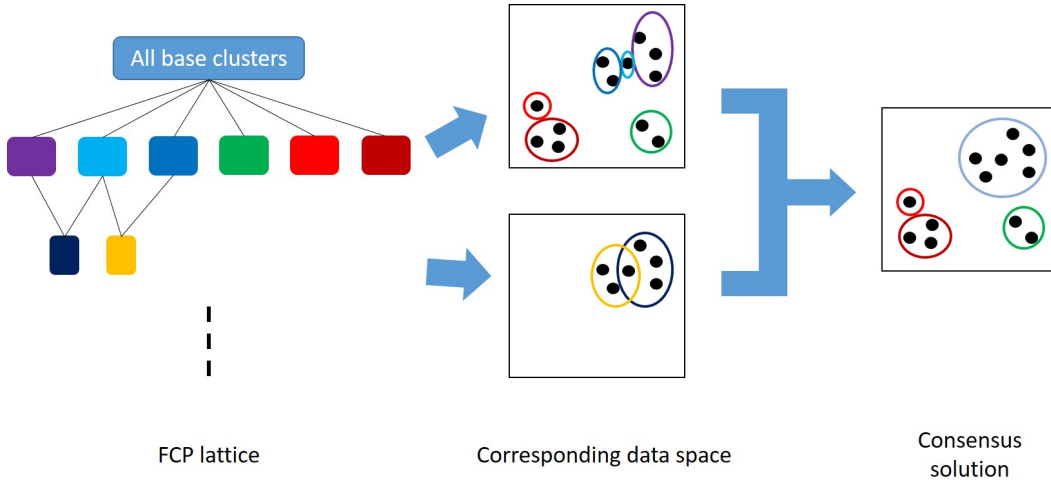


Figure 3.4 – Illustration of the relationship between the pattern space and the original data space.

Algorithms 9 and 10 explain the MultiCons method and approach 1 for the consensus function. Going back to the running example:

- We start with $DT=5$. The clustering patterns with itemsets of size 5 are 1, 2, 3, and 4 (table 3.4). The instance sets of these patterns are all unique, and become the clusters of the first consensus.
- At $DT=4$, FCP 5 is added to the previous sets. FCPs 1 and 2 are removed as they become subsets of FCP 5. This represents a new clustering of the instances as suggested by 4 base clusterings. As there are no new grouping

Algorithm 9: MultiCons

Input : Dataset to cluster
Output : ConsTree tree of consensuses, list of consensus clustering vectors

- 1 Generate clustering ensemble of the dataset and store the clustering vectors in a list *BaseClusterings*
- 2 Calculate in-ensemble similarity (Algo. 8)
- 3 Build the cluster membership matrix $\mathcal{M}$
- 4 Generate FCPs from $\mathcal{M}$ for *minsupport* = 0
- 5 Sort the FCPs in ascending order according to the size of the instance sets
- 6 $MaxDT \leftarrow \text{length}(BaseClusterings)$
- 7 $BiClust \leftarrow \{\text{instance sets of FCPs built from } MaxDT \text{ base clusters}\}$
- 8 Assign a label to each set in *BiClust* to build the first consensus vector and store it in a list of vectors *ConsVctrs*
- 9 **/* Build the remaining consensuses */**
- 10 **for** $DT = (MaxDT - 1)$ **to** 1 **do**
- 11 $BiClust \leftarrow BiClust \cup \{\text{instance sets of FCPs built from } DT \text{ base clusters}\}$
- 12 Call the consensus function (Algo. 10)
- 13 Assign a label to each set in *BiClust* to build a consensus vector and add it to *ConsVctrs*
- 14 **end**
- 15 **/* Remove similar consensuses */**
- 16 $ST \leftarrow \text{Vector of '1's of length } MaxDT$
- 17 **for** $i = MaxDT$ **to** 2 **do**
- 18 $V_i \leftarrow i^{\text{th}}$ consensus in *ConsVctrs*
- 19 **for** $j = (i - 1)$ **to** 1 **do**
- 20 $V_j \leftarrow j^{\text{th}}$ consensus in *ConsVctrs*
- 21 **if** $Jaccard(V_i, V_j) = 1$ **then**
- 22 $ST[i] \leftarrow ST[i] + 1$
- 23 Remove $ST[j]$
- 24 Remove V_j from *ConsVctrs*
- 25 **end**
- 26 **end**
- 27 **/* Find the consensus the most similar to the ensemble */**
- 28 $L \leftarrow \text{length}(ConsVctrs)$
- 29 $TSim \leftarrow \text{Vector of '0's of length } L$
- 30 **for** $i = 1$ **to** L **do**
- 31 $C_i \leftarrow i^{\text{th}}$ consensus in *ConsVctrs*
- 32 **for** $j = 1$ **to** $MaxDT$ **do**
- 33 $C_j \leftarrow j^{\text{th}}$ clustering in *BaseClusterings*
- 34 $TSim[i] \leftarrow TSim[i] + Jaccard(C_i, C_j)$
- 35 **end**
- 36 $TSim[i] \leftarrow \frac{TSim[i]}{MaxDT}$
- 37 **end**
- 38 $RecommCons \leftarrow \text{which.max}(TSim)$
- 39 Build the ConsTree (Algo. 11)

Algorithm 10: Consensus clustering approach 1	
	Input : $BiClust$ (set of instance sets) Output: Modified $BiClust$ (set of unique instance sets) <pre> 1 $N \leftarrow BiClust$ 2 repeat 3 for $i = 1$ to N do 4 $B_i \leftarrow i^{\text{th}}$ set in $BiClust$ 5 for $j = 1$ to N, $j \neq i$ do 6 $B_j \leftarrow j^{\text{th}}$ set in $BiClust$ 7 $IntrscSz \leftarrow B_i \cap B_j$ 8 if $IntrscSz = 0$ then 9 Next j 10 else if $IntrscSz = B_i$ then 11 /* $B_i \subset B_j$ */ 12 Remove B_i from $BiClust$ 13 Next i 14 else if $IntrscSz = B_j$ then 15 /* $B_j \subset B_i$ */ 16 Remove B_j from $BiClust$ 17 Next j 18 else 19 $B_j \leftarrow B_i \cup B_j$ 20 Remove B_i from $BiClust$ 21 Next i 22 end 23 end 24 until All sets in $BiClust$ are unique </pre>

for the instances in FCPs 3 and 4, they will remain unchanged. Consequently, the consensus solution at $DT=4$ is $\{\{1,2,3\},\{4,5,6,7\},\{8,9\}\}$.

- We have no patterns at $DT=3$. This means that there are no other instance sets that can be suggested by 3 base clusterings, other than the instance sets that we already processed. Thus, the consensus at $DT=3$ will be exactly the one at $DT=4$. To prevent generating redundant solutions, the new consensus is removed, and the one at $DT=4$ will have $ST=2$.
- The instance set of FCP 7 is added at $DT=2$, resulting in the following consensus: $\{\{1,2,3\},\{4,5,6,7,8,9\}\}$.
- The instance set of FCP 6 is added at $DT=1$, resulting in grouping all the instances in one cluster.

3.2.5 The ConsTree

The final step is presenting all the generated consensus 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 value. 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 cluster size.

Definition 10 (The ConsTree) A tree of consensus is an ordered set $(\mathcal{P}, \preceq)$ of consensus $\mathcal{P} = \bigcup_{DT=MinDT}^{DT=MaxDT} \Pi^{DT}$ ordered in descending order of DT values. Let's denote $\Pi^\alpha = \{\pi_1^\alpha, \pi_2^\alpha, \dots, \pi_{k_\alpha}^\alpha\}$ and $\Pi^\beta = \{\pi_1^\beta, \pi_2^\beta, \dots, \pi_{k_\beta}^\beta\}$ the consensus generated for α and β DT values respectively. Let's denote π_q^α the q^{th} cluster in Π^α and π_r^β the r^{th} cluster in Π^β , with $1 \leq q \leq k_\alpha$ and $1 \leq r \leq k_\beta$. For $\alpha > \beta$ we have $\Pi^\alpha \preceq \Pi^\beta$, that is $\forall \pi_q^\alpha \in \Pi^\alpha, \exists \pi_r^\beta \in \Pi^\beta$ such that $\pi_q^\alpha \cap \pi_r^\beta \neq \emptyset$. Π^α is a predecessor of Π^β in the tree of consensus.

The ConsTree of the running example is shown in figure 3.5. It consists of 4 levels, each corresponds to a consensus identified by the DT value. It clearly shows how the clusters are formed from the bottom level (the first consensus) to the root. The cluster of 3 instances $\{1,2,3\}$ is a stable cluster, as the grouping of these instances didn't change, except at the root. Recognizing stable clusters is important as they point to the strong similarity among their instances. The recommended solution is shown in the tree by the red thin line that surrounds it. It is also the most stable solution.

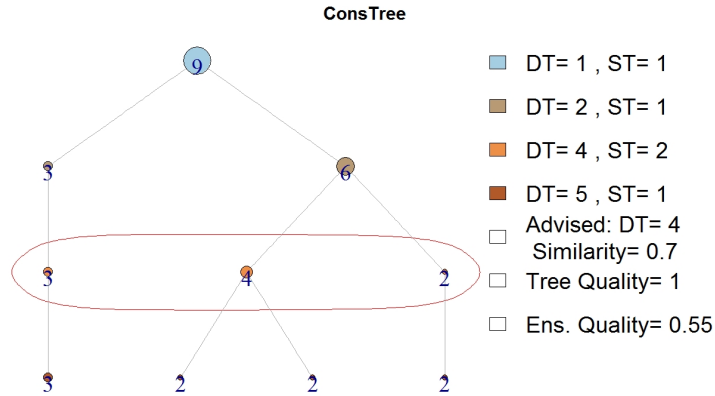


Figure 3.5 – The ConsTree of the running example.

A possible bad result of the consensus clustering approach 1 is the grouping of all instances in one cluster at a high DT value. This will prevent generating useful consensus solutions for sequel values of DT . To illustrate this case, suppose that the instance sets at a specific DT represent nodes in a graph. An edge exists between two nodes if the corresponding instance sets intersect by at least one instance. Approach 1 finds communities of connected nodes, and fuse them into one cluster. If

all nodes are linked, then the consensus solution consists of only one cluster. Therefore, a *tree quality* measure is proposed. It penalizes generating a consensus of one cluster more than once. Tree quality is calculated as follows:

$$\text{Tree quality} = \begin{cases} 1 - \frac{ST[\text{root}] - 1}{MaxDT} & \text{if the consensus at the root of the tree} \\ & \text{consists of one cluster} \\ 1 & \text{otherwise} \end{cases}$$

where $ST[\text{root}]$ is the stability of the consensus at the root of the tree.

If we compare two ConsTrees generated from two different ensembles for the same dataset, then tree quality, the quality of the ensemble, and the similarity of the recommended solution to the ensemble, all can be considered to choose the better solution.

Algorithm 11 shows the pseudo code for building a ConsTree. It is based on the idea of using a table, called in the algorithm *GraphFrm*, to define edges between pairs of nodes in the tree. Each row in the table defines an edge in the form (from, to), where both entries are symbolic names of the nodes. The symbolic name is defined as (“L”, N1, N2) where: “L” stands for level; N1: Numeral identifies the consensus to which the node (cluster) belongs; and N2: Numeral identifies the cluster in the consensus N1, that is, cluster label.

The ConsTree may serve as a data analysis tool. Let’s consider the example tree shown in figure 3.6. It shows the result of applying MultiCons to a dataset of 399 instances, clustered using 10 base clusterings. This tree consists of 7 levels, instead of 10 without merging of identical levels, as duplicated levels for $DT = 2$ and 3, for $DT = 4, 5$ and 6 are merged. Visualizing the tree enables the analyst to understand how the consensus 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 sequence of consensus, which reflect strong similarity between their instances (as the ones circled in blue and red in figure 3.6). It may also highlight groups of instances that are far from being similar to other instances, such as the column of stable cluster circled in red and the similar column beside. 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 more insight on the peculiar information they hold. The ST value can also point to the result that is more stable than others (the consensus for $DT = 6$), but as the clustering task is more related to the relevance of the found patterns to analyst preferences, the analyst may prefer to select the consensus at $DT = 7$ for example, because he/she prefers to separate the cluster of 83 instances at $DT = 6$ into 2 stable clusters as in $DT = 7$, as these 2 may reflect better patterns for him/her. Compared to other consensus methods, that only provide one solution representing the clustering that is most similar to the ensemble, the tree of consensus not only provides more information for the user to understand the strong relations in the

Algorithm 11: ConsTree

Input : *ConsVctrs* (List of consensus clustering vectors), *RecommCons*
(The id of the recommended consensus solution)

Output: ConsTree plot

```

1  $L \leftarrow \text{length}(\text{ConsVctrs})$ 
2  $\text{ConsSize} \leftarrow \text{list}(\text{NULL})$ 
3 for  $i = 1$  to  $L$  do
4    $\text{ConsSize}[i] \leftarrow$  calculate the size of the clusters in  $\text{ConsVctrs}[i]$ , and store
   it in a table of 2 columns  $\text{table}(\text{ClustLabl}, \text{Size})$ 
5 end
6  $\text{ClusConn} \leftarrow$  table of  $L$  columns, each correspond to a vector in  $\text{ConsVctrs}$ 
7  $\text{ClusConn} \leftarrow \text{unique}(\text{ClusConn})$ , that is, remove duplicated rows
8  $\text{GraphFrm} \leftarrow \text{table}(f = \text{NULL}, t = \text{NULL})$  /* Stores the edge
   information to link nodes */
9  $\text{GraphVal} \leftarrow \text{NULL}$  /* Stores the value of each node */
10  $\text{GraphCol} \leftarrow \text{NULL}$  /* Stores the color of each node */
11 for  $i = 1$  to  $L$  do
12   if  $i < L$  then
13      $\text{SubConn} \leftarrow \text{unique}(\text{table}(f = \text{ClusConn}[ ,i], t = \text{ClusConn}[ ,i+1])),$ 
     where  $\text{ClusConn}[ ,i]$  returns all the values of  $\text{ClusConn}$  in column  $i$ 
14      $\text{GraphFrm} \leftarrow \text{attach}(\text{GraphFrm}, \text{table}(f = \text{paste}("L", i, \text{SubConn}[ ,1]),$ 
      $t = \text{paste}("L", i+1, \text{SubConn}[ ,2])))$  /* Symbolic ids to
     distinguish nodes are built with the paste operation */
15   for  $j$  in  $\text{unique}(\text{ClusConn}[ ,i])$  do
16      $\text{GraphVal} \leftarrow \text{attach}(\text{GraphVal}, \text{size of cluster } j \text{ in } \text{ConsSize}[i])$ 
17   end
18    $\text{GraphCol} \leftarrow \text{attach}(\text{GraphCol}, \text{vector of } i \text{ values repeated } \# \text{ of clusters in}$ 
    $\text{ConsVctrs}[i])$ 
19 end
20 Plot the ConsTree by drawing an edge between each pair of nodes defined in
    $\text{GraphFrm}$ , with node's value and color as defined in  $\text{GraphVal}$  and
    $\text{GraphCol}$ . Highlight the nodes in tree level  $\text{RecommCons}$ 

```

data, but it also assists him/her to choose a final clustering based on his/her prior knowledge and preferences.

3.3 Example

Let's consider the *cassini* dataset. This synthetic dataset is already used as an example for testing consensus clustering methods in [Dimitriadou 2002], [Leisch 1999], and [Lozano 2011]. The dataset consists of 1000 instances, each represents a point in a 2D space, forming a structure of three clusters, as shown in figure 3.7a. Let's

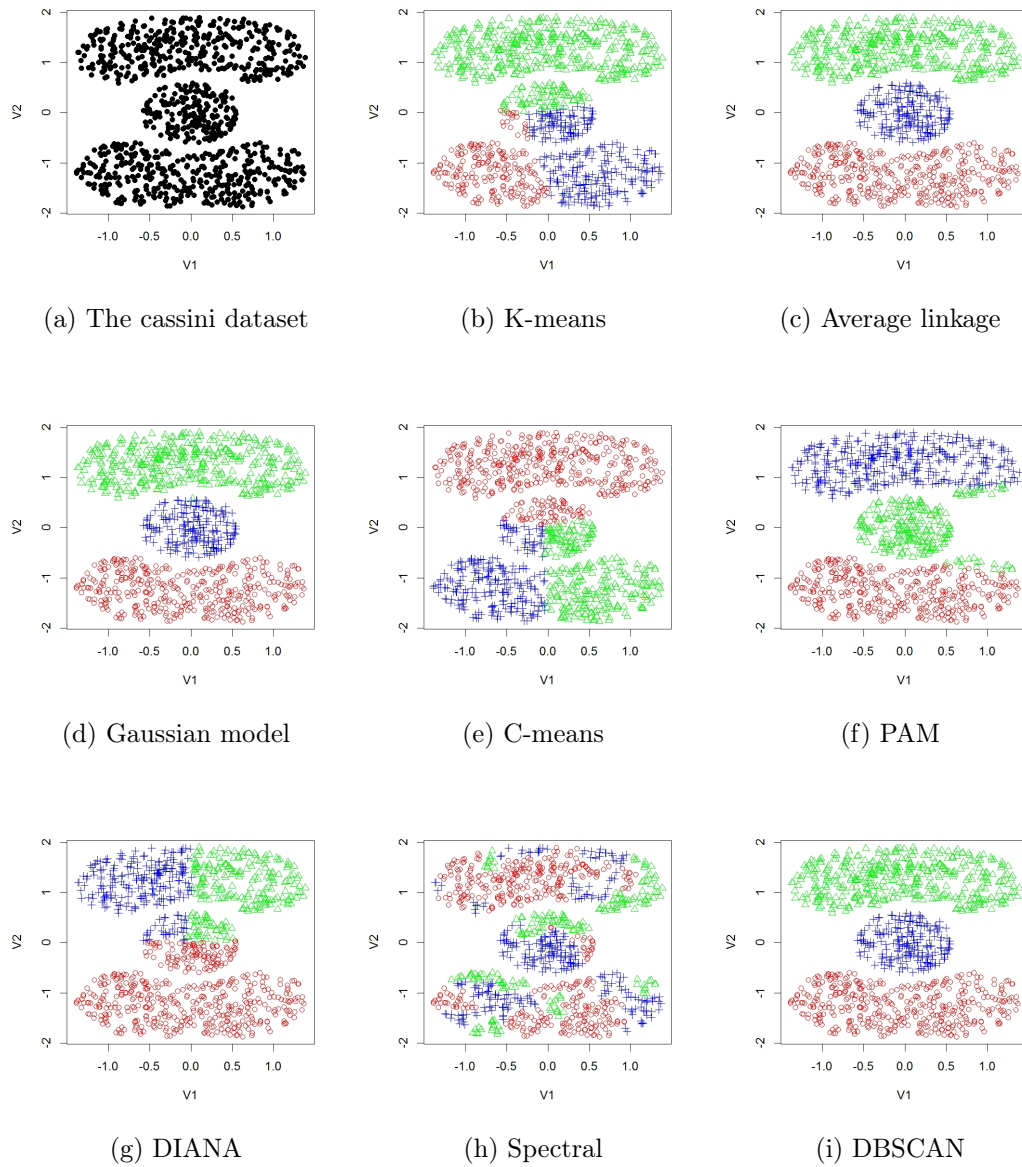


Figure 3.7 – The cassini dataset with 8 different base clusterings.

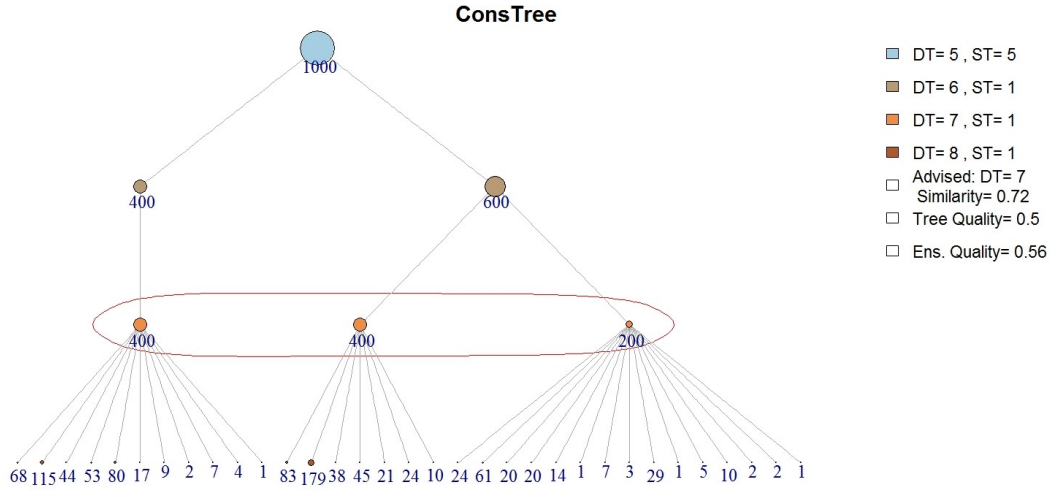


Figure 3.8 – The ConsTree for the cassini dataset.

life problems, K is difficult to predict, especially when there is no domain knowledge about the hidden cluster structure in the dataset. Hence, the ensemble is normally built by choosing a set of clustering algorithms from different categories, to ensure better discovery of cluster boundaries. Each base clustering partitions the dataset into K clusters, where K is chosen randomly within a range of values that define the imaginary or the preferred number of clusters required by the analyst. Identical partitions in the ensemble are removed to not bias the consensus towards them.

Let's try to redo the cassini dataset test using the ensemble generation process presented above. Considering that we do not know the actual number of clusters, the ensemble consists of the following partitions with K in the range $[2, 7]$: K-means with $K = 6$, average linkage with $K = 5$, Gaussian model with $K = 3$, C-means

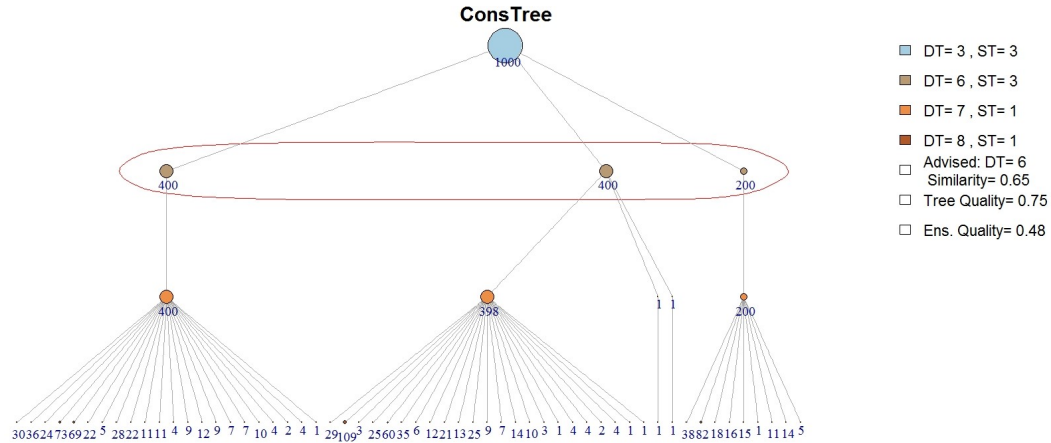


Figure 3.9 – Another ConsTree for the cassini dataset.

with $K = 7$, PAM with $K = 3$, DIANA with $K = 2$, spectral clustering with $K = 6$, and DBSCAN with $Eps = 0.13$ and $MinPts = 3$ which resulted in generating 4 clusters with 2 instances identified as noise. The result is shown in the ConsTree of figure 3.9. The recommended solution is also identical to the correct clustering, however, we have now a better tree, as this solution is clearly the most stable one.

More tests are presented in chapter 5. In the next chapter, other consensus clustering approaches are proposed to overcome the problem of generating a one cluster consensus, based on defining a new measure for merging intersecting instance sets.

Enhancements

Contents

4.1	A Measure of Similarity Between Instance Sets	57
4.2	MultiCons Approach 2	58
4.3	MultiCons Approach 3	60
4.4	MultiCons Approach 4	62
4.5	MultiCons Approach 5	64
4.6	Enhancing the ConsTree	64
4.7	Default values for MT	66
4.8	Example	66

As explained in the previous chapter, approach 1 may, in some situations, fail to identify groups of instances and result in a unique cluster containing all instances. Let's take as an illustrative example a dataset of nine instances $\mathcal{D} = \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$. Suppose that we used 5 base clustering algorithms to produce 5 partitions of $\mathcal{D}$ as follows: $P1 = \{\{1, 2, 3, 4\}, \{5, 6, 7, 8, 9\}\}$, $P2 = \{\{1, 2, 3\}, \{4, 5, 6, 7, 8, 9\}\}$, $P3 = \{\{1, 2, 3, 4, 5\}, \{6, 7, 8, 9\}\}$, $P4 = \{\{6, 7\}, \{1, 2, 3, 4, 5, 8, 9\}\}$, and $P5 = \{\{4, 5, 6, 7, 8, 9\}, \{1, 2\}\}$. Tables 4.1 and 4.2 present the corresponding membership matrix and the clustering patterns. Imagine each instance set represents a node in a graph. Nodes are linked by edges if there are shared instances between them. In figure 4.1, each node is identified by the FCP ID from table 4.2. At $DT = 5$, all the nodes are disconnected. At $DT = 4$, FCPs 4, 5, 10, 11, and 14 are added. The nodes from $DT = 5$ (green nodes) become subsets of the nodes at $DT = 4$ (blue nodes), thus will be removed. For the remaining nodes, we can see that they are linked together (the red lines) by the shared instances between adjacent nodes. Hence, they will become a single cluster that groups all the instances.

Normally, having all the nodes connected occur at low DT values, whereas at high DT , approach 1 will be able to generate clusters from unlinked communities of connected nodes. However, this example shows one of the difficult cases where approach 1 may fail. Thus, to overcome this problem, new measures and approaches are proposed in this chapter.

Table 4.1 – Example membership matrix.

Instance ID	P_1^1	P_2^1	P_1^2	P_2^2	P_1^3	P_2^3	P_1^4	P_2^4	P_1^5	P_2^5
1	1	0	1	0	1	0	0	1	0	1
2	1	0	1	0	1	0	0	1	0	1
3	1	0	1	0	1	0	0	1	1	0
4	1	0	0	1	1	0	0	1	1	0
5	0	1	0	1	1	0	0	1	1	0
6	0	1	0	1	0	1	1	0	1	0
7	0	1	0	1	0	1	1	0	1	0
8	0	1	0	1	0	1	0	1	1	0
9	0	1	0	1	0	1	0	1	1	0

Table 4.2 – Clustering patterns extracted from table 4.1.

FCP ID	Itemset (FCIs)	Instance ID set
1	$\{P_1^1, P_2^1, P_1^3, P_2^4, P_1^5\}$	$\{3\}$
2	$\{P_1^1, P_2^2, P_1^3, P_2^4, P_1^5\}$	$\{4\}$
3	$\{P_2^1, P_2^2, P_1^3, P_2^4, P_1^5\}$	$\{5\}$
4	$\{P_1^1, P_1^3, P_2^4, P_1^5\}$	$\{3,4\}$
5	$\{P_2^2, P_1^3, P_2^4, P_1^5\}$	$\{4, 5\}$
6	$\{P_1^1, P_2^1, P_1^3, P_2^4, P_2^5\}$	$\{1,2\}$
7	$\{P_2^1, P_2^2, P_2^3, P_1^4, P_1^5\}$	$\{6,7\}$
8	$\{P_2^1, P_2^2, P_2^3, P_2^4, P_1^5\}$	$\{8,9\}$
9	$\{P_1^3, P_2^4, P_1^5\}$	$\{3,4,5\}$
10	$\{P_1^1, P_2^1, P_1^3, P_2^4\}$	$\{1,2,3\}$
11	$\{P_2^1, P_2^2, P_2^4, P_1^5\}$	$\{5,8,9\}$
12	$\{P_1^1, P_1^3, P_2^4\}$	$\{1,2,3,4\}$
13	$\{P_2^2, P_2^4, P_1^5\}$	$\{4,5,8,9\}$
14	$\{P_2^1, P_2^2, P_2^3, P_1^5\}$	$\{6,7,8,9\}$
15	$\{P_1^3, P_2^4\}$	$\{1,2,3,4,5\}$
16	$\{P_2^4, P_1^5\}$	$\{3,4,5,8,9\}$
17	$\{P_2^1, P_2^2, P_1^5\}$	$\{5,6,7,8,9\}$
18	$\{P_2^2, P_1^5\}$	$\{4,5,6,7,8,9\}$
19	$\{P_2^4\}$	$\{1,2,3,4,5,8,9\}$
20	$\{P_1^5\}$	$\{3,4,5,6,7,8,9\}$

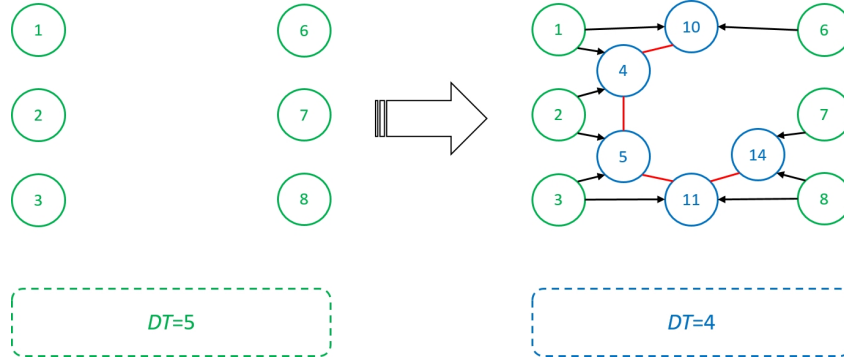


Figure 4.1 – Illustrative example of the relations between instance sets as nodes in a graph.

4.1 A Measure of Similarity Between Instance Sets

Instead of merging two intersecting sets whatever the number of shared instances between them, we can use the size of intersection to determine whether to merge them or not. The idea of measuring the similarity between sets based on their intersection is not new. Jaccard index [Jaccard 1912] is based on calculating the ratio between intersection size over the size of the union of 2 sets:

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

Consider the three cases of sets intersection in figure 4.2. By calculating the Jaccard score for each case we get:

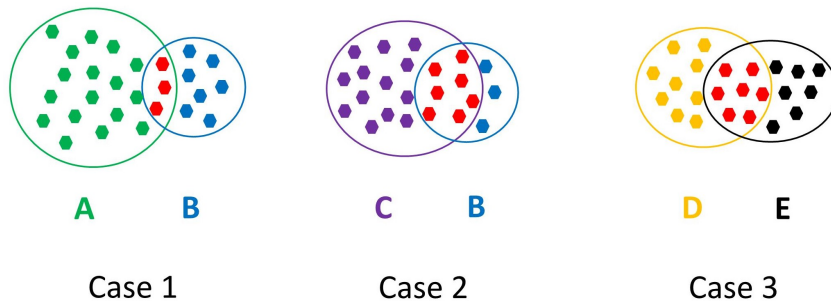


Figure 4.2 – Examples of sets intersection.

$$Jaccard(A, B) = \frac{3}{27}$$

$$Jaccard(B, C) = \frac{7}{23}$$

$$Jaccard(D, E) = \frac{7}{23}$$

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 . Therefore, instead of using Jaccard, a new measure is proposed:

Let $I(X|Y)$ defines how much of set X is part of set Y as the ratio of intersection between the two sets over the size of set X , that is:

$$I(X|Y) = \frac{|X \cap Y|}{|X|}$$

Calculating $I(X|Y)$ for each case in figure 4.2, we get:

$$\text{Case 1: } I(B|A) = \frac{|A \cap B|}{|B|} = \frac{3}{10}, I(A|B) = \frac{|A \cap B|}{|A|} = \frac{3}{20}$$

$$\text{Case 2: } I(B|C) = \frac{|B \cap C|}{|B|} = \frac{7}{10}, I(C|B) = \frac{|B \cap C|}{|C|} = \frac{7}{20}$$

$$\text{Case 3: } I(E|D) = \frac{|E \cap D|}{|E|} = \frac{7}{14}, I(D|E) = \frac{|D \cap E|}{|D|} = \frac{7}{16}$$

From the above scores, we can see that $I(B|C)$ is the highest, and that the scores provided for cases 2 and 3 are different, compared to Jaccard which assigned them the same score.

Based on this measure of intersection between sets, the following approaches are proposed.

4.2 MultiCons Approach 2

Using a *Merging Threshold* (**MT**), two intersecting sets X and Y will be merged only if either $I(X|Y)$ or $I(Y|X)$ scores greater than or equal to MT . Otherwise, the two sets are split, that is, the shared instances are removed from one of them. The split operation involves keeping the shared instances in the smaller size set (more compact cluster) and remove them from the other bigger size set (more spread cluster). For the sets that do not intersect with others, or those that are subsets, they are treated like in approach 1. To apply approach 2, algorithm 9 is updated by adding MT as input, and algorithm 12 is called in line 12 instead of algorithm 10.

Going back to the example dataset presented in the beginning of this chapter, the first consensus ($DT = 5$) consists of 6 clusters which are the instance sets of FCPs 1, 2, 3, 6, 7, and 8. For the next consensus ($DT = 4$), we add the instance sets of FCPs 4, 5, 10, 11, and 14. However, the clusters of the previous consensus will be removed because they are just subsets of the new sets. Therefore, we will have the following sets: $\{3,4\}$, $\{4,5\}$, $\{1,2,3\}$, $\{5,8,9\}$, and $\{6,7,8,9\}$. With $MT = 0.5$, we start with set $\{3,4\}$ that intersects with set $\{4,5\}$ by 0.5 of their instances, thus they are merged to form the set $\{3,4,5\}$. Next, set $\{3,4,5\}$ intersects with set $\{1,2,3\}$ but

Algorithm 12: Consensus clustering approach 2

```

Input : BiClust (set of instance sets)
Output: Modified BiClust (set of unique instance sets)
1  $N \leftarrow |BiClust|$ 
2 repeat
3   for  $i = 1$  to  $N$  do
4      $B_i \leftarrow i^{\text{th}}$  set in BiClust
5     for  $j = 1$  to  $N$ ,  $j \neq i$  do
6        $B_j \leftarrow j^{\text{th}}$  set in BiClust
7        $IntrscSz \leftarrow |B_i \cap B_j|$ 
8       if  $IntrscSz = 0$  then
9         | Next  $j$ 
10      else if  $IntrscSz = |B_i|$  then
11        | /*  $B_i \subset B_j$  */
12        | Remove  $B_i$  from BiClust
13        | Next  $i$ 
14      else if  $IntrscSz = |B_j|$  then
15        | /*  $B_j \subset B_i$  */
16        | Remove  $B_j$  from BiClust
17        | Next  $j$ 
18      else if  $(IntrscSz \geq |B_i| \times MT)$  or  $(IntrscSz \geq |B_j| \times MT)$  then
19        | /* merge intersecting sets */
20        |  $B_j \leftarrow B_i \cup B_j$ 
21        | Remove  $B_i$  from BiClust
22        | Next  $i$ 
23      else
24        | /* split intersecting sets */
25        | if  $|B_i| \leq |B_j|$  then
26        | |  $B_j \leftarrow B_j \setminus B_i$ 
27        | else
28        | |  $B_i \leftarrow B_i \setminus B_j$ 
29      end
30    end
31 until All sets in BiClust are unique

```

by 0.3 which is less than MT , thus they are split into sets $\{3,4,5\}$ and $\{1,2\}$. The same split process is performed for sets $\{3,4,5\}$ and $\{5,8,9\}$ to form sets $\{3,4,5\}$ and $\{8,9\}$. For the remaining sets, we find that set $\{8,9\}$ is a subset of $\{6,7,8,9\}$, thus it is removed. The final clusters at $DT = 4$ are: $\{3,4,5\}$, $\{1,2\}$, and $\{6,7,8,9\}$. The same process is performed for the remaining DT values. The difference between the results of approach 1 and approach 2 with $MT = 0.5$ is shown in figure 4.3. The recommended solution in approach 2 is a good partitioning of the dataset if we

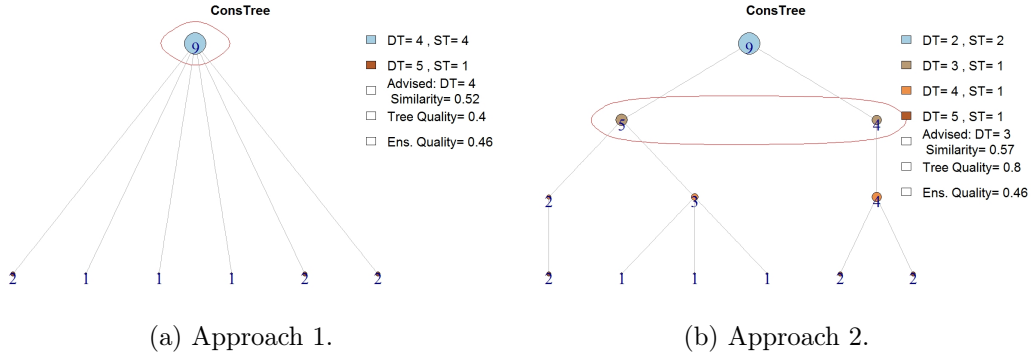


Figure 4.3 – The difference in results between approaches 1 and 2 for the example in table 4.1.

consider the similarities in table 4.1.

4.3 MultiCons Approach 3

In this approach, more restriction is added to the merging process: Two intersecting sets X and Y are merged only if their average intersection ratio is greater than or equal to MT . The average intersection ratio is defined as:

$$Avg_I(X, Y) = \left(\frac{|X \cap Y|}{|X|} + \frac{|X \cap Y|}{|Y|} \right) \times 0.5$$

Going back to the sets example in figure 4.2:

$$Avg_I(A, B) = \left(\frac{3}{20} + \frac{3}{10} \right) \times 0.5 = 0.225$$

$$Avg_I(C, B) = \left(\frac{7}{20} + \frac{7}{10} \right) \times 0.5 = 0.525$$

$$Avg_I(D, E) = \left(\frac{7}{16} + \frac{7}{14} \right) \times 0.5 = 0.469$$

Thus, the average intersection ratio still recognizes case 2 as a better merging decision than the other cases.

Approach 2 checks intersecting sets against a merging threshold, MT , to decide whether to merge or split them. However, it directly merges or splits an instance set with the first one that intersects with it, and then continues to check the other intersecting sets. This is enhanced in approach 3, as it may seem better to search, for each instance set, which of the remaining sets has the best average intersection ratio with it. Then, the two sets are merged if $Avg_I \geq MT$, otherwise, they are split. Algorithm 13 presents the pseudo code of approach 3.

Algorithm 13: Consensus clustering approach 3

Input : $BiClust$ (set of instance sets)
Output: Modified $BiClust$ (set of unique instance sets)

```

1  $N \leftarrow |BiClust|$ 
2 repeat
3   for  $i = 1$  to  $N$  do
4      $B_i \leftarrow i^{\text{th}}$  set in  $BiClust$ 
5      $BestIntrsc \leftarrow 0$ 
6      $Index \leftarrow 0$ 
7     for  $j = 1$  to  $N$ ,  $j \neq i$  do
8        $B_j \leftarrow j^{\text{th}}$  set in  $BiClust$ 
9        $IntrscSz \leftarrow |B_i \cap B_j|$ 
10      if  $IntrscSz = 0$  then
11        Next  $j$ 
12      else if  $IntrscSz = |B_i|$  then
13        /*  $B_i \subset B_j$  */
14        Remove  $B_i$  from  $BiClust$ 
15        Next  $i$ 
16      else if  $IntrscSz = |B_j|$  then
17        /*  $B_j \subset B_i$  */
18        Remove  $B_j$  from  $BiClust$ 
19        Next  $j$ 
20      else
21         $IntrscRatio \leftarrow \left( \frac{IntrscSz}{|B_i|} + \frac{IntrscSz}{|B_j|} \right) \times 0.5$ 
22        if  $IntrscRatio > BestIntrsc$  then
23           $BestIntrsc \leftarrow IntrscRatio$ 
24           $Index \leftarrow j$ 
25      end
26    if  $BestIntrsc > 0$  then
27       $j \leftarrow Index$ 
28       $B_j \leftarrow j^{\text{th}}$  set in  $BiClust$ 
29      if  $BestIntrsc \geq MT$  then
30        /* merge */
31         $B_j \leftarrow B_i \cup B_j$ 
32        Remove  $B_i$  from  $BiClust$ 
33      else
34        /* split */
35        if  $|B_i| \leq |B_j|$  then
36           $B_j \leftarrow B_j \setminus B_i$ 
37        else
38           $B_i \leftarrow B_i \setminus B_j$ 
39      end
40  until All sets in  $BiClust$  are unique

```

4.4 MultiCons Approach 4

Although approach 3 tries to enhance the merging process by searching, for each instance set, which other set best intersect with it, it reads the instance sets sequentially. One can think of enhancing this by searching for the best intersecting sets among the available ones. Such approach requires a matrix of intersection ratios and using a pointer vector to point to the sequence of the merging/splitting process. The matrix consists of $S \times S$ cells, where S is the number of instance sets at iteration DT . Either the intersection ratio measure (approach 2) or the average intersection ratio (approach 3) can be used to define the value of each cell.

Algorithm 14 explains the approach in detail. It starts by constructing the *Intersection Matrix*, where each cell defines the intersection ratio $I(i|j)$. Next, the maximum intersection ratio for each instance set is assigned to a pointer vector, which is sorted according to these values to ensure processing the best intersections first. If the best intersection ratio for a set is greater than or equal to the merging threshold MT , then it is merged, else it is split as in the previous approaches. If an instance set is updated by a merge or split operation, then all its connections to other sets will be ignored until updating the intersection matrix in the next loop. Figure 4.4 shows an example of how approach 4 works.

	1	2	3	4	5
1	0	0.1	0.2	0.5	0.3
2	0.9	0	0.8	0.2	0.4
3	0.6	0.1	0	0.3	0.85
4	0.2	0.92	0.1	0	0.5
5	0.4	0.3	0.7	0.11	0

Intersection matrix

1	4	0.5
2	1	0.9
3	5	0.85
4	2	0.92
5	3	0.7

Pointer

(a) Example intersection matrix.

	1	2	3	4	5
1	0	0.1	0.2	0.5	0.3
2	0.9	0	0.8	0.2	0.4
3	0.6	0.1	0	0.3	0.85
4	0.2	0.92	0.1	0	0.5
5	0.4	0.3	0.7	0.11	0

Intersection matrix

4	2	0.92
2	1	0.9
3	5	0.85
5	3	0.7
1	4	0.5

Pointer

→

	1	2	3	4	5
1	0	0	0.2	0	0.3
2	0	0	0	0	0
3	0.6	0	0	0	0.85
4	0	0	0	0	0
5	0.4	0	0.7	0	0

(b) Processing: Sort the pointer, merge/split the best intersecting sets, and remove corresponding matrix entries. After processing all the pointers, a new matrix is created for the newly generated sets. This is repeated until creating a matrix of NULL values, that is, the remaining instance sets do not intersect.

Figure 4.4 – Illustrative example of approach 4.

Algorithm 14: Consensus clustering approach 4

```

Input : BiClust (set of instance sets)
Output : Modified BiClust (set of unique instance sets)

1 repeat
2   Remove instance sets in BiClust that are subsets of others
3    $N \leftarrow |BiClust|$ 
4   if  $N = 1$  then
5     Break
6    $IMatrix \leftarrow$  matrix of size  $N \times N$  of NULL values
7   for  $i = 1$  to  $N - 1$  do
8     for  $j = i + 1$  to  $N$  do
9        $B_i \leftarrow i^{\text{th}}$  set in BiClust
10       $B_j \leftarrow j^{\text{th}}$  set in BiClust
11       $IntrscSz \leftarrow |B_i \cap B_j|$ 
12      if  $IntrscSz = 0$  then
13        Next j
14       $IMatrix[i, j] \leftarrow \frac{IntrscSz}{|B_i|}$  ,  $IMatrix[j, i] \leftarrow \frac{IntrscSz}{|B_j|}$ 
15    end
16  end
17  if all IMatrix is NULL then
18    Break
19   $Pointer \leftarrow \text{table}(Indx\_row=(1:N), Indx\_col=0, Val=0)$ 
20  for  $i = 1$  to  $N$  do
21    if Not all the cells in row  $i$  in IMatrix are NULL then
22       $Pointer\$Val \leftarrow \max(IMatrix[i, :])$ 
23       $Pointer\$Indx\_col \leftarrow \text{which.max}(IMatrix[i, :])$ 
24    end
25  Sort Pointer in descending order of the Val column
26  Remove the rows of Pointer whose Val=0
27  for  $k = 1$  to # of rows of Pointer do
28     $i \leftarrow Pointer\$Indx\_row[k]$ 
29     $j \leftarrow Pointer\$Indx\_col[k]$ 
30     $BstIntrscVal \leftarrow IMatrix[i, j]$ 
31    if BstIntrscVal is not NULL then
32      if  $BstIntrscVal \geq MT$  then
33         $BiClust_i \leftarrow BiClust_i \cup BiClust_j$ 
34        Remove BiClustj from BiClust
35        Set columns and rows  $i$  and  $j$  in IMatrix to NULL
36      else
37        if  $|BiClust_i| \leq |BiClust_j|$  then
38           $BiClust_j \leftarrow BiClust_j \setminus BiClust_i$ 
39          Set column and row  $j$  in IMatrix to NULL
40        else
41           $BiClust_i \leftarrow BiClust_i \setminus BiClust_j$ 
42          Set column and row  $i$  in IMatrix to NULL
43      end
44  until All sets in BiClust are unique

```

4.5 MultiCons Approach 5

We can think about the intersection matrix as an adjacency matrix of a graph. Each cell in the matrix defines the weight of an edge that connects instance set i with j . Instead of passing this matrix into a graph partitioning algorithm that requires the parameter K to generate K clusters, the merge/split process is used. The idea is simple: Remove the edges with weight less than MT , then merge all connected nodes (sets) into a cluster. Thus, the idea is similar to the idea in approach 1, but here the split operation is added. Algorithm 15 presents the approach using the intersection ratio measure, however, the average intersection ratio can be used instead. Note that after removing the weak edges and making clusters of connected nodes, the actual intersecting instance sets with ratio $< MT$ can still have shared instances between them. Therefore, a split operation is performed to build the final non-overlapping clusters.

The idea of approach 5 is illustrated in figure 4.5. Suppose that at a specific DT , we have 6 instance sets, as shown on the left hand side of the figure. The intersection ratio between a pair of sets is shown as edge values. Using approach 1 will result in grouping all these sets in one cluster. However, with approach 5, if we use $MT = 0.5$, all the edges less than this value are removed, resulting in 2 clusters, as shown on the right hand side of the figure.

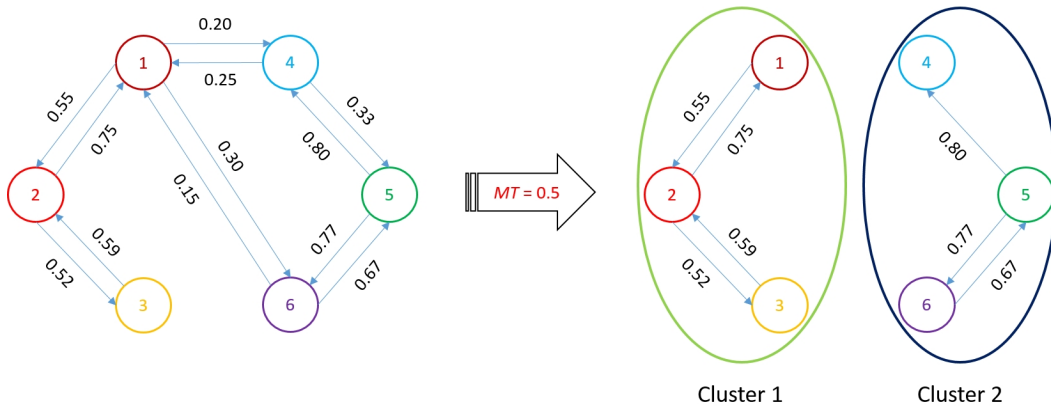


Figure 4.5 – The idea behind approach 5.

4.6 Enhancing the ConsTree

In the previous chapter, the ConsTree is presented as a tool to understand how the instances are grouped at different consensus solutions, and the relationships between successive consensus clusters. As only a merging process is performed in approach 1, the tree is easy to visualize. A cluster at a low tree level is linked to only one cluster at the next upper level. In this chapter, the split operation is added to enhance the consensus clustering result. However, this process yields in different grouping of the

Algorithm 15: Consensus clustering approach 5

```

Input : BiClust (set of instance sets)
Output: Modified BiClust (set of unique instance sets)
1 Remove instance sets in BiClust that are subsets of others
2  $N \leftarrow |BiClust|$ 
3 IMatrix  $\leftarrow$  matrix of size  $N \times N$  of 0 values
4 if  $N > 1$  then
5   for  $i = 1$  to  $N - 1$  do
6     for  $j = i + 1$  to  $N$  do
7        $B_i \leftarrow i^{\text{th}}$  set in BiClust
8        $B_j \leftarrow j^{\text{th}}$  set in BiClust
9        $IntrscSz \leftarrow |B_i \cap B_j|$ 
10      if  $IntrscSz = 0$  then
11        | Next j
12       $IMatrix[i, j] \leftarrow \frac{IntrscSz}{|B_i|}$  ,  $IMatrix[j, i] \leftarrow \frac{IntrscSz}{|B_j|}$ 
13    end
14  end
15 Set to 0 all the cells in IMatrix with value less than MT
16 Build a graph from IMatrix, and make clusters from connected nodes
17 Merge all the instance sets in each cluster
18  $N \leftarrow$  number of clusters
19 /* split the actual intersecting instance sets */
20 if  $N > 1$  then
21   for  $i = 1$  to  $N - 1$  do
22      $C_i \leftarrow i^{\text{th}}$  cluster
23     for  $j = i + 1$  to  $N$  do
24        $C_j \leftarrow j^{\text{th}}$  cluster
25       if  $|C_i \cap C_j| = 0$  then
26         | Next j
27       if  $|C_i| \leq |C_j|$  then
28         |  $C_j \leftarrow C_j \setminus C_i$ 
29       else
30         |  $C_i \leftarrow C_i \setminus C_j$ 
31     end
32   end
33 BiClust  $\leftarrow$  the resulted clusters after the split operation

```

instances when we go up in the tree. That is, a cluster at a low tree level may be linked to two or more clusters at the next level, because some of its instances are grouped differently when we consider a lower number of agreements between the base clusterings. Consequently, the tree may become difficult to visualize, because

of the many links between clusters.

To solve this, a *tree refinement* process is proposed. A cluster at a low tree level x is linked to the cluster at the next upper level y to which most of its instances belong. All other links are removed, which corresponds to removing the instances of x that change their group, considering the majority of instances in x . The refinement process is not intended to alter the consensus solution. It is just to enhance the visualization of the tree to serve for its purpose as an analysis tool. The two trees in figure 4.6 explain the result of this process. The upper tree consists of very few shifts (for clusters in $DT = 8$ to clusters in $DT = 7$), and the lower tree shows the result of tree refinement. The Removed Instances (RI) at the bottom tells how many instances are removed. Algorithm 16 shows the pseudo code for building a refined ConsTree.

4.7 Default values for MT

MT is the only parameter for the MultiCons consensus clustering approaches 2 to 5. It takes any value in the range $[0,1]$, where $MT = 0$ produces the same results of approach 1, that is, merge any intersecting sets, whereas $MT = 1$ splits all intersecting sets, resulting in generating too many small clusters which is not recommended. We can choose $MT = 0.5$ as a good default value to produce moderate merging/splitting of sets. According to extensive tests, this default was found to usually build the best consensus solution (or close to it) for MultiCons approach 2. As approaches 3 and 4 enhance the merging process, tests showed that a lower value of MT can be used to produce the best result. That is, for approach 3, the default MT is 0.4, whereas for approach 4, the default is 0.3. For approach 5, which do not perform iterative merging/splitting as the other approaches, the default is 0.6. It is also found from the analysis of tests results, that using the average intersection ratio, as in approach 3, will limit the range of MT values that can produce good results into the range $[0, 0.7]$.

Therefore, in the following test and the tests in the next chapter, these defaults are used.

4.8 Example

Let's take as an example a synthetic dataset where the instances represent 5 overlapping Gaussian distributed points in a 2D features space, as shown in figure 4.7. Suppose we do not know the real number of possible clusters, thus we tried 6 different clustering algorithms with random choice of K values in the range $[2,9]$ as shown in figure 4.8. Note that two clustering algorithms are not applicable for this dataset of 2000 instances: Spectral clustering because of its high time complexity, and DBSCAN as it failed to find clusters for this dataset (either all instances in 1 cluster, or too many clusters of few instances).

Algorithm 16: Refined ConsTree

Input : *ConsVctrs* (List of consensus clustering vectors), *RecommCons*
(The id of the recommended consensus solution)

Output: ConsTree plot

```

1  $L \leftarrow \text{length}(\text{ConsVctrs})$ 
2  $\text{ClusConn} \leftarrow$  table of  $L$  columns, each corresponds to a vector in ConsVctrs
3  $\text{RmvInst} \leftarrow \text{NULL}$ 
4 for  $i = L$  to 2 do
5    $\text{SubConn} \leftarrow \text{table}(f = \text{ClusConn}[ ,i], t = \text{ClusConn}[ ,i - 1])$ 
6    $\text{FreqTable} \leftarrow \text{table}(f, t, \text{freq})$ , where  $f$  and  $t$  are unique pairs from
    $\text{SubConn}$ , and  $\text{freq}$  counts the frequency of the pair in  $\text{SubConn}$ 
7   Sort  $\text{FreqTable}$  in ascending order of  $f$ , and in descending order of  $\text{freq}$ 
8    $\text{Prev} \leftarrow 0$ 
9   for  $j = 1$  to # of rows of  $\text{FreqTable}$  do
10    if  $\text{FreqTable}\$f[j] = \text{Prev}$  then
11      Add to  $\text{RmvInst} \leftarrow \text{which}(\text{ClusConn}[ ,i] = \text{FreqTable}\$f[j] \ \& \$ 
        $\text{ClusConn}[ ,i - 1] = \text{FreqTable}\$t[j])$ 
12       $\text{Prev} \leftarrow \text{FreqTable}\$f[j]$ 
13    end
14 end
15  $\text{RmvInst} \leftarrow \text{unique}(\text{RmvInst})$  ,  $\text{ConsSize} \leftarrow \text{list}(\text{NULL})$ 
16 for  $i = 1$  to  $L$  do
17    $\text{ConsSize}[i] \leftarrow$  calculate the size of the clusters in  $\text{ConsVctrs}[i]$  after
   removing the instances in  $\text{RmvInst}$ , and store it in a table of 2 columns
    $\text{table}(\text{ClustLabl}, \text{Size})$ 
18 end
19 Remove the rows  $\text{RmvInst}$  from  $\text{ClusConn}$ 
20  $\text{GraphFrm} \leftarrow \text{table}(f = \text{NULL}, t = \text{NULL})$ 
21  $\text{GraphVal} \leftarrow \text{NULL}$ ,  $\text{GraphCol} \leftarrow \text{NULL}$ 
22 for  $i = 1$  to  $L$  do
23   if  $i < L$  then
24      $\text{SubConn} \leftarrow \text{unique}(\text{table}(f = \text{ClusConn}[ ,i], t = \text{ClusConn}[ ,i + 1]))$ 
25      $\text{GraphFrm} \leftarrow \text{attach}(\text{GraphFrm}, \text{table}(f = \text{paste}("L", i, \text{SubConn}[ , 1]),$ 
        $t = \text{paste}("L", i + 1, \text{SubConn}[ , 2])))$  /* Symbolic ids to
       distinguish nodes are built with the paste operation */
26   for  $j$  in  $\text{unique}(\text{ClusConn}[ ,i])$  do
27      $\text{GraphVal} \leftarrow \text{attach}(\text{GraphVal}, \text{size of cluster } j \text{ in } \text{ConsSize}[i])$ 
28   end
29    $\text{GraphCol} \leftarrow \text{attach}(\text{GraphCol}, \text{vector of } i \text{ values repeated } \# \text{ of clusters in } \text{ConsVctrs}[i])$ 
30 end
31 Plot the ConsTree by drawing an edge between each pair of nodes defined in
    $\text{GraphFrm}$ , with node's value and color as defined in  $\text{GraphVal}$  and
    $\text{GraphCol}$ . Highlight the nodes in tree level RecommCons

```

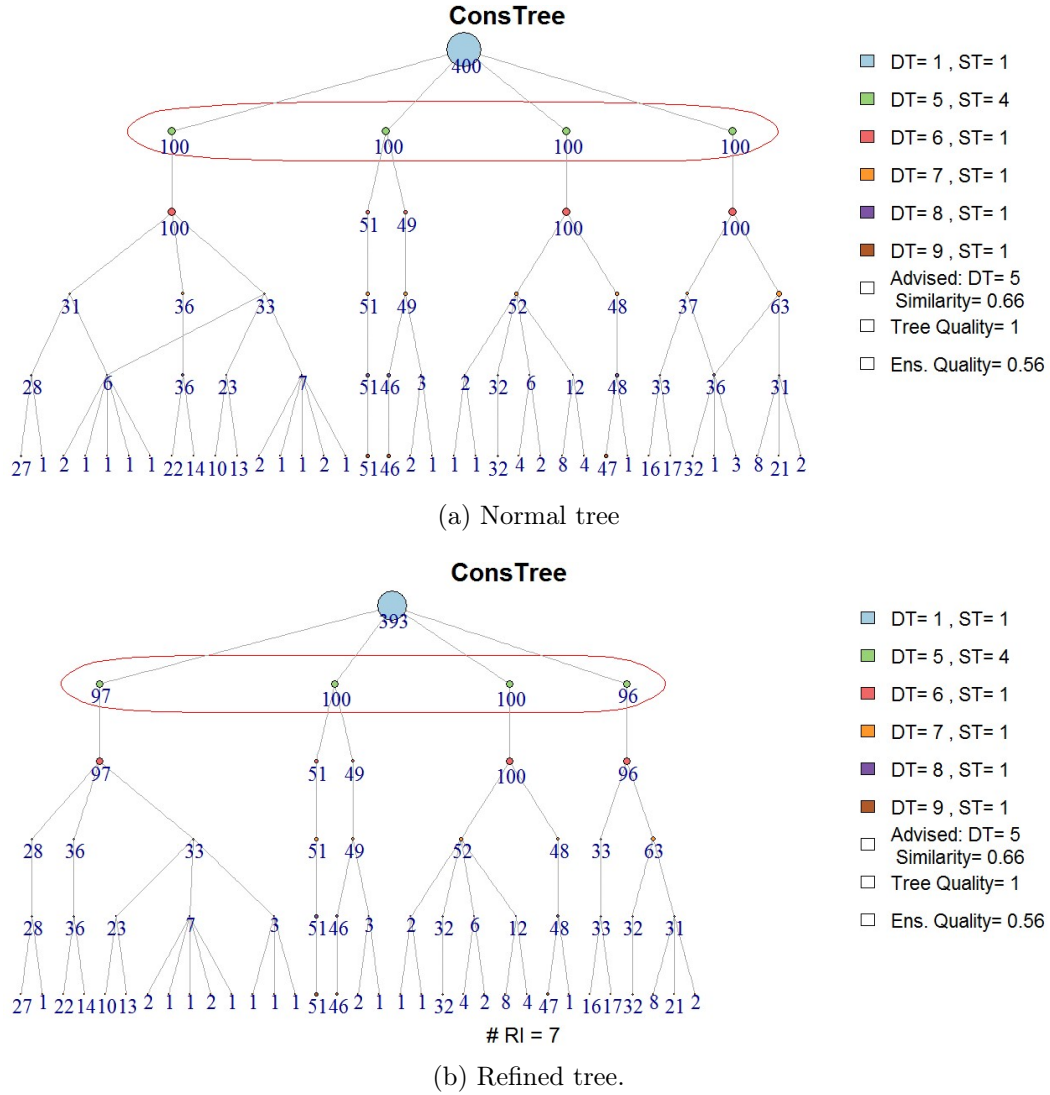


Figure 4.6 – The result of tree refinement.

Applying the 5 MultiCons approaches will produce the ConsTrees shown in figures 4.9 to 4.13. We can see that approaches 1 and 5 failed in generating the correct number of clusters, whereas approaches 2 and 4 were the best. Approach 3 generated 6 clusters instead of 5. We can see in figure 4.15 that these three solutions are much better than the clusterings available in the ensemble.

A refined ConsTree for the results of approach 2 is shown in figure 4.14. It is more clear that the 5 clusters solution is more stable, as the cores of the 5 clusters are formed starting from $DT = 5$ to $DT = 3$. The refined ConsTree eases the understanding of the consensus clustering results when the construction process cannot be easily apprehended from the unrefined ConsTree.

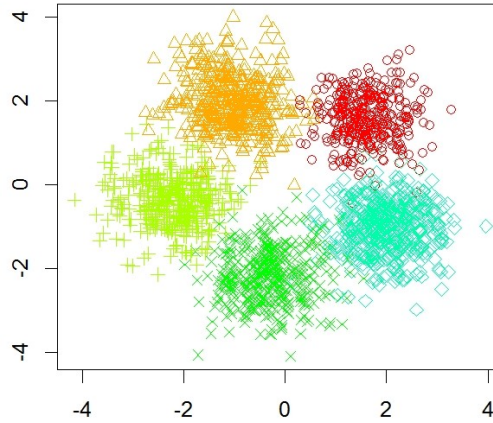
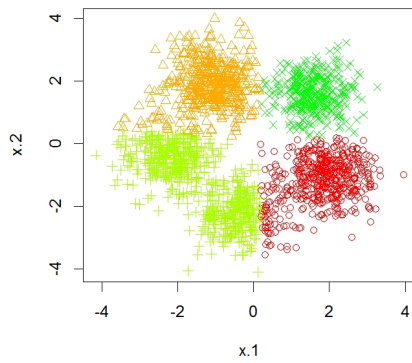


Figure 4.7 – An example dataset of 5 Gaussian distributions in a 2D features space.

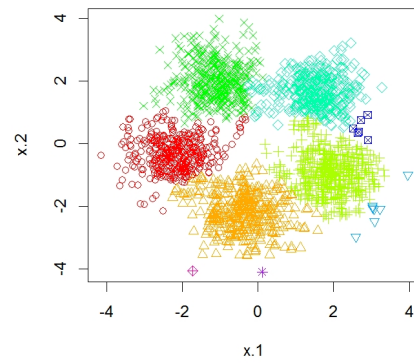
Using external validation measures (section 2.3.2), we can compare the quality of the base clustering algorithms, and the quality of the recommended consensus for the 5 MultiCons approaches. Table 4.3 shows the validation scores using Jaccard and NMI measures. In the next chapter, more tests are presented.

Table 4.3 – Comparison of the quality of the clustering results.

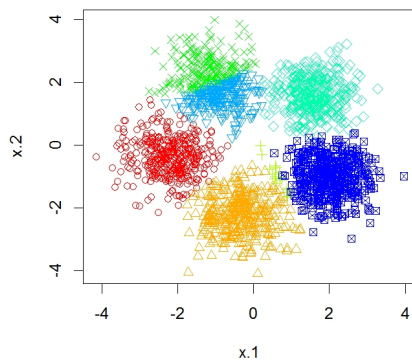
Algorithm	Jaccard	NMI
Kmeans	0.63	0.78
Average linkage	0.80	0.85
Gaussian model	0.81	0.88
Cmeans	0.36	0.52
PAM	0.51	0.71
DIANA	0.66	0.79
Approach1	0.50	0.74
Approach2	0.90	0.92
Approach3	0.84	0.88
Approach4	0.90	0.91
Approach5	0.65	0.81



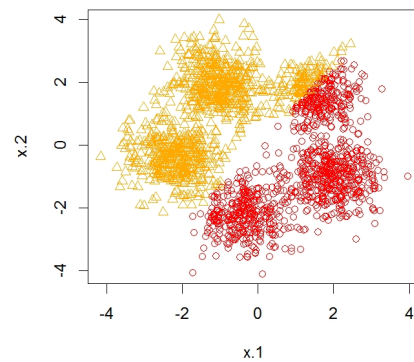
(a) K-means (4 clusters)



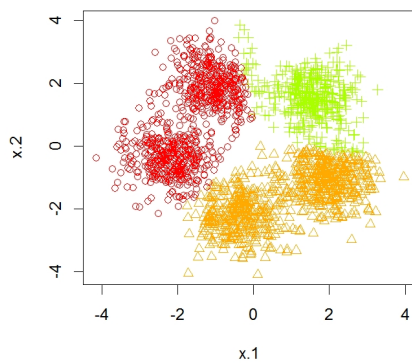
(b) Average linkage (9 clusters)



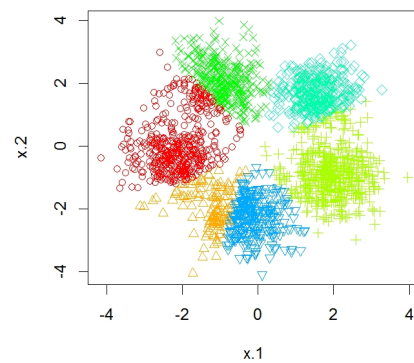
(c) Gaussian model (8 clusters)



(d) C-means (2 clusters)



(e) PAM (3 clusters)



(f) DIANA (6 clusters)

Figure 4.8 – Different clusterings for the dataset in figure 4.7.

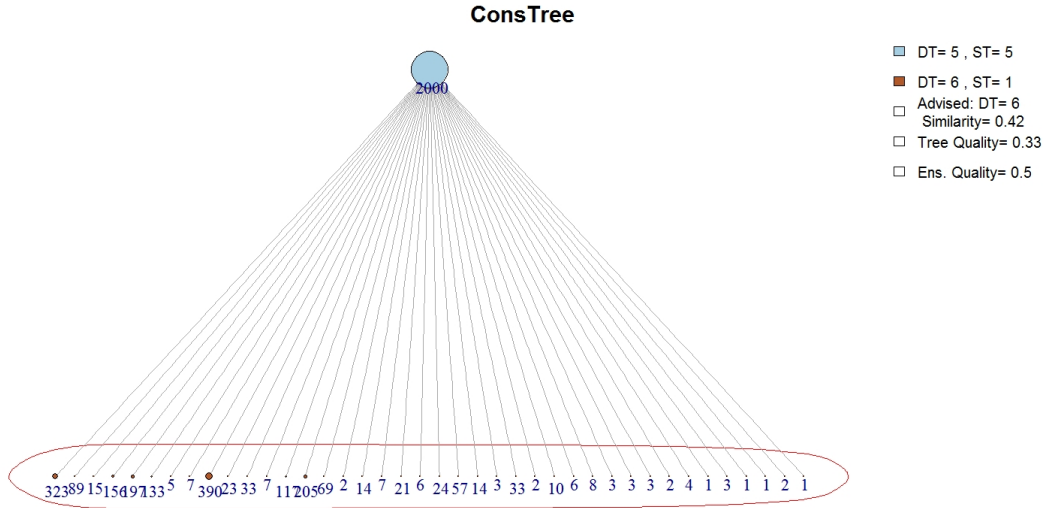


Figure 4.9 – The ConsTree of approach 1 for the ensemble in figure 4.8.

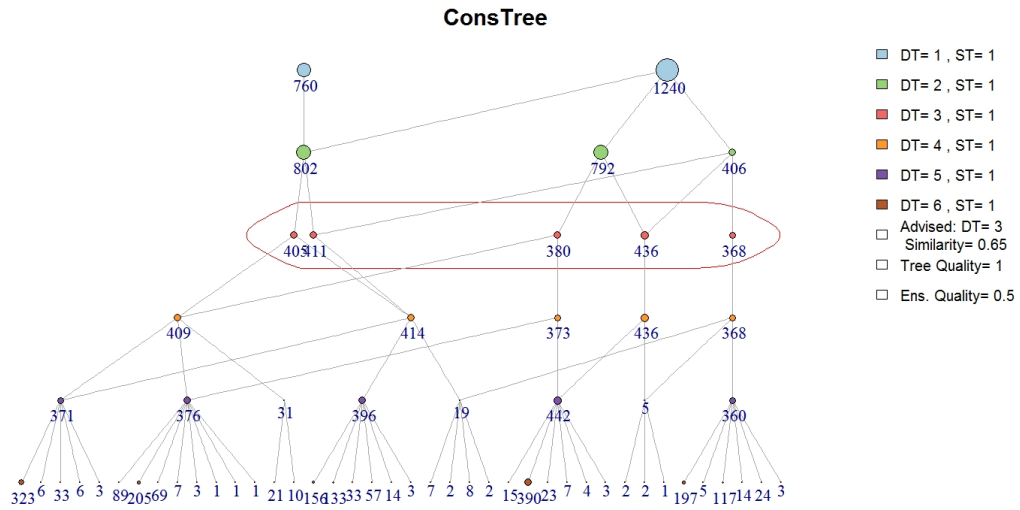


Figure 4.10 – The ConsTree of approach 2 for the ensemble in figure 4.8.

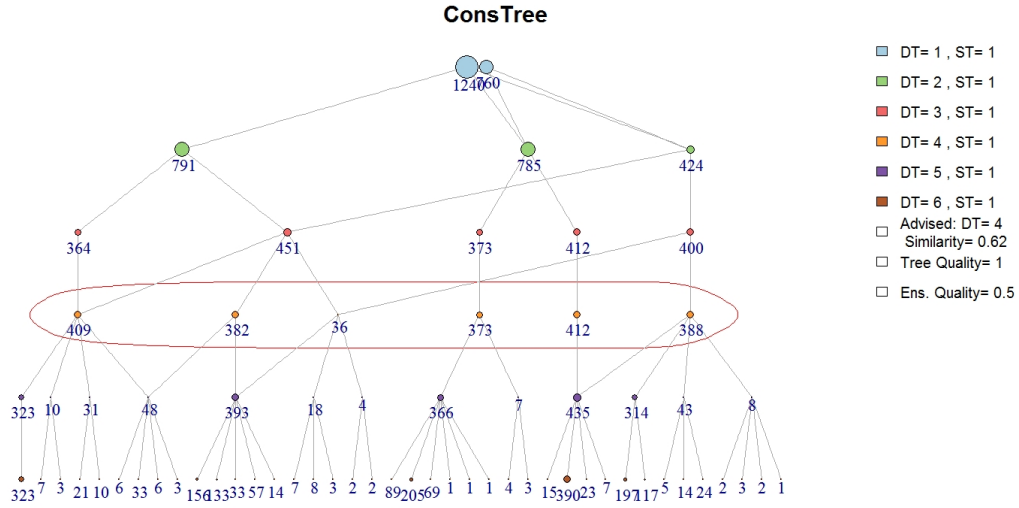


Figure 4.11 – The ConsTree of approach 3 for the ensemble in figure 4.8.

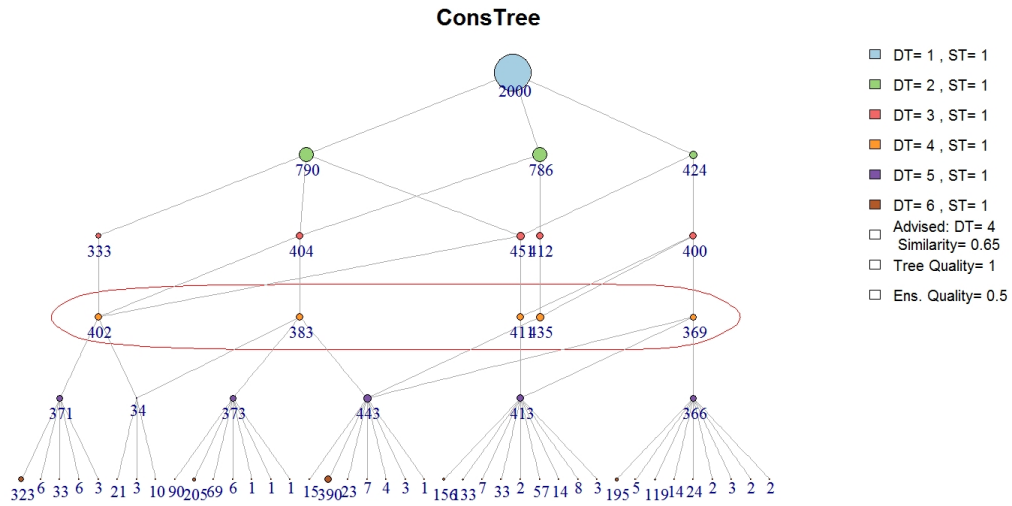


Figure 4.12 – The ConsTree of approach 4 for the ensemble in figure 4.8.

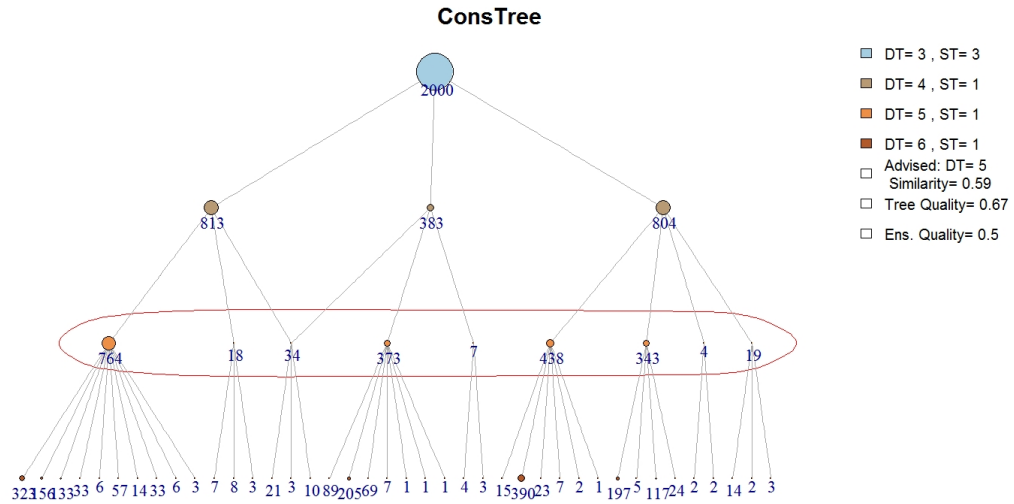


Figure 4.13 – The ConsTree of approach 5 for the ensemble in figure 4.8.

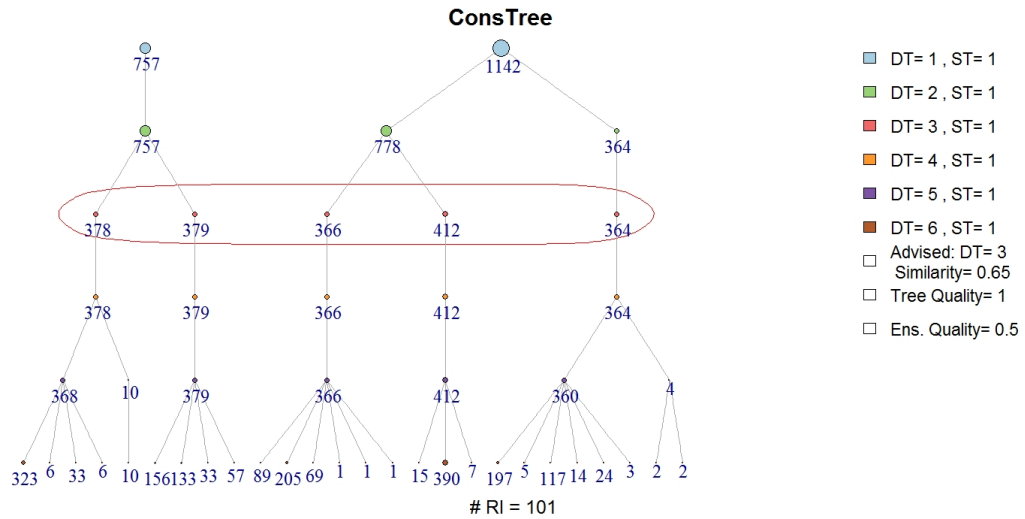
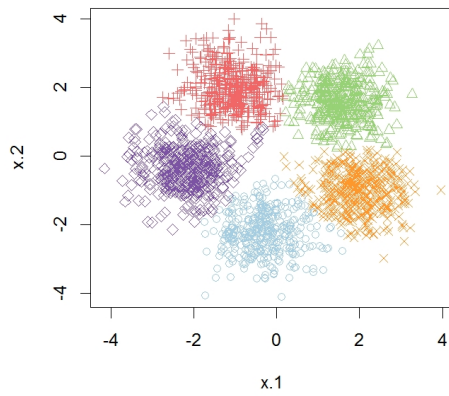
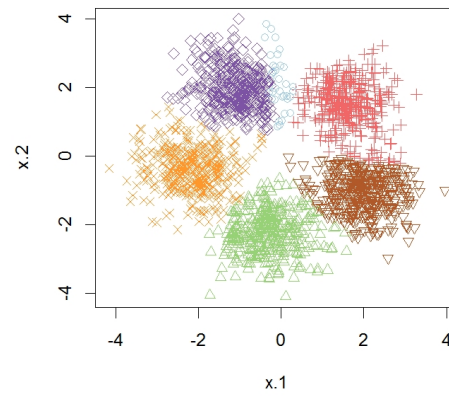


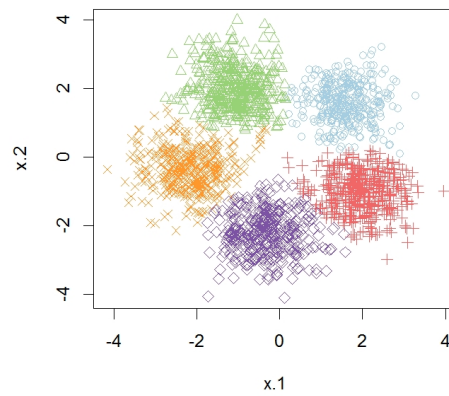
Figure 4.14 – The refined ConsTree for the results in figure 4.10.



(a) Approach 2 (5 clusters)



(b) Approach 3 (6 clusters)



(c) Approach 4 (5 clusters)

Figure 4.15 – The recommended consensus solutions of approaches 2, 3, and 4 for the dataset in figure 4.7.

Experiments

Contents

5.1 Synthetic Datasets	77
5.2 Real Datasets	85
5.3 Summary	90

In the previous chapters, two example datasets that consist of 2D features space are presented to illustrate, with visualization, the performance of clustering algorithms and the achieved consensus solution of MultiCons. In this chapter, more synthetic and real-world datasets with higher number of dimensions and different clustering problems are used to validate the performance of MultiCons.

Tests were run on a DELL PRECISION M4800 with Intel® Core™ i7-4710MQ @ 2.50GHz, 32 GB of RAM, and Microsoft Windows 10 Professional (64-bit) operating system. MultiCons algorithms were implemented using R language [R Core Team 2016]. Function *apriori* in *arules* R package [Hahsler 2005, Hahsler 2011, Hahsler 2016] was used to discover the FCPs, by setting the *target* parameter to “closed frequent itemsets” and *support* = 1 / Datasize.¹ The ConsTree was drawn using the *plot* function in the *igraph* R package [Csardi 2006].

The *CLUE* package [Hornik 2005a, Hornik 2016] in R consists of several relabeling-based consensus clustering methods (see section 2.4.2). Therefore, they are used to compare against MultiCons. For these methods, the *K* parameter is set to the true number of clusters (or classes) known for the tested datasets, whereas MultiCons will try to generate the true clusters (or close to them) based on its clustering process. Other settings for CLUE methods are left to the default values. For MultiCons approaches, *MT* is set to the default values presented in the previous chapter, despite that for some tests, better solutions can be achieved with different *MT* value. The CLUE package includes the following consensus methods [Hornik 2016]:

- DWH: An extension of the greedy algorithm in Dimitriadou *et al.* [Dimitriadou 2002] (see section 2.4.2) for obtaining soft least squares Euclidean consensus partition.

¹A faster algorithm for generating the FCPs called FIST is proposed by [Mondal 2012], and an implementation of it in Java is available on the website of the authors. In addition, [Fournier-Viger 2016] present the SPMF library that contains a large list of Java implemented pattern mining algorithms.

- GV1: Model 1 in Gordon and Vichi [Gordon 2001] (see section 2.4.2) for obtaining soft consensus partition. In this model, a variant of the Euclidean dissimilarity is used: Only matching cluster labels are used in calculating the sum of squared differences between the membership matrices of the ensemble of soft partitions [Hornik 2007].
- SE (Soft/Euclidean): Using all clusters in calculating the Euclidean dissimilarity between soft base partitions, not just the matching ones as in GV1. The result is a soft least squares Euclidean consensus partition. Unlike DWH that performs a single pass over the ensemble, this (and the other methods) perform several passes to enhance the resulted consensus.
- HE (Hard/Euclidean): Like SE, but performed on hard base partitions to obtain hard least squares Euclidean consensus partition.
- SM (Soft/Manhattan): Like SE, but Manhattan dissimilarity is used for obtaining soft median Manhattan consensus.
- HM (Hard/Manhattan): Like SM, but to obtain hard median Manhattan consensus from an ensemble of hard partitions.
- GV3: Model 3 in Gordon and Vichi [Gordon 2001], where co-membership matrices are used to obtain least squares consensus.
- soft/symdiff: Manhattan dissimilarity is used on co-membership matrices of soft base partitions.²
- medoid: One of the partitions in the ensemble that minimizes the distance to the other partitions is chosen as a consensus. Default distance measure is Euclidean.

As the above methods start with an initial guess of the consensus, that is updated in each iteration to better represent the ensemble (like in partitioning based clustering methods where the centroids are updated iteratively), they may result in a different consensus solution at each run. Thus, they are not guaranteed to provide the best result from the first run. This is different from MultiCons results as they do not change over several runs of the algorithm with the same settings. However, finding the best result of CLUE methods is not the objective of the tests. Hence, the presented results are of one run, to be equivalent to one run of MultiCons.

In the following tests, the quality of the clustering solution, whether of the base clusterings, MultiCons, or CLUE methods, is estimated using Jaccard [Jaccard 1912] external validation measure (see section 2.3.2). That is, the resulted clustering is compared against the true clustering available for the datasets in tests. The reason

²Another method is called hard/symdiff, where the distance is the number of distinct pairs of instances in the same cluster in one of the hard partitions. However, this and the soft/symdiff method are very time consuming, thus only the latter is included in the tests.

for choosing Jaccard is that it gives a moderate trade-off between the similarity to the true class and the number of generated clusters.

Each test is defined by the following properties:

- Dataset properties: Size, number of attributes, number of classes, the challenging clustering problem that the dataset imposes (if exist).
- Ensemble size: The number of base clusterings used to build the ensemble.
- K-range: The variable number of clusters generated by the base clusterings is identified by the minimum and maximum K values.
- In-ensemble similarity: The similarity between the base clusterings (see section 3.2.1).
- Ensemble min.: The minimum quality clustering result among the base clusterings.
- Ensemble max.: The maximum quality clustering result among the base clusterings.
- The quality of the recommended solution of a MultiCons approach.
- The number of clusters in the recommended solution of a MultiCons approach.
- The quality of the CLUE methods.

The following clustering algorithms are used (all or some) for building the ensemble for each test: K-Means, PAM (see section 2.2.1), C-Means (see section 2.2.3), agglomerative hierarchical clustering with different linkage, DIANA (see section 2.2.2), Gaussian model-based (see section 2.2.5), DBSCAN (see section 2.2.4), spectral clustering (see section 2.2.8), and bagged clustering [Leisch 1999].³ These clustering methods are either part of R [R Core Team 2016] or available in other packages that need to be added: cluster [Maechler 2016], e1071 [Meyer 2015], flashClust [Langfelder 2012], mclust [Fraley 2002, Fraley 2012], fpc [Hennig 2015], and clusterSim [Dudek 2015]. It is possible to repeat the same algorithm in the ensemble but with different setting, like different K values, to increase the size and the diversity in the ensemble.

5.1 Synthetic Datasets

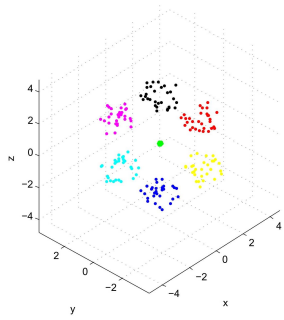
Synthetic datasets are designed to define challenging clustering problems, as clustering algorithms have different responses to situations like variable densities in

³K-means is run over bootstrap samples of the dataset, then all the resulted centroids are passed into agglomerative hierarchical clustering. Although this method can be considered as a consensus clustering approach, we preferred to consider it as an enhanced base clustering, because it uses different ensemble than MultiCons or CLUE methods.

the dataset, overlapping clusters, variable distances between clusters, the existence of noise, etc. Following are some synthetic datasets from Ultsch [Ultsch 2005a, Ultsch 2005b] (the fundamental clustering problem suite):

Hepta:

The dataset is shown in figure 5.1. It represents an easy clustering problem because the clusters are well separated. In such case, using internal validation measures (see section 2.3.1) against different runs of a clustering algorithm with sequential values of K can easily point out to the correct number of clusters (and the correct clustering for this dataset) as the one that results in the best validation score. However, we will not use internal validation, but instead we will combine different clustering results with different K values. From the results in table 5.1, we can recognize that all MultiCons approaches and CLUE methods (except SM) succeeded in generating the correct clustering for this dataset (text in bold represents the best result).



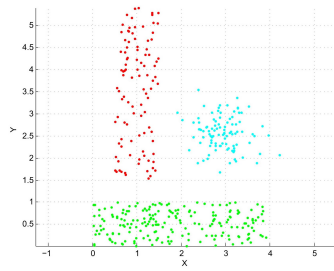
Dataset Size: 212
 # of attributes: 3
 # of true classes: 7
 Clustering problem: Well separated clusters.

Figure 5.1 – Hepta dataset [Ultsch 2005b].

Lsun:

The dataset is shown in figure 5.2. This dataset is more challenging: From the list of clustering algorithms that we use in the tests, only single-linkage, DBSCAN, and spectral can produce the correct clustering, whereas all the others will produce lower quality clustering result when K is set to 3. Therefore, two tests for this dataset are presented in table 5.1. In test Lsun1, K is set to 3 for all base clusterings, with the removal of model-based, DBSCAN and spectral because they yield in exactly the results of single linkage.⁴ In Lsun2, a range of K values is used, and model-based, DBSCAN and spectral are set to not generate 3 clusters. We can recognize that Lsun2 test produced the best consensus solution for all MultiCons approaches and most CLUE methods. The reason of this compared to Lsun1 is that in Lsun2, better quality base clusterings are added.

⁴Removal of identical base clusterings in the ensemble is a main strategy in our ensemble generation process, to prevent generating a consensus biased towards the identical results.

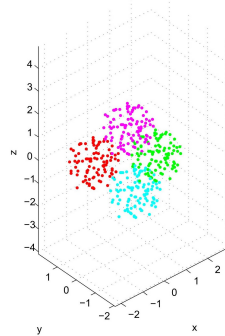


Dataset Size: 400
 # of attributes: 2
 # of true classes: 3
 Clustering problem: Different variances and
 inter cluster distances.

Figure 5.2 – Lsun dataset [Ultsch 2005b].

Tetra:

Tetra dataset is shown in figure 5.3. Although the clusters are not well separated, most of the clustering algorithms can detect the correct partition if K is set to 4, except for single-linkage that grouped all the instances in one cluster. Thus, the test in table 5.1 considers different values of K in the ensemble. All MultiCons approaches and CLUE methods (except soft/symdiff) succeeded in generating the correct partition (some differ in 1 mis-clustered instance).

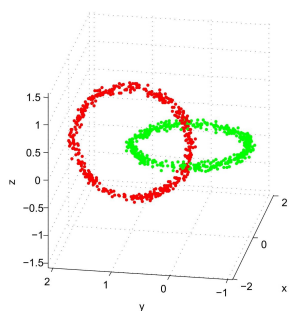


Dataset Size: 400
 # of attributes: 3
 # of true classes: 4
 Clustering problem: Very close clusters.

Figure 5.3 – Tetra dataset [Ultsch 2005b].

Chainlink:

This dataset consists of two linearly non separable clusters as shown in figure 5.4. This is another challenging dataset, as only single linkage, spectral, and DBSCAN can produce exact clustering, whereas partitioning and other hierarchical based clustering methods generated very poor partitions when K is set to 2. In Chainlink1 in table 5.1, all the base clusterings generate two clusters. As the quality of most of the base clusterings is low, neither MultiCons approaches nor CLUE methods were able to produce a good consensus solution. In Chainlink2, the ensemble contained three high quality partitions, which resulted in high quality consensus by MultiCons approaches and some CLUE methods.

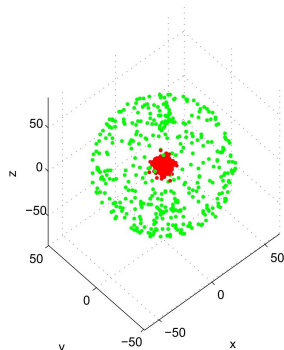


Dataset Size: 1000
 # of attributes: 3
 # of true classes: 2
 Clustering problem: Linearly non separable.

Figure 5.4 – Chainlink dataset [Ultsch 2005b].

Atom:

The Atom dataset consists of two clusters, one with high density represents a core, and the other of lower density represents a shell that surrounds the core, as shown in figure 5.5. As this dataset is linearly non separable, all partitioning and hierarchical clustering methods (except single linkage) cannot build the correct clustering when K is set to 2. In test Atom1 in table 5.1, we can see that grouping such weak base clusterings affected the resulted consensus. On the other hand, in test Atom2 where different K values were used, all MultiCons methods succeeded in generating the correct partition of this dataset.



Dataset Size: 800
 # of attributes: 3
 # of true classes: 2
 Clustering problem: Different variance and linearly non separable.

Figure 5.5 – Atom dataset [Ultsch 2005b].

EngyTime:

The instances in this dataset represent two overlapping Gaussian distributions in a 2D space, as shown in figure 5.6. DBSCAN, average and single linkage hierarchical clustering group all the instances (or the majority of them) in one big cluster. On the other hand, most of the other clustering algorithms will cluster correctly the majority of the instances when K is set to 2.⁵ Therefore, test EngyTime in table

⁵Note that spectral clustering is not applicable for this dataset because of its high time complexity of $O(N^3)$.

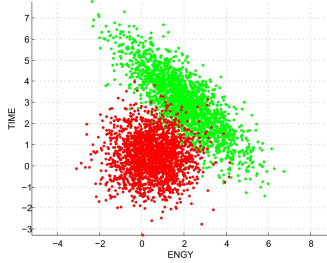
Table 5.1 – Tests validation table 1.

Dataset	Hepta	Lsun1	Lsun2	Tetra	Chainlink1	Chainlink2	Atom1	Atom2
Ensemble size	8	8	12	10	9	12	10	10
K-range	[4,11]	[3]	[2,7]	[3,8]	[2]	[2,7]	[2]	[2,6]
In-ensemble similarity	0.65	0.64	0.57	0.52	0.56	0.38	0.56	0.52
Ensemble min.	0.35	0.43	0.43	0.49	0.37	0.21	0.46	0.34
Ensemble max.	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
# FCPs	68	113	150	307	247	511	736	669
Approach1	1.00	0.64	1.00	0.99	0.50	1.00	0.50	1.00
# cluster app.1	7	6	3	5	1	2	1	2
Approach2	1.00	0.54	1.00	1.00	0.38	1.00	0.50	1.00
# cluster app.2	7	3	3	4	2	2	1	2
Approach3	1.00	0.48	1.00	1.00	0.40	0.96	0.50	1.00
# cluster app.3	7	4	3	4	2	2	1	2
Approach4	1.00	0.48	1.00	0.99	0.40	0.96	0.50	1.00
# cluster app.4	7	3	3	5	3	2	1	2
Approach5	1.00	0.48	1.00	0.99	0.50	1.00	0.50	1.00
# cluster app.5	7	3	3	5	1	2	1	2
SE	1.00	0.57	1.00	0.99	0.35	0.75	0.55	0.97
GV1	1.00	0.56	1.00	1.00	0.40	0.85	0.54	0.92
DWH	1.00	0.57	0.48	1.00	0.40	0.94	0.55	0.93
HE	1.00	0.56	1.00	1.00	0.40	0.74	0.56	0.97
GV3	1.00	0.54	1.00	0.99	0.39	1.00	0.60	1.00
SM	0.76	0.60	0.69	0.99	0.40	0.72	0.55	0.97
HM	1.00	0.56	1.00	1.00	0.40	0.75	0.56	0.97
soft/symdiff	1.00	0.60	1.00	0.65	0.38	1.00	0.63	0.97
medoids	1.00	0.60	1.00	1.00	0.37	1.00	0.48	1.00

Table 5.2 – Execution time table 1 (in seconds).

Dataset	Hepta	Lsun1	Lsun2	Tetra	Chainlink1	Chainlink2	Atom1	Atom2
FCP	0.04	0.06	0.09	0.09	0.13	0.25	0.27	0.23
Approach1	0.11	0.13	0.18	0.29	0.15	0.35	0.20	0.43
Approach2	0.18	0.18	0.37	0.51	0.30	0.87	0.97	0.82
Approach3	0.16	0.23	0.42	0.58	0.45	1.07	0.97	1.57
Approach4	0.17	0.25	0.40	0.59	0.37	0.94	1.43	1.71
Approach5	0.17	0.16	0.33	0.41	0.25	0.73	0.89	0.73
SE	0.01	0.01	0.02	0.00	0.01	0.01	0.01	0.02
GV1	0.03	0.01	0.02	0.03	0.02	0.03	0.01	0.03
DWH	0.00	0.00	0.02	0.00	0.01	0.00	0.00	0.02
HE	0.02	0.00	0.00	0.02	0.02	0.04	0.02	0.00
GV3	2.00	3.44	4.79	4.34	25.28	22.40	18.73	19.21
SM	0.97	0.91	1.50	1.66	3.61	2.32	2.65	1.71
HM	0.02	0.01	0.01	0.02	0.01	0.04	0.01	0.01
soft/symdiff	1.21	42.23	52.11	39.35	323.45	347.17	265.98	192.66
Medoid	0.02	0.02	0.06	0.04	0.05	0.12	0.05	0.06

5.3 considers different values of K in the ensemble (the usual case when we do not know which is the correct value of K). We can see that approaches 1 and 5 failed in building an acceptable consensus for this dataset, as they grouped all the instances in one cluster, unlike the other MultiCons approaches and CLUE methods.

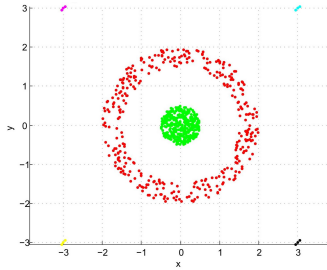


Dataset Size: 4096
 # of attributes: 2
 # of true classes: 2
 Clustering problem: Gaussian mixture.

Figure 5.6 – EngyTime dataset [Ultsch 2005b].

Target:

This dataset contains two linearly non separable clusters, plus four outlier clusters far from the main two, as shown in figure 5.7. This dataset is easy to cluster for single linkage hierarchical clusterings and DBSCAN, whereas difficult for the other clustering algorithms when K is set to 6. Therefore, test Target1 in table 5.3 shows the performance of the consensus methods when $K = 6$. Although MultiCons approach 1 recommended solution was of low quality as all the other consensus methods, the most stable solution in the corresponding ConsTree was the correct result. The same happened in test Target2, as shown in figure 5.8. The reason for the low quality solutions is because of the ring cluster that the majority of the base clusterings in the tests did not recognize. However, in the ConsTrees of all approaches, the middle and the outlier clusters can easily be identified as very stable clusters, as shown in figure 5.9.



Dataset Size: 770
 # of attributes: 2
 # of true classes: 6
 Clustering problem: Outlier clusters.

Figure 5.7 – Target dataset [Ultsch 2005b].

TowDiamonds:

The dataset is shown in figure 5.10. This is a simple clustering problem for clustering algorithms when K is set to 2, except for single linkage and DBSCAN that grouped

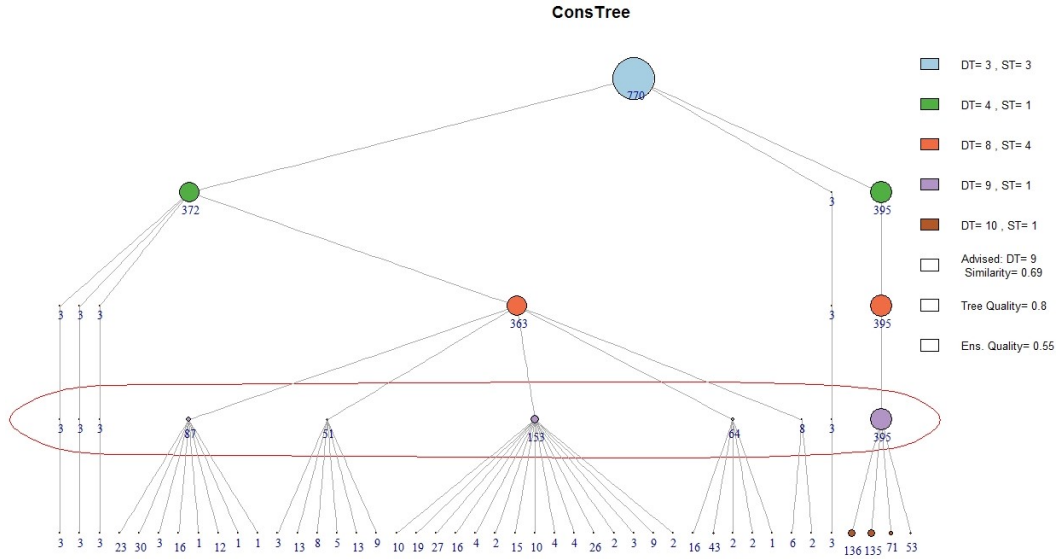


Figure 5.8 – The ConsTree of approach 1 for the Target dataset.

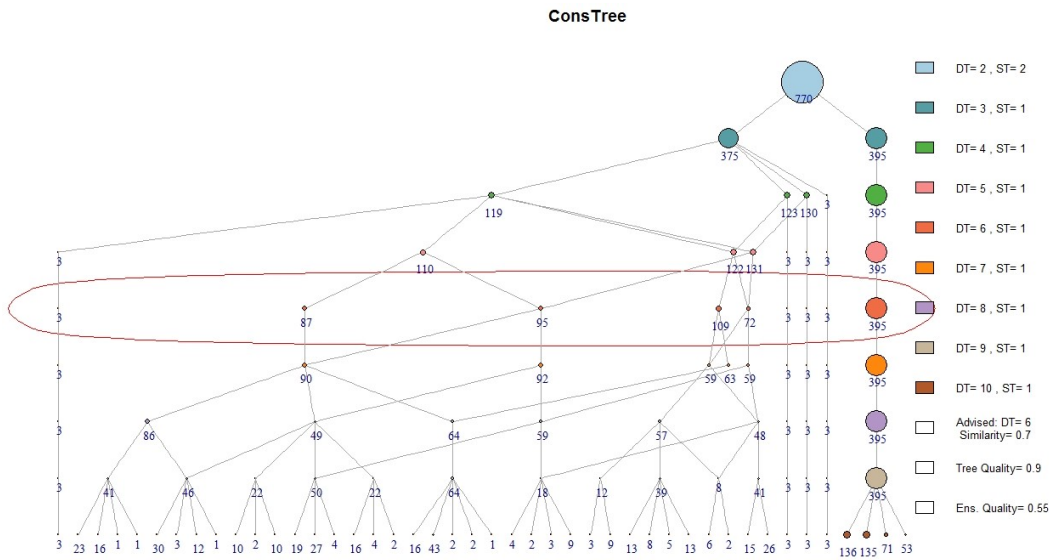


Figure 5.9 – The ConsTree of approach 4 for the Target dataset. We can recognize the stable clusters that represent the noise and the middle core cluster.

the instances in one cluster. Thus, this test will just consider the effect of varying the K value in the ensemble. Although the generated ensemble was of low quality, all MultiCons approaches and most CLUE methods were able to detect the two clusters, as shown in table 5.3.

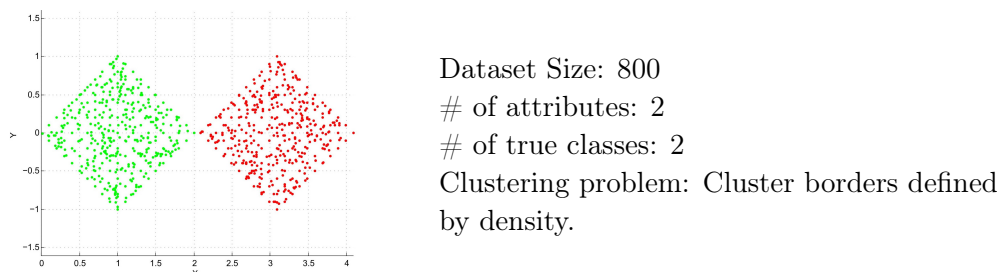


Figure 5.10 – TowDiamonds dataset [Ultsch 2005b].

Wingnut:

The dataset consists of two clusters with different intra-cluster densities against a small distance separating the two clusters, as shown in figure 5.11. Average, single, and complete linkage hierarchical clustering and DBSCAN were able to build the correct clusters, whereas the other algorithms did not when K was set to 2. In table 5.3, test Wingnut1 consists of only one correct clustering (the average linkage) against other lower quality solutions with $K = 2$. MultiCons approach 1 failed in this test as it generated one cluster, and approach 5 built most of the two clusters correctly but didn't group correctly few instances, which resulted in a total of 9 clusters which is unacceptable. On the other hand, the other MultiCons approaches and CLUE methods were able to group correctly most of the instances into two clusters. In test Wingnut2, different K values were used to build the ensemble. We can see that approaches 1, 2, and 5 were able to build the correct partition, whereas the other approaches and some CLUE methods were close to it.

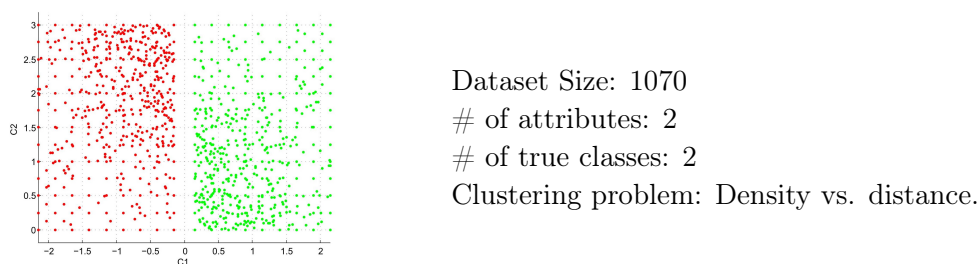


Figure 5.11 – Wingnut dataset [Ultsch 2005b].

2D Normals:

This dataset is already presented as an illustrative example in the previous chapter,

section 4.8. It is built using function *mlbench.2dnormals* in R package *mlbench* [Leisch 2010], by setting $n = 2000$ and $cl = 5$. The validation scores of the test are shown in table 5.3.

Table 5.3 – Tests validation table 2.

Dataset	EngyTime	Target1	Target2	2Diamonds	Wingnut1	Wingnut2	2D Normals
Ensemble size	8	10	10	9	7	11	6
K-range	[2,6]	[6]	[4,9]	[2,7]	[2]	[2,7]	[2,9]
In-ensemble similarity	0.49	0.51	0.57	0.47	0.72	0.44	0.5
Ensemble min.	0.31	0.38	0.35	0.28	0.33	0.25	0.36
Ensemble max.	0.83	1.00	0.74	0.99	1.00	1.00	0.81
# FCPs	817	404	314	406	173	1470	182
Approach1	0.50	0.62	0.63	0.99	0.50	1.00	0.5
# cluster app.1	1	14	14	2	1	2	40
Approach2	0.87	0.66	0.66	1.00	0.87	1.00	0.90
# cluster app.2	2	9	8	2	2	2	5
Approach3	0.82	0.65	0.71	0.99	0.82	0.93	0.84
# cluster app.3	2	11	7	2	2	2	6
Approach4	0.85	0.66	0.71	0.99	0.88	0.98	0.90
# cluster app.4	2	9	7	2	2	2	5
Approach5	0.50	0.72	0.66	0.99	0.88	1.00	0.65
# cluster app.5	1	9	10	2	9	2	9
SE	0.85	0.58	0.60	1.00	0.87	0.33	0.60
GV1	0.85	0.60	0.50	1.00	0.86	0.92	0.64
DWH	0.84	0.59	0.53	0.95	0.86	0.92	0.74
HE	0.85	0.59	0.55	0.99	0.86	0.98	0.75
GV3	0.87	0.72	0.73	1.00	0.86	1.00	0.90
SM	0.85	0.62	0.53	0.99	0.86	0.97	0.88
HM	0.85	0.57	0.54	0.99	0.86	0.97	0.60
soft/symdiff	0.86	0.71	0.71	1.00	0.87	1.00	0.90
medoids	0.83	0.63	0.65	0.42	0.86	0.52	0.80

5.2 Real Datasets

In this section, we will deal with benchmark datasets from the UCI repository [Lichman 2013] and the R package *mlbench* [Leisch 2010]. As these datasets are not designed for clustering problems, achieving the true classification is not possible. Therefore, the validation will consider the highest possible similarity to the true class, compared to the base clusterings and the other consensus methods used in tests. The datasets and their properties are summarized in table 5.5. Attribute type is abbreviated by two letters: N for numeric, and C for categorical.

E.Coli:

Two tests were run with this dataset. E.Coli1 in table 5.6 considers the case when all the base clusterings generate the true number of clusters with different partition

Table 5.4 – Execution time table 2 (in seconds).

Dataset	EngyTime	Target1	Target2	2Diamonds	Wingnut1	Wingnut2	2D Normals
FCP	0.59	0.16	0.12	0.14	0.10	0.43	0.13
Approach1	0.57	0.42	0.29	0.30	0.10	1.20	0.18
Approach2	1.73	0.70	0.53	0.61	0.24	3.60	0.32
Approach3	3.23	0.95	0.62	1.31	0.29	5.75	0.42
Approach4	2.37	0.73	0.63	0.81	0.36	4.99	0.41
Approach5	1.23	0.55	0.43	0.49	0.20	3.09	0.27
SE	0.03	0.02	0.03	0.02	0.01	0.02	0.03
GV1	0.03	0.02	0.07	0.02	0.01	0.03	0.39
DWH	0.02	0.01	0.01	0.00	0.02	0.02	0.01
HE	0.05	0.02	0.02	0.02	0.02	0.02	0.02
GV3	637.98	29.86	27.22	13.27	15.09	25.80	250.16
SM	11.02	6.54	7.11	2.03	15.09	4.54	5.70
HM	0.08	0.03	0.05	0.02	0.01	0.04	0.04
soft/symdiff	5154.39	228.15	178.46	150.81	257.66	335.57	967.31
Medoid	0.12	0.06	0.06	0.04	0.03	0.09	0.04

Table 5.5 – Real-world benchmark datasets.

Dataset	E.Coli	Breast Cancer	Wine	Zoo	Iris	Magic Gamma	Seeds	Mushroom
Area	Life	Life	Physical	Life	Life	Physical	Life	Life
Size	336	699	178	101	150	19020	210	8124
# attributes	7	9	13	16	4	10	7	22
Attr. type	N	N	N	N	N	N	N	C
# classes	8	2	3	7	3	2	3	2
Missing values	No	Yes, 16	No	No	No	No	No	Yes
Source	UCI	UCI, R	UCI	UCI, R	UCI, R	UCI	UCI	UCI

quality. Although this may not be possible in real life tests as we usually do not know how many true clusters are hidden in the dataset, this test is presented to explain the performance of the consensus methods. Test E.Coli2 is more similar to real life as we use a range of K values in the ensemble pretending that we do not know the true K . We can see that MultiCons approaches achieved good scores compared to the CLUE methods and the ensemble.

Breast Cancer: [Bennett 1992]

The 16 missing values in this dataset were replaced by 0 before doing the tests. Test Breast Cancer1 in table 5.6 considers the case of setting K to the true number of classes for all the base clusterings, whereas test Breast Cancer2 considers the case of different K values in the ensemble. Note that single linkage hierarchical method is not a good choice for this dataset as it grouped the instances in one big cluster in both tests, thus it was removed from the ensemble.

Wine:

Setting $K = 3$ in test Wine1 in table 5.6, single and average linkage hierarchical methods didn't generate useful clusters as they grouped most of the instances in one cluster. Thus, they are removed from the ensemble. The low quality result of MultiCons approach 2 was because the default MT was high for this dataset, as a quality score of 0.91 was achievable with $MT = 0.3$. The same happened in test Wine2, as reducing MT can achieve higher quality result for approach 2. On the other hand, the other MultiCons approaches that uses lower MT as default produced higher quality consensus.

Zoo:

All the attributes of this dataset are logical, except one that defines the number of legs of the animal (numerical integer value). As with other datasets, two tests were run: Zoo1 with $K = 7$, and Zoo2 with a range of K values in the ensemble. We can see from table 5.6 that the latter test resulted in better consensus solution for most methods.

Iris:

For this dataset, all MultiCons approaches agreed on a recommended consensus of two clusters instead of the real three when tested with two different ensembles: Iris1 ($K = 3$) and Iris2 (K in a range). The reason for the low consensus results is that all the base clusterings, except one, generated low quality partitions. Figure 5.12 shows the ConsTree of approach 2 with the ensemble of the Iris1 test (which is somehow similar to the ConsTrees of the other approaches in both tests). We can recognize a well separated cluster of 50 instances (corresponding to the setosa class) from the cluster of 100 instances that merges the other two classes (virginica and versicolor). However, we can see also that the 100 instances cluster was built from two clusters. This is a case where the analyst may prefer to take another consensus, not the recommended one, as he may prefer separating the big cluster into smaller two. It is clear how the ConsTree can help in understanding the data space.

Magic Gamma:

Despite the size of this dataset that makes some clustering algorithms inappropriate (like DIANA and spectral clustering), this dataset is not suitable for clustering tests: DBSCAN, single, average, and complete linkage clustering algorithms grouped the majority of the instances in one big cluster when $K = 2$. However, test Magic Gamma in table 5.8 is just to present how the consensus methods perform with a large dataset. For MultiCons, the dataset and its ensemble was summarized by just 144 patterns, which is a huge pruning of the search space. On the other hand, CLUE methods GV3 and soft/symdiff were inapplicable as they required more memory than the 32 GB of RAM available on the testing computer.

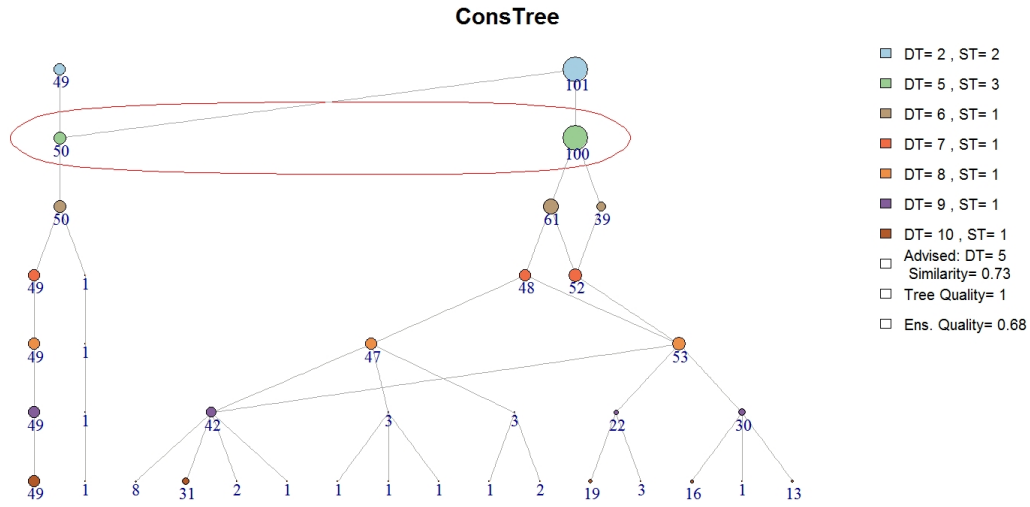


Figure 5.12 – The ConsTree of approach 2 for the Iris dataset using the ensemble of the Iris1 test.

Table 5.6 – Tests validation table 3.

Dataset	E.Coli1	E.Coli2	Breast Cancer1	Breast Cancer2	Wine1	Wine2	Zoo1	Zoo2	Iris1	Iris2
Ensemble size	7	8	8	10	8	10	10	10	10	10
K-range	[8]	[5,11]	[2]	[2,6]	[3]	[2,6]	[7]	[3,12]	[3]	[2,7]
In-ensemble similarity	0.44	0.59	0.74	0.67	0.55	0.48	0.51	0.43	0.68	0.52
Ensemble min.	0.33	0.41	0.64	0.28	0.47	0.41	0.41	0.36	0.57	0.37
Ensemble max.	0.71	0.68	0.87	0.87	0.91	0.87	0.87	0.74	0.88	0.60
# FCPs	384	452	125	1456	362	674	110	143	113	187
Approach1	0.70	0.63	0.58	0.82	0.87	0.86	0.58	0.56	0.60	0.60
# cluster app.1	13	15	16	15	7	9	10	11	2	2
Approach2	0.66	0.64	0.86	0.88	0.76	0.67	0.59	0.81	0.60	0.60
# cluster app.2	6	8	2	2	6	7	8	8	2	2
Approach3	0.68	0.67	0.85	0.83	0.88	0.84	0.74	0.80	0.60	0.60
# cluster app.3	11	6	2	6	4	6	6	10	2	2
Approach4	0.68	0.66	0.88	0.84	0.91	0.86	0.59	0.81	0.60	0.60
# cluster app.4	9	11	2	3	3	4	7	7	2	2
Approach5	0.65	0.65	0.58	0.82	0.77	0.84	0.80	0.80	0.60	0.60
# cluster app.5	15	15	16	18	11	8	7	11	2	2
SE	0.54	0.65	0.87	0.89	0.82	0.52	0.62	0.65	0.62	0.59
GV1	0.58	0.65	0.88	0.87	0.78	0.52	0.63	0.80	0.58	0.54
DWH	0.44	0.66	0.87	0.87	0.78	0.52	0.62	0.78	0.62	0.66
HE	0.68	0.66	0.87	0.89	0.80	0.56	0.61	0.79	0.60	0.56
GV3	0.46	0.65	0.88	0.89	0.87	0.87	0.60	0.76	0.60	0.66
SM	0.43	0.64	0.89	0.89	0.77	0.52	0.59	0.80	0.60	0.58
HM	0.45	0.65	0.88	0.89	0.75	0.54	0.59	0.78	0.63	0.59
soft/symdiff	0.36	0.65	0.85	0.88	0.53	0.80	0.59	0.59	0.60	0.58
medoids	0.71	0.68	0.86	0.86	0.87	0.48	0.65	0.71	0.60	0.58

Table 5.7 – Execution time table 3 (in seconds).

Dataset	E.Coli1	E.Coli2	Breast Cancer1	Breast Cancer2	Wine1	Wine2	Zoo1	Zoo2	Iris1	Iris2
FCP	0.08	0.13	0.09	0.32	0.10	0.13	0.07	0.06	0.07	0.09
Approach1	0.40	0.35	0.08	0.57	0.18	0.29	0.18	0.22	0.14	0.18
Approach2	0.70	0.57	0.19	3.79	0.41	0.79	0.26	0.29	0.19	0.30
Approach3	1.08	0.98	0.21	4.49	0.64	1.11	0.27	0.33	0.22	0.33
Approach4	0.84	0.84	0.22	4.15	0.64	1.19	0.29	0.35	0.22	0.32
Approach5	0.56	0.63	0.13	3.63	0.45	0.86	0.27	0.28	0.19	0.22
SE	0.01	0.00	0.01	0.01	0.01	0.01	0.01	0.03	0.01	0.01
GV1	0.01	0.16	0.04	0.02	0.00	0.02	0.01	0.07	0.01	0.03
DWH	0.00	0.01	0.01	0.01	0.00	0.02	0.00	0.00	0.00	0.01
HE	0.02	0.01	0.01	0.01	0.02	0.00	0.01	0.01	0.01	0.01
GV3	5.11	4.54	11.84	10.89	0.83	0.99	0.54	0.50	0.63	0.67
SM	5.25	3.71	1.89	2.90	0.83	1.06	0.92	0.92	0.83	0.76
HM	0.02	0.02	0.01	0.02	0.01	0.01	0.01	0.03	0.01	0.01
soft/symdiff	25.62	32.50	160.58	134.24	7.07	7.77	3.41	3.07	6.32	6.11
Medoid	0.02	0.03	0.03	0.06	0.02	0.03	0.03	0.03	0.03	0.03

Seeds:

Single linkage hierarchical clustering algorithm is removed from the ensemble of the two tests for this dataset because it grouped most of the instances in one big cluster. The results of the consensus methods on two different ensembles are shown in table 5.8.

Mushroom:

All attributes of this dataset are categorical and contain many missing values. Hence, Gower distance (see section 2.1) was used in the test. Only the clustering algorithms that deal with distance matrix calculated with Gower distance can be applied to partition this dataset. In addition, DBSCAN, average and single linkage algorithms grouped most of the instances in one big cluster. Having a small ensemble, test Mushroom in table 5.8 considers the case of the same true K in all base clusterings.

Big Data:

This is a special test where the objective is not clustering, but rather the performance of the FCI technique on a relatively big data. A dataset of 2,210,084 instances each has 10 attributes is used. In the first test, the dataset is partitioned into 10 clusters using K-means. This partition is repeated 10 times (that is, the ensemble consists of 10 identical partitions) producing a binary membership matrix of 100 columns. Applying the FCI technique to discover the patterns,⁶ which are obviously the 10 clusters generated by K-means, took 1990 seconds (about 30 minutes). In the second test, also K-means is used, but by setting $K = 2$ and repeating this partition 5 times

⁶The FCPs were generated using FIST algorithm [Mondal 2012] implemented in Java.

Table 5.8 – Tests validation table 4.

Dataset	Magic Gamma	Seeds1	Seeds2	Mushroom
Ensemble size	5	10	10	4
K-range	[2]	[3]	[2,7]	[2]
In-ensemble similarity	0.71	0.63	0.45	0.70
Ensemble min.	0.35	0.44	0.35	0.39
Ensemble max.	0.40	0.75	0.66	0.69
# FCPs	144	562	797	19
Approach1	0.27	0.74	0.55	0.50
# cluster app.1	21	12	3	6
Approach2	0.35	0.76	0.71	0.69
# cluster app.2	2	3	3	2
Approach3	0.34	0.75	0.71	0.69
# cluster app.3	3	3	3	2
Approach4	0.35	0.76	0.54	0.70
# cluster app.4	2	3	2	2
Approach5	0.27	0.74	0.55	0.69
# cluster app.5	21	10	3	2
SE	0.36	0.76	0.73	0.69
GV1	0.36	0.75	0.68	0.69
DWH	0.36	0.74	0.70	0.69
HE	0.36	0.77	0.66	0.69
GV3	NA	0.75	0.74	0.70
SM	0.36	0.74	0.64	0.69
HM	0.36	0.77	0.75	0.69
soft/symdiff	NA	0.76	0.55	0.70
medoids	0.36	0.74	0.66	0.69

in the ensemble, resulted in a membership matrix of just 10 columns. While this matrix is much smaller in dimensions than in the first test, the time required to generate the patterns (which are the 2 clusters of K-means) is much longer: 5754 seconds (about 1.5 hours), despite that the two matrices have the same number of rows. These tests show that increasing the ensemble (in terms of the number of clusters) may not necessarily increase the time required to generate the patterns, and that the performance of the FCI technique does not depend directly on the dimensionality of the binary matrix, as matrix density and correlation (rates of co-occurrences of values in instances) are major criteria for the performance of closed pattern extraction.

5.3 Summary

In this chapter, several tests were performed on synthetic datasets that define some clustering problems, as well as on real-world datasets where the true classification of the instances is known. However, the latter case is not a good practice to validate

Table 5.9 – Execution time table 4 (in seconds).

Dataset	Magic Gamma	Seeds1	Seeds2	Mushroom
FCP	0.85	0.14	0.18	0.16
Approach1	0.47	0.27	0.36	0.10
Approach2	0.76	0.48	1.21	0.11
Approach3	1.14	0.96	1.70	0.25
Approach4	1.29	0.88	1.52	0.19
Approach5	0.71	0.64	1.11	0.19
SE	0.07	0.01	0.01	0.02
GV1	0.08	0.01	0.08	0.03
DWH	0.05	0.01	0.01	0.01
HE	0.08	0.01	0.01	0.03
GV3	NA	1.53	1.49	2830.27
SM	44.87	1.70	1.61	19.47
HM	0.08	0.01	0.02	0.06
soft/symdiff	NA	11.29	12.76	14929.07
Medoid	0.19	0.03	0.03	0.04

clustering algorithms, as the class labels do not necessarily follow or define cluster structures. The K parameter for CLUE methods was set to the true number of classes known for each tested dataset, whereas MultiCons approaches usually generated the true number of clusters or very close to it, except approaches 1 and 5 that do not perform iterative merging/splitting of clustering patterns.

It is obvious that the quality of the resulted consensus depends on the quality of the ensemble. We considered in all tests randomly created ensembles, that is, they included good and bad quality partitions, not processed to add only relatively good partitions. This is to mimic unexperienced analyst, and the fact that, in real life, we do not have the true class labels to compare with. However, if the analyst has good experience on clustering algorithms, he may achieve higher quality consensus solutions than the ones produced in the tests.

Regarding execution time shown in tables 5.2, 5.4, 5.7, and 5.9, MultiCons methods were not the fastest, but their execution times were acceptable. On the other hand, CLUE methods SM, GV3, and soft/symdiff were the slowest in all tests, and the latter two were inapplicable for large datasets because of their demand for large amount of memory. While MultiCons approach 1 was the fastest among the other approaches, the search in approaches 3 and 4 may increase their execution time compared to approach 2.

Additional tests can be found in the following papers: [Al-Najdi 2016a] (approach 1), [Al-Najdi 2016b] (approach 2), and [Al-Najdi 2016c] (approach 3).

Conclusions and Future Work

A new consensus clustering method called MultiCons was developed in this work. It uses the frequent closed pattern mining technique in order to transform the relationship between the instances and the partition ensemble into clustering patterns. By dividing this pattern space into subspaces, MultiCons is able to generate multiple consensus from different combinations of base clusterings. In each subspace, it tries to re-cluster the instances based on the information provided in the patterns. 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. Using the ConsTree, the analysts are not limited to one final solution, but they rather can choose the solution which best fits their needs 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.

Five different approaches to cluster the patterns were explained. The following points summarize the conclusions realized from the conducted tests:

- Approach 2 is recommended as the standard for MultiCons for the following reasons: *i)* It produced good consensus throughout the various tests in terms of quality and number of generated clusters. *ii)* Although approaches 3 and 4 were generally as good as approach 2, the difference in the quality of their results is generally low. However, approaches 3 and 4 may spend more execution time than approach 2, because they try to find more similar patterns before taking a merge/split decision, whereas approach 2 do the merge/split process directly. This extra execution time is unnecessary if the final result can be produced by a simpler approach. *iii)* Approaches 1 and 5 sometimes fail to generate acceptable solutions, especially in the number of generated clusters, because of their process of dealing with similar clustering patterns.
- Tests showed that when the ensemble consists of partitions that have different number of clusters, then a better consensus solution can be found, compared to using an ensemble whose partitions consist of the same number of clusters.
- It was found on the majority of the tests that using low merging threshold has a lower execution time compared to a higher value. This is because low MT means more merging, and this reduces the number of instance sets at each iteration of the MultiCons approaches.

As recommendations for future work:

- A possible enhancement for the overall MultiCons execution time would consist to introduce a *minimum agreement threshold* in the FCI mining phase, so as not to generate patterns of itemset size lower than this threshold. For example, if the agreement threshold is set to 40% of the base clusterings and we have 10 partitions in the ensemble, then the FCI mining phase would not generate patterns of itemset size less than 4. This threshold can be higher to highly optimize FCP generation time especially for large datasets and/or ensembles. However, the ensemble needs to be of sufficient quality to ensure achieving good quality consensus from agreements among high number of base clusterings.
- Domain-specific experiments can highlight, with the help of domain experts, which of the MultiCons approaches is more useful in terms of information retrieval. That is, the quality of the consensus candidates (not just the recommended) and the ConsTree in providing interesting information in that domain. This may also identify future developments for MultiCons. In addition, although the default *MT* values of the MultiCons approaches achieved generally the best of the methods (or close to it) during the tests, it is possible to change *MT* when MultiCons is applied on a specific domain. The idea is that may be on a certain domain, low (or equivalently high) *MT* value may achieve better results because of the properties of the data space. In such cases, and to some extent, it may also be possible to use the simpler approaches 1 or 5 to get interesting results.
- Investigating the usefulness of applying other bi-clustering¹ algorithm on the binary membership matrix, instead of using closed pattern mining, to generate clustering patterns. Actually, the FCPs are bi-clusters, and it thus seems interesting to compare both techniques in terms of their performance for deriving a consensus solution, for clustering or classification task.

¹Bi-clustering is first defined by Mirkin [Mirkin 1998] as “the simultaneous clustering of both row and column sets in a data matrix”.

Bibliography

- [Aggarwal 2013] Charu C Aggarwal and Chandan K Reddy. Data clustering: algorithms and applications. CRC Press, 2013. (Cited on pages [iii](#), [7](#), [8](#), [10](#), [11](#), [13](#), [15](#), [16](#), [17](#), [18](#), [19](#), [20](#), [21](#), [22](#), [23](#), [24](#), [26](#), [29](#) and [33](#).)
- [Aggarwal 2014] Charu C Aggarwal and Jiawei Han. Frequent pattern mining. Springer, 2014. (Cited on pages [37](#) and [38](#).)
- [Aggarwal 2015] Charu C Aggarwal. Data mining: The textbook. Springer, 2015. (Cited on pages [10](#), [12](#), [13](#) and [38](#).)
- [Agrawal 1998] Rakesh Agrawal, Johannes Gehrke, Dimitrios Gunopulos and Prabhakar Raghavan. Automatic subspace clustering of high dimensional data for data mining applications, volume 27. ACM, 1998. (Cited on page [16](#).)
- [Al-Najdi 2016a] Atheer Al-Najdi, Nicolas Pasquier and Frédéric Precioso. *Frequent Closed Patterns Based Multiple Consensus Clustering*. In Proceedings of the ICAISC’2016 International Conference on Artificial Intelligence and Soft Computing, Part II, LNCS 9693, pages 14–26, Zakopane, Poland, June 2016. Springer International Publishing. (Cited on page [91](#).)
- [Al-Najdi 2016b] Atheer Al-Najdi, Nicolas Pasquier and Frédéric Precioso. *Multiple Consensuses Clustering by Iterative Merging/Splitting of Clustering Patterns*. In Machine Learning and Data Mining in Pattern Recognition. Springer International Publishing, 2016. (Cited on page [91](#).)
- [Al-Najdi 2016c] Atheer Al-Najdi, Nicolas Pasquier and Frédéric Precioso. *Using Frequent Closed Pattern Mining to Solve a Consensus Clustering Problem*. In Proceedings of the SEKE’2016 International Conference on Software Engineering and Knowledge Engineering, pages 454–461, Redwood City, USA, 2016. KSI Research Inc. (Cited on page [91](#).)
- [Asur 2007] Sitaram Asur, Duygu Ucar and Srinivasan Parthasarathy. *An ensemble framework for clustering protein–protein interaction networks*. Bioinformatics, vol. 23, no. 13, pages i29–i40, 2007. (Cited on pages [32](#) and [33](#).)
- [Bennett 1992] Kristin P Bennett and Olvi L Mangasarian. *Robust linear programming discrimination of two linearly inseparable sets*. Optimization methods and software, vol. 1, no. 1, pages 23–34, 1992. (Cited on page [86](#).)
- [Berkhin 2006] Pavel Berkhin. *A survey of clustering data mining techniques*. In Grouping multidimensional data, pages 25–71. Springer, 2006. (Cited on page [16](#).)

- [Calinski 1974] Tadeusz Calinski and Jerzy Harabasz. *A dendrite method for cluster analysis*. Communications in Statistics-theory and Methods, vol. 3, no. 1, pages 1–27, 1974. (Cited on page 21.)
- [Caruana 2006] Rich Caruana, Mohamed Elhawary, Nam Nguyen and Casey Smith. *Meta clustering*. In Proc. IEEE ICDM Conf., pages 107–118, 2006. (Cited on page 32.)
- [Ceglar 2006] Aaron Ceglar and John F. Roddick. *Association Mining*. ACM Computing Surveys, vol. 38, no. 2, 2006. (Cited on page 38.)
- [Chung 2014] Chih-Heng Chung and Bi-Ru Dai. *A fragment-based iterative consensus clustering algorithm with a robust similarity*. Knowledge and Information Systems, vol. 41, no. 3, pages 591–609, 2014. (Cited on page 31.)
- [Cichosz 2014] Pawel Cichosz. *Data mining algorithms: Explained using r*. John Wiley & Sons, 2014. (Cited on page 18.)
- [Csardi 2006] Gabor Csardi and Tamas Nepusz. *The igraph software package for complex network research*. InterJournal, vol. Complex Systems, page 1695, 2006. (Cited on page 75.)
- [Dalton 2009] Lori Dalton, Virginia Ballarin and Marcel Brun. *Clustering algorithms: on learning, validation, performance, and applications to genomics*. Current Genomics, vol. 10, no. 6, page 430, 2009. (Cited on pages 3, 18, 21 and 107.)
- [Davies 1979] David L Davies and Donald W Bouldin. *A cluster separation measure*. Pattern Analysis and Machine Intelligence, IEEE Transactions on, no. 2, pages 224–227, 1979. (Cited on page 18.)
- [Delen 2014] Dursun Delen. *Real-world data mining: Applied business analytics and decision making*. FT Press, 2014. (Cited on page 21.)
- [Dempster 1977] Arthur P Dempster, Nan M Laird and Donald B Rubin. *Maximum likelihood from incomplete data via the EM algorithm*. Journal of the royal statistical society. Series B (methodological), pages 1–38, 1977. (Cited on page 13.)
- [Dimitriadou 2002] Evgenia Dimitriadou, Andreas Weingessel and Kurt Hornik. *A combination scheme for fuzzy clustering*. International Journal of Pattern Recognition and Artificial Intelligence, vol. 16, no. 07, pages 901–912, 2002. (Cited on pages 28, 49 and 75.)
- [Domeniconi 2007] Carlotta Domeniconi, Dimitrios Gunopulos, Sheng Ma, Bojun Yan, Muna Al-Razgan and Dimitris Papadopoulos. *Locally adaptive metrics for clustering high dimensional data*. Data Mining and Knowledge Discovery, vol. 14, no. 1, pages 63–97, 2007. (Cited on page 26.)

- [Domeniconi 2009] Carlotta Domeniconi and Muna Al-Razgan. *Weighted cluster ensembles: Methods and analysis*. ACM Transactions on Knowledge Discovery from Data (TKDD), vol. 2, no. 4, page 17, 2009. (Cited on pages 26 and 27.)
- [Dudek 2015] Marek Walesiak Andrzej Dudek. *clusterSim: Searching for Optimal Clustering Procedure for a Data Set*, 2015. R package version 0.44-2. (Cited on page 77.)
- [Dudoit 2003] Sandrine Dudoit and Jane Fridlyand. *Bagging to improve the accuracy of a clustering procedure*. Bioinformatics, vol. 19, no. 9, pages 1090–1099, 2003. (Cited on page 28.)
- [Dunn 1974] Joseph C Dunn. *Well-separated clusters and optimal fuzzy partitions*. Journal of cybernetics, vol. 4, no. 1, pages 95–104, 1974. (Cited on page 18.)
- [Ester 1996] Martin Ester, Hans-Peter Kriegel, Jörg Sander and Xiaowei Xu. *A density-based algorithm for discovering clusters in large spatial databases with noise*. In Kdd, volume 96, pages 226–231, 1996. (Cited on page 12.)
- [Färber 2010] Ines Färber, Stephan Günnemann, Hans-Peter Kriegel, Peer Kröger, Emmanuel Müller, Erich Schubert, Thomas Seidl and Arthur Zimek. *On using class-labels in evaluation of clusterings*. In KDD MultiClust International Workshop on Discovering, Summarizing and Using Multiple Clusterings, page 1, 2010. (Cited on page 24.)
- [Fayyad 1996] Usama Fayyad, Gregory Piatetsky-Shapiro and Padhraic Smyth. *From data mining to knowledge discovery in databases*. AI magazine, vol. 17, no. 3, page 37, 1996. (Cited on pages iii, iv, 1, 2, 105 and 106.)
- [Fern 2004] 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. (Cited on pages 26 and 27.)
- [Filkov 2004] Vladimir Filkov and Steven Skiena. *Integrating microarray data by consensus clustering*. International Journal on Artificial Intelligence Tools, vol. 13, no. 04, pages 863–880, 2004. (Cited on pages 30 and 32.)
- [Fischer 2003] Bernd Fischer and Joachim M Buhmann. *Bagging for path-based clustering*. Pattern Analysis and Machine Intelligence, IEEE Transactions on, vol. 25, no. 11, pages 1411–1415, 2003. (Cited on page 28.)
- [Fournier-Viger 2016] Philippe Fournier-Viger, Jerry Chun-Wei Lin, Antonio Gomariz, Ted Gueniche, Azadeh Soltani, Zhihong Deng and Hoang Thanh Lam. *The SPMF Open-Source Data Mining Library Version 2*. In Joint European Conference on Machine Learning and Knowledge Discovery in Databases, pages 36–40. Springer, 2016. (Cited on page 75.)

- [Fowlkes 1983] Edward B Fowlkes and Colin L Mallows. *A method for comparing two hierarchical clusterings*. Journal of the American statistical association, vol. 78, no. 383, pages 553–569, 1983. (Cited on page 22.)
- [Fraley 2002] Chris Fraley and Adrian E. Raftery. *Model-based Clustering, Discriminant Analysis and Density Estimation*. Journal of the American Statistical Association, vol. 97, pages 611–631, 2002. (Cited on page 77.)
- [Fraley 2012] Chris Fraley, Adrian E. Raftery, Thomas Brendan Murphy and Luca Scrucca. *mclust Version 4 for R: Normal Mixture Modeling for Model-Based Clustering, Classification, and Density Estimation*, 2012. (Cited on page 77.)
- [Fred 2002] 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. (Cited on pages 29 and 30.)
- [Gan 2007] Guojun Gan, Chaoqun Ma and Jianhong Wu. Data clustering: theory, algorithms, and applications, volume 20. Siam, 2007. (Cited on pages 6, 10, 11, 13, 16, 18, 19 and 20.)
- [Ghaemi 2009] Reza Ghaemi, Md Nasir Sulaiman, Hamidah Ibrahim and Norwati Mustapha. *A survey: Clustering ensembles techniques*. WASET, vol. 50, pages 636–645, 2009. (Cited on pages 24 and 33.)
- [Gionis 2007] Aristides Gionis, Heikki Mannila and Panayiotis Tsaparas. *Clustering aggregation*. ACM Transactions on Knowledge Discovery from Data (TKDD), vol. 1, no. 1, page 4, 2007. (Cited on pages 30 and 31.)
- [Gordon 2001] AD Gordon and M Vichi. *Fuzzy partition models for fitting a set of partitions*. Psychometrika, vol. 66, no. 2, pages 229–247, 2001. (Cited on pages 28, 29 and 76.)
- [Hahsler 2005] Michael Hahsler, Bettina Gruen and Kurt Hornik. *arules – A Computational Environment for Mining Association Rules and Frequent Item Sets*. Journal of Statistical Software, vol. 14, no. 15, pages 1–25, 2005. (Cited on page 75.)
- [Hahsler 2011] Michael Hahsler, Sudheer Chelluboina, Kurt Hornik and Christian Buchta. *The arules R-Package Ecosystem: Analyzing Interesting Patterns from Large Transaction Datasets*. Journal of Machine Learning Research, vol. 12, pages 1977–1981, 2011. (Cited on page 75.)
- [Hahsler 2016] Michael Hahsler, Christian Buchta, Bettina Gruen and Kurt Hornik. *arules: Mining Association Rules and Frequent Itemsets*, 2016. R package version 1.4-1. (Cited on page 75.)

- [Halkidi 2000] Maria Halkidi, Michalis Vazirgiannis and Yannis Batistakis. *Quality scheme assessment in the clustering process*. In Principles of Data Mining and Knowledge Discovery, pages 265–276. Springer, 2000. (Cited on page 19.)
- [Halkidi 2001a] Maria Halkidi, Yannis Batistakis and Michalis Vazirgiannis. *On clustering validation techniques*. Journal of Intelligent Information Systems, vol. 17, no. 2, pages 107–145, 2001. (Cited on pages 3, 8, 18, 19 and 107.)
- [Halkidi 2001b] Maria Halkidi and Michalis Vazirgiannis. *Clustering validity assessment: Finding the optimal partitioning of a data set*. In Data Mining, 2001. ICDM 2001, Proceedings IEEE International Conference on, pages 187–194. IEEE, 2001. (Cited on page 20.)
- [Han 2011] Jiawei Han, Micheline Kamber and Jian Pei. Data mining: concepts and techniques. Elsevier, 2011. (Cited on pages 6, 7, 8, 9, 12, 15, 16, 17, 20, 21, 36 and 38.)
- [Hartigan 1979] John A Hartigan and Manchek A Wong. *Algorithm AS 136: A k-means clustering algorithm*. Journal of the Royal Statistical Society. Series C (Applied Statistics), vol. 28, no. 1, pages 100–108, 1979. (Cited on page 7.)
- [Hennig 2015] Christian Hennig. *fpc: Flexible Procedures for Clustering*, 2015. R package version 2.1-10. (Cited on page 77.)
- [Hinneburg 1998] Alexander Hinneburg and Daniel A Keim. *An efficient approach to clustering in large multimedia databases with noise*. In KDD, volume 98, pages 58–65, 1998. (Cited on page 12.)
- [Hornik 2005a] Kurt Hornik. *A CLUE for CLUster Ensembles*. Journal of Statistical Software, vol. 14, no. 12, September 2005. (Cited on page 75.)
- [Hornik 2005b] Kurt Hornik. *Cluster ensembles*. In Classification - the Ubiquitous Challenge, pages 65–72. Springer, 2005. (Cited on page 28.)
- [Hornik 2007] Kurt Hornik. *A CLUE for CLUster Ensembles*, 2007. (Cited on pages 29 and 76.)
- [Hornik 2016] Kurt Hornik. *clue: Cluster ensembles*, 2016. R package version 0.3-51. (Cited on page 75.)
- [Jaccard 1912] Paul Jaccard. *THE DISTRIBUTION OF THE FLORA IN THE ALPINE ZONE.1*. New Phytologist, vol. 11, no. 2, pages 37–50, 1912. (Cited on pages 22, 57 and 76.)
- [Jain 2010] Anil K Jain. *Data clustering: 50 years beyond K-means*. Pattern recognition letters, vol. 31, no. 8, pages 651–666, 2010. (Cited on pages 5 and 7.)
- [James 1981] C Bedzek James. *Pattern recognition with fuzzy objective function algorithms*, 1981. (Cited on page 12.)

- [Karypis 1998] George Karypis and Vipin Kumar. *A fast and high quality multilevel scheme for partitioning irregular graphs*. SIAM Journal on scientific Computing, vol. 20, no. 1, pages 359–392, 1998. (Cited on pages 25 and 27.)
- [Karypis 1999a] George Karypis, Rajat Aggarwal, Vipin Kumar and Shashi Shekhar. *Multilevel hypergraph partitioning: applications in VLSI domain*. Very Large Scale Integration (VLSI) Systems, IEEE Transactions on, vol. 7, no. 1, pages 69–79, 1999. (Cited on page 25.)
- [Karypis 1999b] George Karypis, Eui-Hong Han and Vipin Kumar. *Chameleon: Hierarchical clustering using dynamic modeling*. Computer, vol. 32, no. 8, pages 68–75, 1999. (Cited on pages 13, 14 and 15.)
- [Kaufman 1990] Leonard Kaufman and Peter J Rousseeuw. Finding groups in data: an introduction to cluster analysis, volume 344. John Wiley & Sons, 1990. (Cited on pages 8, 11 and 21.)
- [Langfelder 2012] Peter Langfelder and Steve Horvath. *Fast R Functions for Robust Correlations and Hierarchical Clustering*. Journal of Statistical Software, vol. 46, no. 11, pages 1–17, 2012. (Cited on page 77.)
- [Legány 2006] Csaba Legány, Sándor Juhász and Attila Babos. *Cluster Validity Measurement Techniques*. In Proceedings of the 5th WSEAS International Conference on Artificial Intelligence, Knowledge Engineering and Data Bases, AIKED’06, pages 388–393, 2006. (Cited on pages 18, 19 and 20.)
- [Leisch 1999] Friedrich Leisch. *Bagged clustering*. 1999. (Cited on pages 49 and 77.)
- [Leisch 2010] Friedrich Leisch and Evgenia Dimitriadou. *mlbench: Machine Learning Benchmark Problems*, 2010. R package version 2.1-1. (Cited on page 85.)
- [Li 2008] Tao Li and Chris Ding. *Weighted consensus clustering*. In Proc. SIAM Conf. on Data Mining, pages 798–809, 2008. (Cited on page 33.)
- [Lichman 2013] M. Lichman. *UCI Machine Learning Repository*, 2013. (Cited on page 85.)
- [Lozano 2011] Elio Lozano and Edgar Acuña. *Comparing clustering and metaclustering algorithms*. In Machine Learning and Data Mining in Pattern Recognition, pages 306–319. Springer, 2011. (Cited on page 49.)
- [Macnaughton-Smith 1964] P Macnaughton-Smith, WT Williams, MB Dale and LG Mockett. *Dissimilarity analysis: a new technique of hierarchical subdivision*. 1964. (Cited on page 11.)
- [MacQueen 1967] James MacQueen *et al.* *Some methods for classification and analysis of multivariate observations*. In Proceedings of the fifth Berkeley symposium on mathematical statistics and probability, volume 1, pages 281–297. Oakland, CA, USA., 1967. (Cited on page 7.)

- [Maechler 2016] Martin Maechler, Peter Rousseeuw, Anja Struyf, Mia Hubert and Kurt Hornik. *cluster: Cluster Analysis Basics and Extensions*, 2016. R package version 2.0.4 — For new features, see the 'Changelog' file (in the package source). (Cited on page 77.)
- [Manning 2008] Christopher D Manning, Prabhakar Raghavan, Hinrich Schütze *et al.* Introduction to information retrieval, volume 1. Cambridge university press Cambridge, 2008. (Cited on pages 21, 22 and 23.)
- [Meilă 2007] Marina Meilă. *Comparing clusterings - an information based distance*. Journal of multivariate analysis, vol. 98, no. 5, pages 873–895, 2007. (Cited on pages 22 and 23.)
- [Meyer 2015] David Meyer, Evgenia Dimitriadou, Kurt Hornik, Andreas Weingessel and Friedrich Leisch. *e1071: Misc Functions of the Department of Statistics, Probability Theory Group (Formerly: E1071), TU Wien*, 2015. R package version 1.6-7. (Cited on page 77.)
- [Miller 2001] H Miller and J Han. *Spatial clustering methods in data mining: a survey*. Geographic data mining and knowledge discovery, Taylor and Francis, 2001. (Cited on page 8.)
- [Mirkin 1998] Boris Mirkin. *Mathematical classification and clustering: From how to what and why*. In Classification, Data Analysis, and Data Highways. Springer, 1998. (Cited on pages 94 and 121.)
- [Mondal 2012] Kartick Chandra Mondal, Nicolas Pasquier, Anirban Mukhopadhyay, Ujjwal Maulik and Sanghamitra Bandhopadhyay. *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. (Cited on pages 36, 38, 75 and 89.)
- [Ng 1994] R Ng and J Han. *Efficient and effective clustering method for spatial data mining*. In Proc. 1994 Int. Conf. Very Large Data Bases, pages 144–155, 1994. (Cited on page 9.)
- [Ng 2002] Andrew Y Ng, Michael I Jordan, Yair Weiss *et al.* *On spectral clustering: Analysis and an algorithm*. Advances in neural information processing systems, vol. 2, pages 849–856, 2002. (Cited on pages 17 and 26.)
- [Pasquier 1999] Nicolas Pasquier, Yves Bastide, Rafik Taouil and Lotfi Lakhal. *Efficient mining of association rules using closed itemset lattices*. Inf. Systems, vol. 24, no. 1, pages 25–46, 1999. (Cited on pages iii, v, 35, 36, 37 and 38.)
- [Punera 2008] Kunal Punera and Joydeep Ghosh. *Consensus-based ensembles of soft clusterings*. Applied Artificial Intelligence, vol. 22, no. 7-8, pages 780–810, 2008. (Cited on page 26.)

- [R Core Team 2016] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2016. (Cited on pages 75 and 77.)
- [Rand 1971] William M Rand. *Objective criteria for the evaluation of clustering methods*. Journal of the American Statistical association, vol. 66, no. 336, pages 846–850, 1971. (Cited on page 21.)
- [Rendón 2011] Eréndira Rendón, Itzel Abundez, Alejandra Arizmendi and ElviaM Quiroz. *Internal versus external cluster validation indexes*. International Journal of computers and communications, vol. 5, no. 1, pages 27–34, 2011. (Cited on page 18.)
- [Rousseeuw 1987] Peter J Rousseeuw. *Silhouettes: a graphical aid to the interpretation and validation of cluster analysis*. Journal of computational and applied mathematics, vol. 20, pages 53–65, 1987. (Cited on page 20.)
- [Sarumathi 2013] S Sarumathi, N Shanthi and M Sharmila. *A Comparative Analysis of Different Categorical Data Clustering Ensemble Methods in Data Mining*. IJCA, vol. 81, no. 4, pages 46–55, 2013. (Cited on page 33.)
- [Sheikholeslami 1998] Gholamhosein Sheikholeslami, Surojit Chatterjee and Aidong Zhang. *Wavecluster: A multi-resolution clustering approach for very large spatial databases*. In VLDB, volume 98, pages 428–439, 1998. (Cited on page 16.)
- [Sneath 1957] Peter HA Sneath. *The application of computers to taxonomy*. Microbiology, vol. 17, no. 1, pages 201–226, 1957. (Cited on page 9.)
- [Sokal 1958] Robert R Sokal. *A statistical method for evaluating systematic relationships*. Univ Kans Sci Bull, vol. 38, pages 1409–1438, 1958. (Cited on page 11.)
- [Sokal 1962] Robert R Sokal and F James Rohlf. *The comparison of dendrograms by objective methods*. Taxon, pages 33–40, 1962. (Cited on page 21.)
- [Sørensen 1948] Thorvald Sørensen. {*A method of establishing groups of equal amplitude in plant sociology based on similarity of species and its application to analyses of the vegetation on Danish commons*}. Biol. Skr., vol. 5, pages 1–34, 1948. (Cited on page 10.)
- [Strehl 2003] Alexander Strehl and Joydeep Ghosh. *Cluster ensembles – A knowledge reuse framework for combining multiple partitions*. JMLR, vol. 3, pages 583–617, 2003. (Cited on pages 23, 24 and 33.)
- [Topchy 2004] A.P. Topchy, M.H.C. Law, A.K. Jain and A.L. Fred. *Analysis of consensus partition in cluster ensemble*. In Proc. IEEE ICDM Conf., pages 225–232, 2004. (Cited on page 24.)

- [Ultsch 2005a] Alfred Ultsch. *Clustering with SOM: U*C*. In Proc. WSOM Workshop, pages 75–82, 2005. (Cited on page 78.)
- [Ultsch 2005b] Alfred Ultsch. *Fundamental clustering problem suite*, 2005. http://www.uni-marburg.de/fb12/datenbionik/data?language_sync=11. (Cited on pages iii, iv, 78, 79, 80, 82 and 84.)
- [Vega-Pons 2011] Sandro Vega-Pons and José Ruiz-Shulcloper. *A survey of clustering ensemble algorithms*. IJPRAI, vol. 25, no. 03, pages 337–372, 2011. (Cited on pages 3, 24, 25, 26, 29, 30, 33 and 108.)
- [Vega-Pons 2015] Sandro Vega-Pons and Paolo Avesani. *On pruning the search space for clustering ensemble problems*. Neurocomputing, vol. 150, pages 481–489, 2015. (Cited on page 32.)
- [Wang 1997] Wei Wang, Jiong Yang, Richard Muntz et al. *STING: A statistical information grid approach to spatial data mining*. In VLDB, volume 97, pages 186–195, 1997. (Cited on pages 15 and 16.)
- [Wang 2009] Xi Wang, Chunyu Yang and Jie Zhou. *Clustering aggregation by probability accumulation*. Pattern Recognition, vol. 42, no. 5, pages 668–675, 2009. (Cited on page 29.)
- [Wu 2008] Xindong Wu, Vipin Kumar, J Ross Quinlan, Joydeep Ghosh, Qiang Yang, Hiroshi Motoda, Geoffrey J McLachlan, Angus Ng, Bing Liu, S Yu Philip et al. *Top 10 algorithms in data mining*. Knowledge and information systems, vol. 14, no. 1, pages 1–37, 2008. (Cited on page 8.)
- [Wu 2012] 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., vol. 42, no. 3, pages 913–926, 2012. (Cited on pages 31 and 42.)
- [Xie 1991] Xuanli Lisa Xie and Gerardo Beni. *A validity measure for fuzzy clustering*. IEEE Transactions on Pattern Analysis & Machine Intelligence, no. 8, pages 841–847, 1991. (Cited on page 21.)
- [Xu 2005] Rui Xu and Donald Wunsch. *Survey of clustering algorithms*. IEEE Trans. on Neural Networks, vol. 16, no. 3, pages 645–678, 2005. (Cited on pages 8, 9 and 12.)
- [Xu 2015] Dongkuan Xu and Yingjie Tian. *A Comprehensive Survey of Clustering Algorithms*. Annals of Data Science, vol. 2, no. 2, pages 165–193, 2015. (Cited on pages 7, 8, 12, 13 and 17.)
- [Yoon 2006] Hye-Sung Yoon, Sun-Young Ahn, Sang-Ho Lee, Sung-Bum Cho and Ju Han Kim. *Heterogeneous clustering ensemble method for combining different cluster results*. In Data mining for biomedical applications, pages 82–92. Springer, 2006. (Cited on page 32.)

- [Zhang 2011] Yi Zhang and Tao Li. *Consensus Clustering + Meta Clustering = Multiple Consensus Clustering*. In Proc. FLAIRS Conf., 2011. (Cited on page [33](#).)

Traduction Française

A.1 Aperçu

Le Data Mining (**DM**), ou Fouille de Données, désigne “*l’application d’algorithmes spécifiques pour extraire des modèles de connaissances à partir de données*” [Fayyad 1996]. Le DM représente l’étape principale, algorithmique, du processus plus général appelé Knowledge Discovery in Databases (**KDD**), ou Extraction de Connaissances à partir des Données (**ECD**). Le processus d’ECD implique 5 étapes principales, comme décrit dans la figure A.1 : *i*) Sélection des données pertinentes pour la tâche de fouille de données; *ii*) Pré-traitement des données, comme par exemple le nettoyage de valeurs aberrantes ou le traitement des valeurs manquantes, pour l’extraction de modèles de connaissances pertinents; *iii*) Transformation des données pré-traitées, comme par exemple la normalisation des valeurs de variables numériques, afin de les adapter à la tâche de fouille; *iv*) Exploration des données, par l’utilisation de techniques d’analyse exploratoire, en fonction de la tâche de fouille concernée; et enfin *v*) Interprétation et/ou évaluation des modèles extraits pour les valider et déduire de nouvelles connaissances à partir de ceux-ci. Toutefois, le terme “Data Mining” est de nos jours largement utilisé pour décrire le processus de KDD au lieu d’être une partie de celui-ci.

L’ensemble de données¹ se compose de plusieurs instances (objets), chacune définie par un ensemble de valeurs d’attributs (variables). L’ensemble de données peut être représenté sous forme d’un tableau dans lequel chaque ligne est une instance et chaque colonne est un attribut. Les principales tâches du DM qui peuvent être effectuées sur l’ensemble de données pour découvrir les nouvelles connaissances sont :

- **Classification** : l’ensemble de données contient un attribut, appelée *variable cible* ou variable de réponse, qui catégorise les instances en différentes classes. L’objectif de la tâche de classification est d’apprendre un modèle de prédiction (représenté sous forme d’un arbre, de règles, de fonction de calcul, etc.) qui sera ensuite utilisé pour attribuer une étiquette de classe à de nouvelles instances non catégorisées. La classification est considérée comme une approche d’apprentissage supervisé, car elle utilise la connaissance du domaine (la valeur de la variable de réponse pour des exemples connus) afin d’apprendre à classer d’autres données.

¹Les termes *ensemble* et *jeu de données* sont largement utilisés dans le DM à la place de *base de données*.

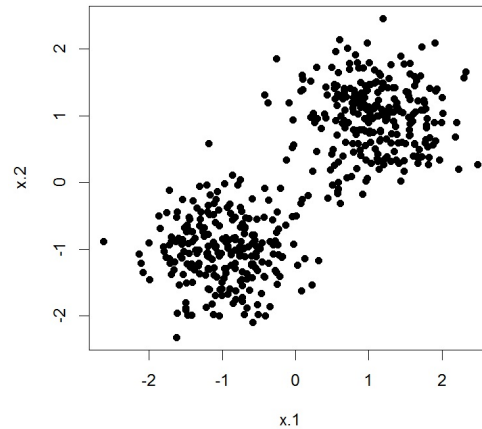


Figure A.2 – Exemple d’ensemble de données bi-dimensionnel.

manière pertinente dans l’espace des données. Autrement dit, nous ne disposons pas initialement d’information sur le nombre de groupes “naturels” présents dans les données, ni sur la forme de ces groupes. Considérons par exemple l’ensemble de données représenté dans la figure A.2. Nous pouvons voir que les instances, chacune représentée par un point, forment deux régions denses dans l’espace des données bi-dimensionnel. Cependant, dans les ensembles de données opérationnels, le nombre de dimensions (attributs) et le plus souvent très grand, ce qui rend impossible la visualisation des instances dans l’espace des données afin d’identifier les clusters.

De nombreux algorithmes de clustering ont été développés au cours des dernières décennies. Le plus souvent, chaque algorithme produit une partition différente lorsqu’il est appliqué au même ensemble de données, car ils sont conçus pour cibler un modèle spécifique (clusters compacts, clusters non-convexes, ...). Un autre facteur qui influe les résultats est le réglage des paramètres : la plupart des algorithmes de clustering exigent de l’utilisateur qu’il spécifie le nombre de clusters recherchés (généralement connu en tant que paramètre K), et/ou d’autres paramètres spécifiques à l’algorithme de clustering considéré. Par exemple, les méthodes basées sur la densité ne nécessitent pas de paramètre K , mais nécessitent d’autres paramètres pour définir ce qui est une région dense dans l’espace de données. Ainsi, la question est : comment choisir un clustering pour un ensemble de données à partir de ces nombreuses possibilités?

La solution la plus courante consiste à utiliser la(les) mesure(s) de validation pour comparer les résultats et sélectionner celui qui obtient le score le plus élevé [Dalton 2009, Halkidi 2001a]. Il existe deux catégories générales de mesures de validation : *validation interne* qui compare le regroupement contre un modèle de clustering spécifique et *validation externe* qui compare le regroupement contre de véritables étiquettes (étiquettes de classe données sur un ensemble d’évaluation en

utilisant les connaissances de domaine). De nombreuses mesures de validation ont été proposées dans ces deux catégories, mais aucune évalue de façon impartiale les résultats de tout algorithme de clustering [Vega-Pons 2011]. L'utilisateur peut ainsi obtenir des scores d'évaluation similaires pour des mesures de validation différentes et/ou pour différents résultats du clustering, alors que les résultats sont dissemblables par de nombreux aspects tels que le nombre de clusters ou la façon de regrouper les instances en clusters.

Plutôt que de dépendre des mesures de validation, une autre approche consiste à combiner les nombreuses solutions de clustering générées par plusieurs algorithmes et/ou paramétrages, afin de produire une partition finale plus pertinente que celle que chaque algorithme peut produire individuellement. Cette technique est appelée *clustering par consensus*, *ensemble de clustering* ou *agrégation de clusterings*, et les algorithmes de clustering combinés sont appelés *algorithmes de clustering de base*. De nombreuses méthodes de clustering par consensus ont été proposés, celles-ci sont présentées dans le chapitre suivant de ce mémoire. Cependant, un certain nombre d'approches de clustering par consensus imposent des limites importantes à l'ensemble de clustering de base, comme par exemple imposer à tous de produire le même nombre de clusters. D'autres approches nécessitent une fonction de consensus d'une grande complexité en terme d'espace de stockage ou de temps de calcul. Dans ce travail, nous présentons une nouvelle catégorie de méthodes de clustering par consensus, qui est basée sur l'utilisation des itemsets fermés fréquents (**IFF**) issue du domaine de l'extraction de motifs fréquents et règles d'association. Les avantages de l'approche proposée sont étudiés dans la section suivante.

A.3 Contribution

- Pas de contraintes imposées à la sélection des algorithmes de clustering de base et leurs paramétrages.
- Inutile de spécifier le paramètre K pour le consensus. La méthode proposée est capable de détecter automatiquement la structure interne des clusters cachés.
- En utilisant les IFF, la recherche pour une solution de consensus est réalisée sur un espace de constitués de motifs fréquents au lieu de l'espace de données des instances. L'espace des motifs fréquents fournit un élagage important de l'espace de données des instances, en particulier pour les grands ensembles de données.
- Le processus de construction du clustering par consensus est expliqué à l'utilisateur final par un diagramme de Hasse appelé ConsTree. Le ConsTree met en évidence des relations de regroupement entre les instances ce qui rend la compréhension de la structure naturelle interne des clusters possible, même dans le cas d'ensembles de données de grande dimension.
- La méthode proposée génère plusieurs clusterings par consensus candidats

et recommande l'un d'eux à l'utilisateur final. La visualisation fournie par le ConsTree donne à l'utilisateur final la possibilité d'envisager l'utilisation d'un autre consensus que celui proposé, en fonction de ses connaissances du domaine, préférences et observations.

A.4 Structure de la Thèse

Le chapitre 2 donne un aperçu général des algorithmes de clustering, des techniques de validation et des catégories de méthodes de clustering par consensus. Le chapitre 3 présente l'extraction de motifs fréquents et l'utilisation des IFF dans la méthode de construction des clusterings par consensus proposée. Dans le chapitre 4, diverses approches algorithmiques pour l'amélioration de la méthode proposée sont étudiées. Les expérimentations menées sur des ensembles de référence de données synthétiques et réelles sont présentées dans le chapitre 5. La conclusion et des perspectives de développement ultérieurs de l'approche sont présentées dans le chapitre 6.

A.5 Résumé du Chapitre 2 : Contexte

Le clustering est le processus de partitionnement d'un ensemble de données en groupes, de sorte que les instances du même groupe sont plus semblables les unes aux autres que des instances des autres groupes. Les algorithmes de clustering peuvent être catégorisés en fonction de leur modèle sous-jacent :

- **Par partitionnement** : dans cette catégorie, également connu sous le nom d'approche par centroïdes, les instances sont regroupées en fonction de leur distance aux centroïdes des clusters. Parmi les algorithmes représentatifs de cette approche, nous pouvons citer : K-means, PAM, CLARA et CLARANS.
- **Hiérarchique** : les instances sont regroupées dans une hiérarchie de clusters. Il existe deux types d'approches parmi celles de clustering hiérarchique : bas-haut et haut-bas. Dans l'approche bas-haut, aussi appelée par agglomérations, le processus de regroupement commence en considérant chaque instance comme un cluster, puis fusionne les deux clusters les plus proches à chaque itération jusqu'à ce que la combinaison de tous les instances dans un cluster. Dans l'approche haut-bas, également appelée par division, le processus débute en considérant l'ensemble des instances comme un unique cluster, puis divise à chaque itération un cluster en deux jusqu'à ce que chaque instance devienne un cluster. Les principaux algorithmes dans cette catégorie sont : single linkage, complete linkage, average linkage, centroid linkage, and DIANA.
- **Ensembles flous** : chaque instance peut appartenir à plusieurs clusters simultanément, avec différents degrés d'appartenance pour chacun. Le clustering flou peut être considéré comme type spécial de clustering par partitionnement. Les algorithmes de clustering flou les plus largement utilisés sont : Fuzzy C-Means et FCM.

- **Par densité** : un cluster est défini comme étant une région dense dans l'espace des données séparée des autres clusters par des régions où la densité est faible. Dans cette hypothèse, les clusters peuvent prendre une forme arbitraire, contrairement à la majorité des approches ci-dessus dont les clusters sont de forme sphérique. Les algorithmes DBSCAN et DENCLUE sont deux algorithmes populaires de clustering par densité.
- **Par distribution** : les instances sont considérées comme étant générées par K distributions de probabilités, chacune représentant un cluster. Nous avons besoin dans cette approche non seulement de déterminer K , mais également les paramètres des modèles de distribution correspondant le mieux aux données. L'algorithme EM (Expectation-Maximization) est la méthode la plus populaire dans cette catégorie.
- **Basée sur les graphes** : les instances sont traitées comme les noeuds d'un graphe reliés par des arêtes en fonction de leurs similitudes. Les clusters sont alors formés par partitionnement de ce graphe. Chameleon est un exemple d'algorithme de clustering basé sur les graphes.
- **Par grille** : l'idée ici consiste à projeter les instances dans les cellules d'une grille divisant l'espace des données, chaque cellule regroupant ainsi plusieurs instances. Les algorithmes STING, CLIQUE et WaveCluster sont des algorithmes représentatifs de cette approche.
- **Spectral** : l'espace de données est transformée en espace de vecteurs propres (Eigenvectors) afin de mieux distinguer les clusters.

Les résultats des algorithmes de clustering dépendent des paramètres utilisés et une modification de ces paramètres peut produire un regroupement différent des instances en clusters, et éventuellement un nombre différent de clusters. Plusieurs difficultés se posent donc pour l'application d'un algorithme de clustering : comment déterminer si une modification des paramètres améliore ou réduit la qualité des clusters résultant? Comment connaître le nombre exact de clusters "naturels" cachés dans l'ensemble de données? En outre, considérant les différentes catégories d'algorithmes de clustering, comment choisir un algorithme spécifique sans connaître la structure interne des clusters cachés? Il est donc nécessaire de comparer les résultats obtenus à partir des différents algorithmes et paramétrages afin d'identifier celui produisant les résultats les plus pertinents. Plusieurs mesures de validation de clustering ont été proposées dans cet objectif.

Les méthodes de validation internes analysent les propriétés des clusters résultant, comme leur séparation et leur compacité. Parmi les mesures de validation interne, nous pouvons citer l'indice Dunn, l'indice Davies-Bouldin (DB), l'indice SD, l'indice S_Dbw , et l'indice de silhouette. Toutes ces mesures ont des performances différentes pour l'identification du "meilleur" regroupement selon différents problèmes liés à l'espace des données, tels que l'existence de bruit, des densités variables, ou des clusters faiblement séparés (clusters très proches). Un inconvénient

lié à l'utilisation de ces mesures est qu'elles sur-évaluent les algorithmes basés sur le même modèle de clustering. Par exemple, une mesure d'évaluation basée sur la notion de distance entre les instances assignera un score élevé à un clustering réalisé par l'algorithme des K-means, celui-ci optimisant naturellement cette distance. En outre, des scores élevés ne reflètent pas nécessairement de bons résultats en termes d'application effective pour la recherche d'information dans les données.

Les mesures externes évaluent les résultats d'un clustering en comparant les clusters et des étiquettes de classe prédéfinies par des experts du domaine. Le nom "externe" est dû au fait que ces classes ne font pas partie des données utilisées dans le processus de regroupement. Parmi les mesures externes largement utilisés : l'indice Rand, l'indice Jaccard, l'indice Fowlkes & Mallows, la pureté, l'information mutuelle, et la variation de l'information. Un problème rencontré avec les mesures de validation externes est qu'elles ne sont applicables que si des connaissances sur les étiquettes de classe existent, ce qui n'est pas toujours le cas pour des applications opérationnelles. De plus, les résultats de ces mesures, principalement appliquées pour des expérimentations sur des ensembles de données synthétiques, ne sont pas toujours fiables pour des données opérationnelles. En effet, les classes peuvent contenir une structure interne ne correspondant pas aux clusters distingués par les attributs, ou encore contenir des anomalies (exceptions).

Chaque mesure de validation, que ce soit interne ou externe, utilise une approche différente pour justifier la qualité d'un résultat de regroupement. Dans la pratique, la(les) mesure(s) de validation appropriée(s) restent inconnue(s). Une autre technique afin d'améliorer la qualité des clusters consiste à combiner différentes solutions de clustering (appelés *clusterings de base*) et construire un partitionnement par consensus potentiellement plus adapté aux données que ce que chaque clustering de base permet de générer. Ce processus est appelé *clustering par consensus*, *ensemble de clustering* ou *agrégation de clusterings*. Il implique deux étapes : tout d'abord, la construction d'un ensemble de partitions (la combinaison de toutes les partitions qui sont fournis par des algorithmes de clustering de base), puis en lui appliquant une fonction de consensus. Le problème du clustering par consensus peut être défini comme suit :

Considérons un ensemble de données D de d dimensions et composé de N instances, $D = \{x_1, x_2, \dots, x_N\}$. Soit $\mathbb{P} = \{P_1, P_2, \dots, P_M\}$ un ensemble de M partitions acquies par l'application de différents scénarios de clustering à l'ensemble D . Chaque partition P_i divise D en K_i clusters, soit $P_i = \{C_1^i, C_2^i, \dots, C_{K_i}^i\}$. C_j^i identifie le cluster j dans la partition i . Soit $\mathbb{P}_D$ l'ensemble de toutes les partitions possibles de D . Nous avons $\mathbb{P} \subset \mathbb{P}_D$. L'objectif du clustering par consensus est de trouver une partition $P^* \in \mathbb{P}_D$ qui représente au mieux les partitions dans $\mathbb{P}$.

Pour résoudre ce problème, différentes méthodes de clustering par consensus ont été proposées. Ces méthodes peuvent être classées en fonction de l'approche sous-jacente utilisée :

- **Basée sur les graphes** : le problème de consensus est formulé comme un problème de partitionnement de graphe (ou hypergraphe). Autrement dit, les

instances (et/ou les clusters) sont représentés par les noeuds d'un graphe reliés par des arêtes dont le poids correspond à la similarité (ou l'appartenance) entre les noeuds. Les méthodes de consensus dans cette catégorie sont : Cluster-based Similarity Partitioning Algorithm (CSPA), HyperGraph-Partitioning Algorithm (HGPA), Meta-CLustering Algorithm (MCLA), Hybrid Bipartite Graph Formulation (HBGF), sCSPA, sMCLA, sHBGF, Weighted Similarity Partitioning Algorithm (WSPA), Weighted Bipartite Partitioning Algorithm (WBPA) et Weighted Subspace Bipartite Partitioning Algorithm (WSBPA).

- **Par ré-étiquetage et vote** : les clusterings de base sont considérés comme des électeurs, où chacun vote pour que l'instance appartienne à un groupe particulier. Cependant, comme chaque étiquette de cluster dans un clustering est sans rapport avec celles des autres clusterings dans l'ensemble, la tâche principale consiste ici à résoudre le problème de correspondance des étiquettes dans l'ensemble. L'approche Bagged Clustering est une méthode de consensus représentative de cette catégorie.
- **Matrice de co-association** : une matrice de co-association est utilisée comme représentation intermédiaire de l'ensemble de clustering de base. Un algorithme de clustering est alors appliqué à cette matrice pour obtenir la partition de consensus. Les algorithmes Evidence Accumulation (EA) et Probability Accumulation (PA) sont des exemples de cette approche.
- **Par distance** : dans les approches basées sur la distance, la partition de consensus est définie comme celle qui a la distance Mirkin minimale à l'ensemble. Parmi les approches de cette catégorie, nous avons par exemple Best Of K (BOK), Simulated Annealing One-element Move (SAOM), Best One-element Move (BOM), Balls algorithm, Agglomerative algorithm, Farthest algorithm et LocalSearch algorithm.
- **Basée sur les fragments** : plutôt que de chercher une solution consensuelle dans l'espace des données complet, des *fragments de données* sont utilisés pour réduire l'espace de recherche. Un fragment de données (défini en considérant l'ensemble de clusterings) est un ensemble d'instances qui ne sont pas divisées par l'un des clusterings de base et qui ne figurent pas dans tout autre ensemble non divisé lui-même par les clusterings de base. Les algorithmes F-Agglomerative, F-Farthest, F-LocalSearch et F-CARS appartiennent à cette catégorie.

Généralement, les dispositions suivantes résument certaines des contraintes dans les méthodes précédentes :

- Le nombre K de clusters générés par la solution de consensus est nécessaire dans les méthodes de consensus basées sur le ré-étiquetage et vote, les graphes et les méthodes qui appliquent un algorithme de clustering pour générer une

solution de consensus. La difficulté vient de l'impossibilité de prédire la valeur optimale de K spécifique à l'ensemble de données traité.

- La majorité de ces méthodes imposent des restrictions sur le processus de génération de l'ensemble de clusterings. C'est par exemple le cas pour les méthodes basées sur le ré-étiquetage et vote dans lesquelles le problème de correspondance des étiquettes est résolu plus efficacement lorsque toutes les partitions de l'ensemble sont constituées du même nombre de clusters.
- Les méthodes de consensus basées sur l'utilisation d'une matrice de co-association ou d'une matrice de distances entre clusters de bases sont inapplicables pour les grands ensembles de données, ces matrices exigeant un espace de stockage important.
- L'utilisation d'un algorithme de clustering pour générer la solution de consensus impose deux restrictions : la complexité temporelle de l'algorithme et la forme des clusters générés par ce dernier.

Dans le chapitre suivant, une nouvelle méthode de clustering par consensus est présentée. Celle-ci permet de s'abstraire de l'ensemble des restrictions évoquées ci-dessus.

A.6 Résumé du Chapitre 3 : Méthode proposée

Dans ce chapitre, une nouvelle méthode de clustering par consensus, appelée MultiCons, est proposée. Cette méthode est la première à transformer le problème de clustering consensuel en un problème d'extraction de motifs (patterns). En utilisant une matrice d'appartenance binaire pour représenter l'ensemble, une technique d'extraction de motifs (basée sur les itemsets fermés fréquents) est appliquée afin d'identifier les similitudes entre les clusterings de base. Le résultat est un réseau de *motifs de clustering*, chacun définissant un agrément entre plusieurs clusterings de base sur le regroupement d'un ensemble d'instances. Considérant ce réseau comme l'espace de recherche pour le problème de consensus, nous pouvons continuer à le diviser en sous-espaces en fonction du nombre de clusterings de base qui définissent les motifs. Cela permet de générer des solutions consensuelles multiples en regroupant les motifs dans chaque sous-espace.

Un élément important de la méthode MultiCons est la représentation graphique générée appelée *arbre de consensus*, ou ConsTree. Chaque niveau du ConsTree représente une solution de consensus dans laquelle chaque noeud représente un cluster. Il permet également de mettre en évidence les relations entre clusters appartenant à des consensus successifs, liés par des arcs représentant la relation d'inclusion, apportant ainsi une meilleure compréhension des relations entre les instances dans l'espace de données.

Clusterings de base : Les clusterings de base sont des partitions strictes d'un ensemble de données, sans limitation sur le nombre de clusters ou la catégorie des algorithmes de clustering utilisés. Ainsi, nous pouvons combiner des clustering basés sur le partitionnement, tels que générés par les K-means ou PAM, avec les résultats d'algorithmes de clustering hiérarchiques, basés sur les modèles gaussien ou basés sur la densité par exemple dans lesquels chaque instance de l'ensemble de données appartient à un unique cluster. Il est préférable d'utiliser différentes valeurs pour le paramètre K pour chaque algorithme de clustering de base, et également différentes valeurs pour les autres paramètres spécifiques à l'algorithme, dans la mesure du possible, afin d'assurer la diversité dans les partitions de base.

Matrice d'appartenance binaire : Une matrice d'appartenance binaire $\mathcal{M}$ se compose de N lignes correspondant chacune à une instance de l'ensemble de données, et N_c colonnes correspondant chacune à un cluster de l'ensemble de clusterings de base. Une entrée $[x, y]$ dans la matrice $\mathcal{M}$ prend la valeur 1 si l'instance x appartient au cluster y , et prend la valeur 0 sinon.

Identification des motifs de clustering : La matrice $\mathcal{M}$ définit l'espace de recherche du problème d'extraction de motifs. En utilisant la technique des IFF, nous pouvons identifier des *motifs de clustering*, ou plus précisément, *motifs fréquents fermés* (MFF). Chaque MFF se compose de deux ensembles : L'ensemble C de clusters de base qui sont en accord sur le regroupement de l'ensemble I d'instances.

Génération de plusieurs consensus : La construction des consensus à partir des MFF est un processus itératif considère lors de chaque itération tous les MFF correspondant à un nombre spécifique, appelés *Decision Threshold* (DT), de clusterings de base. La valeur DT représente le nombre minimum de clusterings de base à considérer pour la construction d'un consensus. Pour le premier consensus, nous avons $DT = MaxDT$, où $MaxDT$ est le nombre de clusterings de base utilisé. La valeur DT est alors décrémentée séquentiellement jusqu'à $DT = 1$, pour intégrer dans le nouveau consensus une autre vue des regroupements générés par un plus petit nombre de clusterings de base. A l'itération n , une solution consensuelle est donc construite à partir des ensembles d'instances I des motifs dont l'ensemble C des clusters de base est de taille n , plus les clusters du consensus précédent (générés à l'itération $n + 1$). Un ensemble d'instances I possède l'une des trois propriétés suivantes :

- i) Unicité : n'a pas d'intersection avec une autre ensemble I' , c'est-à-dire $I \cap I' = \emptyset$.
- ii) Inclusion : est un sous-ensemble d'un autre ensemble I' , c'est-à-dire $I \subseteq I'$.
- iii) Intersection : il possède une intersection avec un autre ensemble I' , c'est-à-dire $I \cap I' \neq \emptyset$, $I \setminus I' \neq \emptyset$ et $I' \setminus I \neq \emptyset$.

L'objectif de la fonction de consensus est de générer des clusters non recouvrant à partir des ensembles d'instances considérés durant l'itération. Pour ce faire, la première approche proposée est décrite dans le paragraphe suivant. D'autres approches seront discutées dans le chapitre suivant.

MultiCons approche 1 :

- Un ensemble d'instances avec la propriété d'unicité signifie que le regroupement des instances basés sur un accord entre DT clusterings de base n'a pas changé en considérant $DT - 1$ clusterings de base. Celui-ci représentant une décision de clustering "forte", cet ensemble d'instances devient un cluster du consensus de l'itération courante $n = DT - 1$.
- Les ensembles d'instances avec la propriété d'inclusion sont supprimés, ceci afin de considérer une nouvelle décision de clustering agréée par $DT - 1$ clusterings de base.
- Les ensembles d'instances avec la propriété d'intersection sont regroupés afin de former un nouveau cluster, l'existence d'instances partagées entre eux étant une indication de proximité de ces ensembles dans l'espace de données. Différents traitements de ces ensembles sont examinés dans le chapitre suivant.

Le processus décrit ci-dessus est répété jusqu'à ce que tous les ensembles possèdent la propriété d'unicité, c'est-à-dire qu'ils ne se chevauchent pas. Après avoir généré tous les clusterings consensuels candidats, ceux qui ont été générés à plusieurs reprises, pour des valeurs successives de DT , sont supprimées. L'index de Jaccard est utilisé pour déterminer si deux candidats clusterings consensuels candidats sont identiques, et celui associé à la valeur de DT la plus faible est éliminé. Un compteur de *STabilité* (ST) est alors utilisé pour indiquer combien de fois un même consensus est généré pour des différentes valeurs de DT .

La grande stabilité d'un clustering consensuel candidat peut refléter sa force et identifier un consensus comme solution finale proposée. Toutefois, ceci n'étant pas toujours le cas, la solution finale proposée correspondra au consensus le plus semblable à l'ensemble. Pour estimer cela, chaque clustering candidat est comparé en utilisant l'indice de Jaccard à chaque clustering dans l'ensemble. Le clustering candidat ayant la similitude moyenne la plus élevée est alors sélectionné comme solution finale proposée.

ConsTree : La dernière étape du processus consiste à présenter les consensus générés dans une structure arborescente, appelée ConsTree, explicitant comment les regroupements des instances dans chaque sous-espace DT . Chaque niveau dans le ConsTree représente les clusters du consensus obtenu pour une valeur de DT spécifique. Les niveaux sont ordonnés en fonction de la valeur de DT , le premier consensus étant représenté par le niveau le plus bas de l'arbre. Dans chaque niveau de l'arbre, l'étiquette et la taille (diamètre) du noeud reflètent la taille (nombre d'instances) du cluster.

La visualisation du ConsTree permet à l'analyste de mieux appréhender comment les consensus ont été construits en fonction des différentes combinaisons de clusterings de base. Ceci permet d'identifier les partitions "fortes" dans l'ensemble de données, représentées par les clusters n'ayant pas été fusionnés avec d'autres sur une succession de consensus, ce qui reflète une forte similitude entre leurs instances. Par rapport aux autres méthodes de consensus, qui fournissent une seule solution représentant le clustering le plus similaire à l'ensemble de clusterings de base, le ConsTree fournit non seulement des informations complémentaires pour la compréhension des relations fortes dans les données, mais aussi pour le choix par l'utilisateur d'un clustering final basé sur ses connaissances antérieures et ses préférences.

A.7 Résumé du Chapitre 4 : Améliorations

Le regroupement systématique des instances d'ensembles qui s'intersectent dans un seul cluster de l'approche 1 de MultiCons génère parfois un résultat inapproprié. Afin d'améliorer les regroupements, au lieu de fusionner systématiquement deux ensembles qui s'intersectent quel que soit le nombre d'instances communes entre eux, nous pouvons utiliser la taille de l'intersection afin de déterminer s'il est plus approprié de les fusionner ou non.

Soit $I(X|Y)$ définissant la taille de l'ensemble X qui constitue une partie de l'ensemble Y comme le ratio d'intersection entre les deux ensembles en fonction de la taille de l'ensemble X , soit:

$$I(X|Y) = \frac{|X \cap Y|}{|X|}$$

Les approches suivantes proposées sont basées sur cette mesure.

MultiCons approche 2 : En utilisant un *Merging Threshold* (MT), deux ensembles X et Y qui s'intersectent seront fusionnés seulement si $I(X|Y)$ ou $I(Y|X)$ est supérieur ou égal MT . Dans le cas contraire, les deux ensembles sont divisés et les instances partagées sont conservées dans l'ensemble de plus petite en taille (le cluster le plus compact) et retirées de l'autre ensemble de plus grand en taille (le cluster le plus éparse). Les autres ensembles, qui ne possèdent pas la propriété d'intersection, sont traités de manière identique à l'approche 1.

MultiCons approche 3 : Dans cette approche, davantage de restrictions sont ajoutées au processus de fusion : deux ensembles X et Y qui s'intersectent sont fusionnées si et seulement si leur ratio moyen d'intersection est supérieur ou égal à MT , le ratio moyen d'intersection étant défini comme suit:

$$Avg_I(X, Y) = \left(\frac{|X \cap Y|}{|X|} + \frac{|X \cap Y|}{|Y|} \right) \times 0.5$$

L'approche 2 examine les ensembles qui s'intersectent vis à vis d'un seuil de fusion, MT , pour décider de les fusionner ou les diviser. Cependant, il le fait d'une manière

péremptoire : il fusionne ou divise un ensemble d'instances avec le premier ensemble qui s'intersecte, puis continue à traiter les autres ensembles qui s'intersectent. L'approche 3 est basée sur l'idée qu'il semble préférable de rechercher pour chaque ensemble d'instances celui qui présente le meilleur ratio d'intersection parmi les ensembles qui s'intersectent avec lui. Les deux ensembles ainsi identifiés sont alors fusionnés si $Avg_I \geq MT$, et divisés sinon.

MultiCons approche 4 : Bien que l'approche 3 se propose d'améliorer le processus de fusion, elle parcourt les ensembles d'instances séquentiellement. Une amélioration possible consiste donc à rechercher en priorité les meilleurs ratio d'intersection parmi tous ceux disponibles. Cette approche nécessite une matrice de ratios d'intersection et l'utilisation d'un vecteur de pointeurs pour définir la séquence du processus de fusion/division. Cette matrice est constituée de $S \times S$ cellules, où S est le nombre d'ensembles d'instances de l'itération DT . Le ratio d'intersection (approche 2) ou le ratio moyen d'intersection (approche 3) peut alors être utilisé pour définir la valeur de chaque cellule.

MultiCons approche 5 : Dans cette approche, la matrice d'intersection est utilisée comme matrice d'adjacence d'un graphe, chaque cellule de la matrice d'intersection définissant le poids d'un arc qui relie les ensembles d'instances i et j . Les algorithmes de partitionnement de graphes nécessitant le paramètre K afin de générer K clusters, un processus de fusion/division est appliquée sur cette matrice. L'idée est de retirer les arcs de poids inférieur à MT et fusionner tous les noeuds restants connectés dans un cluster. Le processus est ainsi similaire à celui de l'approche 1, avec l'ajout de l'opération de division.

Amélioration du ConsTree : Nous avons présenté dans le chapitre précédent le ConsTree, qui est un outil pour la compréhension des regroupements des instances menant à des solutions consensuelles différentes, et des relations entre les clusters de consensus successifs. Comme seul le processus de fusion est réalisé dans l'approche 1, l'arbre est simple à visualiser. Un cluster à un niveau bas dans l'arbre est lié à un seul cluster au niveau supérieur successif. Dans ce chapitre, l'opération de division est ajoutée pour améliorer le résultat de clustering par consensus, ce processus conduisant à différents regroupements des instances lorsque l'on parcourt l'arbre vers le haut. Autrement dit, un cluster à un niveau inférieur peut être lié à plusieurs clusters du niveau suivant, si certaines de ses instances sont regroupées différemment lorsque l'on considère un nombre inférieur d'agréments entre les clusterings de base. Ceci peut compliquer la lecture de l'arbre si de nombreux liens multiples entre clusters successifs sont générés.

Afin de simplifier la lecture de l'arbre dans cette situation, un processus de *raffinement de l'arbre* est proposé : un cluster x à un niveau inférieur de l'arbre est lié au cluster y du niveau immédiatement supérieur auquel la majorité de ses instances appartiennent. Les autres liens sont supprimés, ce qui correspond à la suppression

des instances de x qui changent de groupe, en considérant la majorité des instances de x . Le processus de raffinement ne modifie pas les solutions consensuelles, mais améliore simplement la visualisation de l'arbre en tant qu'outil d'analyse des regroupements.

Les valeurs par défaut pour MT : La paramètre MT , qui est le seul paramètre pour les approches 2 à 5, prend une valeur dans l'intervalle $[0,1]$. Pour $MT = 0$, les consensus résultants sont identiques à ceux de l'approche 1, c'est à dire la fusion des ensembles d'instances qui s'intersectent. Pour $MT = 1$, la division systématique de tous les ensembles qui s'intersectent entraîne la génération d'un grand nombre de petits clusters, ce qui n'est usuellement pas recommandé. Selon les résultats de tests expérimentaux intensifs, la valeur $MT = 0.5$ est proposée comme valeur par défaut afin de produire une fusion/division modérée des ensembles. Cette valeur a permis de générer la meilleure solution de consensus dans la quasi-totalité des expérimentations, et une solution très proche de la meilleure sinon, pour l'approche 2. Du fait de la modification du processus de fusion dans les approches 3 et 4, les expérimentations ont montré qu'une plus faible valeur de MT doit être utilisée pour produire de meilleurs résultats. Pour l'approche 3, la valeur par défaut pour MT est de 0.4, alors que pour l'approche 4, la valeur par défaut est de 0.3. Pour l'approche 5, qui ne réalise pas de processus itératif de fusion/division comme les autres approches, la valeur par défaut est de 0.6.

A.8 Résumé du Chapitre 5 : Expérimentations

Les tests ont été réalisés sur un DELL PRECISION M4800 comportant un processeur Intel® Core™ i7-4710MQ @ 2.50GHz, 32 Go de RAM, et utilisant Microsoft Windows 10 Professional 64-bits comme système d'exploitation. Les algorithmes correspondant aux différentes approches de MultiCons ont été implantés en langage R. Le package *CLUE* de R qui comporte plusieurs méthodes de clustering par consensus basées sur le ré-étiquetage notamment, a été utilisé afin de comparer les résultats des différentes méthodes. Cependant, les méthodes du package *CLUE* nécessitent K en tant que paramètre afin de générer des consensus de K clusters, paramètre qui n'est pas requis par les approches MultiCons. Ainsi, pour les méthodes du package *CLUE*, K est fixé au nombre réel de clusters dans l'ensemble de données, alors que l'approche MultiCons essaye de générer les clusters "naturels" (ou un résultat proche) dans son processus de regroupement.

La qualité de la solution des différentes solution de clustering manipulées, que ce soit des clusterings de base, générés par MultiCons, ou des méthodes du package *CLUE*, est estimée en utilisant l'indice de Jaccard calculé en comparant les étiquettes des classes réelles disponibles pour les ensembles de données de test.

Pour la construction de l'ensemble de clusterings de base pour chaque test, les algorithmes de clustering suivants sont utilisés : K-Means, PAM, C-Means, clustering hiérarchique avec différentes liaisons, DIANA, modèles gaussien, DBSCAN,

clustering spectral et bagged clustering.

Les tests ont été effectués sur deux types d'ensembles de données : synthétiques et réelles. Les ensembles de données synthétiques sont conçus afin d'évaluer les résultats des algorithmes de clustering dans différentes situations correspondant à des problèmes de clustering difficiles, tels que des densités variables dans l'espace des données, des clusters recouvrants, des distances variables entre clusters, l'existence de bruit dans les données, etc. Les ensembles de données réelles sont issues du référentiel *UCI* et le package *mlbench* de R. Ces ensembles de données n'étant pas conçus initialement pour la tâche de clustering, l'évaluation des clusterings sera basée sur la similitude des clusters du clustering et les classes réelles de l'ensemble de données.

Dans tous les tests, la définition du paramètre K au véritable nombre de classes dans l'ensemble de données pour les méthodes du package *CLUE* constitue un avantage par rapport à la méthode MultiCons. Sans cela, la valeur par défaut pour K est usuellement choisie comme la valeur maximale de K dans l'ensemble de clusterings de base, ce qui, dans le cas d'ensembles où la valeur maximale de K est plus élevée que la valeur réelle, peut résulter en des clusterings de qualité très inférieure au consensus générés par MultiCons. D'autre part, les approches de MultiCons se sont avérées capables de générer le plus souvent le nombre réel de clusters, ou un nombre très proche sinon, à l'exception des approches 1 et 5 qui ne réalisent pas de processus itératif de fusion/division des motifs de clustering.

La qualité des consensus résultant dépend fortement de la qualité de l'ensemble de clusterings de base. Dans tous les tests, les ensembles de clusterings de base ont été créés aléatoirement, incluant ainsi des partitions de bonne et mauvaise qualité, sans qu'aucun traitement ne soit réalisé pour ne générer que de relativement bonnes partitions dans l'ensemble. Ceci dans le but d'imiter le fait que, dans le cas d'applications opérationnelles, nous ne disposons pas toujours des étiquettes de classe réelle afin de comparer les résultats. Toutefois, un analyste ayant une bonne expérience des algorithmes de clustering pourrait parvenir à des solutions consensuelles de qualité supérieure que ceux produits dans les tests.

En ce qui concerne les temps d'exécution, bien que les approches de MultiCons ne soient pas les plus rapides lors des tests, leurs temps d'exécution, de l'ordre de quelques secondes, restent dans tous les cas acceptables. Alors que l'approche MultiCons 1 est la plus rapide parmi, la recherche effectuée dans les approches 3 et 4 peut augmenter sensiblement les temps d'exécution par rapport à l'approche 2.

A.9 Conclusions et travaux futur

Une nouvelle méthode de clustering par consensus, appelée MultiCons, a été présentée dans cet ouvrage. Elle utilise la technique d'extraction des motifs fermés fréquents afin de transformer les relations entre les instances et l'ensemble de clusterings de base en motifs de clustering. En divisant l'espace des motifs en sous-espaces, MultiCons est capable de générer différents clustering consensuels par des combinaisons

différentes des clusterings de base. Dans chaque sous-espace, MultiCons regroupe les instances en fonction des informations fournis dans les motifs. Le traitement successif des différentes vues de regroupement, chacune correspondant à un nombre différent d'agrément entre les clusterings de base, permet de découvrir le nombre de clusters cachés dans l'ensemble de données, sans avoir ainsi besoin de le spécifier explicitement, et de construire une représentation graphique arborescente, appelée ConsTree, de visualisation des relations entre les instances. À l'aide du ConsTree, l'utilisateur n'est pas limité à une solution unique finale, mais il peut choisir la solution qui correspond au mieux à ses besoins en fonction de ses observations et préférences. Il peut par exemple choisir une solution de clustering où un cluster particulier est divisé en deux clusters, qui reflète un groupement plus significatif ou utile pour l'application concernée.

Cinq approches algorithmiques de regroupement des motifs différentes sont présentées. Les points suivants résument les conclusions découlant des tests effectués :

- L'approche 2 est recommandée comme standard pour MultiCons pour les raisons suivantes: *i)* Elle produit de bons résultats dans l'ensemble des tests en termes de qualité et de nombre de clusters générés. *ii)* Les approches 3 et 4 se sont avérées globalement aussi pertinentes que l'approche 2, la différence dans la qualité de leurs résultats étant généralement très faible. Toutefois, les approches 3 et 4 peuvent nécessiter davantage de temps de calcul que l'approche 2 car elles essayent de trouver les motifs plus semblables avant de prendre une décision de fusion/division, alors que l'approche 2 réalise le processus de fusion/division directement. *iii)* Les approches 1 et 5 échouent parfois à générer des solutions d'aussi bonne qualité que les 3 autres, en particulier dans le nombre de clusters générés. Ceci est dû aux similarités de leurs processus de traitement des motifs de clustering.
- Les tests réalisés ont démontré qu'un ensemble de clusterings de base composé de partitions contenant un nombre différent de clusters permet globalement de générer une meilleure solution de consensus, par rapport à un ensemble dont les partitions sont constituées du même nombre de clusters.
- Les tests ont également démontré qu'un faible seuil de fusion (MT) entraîne le plus souvent des temps d'exécution inférieurs par rapport à des valeurs plus élevées. En effet, une faible valeur de MT signifie davantage de fusions, ce qui réduit le nombre d'ensembles d'instances considérés durant chaque itération de l'approche.

Parmi les recommandations concernant de futurs travaux de développement de la méthode :

- Une amélioration possible des temps d'exécution des approches de MultiCons consisterait à introduire *un seuil minimal d'agrément* dans la phase d'extraction des IFF, ceci afin de ne pas générer de motifs dont la taille de l'itemset est inférieure à ce seuil. Par exemple, pour un seuil d'agrément de

40%, et si nous avons 10 clusterings de base dans l'ensemble, l'extraction des IFF ne générerait pas de motif correspondant à un agrément entre moins de 4 clustering de base. Ce seuil peut être augmenté afin d'optimiser les temps de génération des MFF, particulièrement pour des ensembles de données massifs et/ou de très nombreux clusterings de base. Toutefois, l'ensemble doit être de qualité suffisante pour qu'il soit possible d'atteindre un consensus pertinent par agrément d'un grand nombre de clusterings de base.

- Des expérimentations spécifiques à divers domaines d'application peuvent mettre en évidence, avec l'aide des experts du domaine, de l'utilité des approches de MultiCons en termes de recherche d'information. En effet, les mesures de qualité des consensus candidats, et pas seulement de celui recommandé, et du ConsTree peuvent fournir des informations importantes relativement au domaine. Cela peut également permettre d'identifier des évolutions futures de la méthode MultiCons. De plus, bien que les valeurs par défaut de MT ont permis de générer la meilleure solution, ou une solution très proche, durant les expérimentations, il est possible de modifier la valeur de MT en tenant compte des spécificités du domaine d'application. L'idée consiste à adapter la valeur, valeur basse ou haute de MT , afin d'obtenir des résultats plus adaptés en fonction des propriétés de l'espace des données. Dans cette situation, et dans une certaine mesure, il est possible d'utiliser les approches 1 ou 5, plus simples, pour obtenir des résultats pertinents.
- Une perspective intéressante de développement de MultiCons concerne l'étude de l'utilisation d'autres algorithmes de bi-clustering² à la place de l'extraction des MFF, afin de générer des motifs de clustering à partir de la matrice d'appartenance binaire. L'extraction de MFF étant une approche spécifique de bi-clustering, basées sur la fermeture de Galois, il semble intéressant de la comparer avec les autres techniques de bi-clustering en termes de performances pour obtenir une solution de consensus, que ce soit pour une tâche de clustering ou une tâche de classification.

²Le bi-clustering est défini par Mirkin [Mirkin 1998] comme "le regroupement simultané des lignes et colonnes d'une matrice de données".

