Pattern-based Clustering using Unsupervised Decision Trees

by

MSc. Andres Eduardo Gutierrez Rodríguez

Dissertation submitted for the degree of

DOCTOR IN COMPUTER SCIENCE

at the

National Institute for Astrophysics, Optics and Electronics
November 2015,
Tonantzintla, Puebla, México

Advisors:

PhD. José Francisco Martínez Trinidad
PhD. Milton García Borroto

Acknowledgments

I would like to express my gratitude to my advisors PhD. José Francisco Martínez Trinidad, PhD. Milton García Borroto and PhD. Jesús Ariel Carrasco Ochoa. Their support, guidance and patience have been invaluable to me and to the development of this thesis.

I wish to thanks the members of my revision committee for their excellent recommendations during the preparation of this thesis. Thanks to PhD. Leopoldo Altamirano Robles, PhD. Gustavo Rodríguez Gómez, PhD. René Cumplido Parra, PhD. José Enrique Muñoz de Cote, and PhD. Guozhu Dong.

I express my love and gratitude to my wife Maire and my son Andre. They show me their unconditional love every day. My special gratitude is to my mother, father and my families for their loving support.

I would like to special thanks to my brothers Milton (again), Migue and Tavo. We started this adventure some years ago.

I wish to thanks to all the people that kindly reviewed the document and the presentation of this thesis.

I want to acknowledge to my brothers Cosme, Alber, Chang, Raudel, Airel, and all the Cuban family in Puebla. To all my Mexican friends, I thank you deeply.

I also want to acknowledge all the help provided by the INAOE, Bioplantas, University of Ciego de Ávila, Provincial Crime Lab of Ciego de Ávila and CENATAV members.

Finally, I thank to the CONACyT (Scholarship Grant 436765) for their support during the doctoral studies.

Abstract

In clustering, providing an explanation of the results is an important task. Pattern-based clustering algorithms provide, in addition to the list of objects belonging to each cluster, an explanation of the results in terms of a set of patterns that describe the objects grouped in each cluster. It makes these algorithms very attractive from the practical point of view; however, pattern-based clustering algorithms commonly have a high computational cost in the clustering stage. Moreover, the most recent algorithms proposed within this approach, extract patterns from numerical datasets by applying an a priori discretization process, which may cause information loss. In this thesis, we propose new algorithms for extracting only a subset of patterns useful for clustering, from a collection of diverse unsupervised decision trees induced from a dataset. Additionally, we propose a new clustering algorithm based on these patterns. Experimental results show that our pattern-based clustering algorithm obtains better clustering results, and extracts significantly less patterns in a significantly less time, than state-of-the-art pattern-based clustering algorithms. Moreover, the proposed pattern-based clustering algorithm achieves clustering results similar, in quality, to those obtained by traditional clustering algorithms.

Resumen

En problemas de agrupamiento, proporcionar una explicación de los resultados es una tarea importante. Los algoritmos de agrupamiento basado en patrones proveen, además de la lista de objetos que pertenecen a cada grupo, una explicación de los resultados en términos de un conjunto de patrones que describen a los objetos de cada grupo. Esto hace que estos algoritmos sean muy atractivos desde un punto de vista práctico; sin embargo, los agrupadores basados en patrones tienen un alto costo computacional en la etapa de agrupamiento. Por otra parte, los algoritmos más recientes propuestos en este enfoque extraen patrones de conjuntos de datos numéricos realizando una discretización a priori de las variables numéricas, la cual puede causar pérdida de información. En esta tesis se proponen nuevos algoritmos que, a partir de una colección de árboles de decisión no supervisados inducidos a partir de un conjunto de datos, extraen un subconjunto de patrones útiles para agrupar. Adicionalmente, se propone un nuevo algoritmo de agrupamiento basado en estos patrones. Los resultados experimentales muestran que nuestro algoritmo de agrupamiento basado en patrones obtiene mejores resultados, extrayendo significativamente menos patrones en un tiempo significativamente menor, que los algoritmos de agrupamiento basado en patrones reportados en el estado del arte. Por otra parte, el algoritmo de agrupamiento basado en patrones propuesto obtiene agrupamientos con calidad similar a los obtenidos por algoritmos de agrupamiento tradicionales.

Contents

Glossary and Acronyms

Categorical dataset	A dataset with objects described by categorical features.
CLUS	Clustering algorithm based on a single unsupervised decision tree, proposed in (Blockeel H., 1998).
COBWEB	Pattern-based clustering algorithm proposed in (Fisher, 1987).
$cov(P)$	Cover of a pattern P. The set of objects described by P in a dataset.
CPC	Pattern-based clustering algorithm proposed in (Fore and Dong, 2012).
Dataset	Set of n unlabeled objects described through m features.
EM	Expectation Maximization clustering algorithm, proposed in (Moon, 1996).
Equivalence class	Set of patterns which cover exactly the same set of objects.
F-measure	External clustering validity index proposed in (Makhoul et al., 1999).
Feature	Variable used for describing an object in a dataset.
Feature-value item	Combination of a feature and a single value.
FP-growth	Pattern mining algorithm proposed in (Han et al., 2004).
K-means	Traditional clustering algorithm proposed in (MacQueen, 1967).
Mixed dataset	A dataset with objects described by categorical and numerical features.
Numerical dataset	A dataset with objects described by numerical features.
Object	Element of a dataset described through m features values.
Pattern	Conjunction of items that describes an object subset.
PbCA	Proposed pattern-based clustering algorithm.
PMCC	Proposed pattern mining algorithm for categorical datasets.
PMCM	Proposed pattern mining algorithm for mixed datasets.
PMCN	Proposed pattern mining algorithm for numerical datasets.

RandomForest	Strategy to generate a collection of decision trees, proposed in (Breiman, 2001).
$sup(P)$	Support of a pattern P. Fraction of objects in a dataset covered by the pattern P.

Chapter 1

Introduction

Cluster analysis is one of the most important techniques in pattern recognition. It has been widely studied and applied in many areas like computer vision, information retrieval, marketing, and bioinformatics (Jain, 2010). In this thesis, only non-overlapping clustering is considered, which consists in partitioning a set of unlabeled objects into disjoints clusters, according to a certain criterion (Duda et al., 2012). This type of clustering is the most used in cluster analysis.

Several works for solving non-overlapping clustering have been proposed, including: K-means (MacQueen, 1967), EM (Moon, 1996), DBSCAN (Ester et al., 1996), Pairwise Clustering (Hofmann and Buhmann, 1997), Kernel K-means (Schölkopf et al., 1998), DENCLUE (Hinneburg and Keim, 1998), X-means (Pelleg et al., 2000), Minimum-entropy Clustering (Roberts et al., 2001), Normalized Cut (Shi and Malik, 2000), K-medoids (Kaufman and Rousseeuw, 2005), SVM Clustering (Winters-Hilt and Merat, 2007), and ABC Clustering (Karaboga and Ozturk, 2011).

As it has been argued in the literature (Färber et al., 2010), several clustering results can be obtained from the same dataset and many of them can be considered as correct. This is because there is a great diversity of clustering algorithms and the quality of their results is measured in many different ways (Aggarwal and Reddy, 2013). Therefore, validating clustering results is a subjective process. In addition, the lack of comprehensibility of the results is a common issue of traditional clustering algorithms like K-means and EM, because these algorithms only return a list of objects belonging to each cluster. All these issues make difficult the process of validating and understanding clustering results.

In different application areas such as agriculture, bioinformatics, text processing and web mining, users need some explanation about clustering results more than just a list of objects for each cluster (Michalski et al., 2006; Baridam and Owolabi, 2010; Hotho et al., 2003; Tiddi et al., 2014). For example, in text processing, common text clustering techniques do not provide an explanation in terms of the features (terms) of the resulting clusters, which is an important information to understand the topics associated to each cluster (Hotho et al., 2003). Another example, in web mining, applying traditional clustering algorithms becomes a laborious and time-consuming process, involving expertise in possibly different domains for getting an explanation of the results. However, pattern-based clustering (Michalski and Stepp, 1983) aims to provide an explanation of the clustering results in terms of the features used to describe the data.

Several works have been reported within the pattern-based clustering approach (Michalski and Stepp, 1983; Fisher, 1987; Ralambondrainy, 1995; Mishra et al., 2004; Wong and Li, 2008; Fore and Dong, 2012). Algorithms under this approach, in addition to the list of objects belonging to each cluster, return a set of patterns[1] that describe each cluster. These patterns provide information, in terms of feature values, about the characteristics of the objects in each cluster. This description is very important because it allows an easier explanation of the results.

1.1 Description of the problem

Some pattern-based clustering algorithms (Michalski and Stepp, 1983; Fisher, 1987) extract few patterns for clustering, making insufficient the explanation of their results. On the other hand, other pattern-based clustering algorithms (Mishra et al., 2004; Wong and Li, 2008; Fore and Dong, 2012) mines all patterns in a categorical dataset, but computing all patterns makes these algorithms non suitable for medium-large datasets, due to the huge amount of patterns that can be computed. Moreover, using too many patterns produces a high computational cost at the clustering stage. As we will show in our experimental results, not all patterns are useful for clustering, since many of them are redundant, which may negatively affect the clustering quality. In addition, several pattern-based clustering algorithms have been proposed for clustering exclusively categorical datasets. For applying these algorithms

[1] A pattern is a conjunction of feature-value items (Michalski and Stepp, 1983).

over numerical datasets, all numerical features must be discretized a priori. Data discretization is the process of transforming numerical features into a finite set of intervals, causing loss of information (Garcia et al., 2013).

In pattern-based clustering, since extracting all patterns from a dataset is too expensive and may produce too many patterns, it is desirable that mining algorithms return a subset of comprehensible patterns for describing the clusters. One of the problems to solve in this thesis, is how to extract this subset of suitable patterns for clustering, without extracting redundant patterns that negatively affect the quality of the result. In general, obtaining clusters and patterns from data has a high computational cost. Thus, a good algorithm for mining patterns for clustering should extract a subset of patterns in a short time, instead of extracting all patterns, which would allow reducing the computational cost of the clustering phase. Another issue that should be addressed is clustering numerical datasets without applying an a priori discretization on numerical features, aiming to avoid information loss that may affect the clustering quality.

Additionally, a pattern filtering algorithm is necessary to remove duplicate patterns, and, at the same time, simplifying each pattern by joining redundant items of the same feature. Thus, a shorter explanation of the clustering results can be obtained. Finally, a clustering algorithm based on the extracted patterns is required to build clusters of objects. We consider that the idea of first defining a relationship between patterns to cluster the set of patterns, and then clustering the objects of a dataset based on the clusters of patterns, is promissory but the algorithm proposed by Fore and Dong (2012) is too expensive for large datasets. Therefore, another problem to solve is defining a relationship between patterns which allows clustering objects, based on clusters of patterns, more efficiently and effectively than the state-of-the-art pattern-based clustering algorithms.

1.2 Objectives

1.2.1 General objective

To develop a pattern-based clustering algorithm by extracting only a small subset of patterns suitable for clustering, instead of mining all patterns; which allows working with mixed data without an a priori discretization of numerical features. The proposed algorithm should be more efficient and effective

than the state-of-the-art pattern-based clustering algorithms.

1.2.2 Specific objectives

- To develop an algorithm to extract subsets of frequent patterns from mixed and incomplete unsupervised datasets, without an a priori discretization of numerical features.
- To select a filtering algorithm to select only those patterns that are suitable for clustering.
- To develop a clustering algorithm more efficient and effective than the state-of-the-art pattern-based clustering algorithms, based on the patterns selected by the proposed filtering algorithm.

1.3 Main contribution

The main contribution of this thesis is a new pattern-based clustering algorithm for mixed datasets. Our proposal mines only a subset of useful patterns for clustering (instead of mining all patterns) and without applying an a priori discretization on numerical features, from a collection of unsupervised decision trees. The proposed algorithm is more efficient and effective than state-of-the-art pattern-based clustering algorithms. Moreover, the proposed algorithm is competitive, in terms of F-measure, with traditional clustering algorithms.

1.4 Thesis organization

The remaining chapters of this thesis are organized as follows:

Chapter 2 presents the background necessary to understand the remaining chapters.

Chapter 3 contains a review of the related work about pattern-based clustering. This chapter includes some works that are considered to be related to pattern-based clustering, but, in fact, follow a different approach.

Chapter 4 presents the three new pattern mining algorithms developed in this research. For each algorithm, the split evaluation criteria and the strategies to obtain diverse patterns are defined. Additionally, this chapter

introduces a filtering algorithm to select suitable patterns for clustering. Finally, based on the mined patterns, a new pattern-based clustering algorithm is proposed.

Chapter 5 contains the experimental results. It includes a comparison among the proposed algorithms and state-of-the-art pattern-based clustering algorithms, as well as a comparison against traditional clustering algorithms.

Chapter 6 enunciates the conclusions and contributions of this thesis and provides some future research directions.

Chapter 2

Background

In this chapter, the necessary background of the main topics required to understand the contents of this research is presented. First, we provide an introduction to clustering, and a brief description of two well-known traditional clustering algorithms. The problem of clustering validation is also explained. Then, some basic definitions related to patterns are presented. Finally, since we propose new algorithms for inducing unsupervised decision trees with the objective of mining useful patterns for clustering, some concepts related to the induction of unsupervised decision trees are introduced.

2.1 Traditional clustering algorithms

Pattern recognition is about assigning labels to objects, which is known as classification (Kuncheva, 2004). If there is no previous knowledge about the labels of any object in the dataset, the process of grouping objects in clusters according to some predefined criterion is named *clustering*. In this thesis, we will refer only to *non-overlapping* clustering, which consists in partitioning a set of unlabeled objects into disjoints clusters.

In general, clustering algorithms differ in the specific clustering criterion or how this criterion is measured. One of the most used clustering criterion assumes that objects of the same cluster should be more similar than objects from different clusters (Karaboga and Ozturk, 2011). The similarities among objects are determined using a comparison function (Bandyopadhyay and Saha, 2013), which is usually a distance function.

A distance is a function that, for every pair x, y in the function domain,

fulfills (Webb, 2003):

- $d(x, y) \geq 0$, for every x and y,
- $d(x, y) = 0 \leftrightarrow x = y$, for every x and y,
- $d(x, y) = d(y, x)$, for every x and y, and
- $d(x, z) + d(z, y) \geq d(x, y)$, for every x, y and z.

2.1.1 K-means

K-means (MacQueen, 1967) is one of the most commonly used clustering algorithms. K-means finds a partition of objects such that the squared distance between the center of a cluster (the mean) and the objects in the cluster is minimized. This algorithm is defined for numerical data, and it is typically used together with the Euclidean distance. Let μ_i be the center of the cluster $K_i \in K$ (K is the set of clusters), the goal of K-means is to minimize the function defined in Eq. (2.1):

$$E(K) = \sum_{i=1}^{k} \sum_{x \in K_i} d(x, \mu_i)^2. \tag{2.1}$$

The main steps of K-means are as follows:

1. Select an initial set of k cluster centers;
2. Generate a new partition by assigning each object to its closest cluster center;
3. Compute new cluster centers by averaging the feature values of the objects belonging to the same cluster;
4. Repeat steps 2 and 3 until few change in the centers occurs.

Different initializations of K-means can lead to different final clustering because K-means converges to local minima. One way to overcome the local minima is to run the K-means algorithm, for a given k, with multiple different initial cluster centers, and choosing the partition that minimizes Eq. (2.1).

2.1.2 EM

Another well-known clustering algorithm is EM (Moon, 1996). This clustering algorithm defines a cluster as a high density region of objects. EM models a density function by a mixture of a Gaussian probability distributions (Scott, 2009).

The EM clustering algorithm is defined for both numerical and categorical data. It uses a mixture model to represent k clusters and performs iterative refinements to fit the model to the dataset. The mixture model probability density function is defined as in Eq. (2.2):

$$Pr(x) = \sum_{\ell=1}^{k} W_\ell Pr(x|\ell), \tag{2.2}$$

where the coefficients W_ℓ (mixture weights) represent the fraction of the dataset D in the corresponding cluster, and x is an object of the dataset.

EM begins with an initial estimate of the parameter vector (W_ℓ, μ_ℓ, Σ_ℓ) for the density function; where μ represents the cluster centers, and Σ represents the covariance matrix. Then, EM iteratively re-scores objects based on the mixture density and refines the parameters based on the re-scored objects.

The main steps of EM are as follows:

1. Set $j = 0$;
2. Choose an initial mixture model parameters: W_ℓ^j, μ_ℓ^j and Σ_ℓ^j, for $\ell = 1, \ldots, k$;
3. Compute the *membership probability* of all objects in each cluster by: $Pr(\ell|x) = W_\ell^j Pr^j(x|\ell)/Pr^j(x)$;
4. Update the mixture model parameters:
   - $W_\ell^{j+1} = \frac{1}{n} \sum_{x \in D} Pr(\ell|x)$,
   - $\mu_\ell^{j+1} = \sum_{x \in D} x \cdot Pr(\ell|x) / \sum_{x \in D} Pr(\ell|x)$, and
   - $\Sigma_\ell^{j+1} = \sum_{x \in D} Pr(\ell|x) \left(x - \mu_\ell^{j+1}\right) \left(x - \mu_\ell^{j+1}\right)^T / \sum_{x \in D} Pr(\ell|x)$;
5. If $|E^j - E^{j+1}| \leq \epsilon$, stop. Else set $j = j + 1$ and repeat steps 3, 4 and 5. E^j is the log-likelihood of the mixture model at iteration j: $E^j = \sum_{x \in D} \log \left(\sum_{\ell=1}^{k} W_\ell^j Pr^j(x|\ell) \right)$.

2.2 Cluster validation

Validating clustering results is a subjective process, since different results can be considered as correct independently of the criterion used for validating. There are two kinds of clustering quality indices: internal and external (Aggarwal and Reddy, 2013). The selection of the appropriate index is still a challenge (Färber et al., 2010).

Internal indices (Liu and Dong, 2012; Zhao and Fränti, 2014) evaluate the results of clustering algorithms using only quantities and features inherent to the dataset. These indices evaluate whether a clustering result meets certain criterion or not. In our experiments, we avoid using internal indices because internal measures usually reflect the objective function of specific clustering algorithms benefiting these types of algorithms at the validation stage (Färber et al., 2010).

On the other hand, external indices (Rosenberg and Hirschberg, 2007; Amigó et al., 2009) evaluate the results of clustering algorithms based on a pre-specified structure, i.e. the correlation between two partitions. The usual approach using external indices for cluster validation, is selecting labeled datasets based on the assumption that the correct clustering of the dataset is reflected by the class labels (Färber et al., 2010). The external quality index F-measure (Makhoul et al., 1999) is one of the most used indices for clustering validation (Breaban and Luchian, 2011; Fore and Dong, 2012), and CPC uses F-measure to evaluate their clustering results. For this reason, in this thesis we use F-measure to evaluate all algorithms.

F-measure evaluates the dependence between the dataset classes C and a clustering result K; if they are independent, F-measure takes values close to 0. The maximum value of F-measure is 1, which is achieved for identical partitions. F-measure is defined as in Eq. (2.3):

$$F\text{-}measure(C, K) = \sum_{C_i \in C} \frac{|C_i|}{n} \max_{K_j \in K} \{F(C_i, K_j)\}, \tag{2.3}$$

where

$$F(C_i, K_j) = \frac{2 \cdot Recall(C_i, K_j) \cdot Precision(C_i, K_j)}{Recall(C_i, K_j) + Precision(C_i, K_j)}, \tag{2.4}$$

n is the number of objects in the dataset, C is the set of classes, K is the set of clusters built by the clustering algorithm, $Recall(C_i, K_j) = n_{ij}/|C_i|$, and $Precision(C_i, K_j) = n_{ij}/|K_j|$, where n_{ij} is the number of objects of the class $C_i \in C$ belonging to the cluster $K_j \in K$.

2.3 Patterns

In different application areas, an explanation about clustering results is needed. The pattern-based clustering approach provides an explanation of the clustering results in terms of a set of patterns that describe each cluster. In pattern-based clustering, the definition of *pattern*, and how the patterns are extracted from a dataset, are two key issues. In this thesis, a pattern is defined as a conjunction of relational items $X_i \# R_i$ that describes an object subset, where R_i is a value in the domain of the feature X_i, and $\#$ is one of the relational operators "=", "$\leq$" and ">" (Michalski and Stepp, 1983).

Traditional pattern mining algorithms, like FP-growth (Han et al., 2004), only use "=" as relational operator because they are defined for categorical data. For example, the Zoo dataset (Frank and Asuncion, 2010) has 101 objects (animals); an example of a pattern for this dataset could be $P = ([Legs = 2] \wedge [Eggs = true])$. In this example, there are twenty objects into the Zoo dataset that fulfill this pattern; therefore, this pattern *covers* twenty objects in the dataset. Alternatively, these objects are *covered* by the pattern. Given a pattern P, $|P|$ denotes the length of the pattern, and it is defined as the number of items in P, in our example $|P| = 2$; $cov(P)$ denotes the set of objects covered by P; and $|cov(P)|$ denotes the number of these objects, twenty objects in our example. The *support* of a pattern is expressed as a fraction of objects in the dataset that are covered by the pattern; for the pattern P in the example, $sup(P) = |cov(P)|/|dataset| = \frac{20}{101} \approx 0.2$. A pattern P is a *superset* of another pattern Q if all the items in Q appear in P and $|P| > |Q|$. For example, the pattern $P = ([Legs = 2] \wedge [Eggs = true])$ is a superset of the pattern $Q = ([Legs = 2])$.

2.4 Unsupervised decision tree induction

An *unsupervised decision tree* (Basak and Krishnapuram, 2005) represents a hierarchical clustering. The internal nodes of these trees, including the root node, are *decision nodes* which contain a feature-value item assigned to each child node, while leaf nodes only represent clusters of objects.

The main purpose of an unsupervised decision tree induction algorithm is to build a tree structure where each node represents a good cluster according to a certain criterion. Algorithms like C4.5 (Quinlan, 1993), which build supervised decision trees, can be modified to induce unsupervised trees,

but split evaluations must be modified according to an unsupervised quality criterion, i.e., a criterion that does not use class information.

For example, the splits in a tree can be evaluated by using an unsupervised criterion for categorical features, which is based on entropy, defined in Eq. (2.5) (Basak and Krishnapuram, 2005),

$$H = -\sum_{i} n_i \log n_i, \tag{2.5}$$

where n_i is the frequency of each value in a categorical feature. To induce an unsupervised decision tree, at every decision node, the data is split based on a single feature. The feature that reaches the maximum value in Eq. (2.5) is selected for splitting. Then, the objects in the node are partitioned into subsets, based on the selected feature, and each subset is allocated into a child node. The induction process ends when, for every feature, the value of (2.5) is less than an user predefined threshold.

Chapter 3

Related work

This chapter presents a review of the most relevant works on pattern-based clustering. The content of the chapter is split in five sections. In the first section, relevant pattern-based clustering algorithms in the state-of-the-art are reviewed. In sections 3.2, 3.3 and 3.4, those pattern-based clustering algorithms that are used in our experiments are explained in detail. Finally, the last section presents a brief discussion of the related work.

3.1 Related pattern-based clustering algorithms

Pattern-based clustering (or *conceptual clustering*) has been an active area of research since 1980. Although several works have been reported (Michalski and Stepp, 1983; Fisher, 1987; Hanson and Bauer, 1989; Bisson, 1992; Mineau and Godin, 1995; Ralambondrainy, 1995; Carpineto and Romano, 1996; Perkowitz and Etzioni, 1999; Robardet and Feschet, 2001; Jonyer et al., 2002; Ozdal and Aykanat, 2004; Mishra et al., 2004; Jänichen and Perner, 2005; Yang and Padmanabhan, 2005; Romero-Zaliz et al., 2006; Lisi, 2006; Ayaquica-Martínez et al., 2007; Xia and Xi, 2007; Wong and Li, 2008; Funes et al., 2009; Baridam and Owolabi, 2010; Perner and Attig, 2010; Schmidt et al., 2012; Fore and Dong, 2012; Liang and Forbus, 2014), it is important to separate those works that, at first sight, could be considered as related to pattern-based clustering, but indeed they follow a different approach.

Some works use the term pattern-based clustering for *subspace clustering* (Wang et al., 2002; Pei et al., 2003; Agrawal et al., 2005; Kriegel et al., 2009;

Günnemann et al., 2011; Boongoen et al., 2011; Voggenreiter et al., 2012; Adler et al., 2013; Patel et al., 2013; Soltanolkotabi et al., 2014); however, subspace clustering is about clustering objects using subsets of features. In (Kriegel et al., 2009), a distinction has been made between different subproblems such as *projected clustering* and *biclustering*, however both are out of the scope of this doctoral research.

Text clustering, is another field where patterns are used for building clusters (Beil et al., 2002; Fung et al., 2003; Yu et al., 2004; Malik and Kender, 2006; Kryszkiewicz and Skonieczny, 2006; Malik and Kender, 2008; Li et al., 2008; Zhang et al., 2010; Morik et al., 2012; Zheng et al., 2014; Peng and Liu, 2015). Nevertheless, in these works, a pattern is a sequence of words that appears frequently in a text. Following the idea that frequent sequences are more useful for representing documents than only using frequent words, in these works the patterns are commonly used for representing the documents following the bag-of-words model. Once the documents are represented under this model, a comparison function is used to build the clusters applying a traditional clustering algorithm. Therefore, these works are also out of the scope of this doctoral research.

Hereinafter, some relevant pattern-based clustering algorithms are described. CLUSTER/2 was proposed by Michalski and Stepp (1983). This algorithm groups a dataset in k predefined clusters in such a way the descriptions (patterns) of the clusters are simple and describe well the dataset. In a first step, CLUSTER/2 randomly selects k objects as seeds. Then, for each seed, a pattern is built by using the feature-value items in the seed such that these items are different from those items in the patterns built from the other seeds. Based on these patterns, an optimized clustering is built by selecting those patterns that contain intersecting feature-value items and modifying these patterns to make them disjoint. Each pattern, and the objects covered by it, represents a cluster. A quality criterion, based on the sparseness of the clustering and the inter-cluster differences, is finally evaluated. If the obtained partition is not better than the last partition, the algorithm stops; otherwise, new object seeds are selected and a new iteration of the algorithm begins.

Fisher (1987) developed the COBWEB clustering algorithm. This algorithm incrementally builds a pattern tree where each node is a probabilistic (instead of support) pattern that represents an object class. The tree is incrementally built by traversing the pattern tree trough the nodes that cover the new object until reaching a leaf node where the new object is stored. All

the probabilistic patterns associated to nodes, in the path followed to reach a leaf, are updated with the information provided by the description of the new object. During the traversal, if at same level there is no node covering the new object, a new leaf node is built at the same level and the object is stored into it. The traversal for building the pattern tree is guided by a measure that evaluates the clustering at a certain level of the tree, which allows to decide the path to follow during the construction of the pattern tree. COBWEB is an incremental algorithm for hierarchical clustering and it cannot partition the dataset into a predefined number of clusters.

Ralambondrainy (1995) proposed a pattern-based K-means algorithm called CKM. This algorithm groups a dataset using the traditional K-means clustering algorithm. Then, a characterization phase builds patterns for each cluster. In this phase, a generalization for each feature must be defined. For categorical features, the generalization must be previously provided by the user. In each cluster, numerical features are discretized into three qualitative values, for automatically building a generalization. These qualitative values define the *lower*, *typical* and *upper* intervals based on the mean and the variance of the feature values. Then, a set of patterns is computed for each cluster by finding conjuntions of features that cover at least β objects in a cluster, and do not cover more than α objects in other clusters. In CKM the patterns are extracted in a post clustering step, and they are not taken into account for building the clusters, which is out of the main idea of the pattern-based clustering approach. This work was extended by Ayaquica-Martínez et al. (2007) for mixed data using a general function, but still extracting the patterns in a post clustering step.

Mishra et al. (2004) introduced a graph formulation for pattern-based clustering with the objective of identifying a collection of patterns that describe the objects. The authors connect the pattern-based clustering problem with the maximum edge biclique problem (Peeters, 2003) by defining a bipartite graph $G = (U, W, E)$, where U is the set of objects, W is the set of all feature-value combinations, and E is the set of edges between $u \in U$ and $w \in W$ such that u has the w feature-value combination. A biclique is a subgraph $B = (U_B, W_B, E_B)$ where each vertex in U_B is connected to every vertex of W_B. A biclique naturally corresponds to a pattern-based cluster since each object $u \in U_B$ satisfies the conjunction of features in W_B. A maximum edge biclique corresponds to the best conjunctive cluster, where $|W_B|$ is precisely the length of the conjunction and $|U_B|$ is the number of objects that satisfy the conjunction. This algorithm follows an heuristic to discover

the k best conjunctive clusters looking for the k largest clusters that do not overlap too much. However, the clusters discovered by this approach may overlap and they also may not cover all the objects in U.

The Greedy Hierarchical Itemset-based Clustering (GHIC) algorithm was proposed by Yang and Padmanabhan (2005). This algorithm extracts patterns from a dataset with the Apriori (Agrawal et al., 1996) pattern mining algorithm. Then, a new dataset is generated, where the rows represent the objects and the columns represent the presence or abscence of the extracted patterns. The GHIC clustering algorithm creates a hierarchical structure that, at each step, split the dataset in two clusters by maximizing an objective function that sum the difference between clusters and the similarity inside each cluster. The difference between clusters follows the intuition that the support of any pattern to a cluster should be greater than the support to the other cluster. For each pattern, the difference between clusters is computed based on the difference of the support values of the two clusters and the sum of the support values in these clusters. The total difference between clusters is computed as the sum of the differences of all patterns. On the other hand, the similarity inside each cluster K_i is measured as the number of patterns in K_i. A balancing factor is included into the objective function to create clusters with approximately the same number of objects. A greedy heuristic is proposed to select, at each step, the partition of objects covered by a pattern and its complement, such that it maximizes the objective function. It is important to comment that this algorithm does not partition the dataset into a predefined number of clusters.

Wong and Li (2008) proposed a pattern-based clustering algorithm to simultaneously cluster patterns and objects. Unlike traditional pattern mining algorithms, which extract patterns based on feature-value frequencies, this algorithm extracts patterns from categorical data using correlation among objects. The idea is to evaluate if the occurrence of a feature-value item in an object is random or not, by evaluating the difference between the real number of occurrences and the expected number of occurrences. If a priori knowledge about the domain is not available, the authors propose to compute the expected number of occurrences for a feature-value item as the number of objects in the dataset divided by the number of different values in this feature. Each pattern is an itemset, represented as conjunctions of items with the form $[feature = value]$, in which the occurrence of each item is not random. In order to cluster the extracted patterns and their associated objects, this algorithm defines some distance measures between patterns, which

consider both the matched objects and the matched features. Then, a hierarchical K-means is applied by iteratively reducing the number of clusters (starting at n: number of objects, until k: number of desired clusters), and merging the nearest clusters at each level of the hierarchy.

Fore and Dong (2012) reported the CPC algorithm. In the first step, CPC mines all frequent patterns in a dataset using the FP-growth algorithm (Han et al., 2004). Then, the extracted patterns are grouped into equivalence classes. The algorithm has the particular idea of defining a relationship between patterns, which is used to cluster the set of patterns in k clusters. In order to cluster the patterns, CPC computes k initial pattern seeds through a greedy algorithm. The selected seeds are those patterns that are less similar among them. Thus, each pattern seed corresponds to a cluster, and each remaining pattern is associated to the cluster where it reaches the greatest similarity. After building clusters of patterns, the objects in the dataset are assigned to the clusters based on pattern matching.

Given that the pattern-based clustering algorithm proposed in this thesis uses unsupervised decision trees, those clustering algorithms based on unsupervised decision trees (Blockeel H., 1998; Liu et al., 2000; Basak and Krishnapuram, 2005; Basak, 2008; Dimitrovski et al., 2012) are also included as related work. Particularly, from this type of algorithms, the CLUS algorithm proposed by Blockeel H. (1998) is the closest to our work because it generates an unsupervised decision tree. In this tree, each leaf corresponds to a cluster, which is directly interpretable in the form of feature-value items. CLUS induces a binary tree by selecting, at each step, the split that maximizes the distances among the prototypes of the object sets belonging to the two new nodes. This algorithm does not partition the dataset into a predefined number of clusters.

Since in our experiments the algorithms COBWEB, CPC and CLUS are used, in the next sections these clustering algorithms are described in more detail.

3.2 COBWEB

The COBWEB pattern-based clustering algorithm was developed by Fisher (1987). COBWEB is an incremental algorithm for hierarchical clustering. This algorithm incrementally builds clusters using a classification tree where each node is an object class. These nodes are associated to a list of feature-

value items and their respective probabilities, computed in terms of the objects classified under the nodes. The classification of a new object is performed by descending the tree along the appropriate path of nodes, testing the different feature-value items.

COBWEB uses a heuristic evaluation measure, called *category utility*, to guide the search. This measure is a trade-off between intra-cluster similarity and inter-cluster dissimilarity for objects described in terms of nominal feature-value items. The intra-cluster similarity is computed using $Pr(A_i = V_{ij}|K_l)$, where $A_i = V_{ij}$ is a feature-value item and K_l is a cluster. The larger this probability, the greater the proportion of cluster members sharing the value V_{ij} and the more *predictable* this value is for cluster members. The inter-cluster similarity is a function of $Pr(K_l|A_i = V_{ij})$. The larger this probability, the fewer objects in other clusters that share this value and the more *predictive* this value is for the cluster. These probabilities can be combined to give an overall measure of partition quality, as it is defined in Eq. (3.1):

$$H = \frac{\sum_{l=1}^{n} Pr(K_l)[\sum_i \sum_j (Pr(A_i = V_{ij}|K_l)^2 - Pr(A_i = V_{ij}))^2]}{|K_l|}, \qquad (3.1)$$

where the probability $Pr(A_i = V_{ij})$ weights the importance of individual values, and the denominator allows comparing clusters of different sizes.

COBWEB represents patterns as a list of feature-values and their associated probabilities, i.e., a *probabilistic* pattern. In this algorithm, each node in the tree is labeled by a probabilistic pattern which summarizes the objects classified under the node.

COBWEB includes operators for *merging* and *splitting* nodes. When an object is incorporated into the classification tree, only the two best leaf nodes (clusters) of the same level are considered for merging. Merging takes these nodes and combines them if the resultant partition has better quality, a new node is created and the probabilities of the feature-value pairs in the new node are updated. The two original nodes are made children of the newly created node.

Splitting is considered only for the children of the best node among the existing leaf nodes. As in merging, splitting should increase the partition quality. A leaf node (of a partition of n nodes) may be deleted and its children promoted, resulting in a partition of $n + m - 1$ nodes, where m is the number of children of the deleted node.

In COBWEB, the induction of the tree starts with a root node and for each object in the dataset the best of the four operators of the algorithm is

applied (incorporating the object to an existing node, creating a new cluster, merging two nodes, or splitting one node). At the end, the resulting tree corresponds to a hierarchical structure of patterns and objects, where the leaf nodes correspond to the desired partition of the dataset. It is important to remark that in COBWEB the resultant number of clusters cannot be set by the user.

In our experiments, the implementation of COBWEB distributed in Weka (Hall et al., 2009) is used. This implementation includes aspects of CLASSIT (Gennari et al., 1989), which is an extension of COBWEB to handle numerical data.

3.3 CPC

Fore and Dong (2012) reported the CPC algorithm. As a first step, CPC mines all patterns in a dataset using the well-known FP-growth algorithm (Han et al., 2004). Then, to reduce the number of patterns, CPC applies a filtering process based on equivalence classes.

Let W be the set of all the patterns computed by FP-growth, each pattern $P \in W$, is associated with an equivalence class (EC) defined as $EC(P) = \{Q \in W | cov(Q) = cov(P)\}$. A *minimal generator* (MG) pattern in an EC is a pattern that is not a *superset*[1] of any other pattern in the same EC (there could be more than one MG pattern in an EC). A *closed pattern* (CP) is the longest pattern in an EC. Given an EC, $mgLen(EC)$ denotes the average length of the MG patterns in EC, and P_{max} denotes the length of the closed pattern of EC. Given a pattern P from an equivalence class EC, the *length ratio* of P is defined as $|P_{max}|/|P|$. In the rest of the document we treat equivalence classes as patterns.

For efficiency, the CPC algorithm works with equivalence classes, rather than patterns. Given a pattern set PS, $cov(PS)$ denotes $\bigcup_{P \in PS} cov(P)$. We say that the pattern P *overlaps* a set of objects OS if $cov(P) \cap OS \neq \emptyset$. A pattern $X \in W$ different from P_1 and P_2 is a *mutual pattern* of P_1 and P_2 if $cov(X)$ intersects both $cov(P_1)$ and $cov(P_2)$.

In order to determine if two patterns should belong to the same cluster, CPC defines a function to evaluate the relationship between patterns. The key idea is that two patterns must be in the same cluster if they have many

[1] A pattern P_1 is a *superset* of another pattern P_2 if all the items of P_2 appear in P_1.

mutual patterns, even though they support different objects. This function is named *Mutual Pattern Quality*, and it is defined in Eq. (3.2).

$$MPQ(P_1, P_2) = \frac{\sum_{X \in W} \left(\frac{|cov(P_1) \cap cov(X)| \cdot |cov(P_2) \cap cov(X)|}{cov(X)} \cdot \left(\frac{|X_{max}|}{|X|} \right)^2 \right)}{PQ(cov(P_1)) \cdot PQ(cov(P_2))} \quad (3.2)$$

Where PQ is defined in Eq. (3.3).

$$PQ(OS) = \sum_{P} \left(|OS \cap cov(P)| \cdot \left(\frac{|P_{max}|}{|P|} \right)^2 \right) \quad (3.3)$$

Once the patterns have been mined and filtered, CPC randomly selects m different sets having each one k pattern seeds. Each pattern pair P and Q, in each one of the sets, must hold the overlap constraint defined as in Eq. (3.4).

$$|cov(P) \cap cov(Q)| \leq threshold/(k-1) \cdot min(|cov(P)|, |cov(Q)|) \quad (3.4)$$

Eq. (3.4) means that the intersection between the sets of objects supported by the patterns P and Q must not exceed a previously determined percent of the minimum support of those patterns. The authors of CPC suggest to use $threshold = 0.05$. From the m pattern sets, the one with lowest maximum MPQ value between any pair of patterns in the set, is chosen as the initial seed. To refine the initial seed set, the pair of seed patterns having the maximum MPQ value is modified by changing one pattern of the pair by another pattern $R \in W$ such that the MPQ value decreases. This refinement is repeated until no improvement can be found.

For each pattern seed S, which represents the cluster $K_i, 1 \leq i \leq k$, all patterns Q having high $MPQ(S, Q)$ values and fulfilling the overlap constraint are added to the cluster K_i. In practice, it is important to highlight that most of the patterns do not meet the overlap constraint, therefore it is necessary to add the remaining patterns by object overlapping through the *Pattern-Cluster Membership* measure (PMCM), defined in Eq. (3.5). The pattern P will be assigned to the pattern cluster K_i where PMCM takes the maximum value. This step creates a pattern set for each cluster K_i.

$$PMCM(P, K_i) = \frac{|cov(P) \cap cov(K_i)| \cdot \left(\frac{|P_{max}|}{|P|}\right)^2}{PQ(cov(K_i))} \tag{3.5}$$

Once pattern clusters have been built, the objects in the dataset are grouped into the clusters using the vote (defined in Eq. (3.6)) that each pattern P gives to those objects that P supports (matches).

$$vote(P) = \frac{PMCM(P, K_{1st}) - PMCM(P, K_{2nd})}{\sum_{i=1}^{k} PMCM(P, K_i)} \cdot \left(\frac{|P_{max}|}{|P|}\right)^2, \tag{3.6}$$

where K_{1st} and K_{2nd} respectively denote the two clusters associated with the highest and second-highest PMCM values for the pattern P.

In this way, each pattern cluster K_i gives a vote to an object O by adding the votes of those patterns in K_i that support O. Thus, each object is assigned to the cluster where it reaches its maximum *Object Cluster Membership* value (OCM), defined in Eq. (3.7).

$$OCM(O, K_i) = \sum_{P \in K_i, O \in cov(P)} vote(P) \tag{3.7}$$

At the end, CPC obtains a set of clusters, each one associated to a set of patterns that describes the objects in the cluster.

3.3.1 Complexity analysis of CPC

In this subsection, the time and space complexity of CPC is analyzed. CPC has two main steps, first it extracts patterns using FP-growth, and then it build the clusters based on the extracted patterns. Therefore, the complexity of CPC is the sum of the complexity of FP-growth and the complexity of the clustering process.

In (Kosters et al., 2003), a complete analysis of the time and space complexity of FP-growth is provided. Eq. (3.8), provided in (Kosters et al., 2003), shows the number of operations that FP-growth does for extracting the patterns.

$$\sum_{P \in AP} \sum_{|P|}^{m} |cov(P)| \tag{3.8}$$

In Eq. (3.8), AP is the set of all patterns in a dataset, $|P|$ denotes the number of features involved in a pattern P, $|cov(P)|$ is the number of objects covered by P, and m is the number of features in the dataset. Since the value of Eq. (3.8) is related to the number of patterns in the dataset, the time complexity of FP-growth is, in the worse case, related to the maximum number of patterns that can be extracted from a dataset. In a dataset with n objects and m features, considering the minimum support value as $1/n$ (any feature-value combination is frequent), for each object there could be $2^m - 1$ patterns. Then, the maximum number of patterns for a dataset is $n(2^m - 1)$. In addition, the time complexity of $\sum_{|P|}^{m} |cov(P)|$ is $O(mn)$, since the complexity of computing $|cov(P)|$ is $O(nm)$ because all objects must be verified in all the features involved in a pattern, which are at most m. Therefore, the time complexity of FP-growth is $O(m^2n^2 2^m)$.

The space complexity of FP-growth is related to the total number of nodes in the FP tree. This number of nodes is also determined through Eq. (3.8) (Kosters et al., 2003). Thus, the space complexity of FP-growth is $O(m^2n^2 2^m)$.

On the other hand, the complexity analysis of the clustering step of CPC is reported in (Fore and Dong, 2012). The time complexity of clustering with CPC is $O(p^2 n)$, while the space complexity of CPC is $O(p^2 + pn)$, being p the number of patterns. Thus, since the maximum number of patterns is $n(2^m - 1)$, these complexities are $O(n^3(2^m - 1)^2)$ and $O(n^2(2^m - 1)^2 + n^2(2^m - 1))$ respectively.

3.4 CLUS

The CLUS algorithm (Blockeel H., 1998) induces an unsupervised decision tree for partitioning a dataset in a hierarchy of clusters. CLUS defines a distance between different clusters of objects based on a given distance d between objects (any distance can be used). The algorithm assumes the existence of a prototype function p that computes the prototype $p(K_i)$ of a cluster K_i as an object in the same feature space of the dataset objects (for example, the prototype can be the mean). Then, the distance between two clusters K_i and K_j is defined in Eq. (3.9) as the distance between the prototypes of the clusters.

$$d(K_i, K_j) = d(p(K_i), p(K_j)) \tag{3.9}$$

During the induction of the tree, CLUS splits a node (cluster) K into two disjoint nodes (subclusters) K_1 and K_2, if the distance between the two new nodes, computed by using Eq. (3.9), is the maximum among all the candidate splits. A candidate split is any feature-value pair that appears in the node that is divided.

The induction process continues over child nodes until the stopping criterion defined in Eq. (3.10) reaches less values than a predefined threshold.

$$F = \frac{SS/(n-1)}{(SS_L + SS_R)/(n-2)} \tag{3.10}$$

In Eq. (3.10), SS is the sum of squared differences from the mean inside the parent node, SS_L and SS_R are the sums (SS) for the two children, and n is the total number of objects in the parent node.

This algorithm does not partition the dataset into k predefined number of clusters, since the final number of clusters corresponds to the number of leaf nodes in the induced tree.

3.5 Concluding remarks

In pattern-based clustering, it is desirable that algorithms return accurate results in a short time, with just a few patterns for describing the clusters. This is very important, from the user's point of view, in situations when an explanation of the results is needed. In general, obtaining clusters and patterns from data has a high computational cost (Michalski and Stepp, 1983; Wong and Li, 2008; Fore and Dong, 2012). To avoid this, some algorithms follow an approach for sequentially adjusting clusters and patterns (Michalski and Stepp, 1983; Fisher, 1987; Blockeel H., 1998). However, these algorithms select a single pattern per cluster, discarding several patterns that could be useful to obtain better clustering results. In contrast, other algorithms (Mishra et al., 2004; Yang and Padmanabhan, 2005; Wong and Li, 2008; Fore and Dong, 2012) use traditional pattern mining algorithms like FP-growth, but it produces a huge amount of patterns, which leads to a high cost at the clustering stage.

On the other hand, in most of the pattern-based clustering algorithms (Fisher, 1987; Mishra et al., 2004; Yang and Padmanabhan, 2005; Wong and Li, 2008; Fore and Dong, 2012) numerical features must be a priori discretized to extract patterns. This may cause information loss, reducing

the application areas of these algorithms. For this reason, in this thesis, we propose a pattern-based clustering algorithm that extracts a small subset of patterns useful for clustering, instead of mining all patterns, without applying an a priori discretization on numerical data. This aims to obtain better clustering results than those obtained by other state-of-the-art pattern-based clustering algorithms.

Chapter 4

Pattern-based clustering algorithms

This chapter presents four novel results. First, Section 4.1 introduces three new pattern mining algorithms that, instead of extracting all patterns, mine a subset of all patterns, useful for clustering categorical, numerical and mixed datasets, respectively. Second, Section 4.2, presents a new pattern-based clustering algorithm that uses the patterns extracted to build a high quality set of clusters. Third, Section 4.3 introduces the analysis of the computational complexity of the proposed algorithms. Finally, the last section presents our concluding remarks about the algorithms introduced in this chapter.

4.1 New pattern mining algorithms for clustering

Extracting patterns for clustering by using traditional pattern mining algorithms produces a huge amount of patterns, which considerably increases the cost of the clustering process. These algorithms require to apply data discretization over all the numerical features before mining the patterns. Motivated by these drawbacks, we propose three new pattern mining algorithms, aiming at overcoming these limitations.

4.1.1 Mining patterns for clustering categorical datasets

This subsection introduces a new pattern mining algorithm that mines a subset of all patterns useful for clustering exclusively categorical data (PMCC). This algorithm is based on building a collection of diverse unsupervised decision trees and extracting a subset of patterns from them, instead of extracting all patterns.

In order to induce an unsupervised decision tree from a categorical dataset, we define a new split evaluation criterion for building internal nodes in an unsupervised decision tree. Unlike traditional algorithms for building unsupervised decision trees, PMCC introduces a new heuristic to generate more candidate splits than those generated by conventional methods. It returns, as result, a collection of diverse unsupervised decision trees, which allows to get good patterns for clustering.

Inducing an unsupervised decision tree

For building internal nodes of an unsupervised decision tree, the candidate splits must contain appropriate feature-value items to build good patterns. In an unsupervised context, pattern cover is a good quality measure to select patterns (Fore and Dong, 2012), but using only pattern cover creates a bias toward features with fewer values.

For example, suppose that a feature has only two different values in a collection of 100 objects, covering 51 and 49 objects respectively. On the other hand, suppose there is another feature which has seven different values, six items cover 10 objects each one and the seventh item covers 40 objects. In the second feature, the cover of the seventh item is four times greater than the other ones, while the cover values in the first feature are almost the same. This example shows that a feature with few values favors higher cover values, whereas many feature values favors lower cover values. For this reason, the cover of an item, jointly with the number of different values in each feature, are used in order to select appropriate feature-value items as splits.

Following this idea, a split evaluation criterion Q_{cat} for a decision node N is defined in Eq. (4.1). This quality measure reduces the bias towards features with fewer values. It is important to mention that a node N is split, with the feature f_j, by building one child node for each different value of f_j that fulfills the support threshold constraint, imposed by the user.

$$Q_{cat}\left(f_j\right) = (\max_{v_i \in V_j} \{|cov(\text{“}f_j = v_i\text{”})|\})|V_j| \tag{4.1}$$

In Eq. (4.1), V_j is the set of values of feature f_j in the objects of the node N; $|cov(\text{“}f_j = v_i\text{”})|$ is the number of objects, in node N, with value v_i for feature f_j. Greater values for $Q_{cat}(f_j)$ correspond to feature-value items with better quality. Thus, the best candidate split will have at least one child node with many objects having the same value v_i in feature f_j, in comparison to the number of objects in other child nodes of the same split. In the example described in the previous page, the first feature has $Qcat(f_1) = 102$ while the second feature has $Qcat(f_2) = 280$.

Another example: Table 4.1 shows six animals from the *Zoo* dataset, where 3 animals are mammals and 3 are invertebrates. Only two features are selected for our example, *Aquatic* and *Legs*. For this example, the *Aquatic* feature has two values: *true* and *false*, while the *Legs* feature has three values: *0*, *2* and *4*.

Animal name	Aquatic	Legs	Class
sealion	true	2	mammal
elephant	false	4	mammal
leopard	false	4	mammal
crab	true	4	invertebrate
seawasp	true	0	invertebrate
worm	false	0	invertebrate

Table 4.1: Six animals from the *Zoo* dataset.

To illustrate the effect of Eq. (4.1), the results of spliting the data in Table 4.1, with both features *Aquatic* and *Legs*, are shown in Fig. 4.1. This figure shows unsupervised decision trees where the number of objects is represented with m for mammals and i for invertebrate.

For the *Aquatic* feature, both items (patterns) $[Aquatic = false]$ and $[Aquatic = true]$ have a cover of 3. For the *Legs* feature, $[Legs = 2]$ has a cover of 1, $[Legs = 0]$ has a cover of 2, and $[Legs = 4]$ has a cover of 3. Applying the quality measure (4.1), *Aquatic* has a quality of 6, while *Legs* has a quality of 9. Therefore, *Legs* is better than *Aquatic*, according to Eq. (4.1), for splitting the data in Table 4.1, as shown in Fig. 4.1.

For inducing an unsupervised decision tree, PMCC splits the data by selecting the best split at each step, using the proposed split quality measure

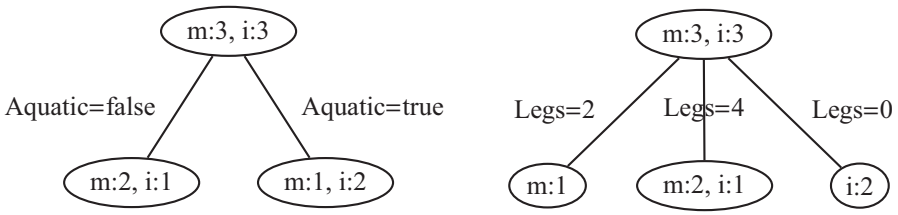

Figure 4.1: Candidate splits using the *Aquatic* feature (left) and the *Legs* feature (right).

(4.1), until meeting the stop criterion. The stop criterion is met when: 1) the number of objects in a new child node is less than $\mu \cdot |T|$ (μ is the minimum support threshold, $|T|$ is the number of objects in the entire dataset), it means that the node does not fulfill the support constraint; or 2) the number of new child nodes is less than two, because each parent node should have at least two child nodes.

The tree induced by this algorithm, using the data in Table 4.1 with $\mu = 0.15$, is shown in Fig. 4.2.

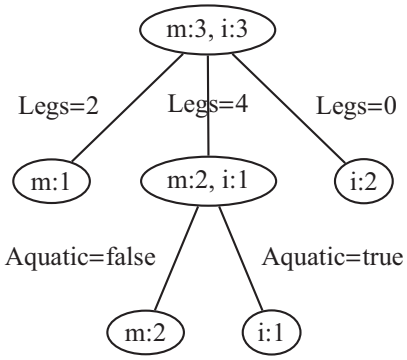

Figure 4.2: Unsupervised decision tree built by Alg. 1, using the data in Table 4.1 with $\mu = 0.15$.

Extracting patterns

Once an unsupervised decision tree has been built, any conjunction of feature-value items in any path from the root node to any other node in the tree is

considered as a pattern. It is important to notice that these patterns have support values greater than the minimum support threshold fixed by the user; otherwise, according to our induction algorithm, the nodes in the path would not have been created. Therefore, once an unsupervised decision tree has been built, we extract all possible patterns from this tree by a depth-first traversal (in order to maintain the feature order in the extracted patterns) by starting at the root of the tree. When a node is visited, the conjunction of items in the path from the root to this node generates a new pattern. Recursively, all the nodes of the tree are visited.

Using this algorithm, the patterns extracted from the tree in Fig. 4.2 are:

- $[Legs = 2]$
- $[Legs = 4]$
- $[Legs = 0]$
- $[Legs = 4 \wedge Aquatic = false]$
- $[Legs = 4 \wedge Aquatic = true]$

Generating several diverse unsupervised decision trees

Extracting the patterns from only one unsupervised decision tree produces few patterns as result. Therefore, we propose to build several different trees (the amount of trees is a parameter). We select a trade-off between the best tree (the one with the highest Q_{cat} value in all splits) and the generation of all possible trees, because the former is unique, and the latter is unfeasible in practice given its time complexity.

Once the user specifies the amount of trees to build, the generation of diverse trees is performed by randomly selecting a subset of features as RandomForest (Breiman, 2001) does. For each single tree, first a subset of $m = \log_2(|F|)$ (Breiman, 2001) features is randomly selected, where F is the set of features in the dataset. Then, it is selected as the best split in the first level of the tree the one with the highest Q_{cat} among all possible splits for these m features. The induction continues in the same way for the next levels of the tree while the stop condition is not met. Following this process, all the generated trees should be different due to the random component of the feature subset selection at every level. Finally, after the

patterns have been extracted from the collection of trees, duplicated patterns are eliminated.

The pseudocode of the induction algorithm appears in Alg. 1; while the pseudocode of the pattern extraction algorithm appears in Alg. 2.

Algorithm 1: UD3cat - Induction of an unsupervised decision tree for categorical datasets.

Data: T - dataset,
μ - minimum support threshold.
Result: UDT - an unsupervised decision tree.
Build the *root node* N, with all the objects in T;
Select randomly a subset F' of $m = \log_2(|F|)$ features, where F is the set of features in T;
foreach *feature* f_j *in* F' *with two or more values that fulfill the support constraint* **do**
  Compute $Q_{cat}(f_j)$ as defined in 4.1;
  foreach *value* v_i *of* f_j *that fulfills the support constraint* **do**
    Create the item "$f_j = v_i$";
if Q_{cat} *values were computed* **then**
  Select the candidate split f_j with the highest Q_{cat} value;
  foreach *item "*$f_j = v_i$*" created* **do**
    Assign $New\_child\_of\_N$ = UD3cat(cov("$f_j = v_i$",T), μ);
return UDT;

4.1.2 Mining patterns for clustering numerical datasets

In this subsection, we introduce a new pattern mining algorithm that extracts only a subset of all patterns, useful for clustering exclusively numerical datasets (PMCN), without applying an a priori discretization on numerical features.

Traditional pattern mining algorithms require to discretize numerical feature. Thus, for example, in a dataset the feature *Age* could be split in the ranges (0,10], (10,20], (20,30], (30,40], etc. If we need a pattern that

Algorithm 2: RPE - Recursive Pattern Extraction.

Data: *node* - current node (initially is the *root* node of the unsupervised decision tree),
path - list of items in the current path (initially is *null*),
patterns - list of patterns (initially is *empty*).

Result: *patterns* - list of patterns.

```
foreach child in node.children do
    path.Add(child.item);
    pattern = new Pattern();
    foreach item in path do
        // Create a pattern from the items of the current path
        pattern.Add(item);
    patterns.Add(pattern);
    // Depth search of new items
    RPE(child, path, patterns);
    path.RemoveLastElement();
return patterns;
```

covers all the persons with an age in the range (0,25], a pattern might be $Age = (0, 10] \vee Age = (10, 20] \vee Age = (20, 30]$, which also covers those persons with age between 26 and 30. On the contrary, PMCN avoids an a priori discretization step using the relational operators "$\leq$" and "$>$" for numerical features, which allows to build the pattern $[Age \leq 25]$.

For mining patterns, we use a collection of binary *unsupervised decision trees* generated through a new induction procedure. The proposed induction procedure includes a new split evaluation criterion involving numerical features to build internal nodes of the tree.

Inducing an unsupervised decision tree

We propose to induce a binary unsupervised decision tree by creating splits that maximize the difference between the feature-value means (centroids) of the child nodes. This reflects the criterion that the objects in different clusters should be as dissimilar as possible, which is a widely used clustering criterion.

For each feature f_j, let V_j be the set of all values that feature f_j takes in the objects of a node N. The induction algorithm creates, for each value $v_i \in V_j$, two new child nodes (one candidate split) with the items $f_j \leq v_i$ and $f_j > v_i$, respectively. We associate to each child node those objects in the parent node that fulfill the corresponding item. Then, for each new node, we compute the mean of the values of feature f_j as in Eq. (4.2).

$$M = \frac{\sum_{v_i \in V_j} v_i}{|V_j|}. \tag{4.2}$$

Eq. (4.3) defines the split evaluation criterion Q_{num}, for a decision node N split with feature f_j and value v_i, as the difference between the means of the left and right child nodes; normalized by the range of values of feature f_j in the node N.

$$Q_{num}(f_j, v_i) = \frac{|M_L - M_R|}{\max\{V_j\} - \min\{V_j\}}, \tag{4.3}$$

In Eq. (4.3), M_L and M_R are the means of the left and right child nodes respectively, that would be produced by the candidate split (f_j, v_i), while $\max\{V_j\}$ and $\min\{V_j\}$ are the maximum and minimum values that feature f_j takes in the *parent* node, respectively. The denominator in Eq. (4.3)

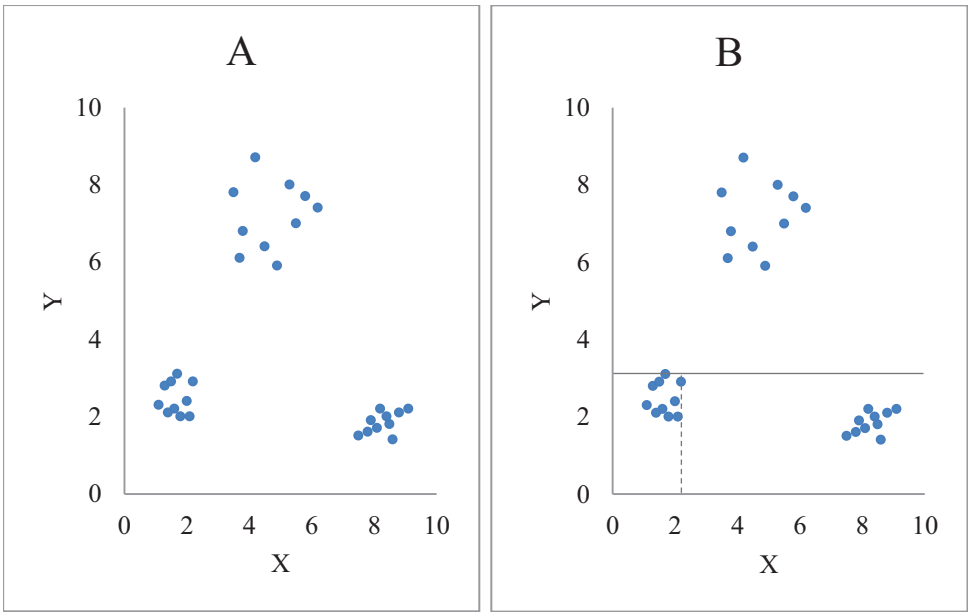

Figure 4.3: A) Synthetic dataset with 30 objects and 3 clusters. B) The best two splits according to Eq. (4.3).

normalizes $Q_{num}(f_j, v_i)$. High values of the quality means that objects from different clusters are dissimilar in feature f_j.

For example, Fig. 4.3A) shows a synthetic dataset with 30 objects grouped in 3 well defined clusters. Some candidate splits for partitioning the dataset are $(X, 2.2)$, $(X, 6.2)$ and $(Y, 3.1)$. In order to select the best split among these three candidates, we compute, for each candidate, the means for the corresponding feature. For the first candidate split $(X, 2.2)$, the means of the values that feature X takes in the child nodes are 1.67 and 6.52; while for the second candidate split $(X, 6.2)$ the means for feature X in the child nodes are 3.21 and 8.29. After applying the split evaluation criterion, the values of $Q_{num}(X, 2.2)$ and $Q_{num}(X, 6.2)$ are 0.54 and 0.57 respectively. For the candidate split $(Y, 3.1)$, the means for feature Y in the child nodes are 2.16 and 7.18. After normalization, the value of $Q_{num}(Y, 3.1)$ is 0.7, which is the highest. For this reason, in this example, the best candidate split for partitioning the dataset, according Q_{num}, is $(Y, 3.1)$. In the next step, the best candidate split is $(X, 2.2)$, like Fig. 4.3B) shows.

For inducing an unsupervised decision tree, each node of the tree is split

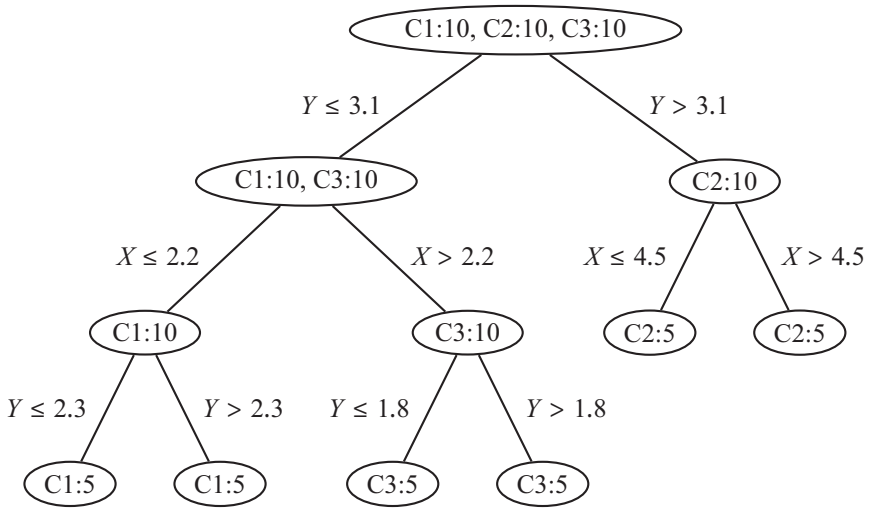

Figure 4.4: Tree induced by Alg. 3, using the data in Fig. 4.3A) with a minimum support threshold $\mu = 0.15$.

by selecting the best candidate split according to (4.3) (i.e. the one with the highest Q_{num} value). The process stops when one of the new child nodes (or both) does not fulfill the support defined by the user, as PMCC does.

The tree induced by Alg. 3, using the data in Fig. 4.3A) with a minimum support threshold equal to 0.15, is shown in Fig. 4.4. The number of objects in each node is represented by $C1$, $C2$ and $C3$ for the three clusters. Note that the number of levels in the tree depends on the support threshold, which is the stopping criterion of the induction process.

Simplifying patterns

Once the patterns have been extracted by applying Alg. 2, each pattern is simplified by joining redundant items of the same feature, in order to obtain more compact patterns. For patterns built from numerical features, two items of the same feature are redundant if one item is more general than the other one. An item I_1 is more general than another item I_2 if all objects in the universe that fulfill I_1 also fulfill I_2, but not vice versa. If there are redundant items in a pattern, the more general item is eliminated. An example of redundant items is:

- $[Y \leq 2.3 \wedge Y \leq 3.1]$, which is simplified as $[Y \leq 2.3]$.

The patterns extracted from the tree in Fig. 4.4, after simplifying redundant items, are:

- $[Y \leq 3.1]$
- $[X \leq 2.2 \wedge Y \leq 3.1]$
- $[X \leq 2.2 \wedge Y \leq 2.3]$
- $[X \leq 2.2 \wedge Y > 2.3 \wedge Y \leq 3.1]$
- $[X > 2.2 \wedge Y \leq 3.1]$
- $[X > 2.2 \wedge Y \leq 1.8]$
- $[X > 2.2 \wedge Y > 1.8 \wedge Y \leq 3.1]$
- $[Y > 3.1]$
- $[X \leq 4.5 \wedge Y > 3.1]$
- $[X > 4.5 \wedge Y > 3.1]$

Generating several diverse unsupervised decision trees

Since a single tree generates too few patterns, in our proposed miner algorithm we propose to build several different trees (the number of trees is specified by the user). To guarantee diversity among trees, we use the same diversity generation strategy previously explained in Subsection 4.1.1; but evaluating the splits with Q_{num}. Again, after extracting the patterns from the trees, PMCN eliminates duplicate patterns.

The pseudocode of the induction algorithm appears in Alg. 3.

4.1.3 Mining patterns for clustering mixed datasets

In this subsection, a new pattern mining algorithm, that extracts patterns for clustering mixed datasets (PMCM), is proposed. The proposed pattern mining algorithm takes the advantages of the split evaluation criteria previously proposed for PMCC and PMCN. Since the proposed split evaluation criteria are not directly comparable, we introduce a new strategy for selecting the best split.

Algorithm 3: UD3num - Induction of an unsupervised decision tree for numerical datasets.

Data: T - dataset,
μ - minimum support threshold.

Result: UDT - an unsupervised decision tree.

Build N the *root node* of UDT, with all the objects in T;

Select randomly a subset F' of $m = \log_2(|F|)$ features, where F is the set of features in T;

foreach *feature* f_j *in* F' **do**
  foreach *value* v_i *of* f_j *in* N **do**
    Create two items: "$f_j \leq v_i$" and "$f_j > v_i$";
    if *the two items fulfill the support constraint* **then**
      Compute $Q_{num}(f_j, v_i)$ as defined in Eq. (4.3);

if Q_{num} *values were computed* **then**
  Select the candidate split with the highest Q_{num} value;
  Assign $Left\_child\_of\_N$ = UD3num(cov("$f_j \leq v_i$",T), μ);
  Assign $Right\_child\_of\_N$ = UD3num(cov("$f_j > v_i$",T), μ);

return UDT;

Inducing an unsupervised decision tree

A new strategy is introduced to select the best split between categorical and numerical features for mixed datasets. The split evaluation criteria $Qcat$ and $Qnum$, introduced in Eq. (4.1) and Eq. (4.3) respectively, are defined in different ranges of values; making them non comparable. This is a problem when we want to select the best split in datasets containing both mixed categorical and numerical features. In order to take into account the advantages of both measures, a modification to RandomForest is proposed to select the best split.

For a single tree in RandomForest, we propose to select the best categorical split and the best numerical split from a subset of $m = \log_2(|F|)$ features, according to the $Qcat$ and $Qnum$ split criteria. Then, one of these two splits is randomly selected, taking into account the proportion of categorical and numerical features in the subset of m features. The probability of selecting the best numerical split is $\frac{num}{num+cat}$, while the probability of selecting the best categorical split is $\frac{cat}{num+cat}$[1]. Thus, we randomly generate a number between 0 and $num + cat - 1$; if the random number is less than num, we select as split for node N the best numerical split. Otherwise, we select the best categorical split.

For example, the Zoo dataset has 17 features. The number of features that are evaluated at each step is $m = \log_2(17) \approx 4$. If one of the four features is numerical and the other three features are categorical, the best numerical split and the best categorical split are taken into account to determine which one of them will be used as split for the node. Then, one split is randomly selected among the two candidate splits, but the categorical split has the priority, since there are more categorical features in m than numerical features. To do that, a number between one and four is randomly selected. In this example, if the random number is one, then the numerical split will be selected for the node. Otherwise, the categorical split will be selected.

A possible tree induced by PMCM, using the data in Table 4.1 with minimum support $\mu = 0.15$, is shown in Fig. 4.5.

Extracting patterns

In order to extract the patterns from a single tree, we apply the same Alg. 2. Additionally, numerical features are simplified to obtain more compact

[1] num and cat are, respectively, the amount of numerical and categorical features in m.

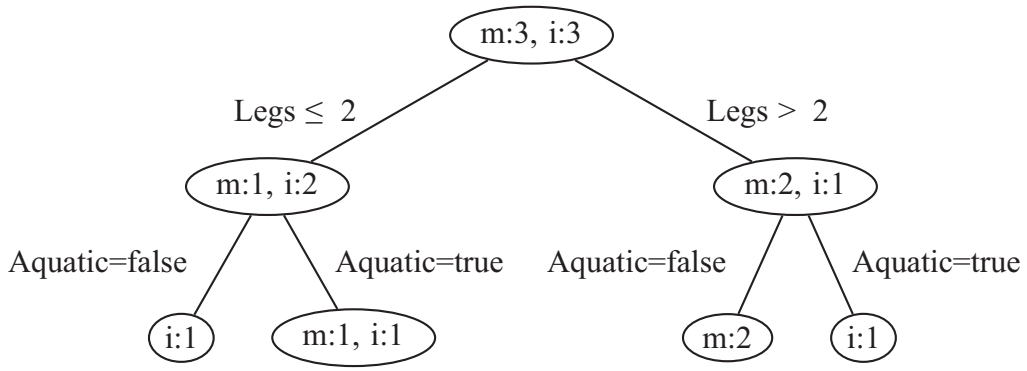

Figure 4.5: Tree induced by Alg. 4, using the data in Table 4.1 with $\mu = 0.15$.

patterns by joining redundant numerical items, as it is done in PMCN.

The patterns extracted from the tree in Fig. 4.5 are:

- $[Legs \leq 2]$
- $[Legs \leq 2 \wedge Aquatic = false]$
- $[Legs \leq 2 \wedge Aquatic = true]$
- $[Legs > 2]$
- $[Legs > 2 \wedge Aquatic = false]$
- $[Legs > 2 \wedge Aquatic = true]$

Generating several diverse unsupervised decision trees

As we have already commented, the number of extracted patterns from a single tree is very small, thus we propose to build several different trees. To guarantee diversity among trees, it is used the same diversity generation strategy employed in the previously proposed algorithms. Again, the amount of trees to generate is specified by the user, and duplicate patterns are eliminated after extracting the patterns from the collection of trees.

The pseudocode of the induction algorithm for building an unsupervised decision tree for mixed datasets appears in Alg. 4. The stopping criterion is the same used for PMCC an PMCN.

Algorithm 4: UD3mix - Induction of an unsupervised decision tree for mixed datasets.

Data: T - dataset,
μ - minimum support threshold.

Result: UDT - an unsupervised decision tree.

Build N the *root node* of UDT, with all the objects in T;

Select randomly a subset F' of $m = \log_2(|F|)$ features, where F is the set of features in N's objects;

Set $cat = 0$ and $num = 0$;

foreach *feature* f_j *in* F' **do**
  if f_j *is categorical* **then**
    $cat++$;
    foreach *value* v_i *of* f_j *that fulfill the support constraint in* N **do**
      Create the item "$f_j = v_i$";
    if *at least two items were created* **then**
      Compute $Q_{cat}(f_j)$ as defined in 4.1;
  else
    $num++$;
    foreach *value* v_i *of* f_j *in* N **do**
      if *the two items fulfill the support constraint* **then**
        Compute $Q_{num}(f_j, v_i)$ as defined in Eq. (4.3);
if Q_{cat} *or* Q_{num} *values were computed* **then**
  Select the two candidate splits with the highest Q_{cat} and Q_{num} values;
  if *random(0, cat + num - 1)* $<$ *num* **then**
    Select the candidate split f_j corresponding to the highest Q_{num} value;
    Assign $Left\_child\_of\_N$ = UD3mix(cov("$f_j \leq v_i$",T), μ);
    Assign $Right\_child\_of\_N$ = UD3mix(cov("$f_j > v_i$",T)), μ);
  else
    Select the candidate split f_j corresponding to the highest Q_{cat} value;
    foreach *item "*$f_j = v_i$*" created* **do**
      Assign $New\_child\_of\_N$ = UD3mix(cov("$f_j = v_i$",T)), μ);
return UDT;

4.2 A new pattern-based clustering algorithm

This section introduces a new pattern-based clustering algorithm (PbCA), with the purpose of obtaining better clustering results than those obtained by the pattern-based clustering algorithms reported in the state-of-the-art, and that allows clustering any dataset[2]. The idea of firstly find out a relationship between the patterns extracted from the dataset is followed. Then, based on this relationship to build clusters of patterns, the objects of a dataset are finally grouped into the clusters of patterns.

4.2.1 Clustering patterns

In this stage, PbCA has two steps. First, based on the idea proposed by Liu and Dong (2012) that a good pattern clustering must have small high-quality pattern subgroups, these subgroups are built from the extracted patterns. Second, the pattern subgroups are clustered to get the final clusters.

In order to find the pattern subgroups, the patterns are filtered using equivalence classes, in the same way as CPC does (see Section 3.3). Then, for each equivalence class, a closed pattern is created as the conjunction of all the items of the patterns in the equivalence class. Each closed pattern can be seen as the representative (or the description) of its equivalence class. PbCA works with closed patterns because it allows reducing the redundancy among patterns and, consequently, the computational cost of the proposed pattern-based clustering algorithm. In the following two subsections, we will use the term "pattern" to refer to a "closed pattern".

Finding small pattern subgroups

The pattern subgroups are found by clustering the patterns using a strategy similar to K-means, using a custom algorithm for computing the centroids and the Jaccard similarity (Pandit et al., 2011) to evaluate the similarity between patterns. The Jaccard similarity between patterns P_i and P_j is computed using the sets of objects covered by P_i and P_j, as defined in Eq. (4.4).

$$J\left(P_i, P_j\right) = \frac{|cov(P_i) \cap cov(P_j)|}{|cov(P_i) \cup cov(P_j)|}. \tag{4.4}$$

[2]Since some algorithms cannot cluster several datasets.

If $cov(P_i) = cov(P_j)$, the similarity takes value 1; and if $cov(P_i) \cap cov(P_j) = \emptyset$, the similarity takes value 0. Based on Eq. (4.4), a similarity matrix SM that contains the similarity values between all pairs of patterns is constructed.

Once the matrix of similarities between patterns has been computed, the modified K-means is run to obtain small pattern subgroups. The number of pattern subgroups must be larger than the number of clusters k predefined by the user. The number of pattern subgroups is $k' = g \cdot k$, where g is a parameter. At first, k' patterns are randomly selected as the initial centroids for the k' pattern subgroups. Then, the remaining patterns are assigned to the pattern subgroup with the most similar centroid. Later, the new centroids for the k' pattern subgroups are re-computed: for each subgroup the new centroid is the pattern that is the most similar to the other patterns in the same subgroup. The process of assigning the patterns and re-computing the centroids of the pattern subgroups is repeated until there is no change in the centroids from one iteration to the next one, or until a predefined number of iterations is reached. Finally, the k' pattern subgroups are those computed in the last iteration of K-means.

For all the datasets, to overcome the local minima of K-means, this algorithm is ran 10 times[3] with different randomly selected initial centers. Then, the partition that maximizes Eq. (4.5), among the 10 results, is selected.

$$B(K) = \min_{i,j} J(\mu_i, \mu_j), \forall i \neq j, \tag{4.5}$$

In Eq. (4.5), K is the set of clusters; μ_i and μ_j are the patterns selected as the centers of the clusters K_i and K_j respectively; and $J(\mu_i, \mu_j)$ is the Jaccard similarity (see Eq. (4.4)) between μ_i and μ_j.

Computing the final pattern subgroups

For clustering the k' pattern subgroups into k pattern clusters, the same modified K-means algorithm is ran, but now the pattern subgroups are clustered rather than the patterns. Thus, a new similarity matrix SM' is computed to evaluate the similarity between the k' pattern subgroups. The similarity between two pattern subgroups G_i and G_j is defined, in Eq. (4.6), as the average similarity between all pairs of patterns $P \in G_i$ and $Q \in G_j$.

[3]We run K-means more than 10 times but it significantly increases its runtime, while running the algorithm less than 10 times causes a decrease in the F-measure values.

$$S\left(G_i, G_j\right) = \frac{\sum_{P \in G_i} \sum_{Q \in G_j} J\left(P, Q\right)}{|G_i| \cdot |G_j|}, \tag{4.6}$$

In Eq. (4.6), P and Q are patterns belonging to G_i and G_j respectively, while $|G_i|$ and $|G_j|$ are, respectively, the number of patterns belonging to G_i and G_j. Based on the new similarity matrix SM', the modified K-means is runned in the same way as in the previous stage to compute k clusters of pattern subgroups.

4.2.2 Clustering objects

Once the clusters of patterns have been built, each object in the dataset is assigned to the cluster with the highest fraction of patterns covering it, as in Eq. (4.7):

$$A(O_i) = \arg\max_{K_j \in K} \frac{\sum_{P \in K_j} |cov(P) \cap \{O_i\}|}{|K_j|}, \tag{4.7}$$

where K represents the pattern clustering and the object O_i is assigned to the cluster K_j that maximizes the argument of Eq. (4.7). If there is a tie in Eq. (4.7) among several pattern clusters, the object is randomly assigned to one of them. In this way, the proposed algorithm returns a clustering of objects where each cluster has associated a pattern set that describes the objects belonging to it. The pseudocode of PbCA is shown in Alg. 5.

4.3 Complexity analysis

The analysis of the computational complexity of the proposed algorithms is divided in two parts. First, the time and space complexities of the algorithms proposed for mining frequent patterns is analyzed. Then, the time and space complexities of the proposed pattern-based clustering algorithm is also analyzed.

4.3.1 Time complexity analysis

In this section, it is assumed that n is the number of objects in a dataset, each one described through m features. In PMCC, for selecting the best categorical split for a single feature, evaluating Eq. (4.1), has a cost of n because

Algorithm 5: PbCA - Pattern-based clustering algorithm.

Data: T - dataset,
PS - set of patterns,
k - number of clusters to build,
g - average number of pattern subgroups in each pattern cluster.

Result: *A clustering where each cluster of objects has associated a pattern set.*

Compute a similarity matrix SM between all the pair of patterns in PS using the similarity measure of Eq. (4.4);

Set $k' = k \cdot g$;

Run the modified K-means for clustering the patterns into k' groups;

Compute a similarity matrix SM', using the similarity measure in Eq. (4.6), between all the pairs of the k' pattern subgroups;

Run the modified K-means for clustering the k' pattern subgroups into k clusters;

Assigning each object in T to the cluster that maximizes the argument in Eq. (4.7);

return *The set of clusters of objects and their patterns*;

we need to compute the number of different values in each feature. Thus, selecting the best categorical split for a single feature is $O(n)$. Therefore, if the dataset has m features, the computational complexity of selecting the best split is $O(\log_2(m)n)$ for the first level, because $\log_2(m)$ features are selected according to the diversity generation strategy. Hence, the time complexity for constructing a tree with PMCC is $O(\log_2(m)nl)$, where l is the depth of the tree, because at each level the sum of the number of objects in each node is n. However, for further levels, since the features cannot be chosen again, the number of candidate features to be evaluated at level i decreases to $\log_2(m-i)$. Thus, if we do not consider the stop criterion, the maximum number of levels is m; then, the total number of selected (evaluated) features is $\sum_{i=0}^{m-1} \log_2(m-i) = \log_2(m!)$. Therefore, the time complexity for inducing a single unsupervised decision tree with PMCC is at most $O(\log_2(m!)n)$. If the number of trees to generate by PMCC is t, then the time complexity for PMCC is $O(t \log_2(m!)n)$.

In PMCN, for selecting the best numerical split for a single feature, Eq. (4.3) is evaluated at most n times, because each different feature value is considered as a candidate split. However, computing Eq. (4.3) has a constant time complexity if the numerical values are sorted and the means of the child nodes are dynamically updated. Therefore, if the dataset has m features, the time complexity of selecting the best numerical split is $O(\log_2(m)n)$ for the first level, due to the diversity generation strategy. For further levels, since the features can be repeated, the number of candidate features to be evaluated is always $\log_2(m)$. Thus, the time complexity for inducing a tree with PMCN is $O(\log_2(m)nl)$, where l is the depth of the tree. If we do not consider the stopping criterion, in the worst case (skewed binary tree) $l = n - 1$, and in the best case (balanced tree) $l = \log_2(n)$. Given that the time complexity of sorting the values of a numerical feature is $O(n \log_2(n))$, for the m features the time complexity is $O(mn \log_2(n))$. Then, assuming that $n \geq m$, the overall time complexity T of inducing a single tree with PMCN is $O(mn \log_2(n)) \leq T \leq O(\log_2(m)n^2)$, and for inducing t trees we have a time complexity $O(tmn \log_2(n)) \leq T' \leq O(t \log_2(m)n^2)$.

In PMCM, $\log_2(m)$ features are only evaluated at each level. In the best case, if all the features of the dataset are categorical, for constructing a single tree, $\log_2(m!)n$ operations are needed. Otherwise, in the worst case, if all the features of the dataset are numerical, we need $\log_2(m)n^2$ operations. Therefore, the time complexity T for inducing a single tree with PMCM is $O(\log_2(m!)n) \leq T \leq O(\log_2(m)n^2)$. For inducing t trees, the time complexity

of PMCM is $O(t \log_2(m!)n) \leq T' \leq O(t \log_2(m)n^2)$

Finally, it is important to comment that extracting the patterns from the induced trees, built by any of the three pattern mining algorithms, does not increase the complexity because the patterns can be directly extracted from the trees during the induction process.

In order to analyze the time complexity of PbCA, the analysis is split in five parts. First, for computing the similarity matrix SM, if each pattern covers all the objects in the dataset, for comparing two patterns n operations are needed. Thus, computing SM is $O(p^2 n)$, where p is the number of patterns to cluster. Second, since in order to find the k' pattern subgroups, the modified K-means algorithm is applied, this part of the algorithm is $O(p^2)$. Third, assuming that each pattern subgroup has approximately p/k' patterns and the similarities between them are already stored in SM, $(k')^2 \cdot (p/k')^2 = p^2$ operations are needed for computing the similarity matrix SM' between the pattern subgroups. Then, computing SM' is $O(p^2)$. Fourth, clustering the k' pattern subgroups by applying the modified K-means algorithm has a computational complexity of $O((k')^2)$. Fifth, assigning the objects to clusters has a complexity of $O(npk)$, because for each object we count the patterns that cover the object in each cluster. Therefore, because the highest time complexity of the five parts is computing SM, the time complexity of PbCA is $O(p^2 n)$.

4.3.2 Space complexity analysis

Since the number of induced trees is t, the maximum number of nodes that a single tree has, multiplied by t, determines the space complexity of the mining process. For a single binary tree induced by PMCN, in the worse case (skewed binary tree) the number of levels is $l = n - 1$, each level containing two nodes. Thus, the total number of nodes in the worse case is $2(n - 1)$. In the best case (balanced tree) the number of levels is $l = \log_2(n)$, and the total number of nodes is $2^{l+1} - 1$; then, the total number of nodes is $2^{\log_2(n)+1} - 1 = 2n - 1$. Therefore, the space complexity for PMCN is $O(tn)$.

In the case of PMCC and PMCM, these miners generate n-ary trees. In (Cha, 2012), a complete analysis of the space complexity of balanced n-ary trees is provided. In a single balanced n-ary tree, the total number of nodes is $\frac{k^{l+1}-1}{k-1}$, where l is the number of levels and k is the number of children for each node. Since the maximum number of leaf nodes is n, the number of levels is $l = \log_k(n)$ (Basak and Krishnapuram, 2005). Thus, the total

number of nodes is $\frac{kn-1}{k-1}$, $2 \leq k \leq n$. If $k = n$ the total number of nodes is $n + 1$, while if $k = 2$ the total number of nodes is $2n - 1$. Therefore, for inducing t trees, the space complexity of PMCC and PMCM is $O(tn)$.

The space complexity of the clustering process depends on the number of patterns (p), objects (n) and of features (m). Storing the extracted patterns has a space complexity of $O(pm)$; while storing the extracted patterns and the index of the objects that each pattern covers has a space complexity $O(pn)$. In addition, storing SM has a space complexity $O(p^2)$. Therefore, the space complexity of the clustering process is $O(p^2 + pn + pm)$. Since, in our trees, we consider as a pattern any path from the root node to any other node, the maximum number of patterns that can be extracted from a single tree is the same as its maximum number of nodes. Thus, in a collection of t trees the maximum number of patterns is $O(tn)$. Finally, the space complexity of PbCA is $O(t^2n^2 + tn^2 + tnm)$; while the time complexity of PbCA, analyzed in the previous subsection, is $O(t^2n^3)$.

4.4 Concluding remarks

In this chapter, three pattern mining algorithms for clustering categorical, numerical and mixed datasets are proposed. The algorithms PMCN and PMCM extract only a subset of patterns useful for clustering, instead of mining all patterns, without applying any a priori discretization of numerical features. Our algorithms mine a subset of patterns by inducing a collection of unsupervised decision trees using RandomForest. In addition, we also proposed a new pattern-based clustering algorithm that first groups the patterns extracted by the proposed mining algorithms, and then the objects of the dataset are clustered into the pattern subgroups. Our proposed miners have a computational and space complexity less than FP-growth. Additionally, PbCA has the same time and space complexity as the clustering step of CPC. In both cases, the complexity depends on the number of patterns (p). However, our proposed miners extract significantly less patterns than FP-growth, since our miners extract $2tn$ patterns in the worse case, while FP-growth extracts $n(2^m - 1)$ patterns. As we will show later in our experiments, our proposed pattern mining algorithms are able to extract just a few patterns in a suitable time for all the tested datasets.

Chapter 5

Experimental results

This chapter shows some experiments designed to evaluate the performance of the proposed algorithms. In the first experiment, we evaluate different number of trees to be generated for our proposed miner algorithms. The second experiment evaluates how good the patterns extracted by PMCC, PMCN and PMCM are for clustering. Third, a comparison between our proposed pattern-based clustering algorithm, PbCA, against CPC (Fore and Dong, 2012), the closest pattern-based clustering algorithm to our proposal, is performed. This comparison is made in terms of F-measure, runtime and number of extracted patterns. The fourth experiment compares PbCA against other state-of-the-art pattern-based clustering algorithms, and some traditional (non pattern-based) clustering algorithms.

For our experiments, well-known datasets taken from the UCI Repository (Bache and Lichman, 2013) are used. For replacing missing values, the algorithm ReplaceMissingValues, implemented in the Weka framework (Hall et al., 2009), is employed. For all the experiments, the datasets are divided into categorical, numerical and mixed. Each section presents a description of the datasets used for each experiment.

For comparing the clustering results obtained by our proposed pattern-based clustering algorithm, five clustering algorithms are selected. Three of them are pattern-based clustering algorithms: COBWEB (Fisher, 1987), CPC (Fore and Dong, 2012) and CLUS (Blockeel H., 1998). The COBWEB algorithm is selected because it is a well-known pattern-based clustering algorithm available in the Weka framework. The implementation provided by the authors of CPC is used. CLUS is also included since it is an algorithm for clustering based on unsupervised decision trees (see implementation details

about CLUS in (Struyf, 2015)). Two traditional clustering algorithms, K-means (MacQueen, 1967) and EM (Moon, 1996), which are also available in the Weka framework, were included in our comparison. For those algorithms taken from Weka, we used the default values for their parameters. In the case of CPC, K-Means, EM and our proposed algorithms, we set the number of groups as the number of classes reported for each dataset in the repository. For CLUS, we fixed the predictive attribute equal to the class attribute in each dataset, and the remaining attributes were used for clustering.

In our experiments, 0.05, 0.06, 0.07, 0.08 and 0.09 are used as support threshold values. Lower values increase the computational cost due to the huge amount of patterns that are computed by FP-growth[1]. Higher values produce only a small amount of patterns. For each miner and for each dataset, the support threshold value, that allow obtaining the best result, is selected. However, since there are many values, these values are not reported. For PbCA, in all the experiments, the average number of pattern subgroups in each pattern cluster is fixed as $g = 5$. This value is selected because values between 2 and 9 were tested in a small number of datasets, and $g = 5$ obtained the best results. Since CPC was designed for categorical features, for using CPC in datasets with numerical features, these features are a priori discretized using the Weka implementation of the Equal Width Binning (Dash et al., 2011) discretization algorithm (with 10 bins), which is the same algorithm used in (Fore and Dong, 2012).

The clustering results are evaluated with the external quality index F-measure (Amigó et al., 2009). To determine the statistical significance of differences among the clustering results, the Friedman test is performed to compare multiple algorithms (Demsar, 2006). For this test, a significance level of 0.05 is fixed. The Friedman test makes a pairwise comparison between all the results and generates a ranking of the compared clustering algorithms. When there were significant differences according to the Friedman test, the Bergmann-Hommel test, with a significance level of 0.1, is performed, in order to determine which clustering results are significantly different. Both significance level values were fixed according to García and Herrera (2008) for this type of experiments.

All the experiments were performed on a desktop PC with an Intel-Core i7 processor at 3.4 GHz and 32 GB of RAM, running Microsoft Windows 7.

[1]FP-growth is the pattern mining algorithm used by CPC

5.1 Evaluating different number of trees to be generated for our miners

This section evaluates different number of induced trees for our proposed miners. To evaluate the quality of the extracted patterns by our miners, the quality (F-measure) of clustering using PbCA is measured.

5.1.1 Evaluating different number of trees for PMCC

For this experiment, 10 categorical datasets, which are shown in Table 5.1, are selected.

Dataset	#Obj	#CatF	#Class
Audiology	226	69	24
Balloons	20	4	2
Hiv-1	6590	8	2
Lenses	24	4	3
Lung-cancer	26	56	3
Post-operative	90	8	3
Primary-tumor	339	17	21
Solar-flare	323	12	6
Spect	267	22	2
Sponge	76	45	3

Table 5.1: Description of the categorical datasets used to evaluate different number of trees for PMCC.

This experiment compares the results of clustering the datasets in Table 5.1 using PbCA with the patterns extracted by inducing 20, 40, 60, 80, and 100 trees. Generating less than 20 trees highly decrease the quality of the clusters, in terms of F-measure. On the other hand, values larger than 100 do not improve the quality of the clusters, but increase the runtime.

Fig. 5.1 shows the best clustering results, according to F-measure, for each number of trees. In this figure, we can see that there are small differences among the different number of trees; even in some datasets the F-measure values are the same. Thus, we compute the Friedman test to compare these results.

Table 5.2 shows the average ranking of the F-measure results according to the Friedman test. The P-value computed by the test is 0.736, this value

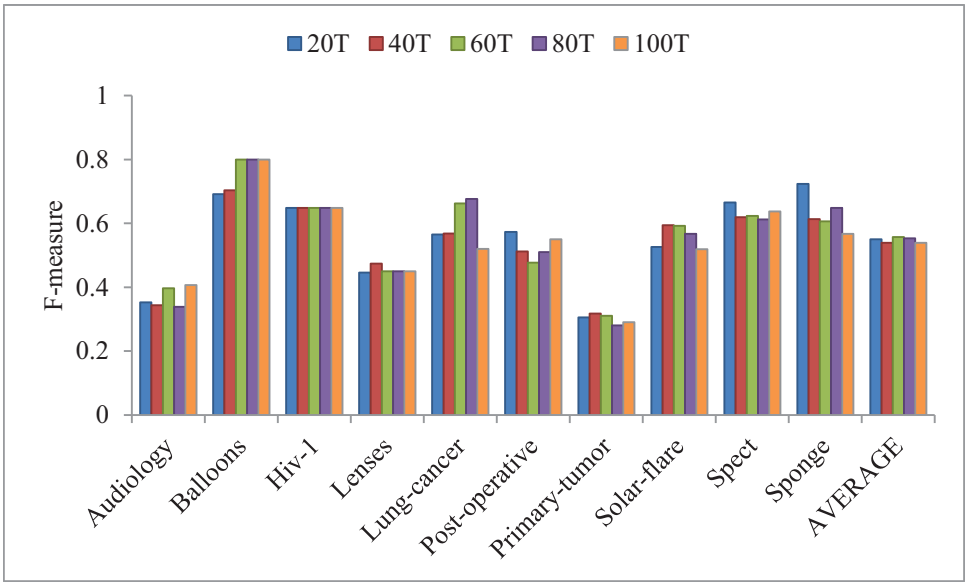

Figure 5.1: Results of F-measure for PbCA with the patterns extracted from PMCC inducing different number of trees.

is not less than, or equal to, 0.05. It means that the differences between the results are not statistically significant. However, from Table 5.2 we can see that the best ranking of F-measure is obtained with 40 trees. Therefore, the number of trees is fixed to 40 for PMCC in the further experiments.

#Trees	Ranking
40	2.7
60	2.8
20	3.0
100	3.2
80	3.3

Table 5.2: Average rankings of PbCA with different number of induced trees for PMCC.

5.1.2 Evaluating different number of trees for PMCN

We perform a similar experiment to evaluate different number of induced trees for PMCN, but in the 10 numerical datasets shown in Table 5.3. Fig.

5.2 shows the best F-measure results for each number of trees. Notice that there are not relevant differences among the different number of trees.

Dataset	#Obj	#NumF	#Class
Biodeg	1055	41	2
Breast-tissue	106	9	6
Cloud	108	4	4
Diabetes	768	8	2
Parkinsons	195	22	2
Segment	2310	19	7
Spectrometer	531	100	4
Vertebral-2	310	6	2
Wholesale	440	7	2
Yeast	1484	8	10

Table 5.3: Description of the categorical datasets used to evaluate different number of trees for PMCC.

Table 5.4 shows the average ranking of the F-measure results according to the Friedman test. The P-value computed by the test is 0.620, this value is not less than, or equal to, 0.05. Thus, the differences between the results are not statistically significant. In Table 5.4, the best ranking of F-measure is also obtained with 40 trees; therefore, we fix the number of trees to 40 for PMCN in the further experiments.

#Trees	Ranking
40	2.6
100	2.7
60	2.9
80	3.2
20	3.6

Table 5.4: Average rankings of PbCA with different number of induced trees for PMCN.

5.1.3 Evaluating different number of trees for PMCM

For evaluating different number of trees for PMCM, the 10 mixed datasets shown in Table 5.5 are selected. Fig. 5.3 shows the best clustering results

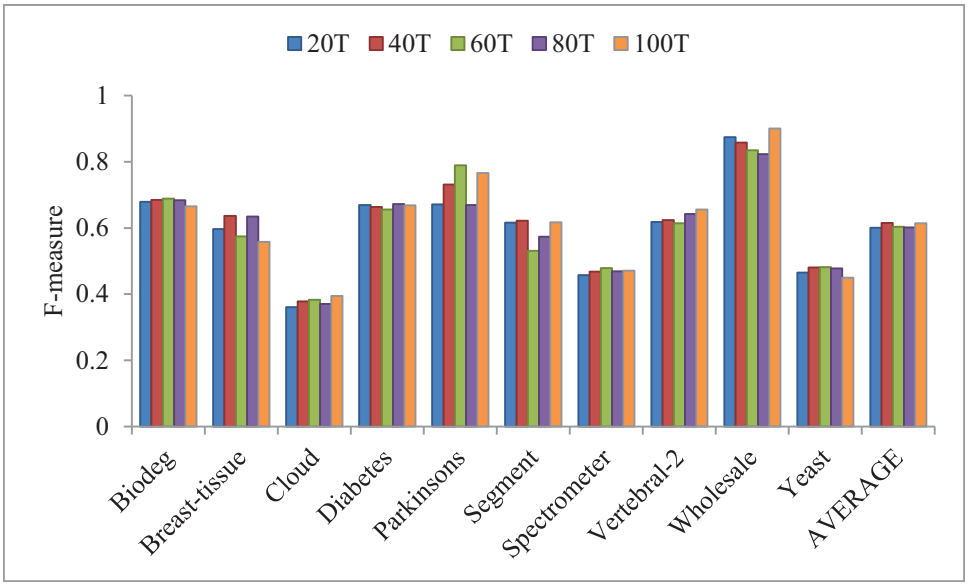

Figure 5.2: Results of F-measure for PbCA with the patterns extracted from PMCN inducing different number of trees.

according to F-measure. In Fig. 5.3, there are not important differences in the results.

Table 5.6 shows the average ranking of the clustering results according to the Friedman test. The P-value computed by the test is 0.592, this value is not less than, or equal to, 0.05. Therefore, the differences of the PMCM results by inducing different number of trees are not statistically significant. However, the best ranking of F-measure is also obtained with 40 trees, and for this reason, we fix the number of trees to 40 for PMCM in further experiments.

Based on the previous experiments, we can fix to 40 the number of trees for all the proposed pattern mining algorithms. It is important to remark that for categorical datasets, PMCC and PMCM extract exactly the same patterns, since for categorical data PMCC and PMCM works in the same way. Also, for the same reason, for numerical datasets PMCN and PMCM extract the same patterns. The only difference in both cases is that PMCM checks the type of feature before evaluating the candidate splits.

Dataset	#Obj	#CatF	#NumF	#Class
Abalone	4177	1	7	28
Anneal	898	32	6	5
Cylinder-bands	540	21	18	2
Dermatology	366	33	1	6
Fertility	100	5	4	2
Horse-colic	368	15	7	2
Hypothyroid	3772	22	7	4
Seismic-bumps	2584	4	14	2
Tae	151	2	3	3
Zoo	101	16	1	7

Table 5.5: Description of the categorical datasets used to evaluate different number of trees for PMCM.

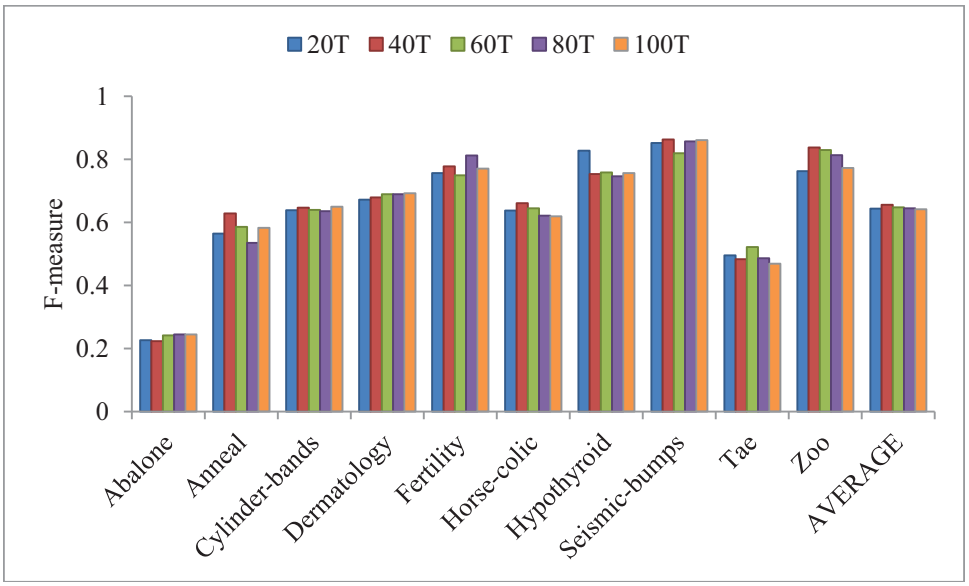

Figure 5.3: Results of F-measure for PbCA with the patterns extracted from PMCM inducing different number of trees.

#Trees	Ranking
40	2.5
60	2.8
100	2.9
80	3.2
20	3.6

Table 5.6: Average rankings of PbCA with different number of induced trees for PMCM.

5.2 Comparing the miners

This section evaluates the suitability for clustering of the patterns extracted by our proposed miners using 40 trees, which is the best amount of trees to be induced, according to the experiments shown in Section 5.1. For this experiment, 15 datasets for each type of data, different from those datasets used for tuning the number of trees parameter, are selected.

In order to use PMCC in datasets with numerical features, the numerical features are a priori discretized using the Equal Width Binning (EWB) discretization algorithm (Dash et al., 2011). EWB is the discretization algorithm that CPC uses. Using EWB allows us to maintain uniformity in the comparisons of the experimental results, since Section 5.3 will compare our proposal against CPC. For using PMCN in categorical datasets, the categorical feature values are transformed replacing them by numerical values. To do that, the categorical values are replaced with non-negative integer numbers (starting at 0), by assigning numbers to categorical values in the order they appeared in the dataset.

5.2.1 Comparison over categorical datasets

For comparing the miners in the 15 categorical datasets shown in Table 5.7, PbCA is used with the patterns obtained by the proposed mining algorithms. Although the results with PMCC and PMCM are the same, both are included in this experiment.

Fig. 5.4 shows that the clustering results of PbCA using PMCC and PMCM as miners are the same, as it was expected since they follow the same induction strategy for categorical datasets. Those results are better than the results of PbCA using PMCN as miner.

Dataset	#Obj	#CatF	#Class
Balance-scale	625	4	3
Breast-cancer	286	9	2
Car	1728	6	4
Chess	3196	36	2
Hayes-roth	160	4	3
Lymphography	148	18	4
Molecular	106	58	2
Monks-1	556	6	2
Mushroom	8124	22	2
Soybean-l	683	35	19
Soybean-s	47	35	4
Splice	3190	61	3
Tic-tac-toe	958	9	2
Trains	10	32	2
Vote	435	16	2

Table 5.7: Description of the categorical datasets used to compare PMCC, PMCN and PMCM.

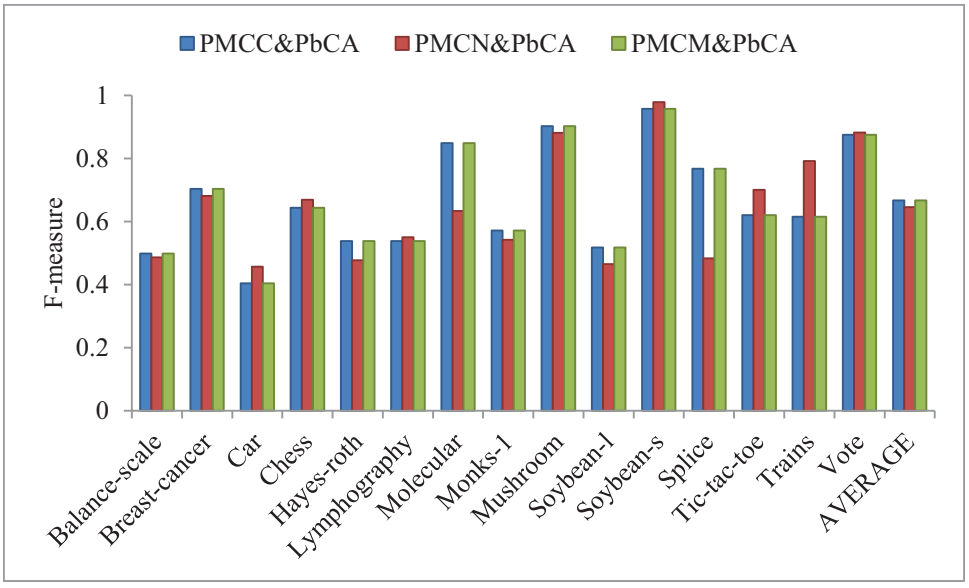

Figure 5.4: Results of F-measure for PbCA with the patterns extracted from PMCC, PMCN and PMCM on categorical datasets.

Table 5.8 shows the average ranking of the clustering results according to the Friedman test. The P-value computed by the test is 0.951, this value is not less than, or equal to, 0.05. Although the difference in the results is not statistically significant, PbCA with the patterns extracted by PMCC and PMCM gets the best ranking of F-measure according to the Friedman test.

Algorithm	Ranking
PMCC&PbCA	1.97
PMCM&PbCA	1.97
PMCN&PbCA	2.06

Table 5.8: Average rankings of PbCA with the patterns extracted from PMCC, PMCN and PMCM on categorical datasets.

These results suggest that clustering categorical data using PMCC is better than transforming categorical data into numerical data in order to apply a clustering algorithm designed for numerical features.

5.2.2 Comparison over numerical datasets

For comparing the proposed pattern mining algorithms on the 15 numerical datasets shown in Table 5.9, PbCA is used with each of the proposed miners. In this experiment, although the results with PMCN and PMCM are the same, we included both results.

Fig. 5.5 shows the clustering results of PbCA using the patterns extracted from the numerical datasets, shown in Table 5.9, by using the miners PMCC, PMCN and PMCM. In Fig. 5.5, we can see that the results of PbCA using PMCN and PMCM as miners are the same since they both use the same strategy for inducing decision trees on numerical datasets. These results are better than the results of PbCA using PMCC as miner.

Table 5.10 shows the average ranking of the clustering results according to the Friedman test. The P-value computed by the test is 0.086, this value is not less than, or equal to, 0.05. Then, the difference in the results is not statistically significant, but PbCA with the patterns extracted by PMCN and PMCM gets the best ranking of clustering results, in terms of the Friedman test.

From these results, we can conclude that clustering numerical data using the patterns mined by PMCN is better than transforming numerical data

Dataset	#Obj	#NumF	#Class
Breast-w	699	9	2
Ecoli	336	7	8
Faults	1941	27	7
Glass	214	10	6
Ilpd	583	11	2
Ionosphere	351	34	2
Iris	150	4	3
Knowledge	403	5	4
Liver-disorders	345	6	2
Mammographic	961	6	2
Sensor-readings	5456	24	4
Sonar	208	60	2
Transfusion	748	4	2
Vehicle	846	18	4
Wine	178	13	3

Table 5.9: Description of the numerical datasets used to compare PMCC, PMCN and PMCM.

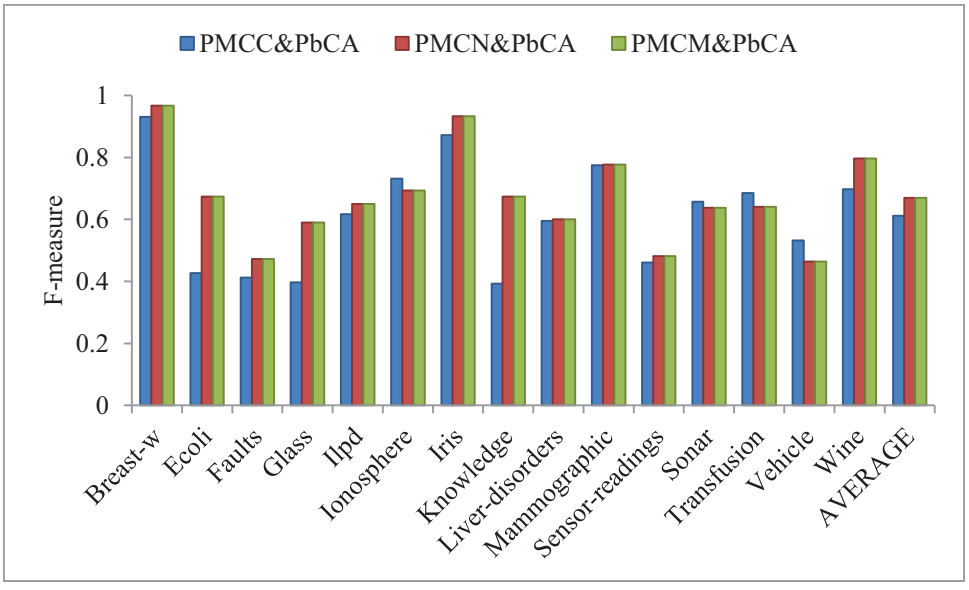

Figure 5.5: Results of F-measure for PbCA with the patterns extracted from PMCC, PMCN and PMCM on numerical datasets.

Algorithm	Ranking
PMCN&PbCA	1.77
PMCM&PbCA	1.77
PMCC&PbCA	2.46

Table 5.10: Average rankings of PbCA with the patterns extracted from PMCC, PMCN and PMCM on numerical datasets.

into categorical data by applying an a priori discretization process (which may cause information loss).

5.2.3 Comparison over mixed datasets

For comparing the proposed pattern mining algorithms over mixed datasets, the 15 mixed datasets shown in Table 5.7 are used. Fig. 5.6 shows the clustering results of PbCA by using the extracted patterns by PMCC, PMCN and PMCM as miners.

Dataset	#Obj	#CatF	#NumF	#Class
Autos	205	10	15	6
Bridges	108	7	4	7
Credit-a	690	9	6	2
Credit-g	1000	13	7	2
Diagnosis	120	6	1	2
Echocardiogram	132	2	7	3
Flags	194	26	2	8
Haberman	306	1	2	2
Heart-c	303	7	6	2
Heart-h	294	7	6	2
Heart-statlog	270	8	5	2
Hepatitis	155	13	6	2
Labor	57	8	8	2
Post-operative	90	7	1	4
Thoracic	470	13	3	2

Table 5.11: Description of the mixed datasets used to compare PMCC, PMCN and PMCM.

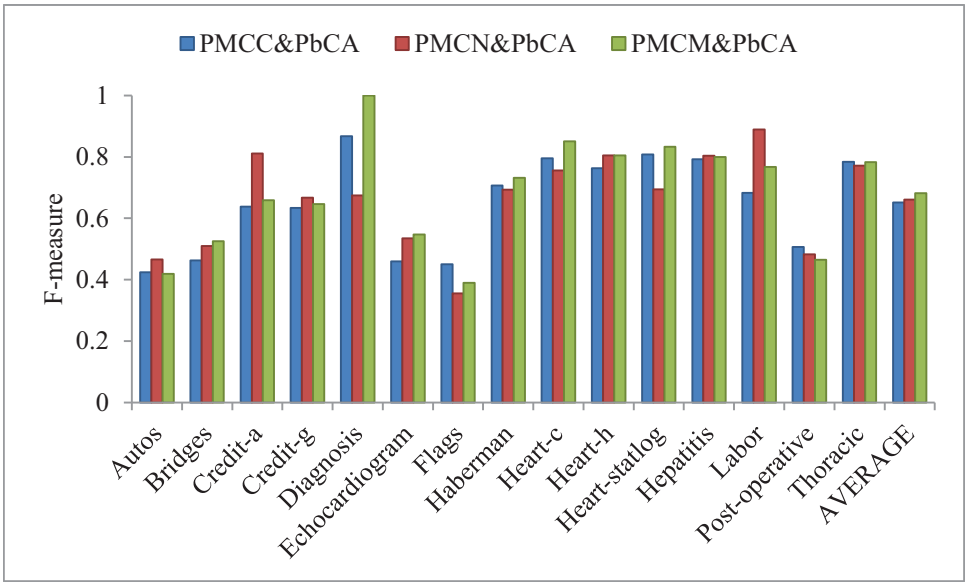

Figure 5.6: Results of F-measure for PbCA with the patterns extracted from PMCC, PMCN and PMCM on mixed datasets.

Table 5.12 shows the average ranking of the clustering results according to the Friedman test. The P-value is 0.296, this value is not less than, or equal to, 0.05. Although the difference in the results is not statistically significant, PbCA with the patterns extracted by using PMCM obtains the best ranking of F-measure according to the Friedman test.

Algorithm	Ranking
PMCM&PbCA	1.70
PMCN&PbCA	2.03
PMCC&PbCA	2.27

Table 5.12: Average rankings of PbCA with the patterns extracted from PMCC, PMCN and PMCM on mixed datasets.

From the previous experiments, we can see that for the three types of datasets, PMCM&PbCA obtained the best results; sometimes tied with PMCC&PbCA (for categorical datasets) or with PMCN&PbCA (for numerical datasets). Therefore, in further experiments, we will use only PMCM as pattern mining algorithm for PbCA.

5.3 Comparing PbCA against CPC

This section compares the results of PbCA and CPC in terms of F-measure, runtime and number of patterns. This section is divided in three subsections, for showing experiments by using categorical, numerical and mixed datasets respectively. In our experiments, the results of PbCA using PMCM as pattern miner, the results of CPC using PMCM as pattern miner, and the results of CPC using FP-growth as pattern miner, as it was originally proposed by Fore and Dong (2012), are included. In our experiments, CPC did not get results for some datasets due to two main reasons: CPC could not handle a huge amount of patterns (more than 100000); and in some other datasets, the CPC overlap constrain was not fulfilled for any subset of k patterns (k is the number of clusters to find) (see Section 3.3).

5.3.1 Comparison over categorical datasets

In order to compare the results of PbCA against CPC over categorical datasets, the 15 datasets shown Table 5.7 are used. The F-measure results are shown in Fig. 5.7. Notice that, in Fig. 5.7, PMCM&CPC could not cluster the Soybean-l dataset, while CPC could not cluster Chess and Soybean-l. For those datasets where an algorithm could not cluster them, we do not show their corresponding bars in the chart.

Table 5.13 shows the average ranking of the clustering results according to the Friedman test. The P-value computed by the test is 0.259, this value is not less than, or equal to, 0.05. The differences of the results are not statistically significant, but using the patterns extracted for PMCM allows obtaining better clustering results than using the patterns extracted by FP-growth. For categorical datasets, PMCM&CPC obtains better results than PMCM&PbCA, and both are better than CPC.

Algorithm	Ranking
PMCM&CPC	1.80
PMCM&PbCA	1.83
CPC	2.37

Table 5.13: Average rankings of PMCM&PbCA, PMCM&CPC and CPC on categorical datasets.

In addition, we also compare the runtime and the number of patterns for

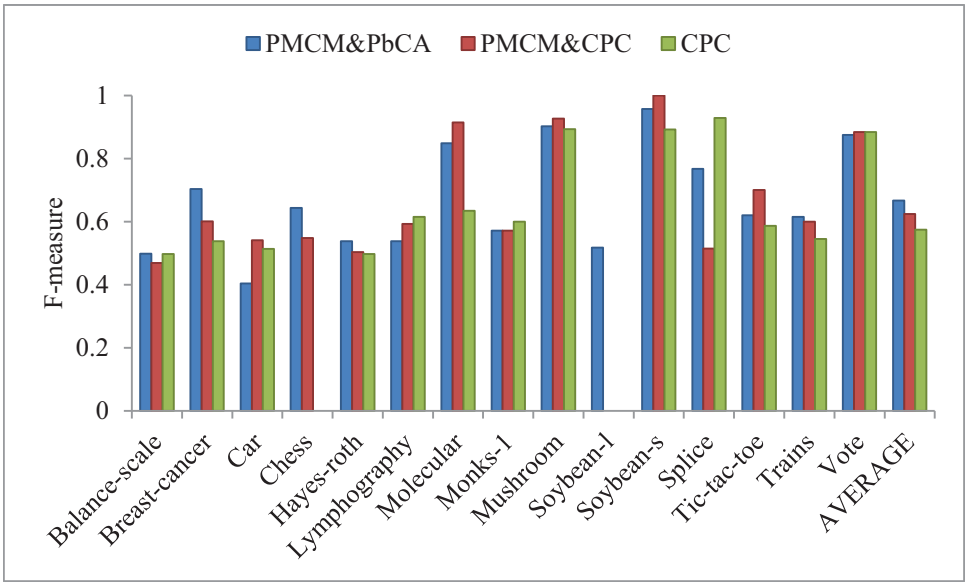

Figure 5.7: Results of F-measure for PMCM&PbCA, PMCM&CPC and CPC on categorical datasets.

the three algorithms. For clarity, in all the comparisons, the datasets are arranged in ascending order according to the number of patterns extracted by FP-growth.

Fig. 5.8 shows the runtime (in seconds) in logarithmic scale for PMCM& CPC, PMCM&PbCA and CPC. Since the clustering algorithm of CPC could not build the clusters in Soybean-l and Chess datasets, the runtime of PMCM &CPC and CPC for these datasets are not shown in Fig 5.8. In the Trains dataset, PMCM&PbCA has a runtime shorter than a tenth of second; therefore, the corresponding bar is too small to be noticed. In average, PMCM&PbCA is 105 times faster than CPC for these categorical datasets, because the average runtime of PMCM&PbCA is 15 seconds while the average runtime of CPC is 1582 seconds. Fig. 5.8 shows that PMCM&PbCA is faster than PMCM&CPC and CPC in 9 of the 15 datasets. PMCM&PbCA was up to 10000 times faster than CPC in the Molecular dataset.

The number of patterns (in logarithmic scale) extracted by PMCM and FP-growth in categorical datasets are shown in Fig. 5.9. In all the datasets, PMCM extracted a less number of patterns than FP-growth, which is desirable in applications that need an explanation of the results, since a less

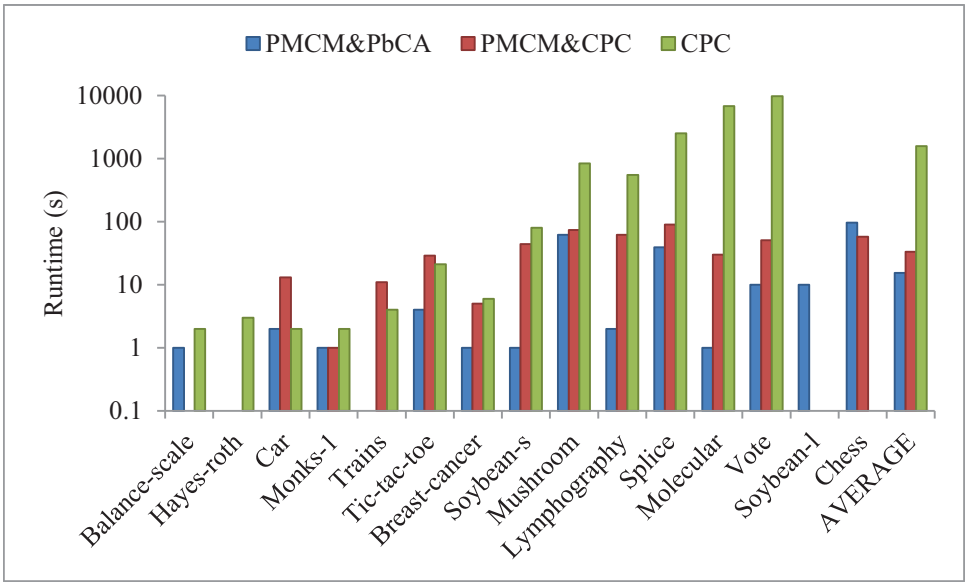

Figure 5.8: Runtime (in seconds) in logarithmic scale for PMCM&PbCA, PMCM&CPC and CPC on categorical datasets.

amount of patterns makes easier the explanation of the results than by using a lot of patterns. In average, PMCM extracts 332 patterns while FP-growth extracts 1150983.

5.3.2 Comparison over numerical datasets

For comparing the results of CPC and PbCA over numerical datasets, the 15 datasets in Table 5.9 are used. The F-measure results for PMCM&PbCA, PMCM&CPC and CPC are shown in Fig. 5.10. Notice that, in this figure, PMCM&CPC y CPC could not cluster the Ecoli dataset.

Table 5.14 shows the average ranking of clustering results according to the Friedman test. The P-value computed by the test is 0.522, this value is not less than, or equal to, 0.05. The differences in the results are not statistically significant, but the patterns extracted with PMCM allow PbCA to obtain the best ranking of F-measure.

Fig. 5.11 shows the runtime (in seconds) in logarithmic scale for PMCM& PbCA, PMCM&CPC and CPC. In Iris, Mammographic and Transfusion, the runtime of CPC are shorter than a tenth of second, thus the corresponding

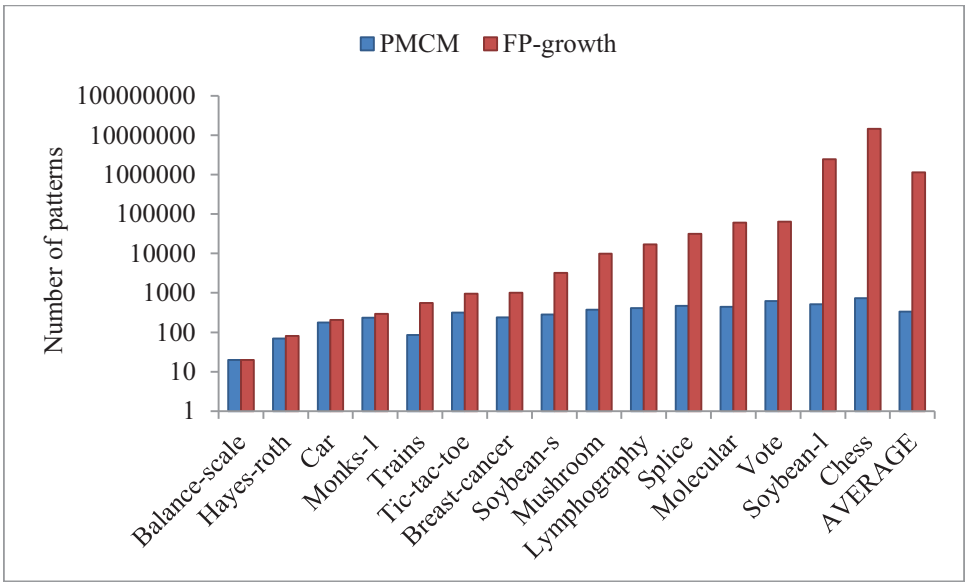

Figure 5.9: Number of patterns in logarithmic scale for PMCM and FP-growth on categorical datasets.

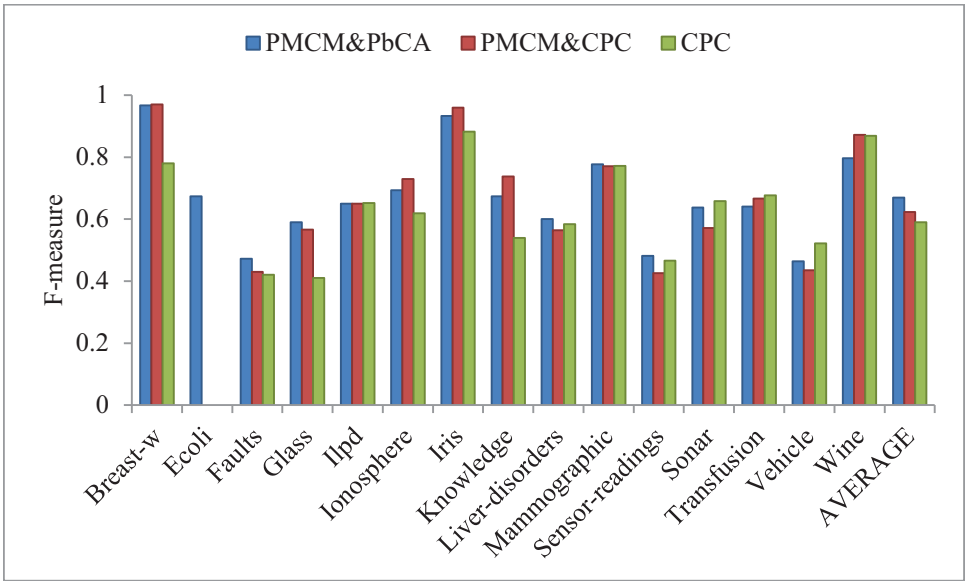

Figure 5.10: Results of F-measure for PMCM&PbCA, PMCM&CPC and CPC on numerical datasets.

Algorithm	Ranking
PMCM&PbCA	1.7
PMCM&CPC	2.1
CPC	2.2

Table 5.14: Average rankings of PMCM&PbCA, PMCM&CPC and CPC on numerical datasets.

bars are not visible in Fig. 5.11. As we have already mentioned, Ecoli could not be clustered by CPC. Since the average runtime of PMCM&PbCA is 75 second and the average runtime of CPC is 1015 seconds, PMCM&PbCA is 13 times faster than CPC for these numerical datasets. Although the runtime of CPC is the shortest in most of the datasets, in Ionosphere the runtime of CPC is close to 10000 seconds, while the runtime of PMCM&PbCA never reach 1000 seconds. In numerical datasets, since CPC uses the FP-growth algorithm for mining patterns, the datasets must be discretized. This discretization usually produces discretized features with fewer different values than the original numerical features, and consequently less patterns are extracted. Therefore, in this experiment, CPC is faster than PMCM&PbCA.

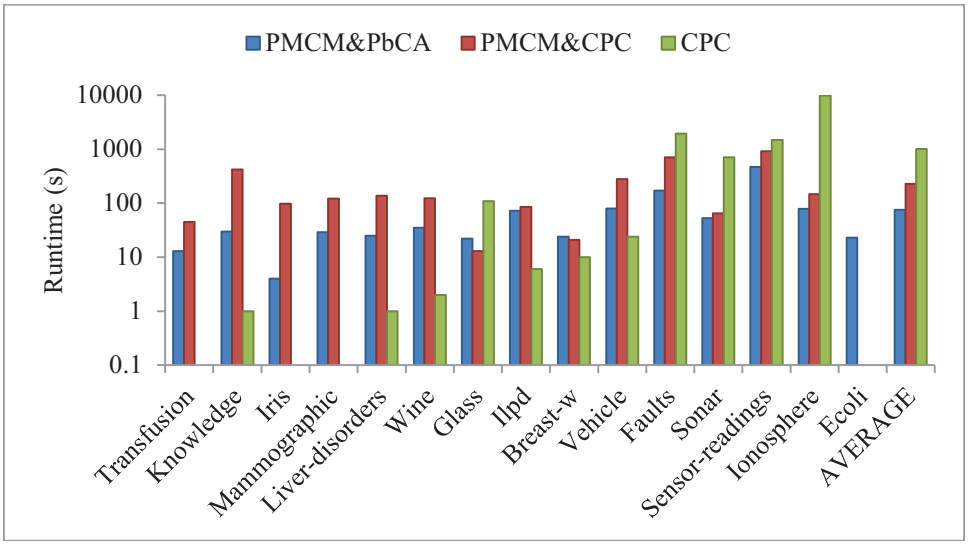

Figure 5.11: Runtime (in seconds) in logarithmic scale for PMCM&PbCA, PMCM&CPC and CPC on numerical datasets.

The number of patterns (in logarithmic scale) extracted by PMCM and

FP-growth on numerical datasets is shown in Fig. 5.12. In these datasets, FP-growth extracted a smaller number of patterns Than PMCM in 6 of the 15 datasets; this happened because of the a priori discretization, used by FP-Growth, highly reduces the number of different values for each numerical feature. Thus, CPC is faster than PbCA for some numerical datasets. On the other hand, in this experiment, in some numerical datasets (Faults, Sonar, Sensor-readings and Ionosphere) FP-growth extracts more than 10000 patterns, while PMCM only extracts around 1000.

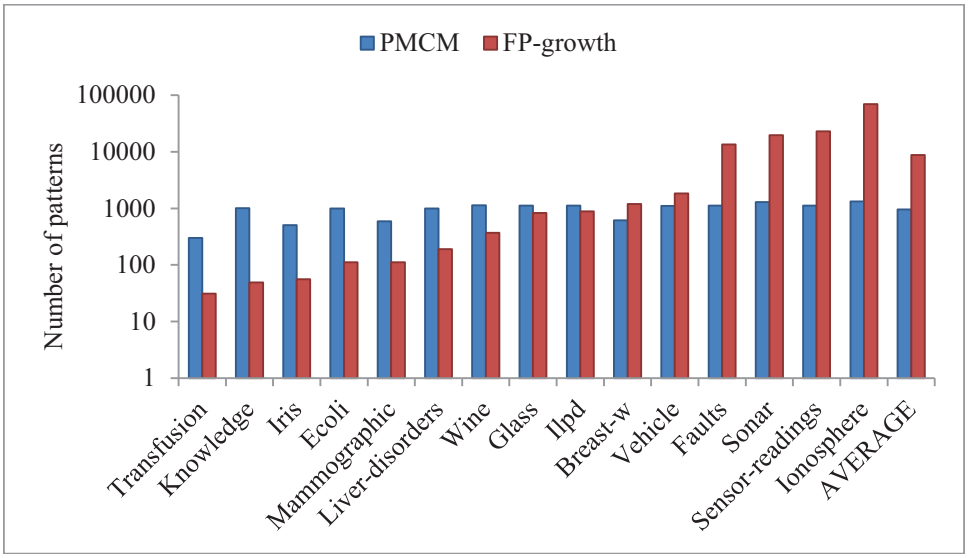

Figure 5.12: Number of patterns in logarithmic scale for PMCM&PbCA, PMCM&CPC and CPC on numerical datasets.

5.3.3 Comparison over mixed datasets

This experiment compares the results of PbCA and CPC over mixed datasets, using the 15 datasets shown in Table 5.11. Fig. 5.13 shows the F-measure results for PMCM&PbCA, PMCM&CPC and CPC. Notice that, in Fig. 5.13, PMCM&CPC could not cluster the datasets Bridges and Flags, while CPC could not cluster Flags.

The average ranking of the clustering results according to the Friedman test are shown in Table 5.15. The P-value computed by the test is 0.012, this value is less than 0.05. It means that the differences among the results are

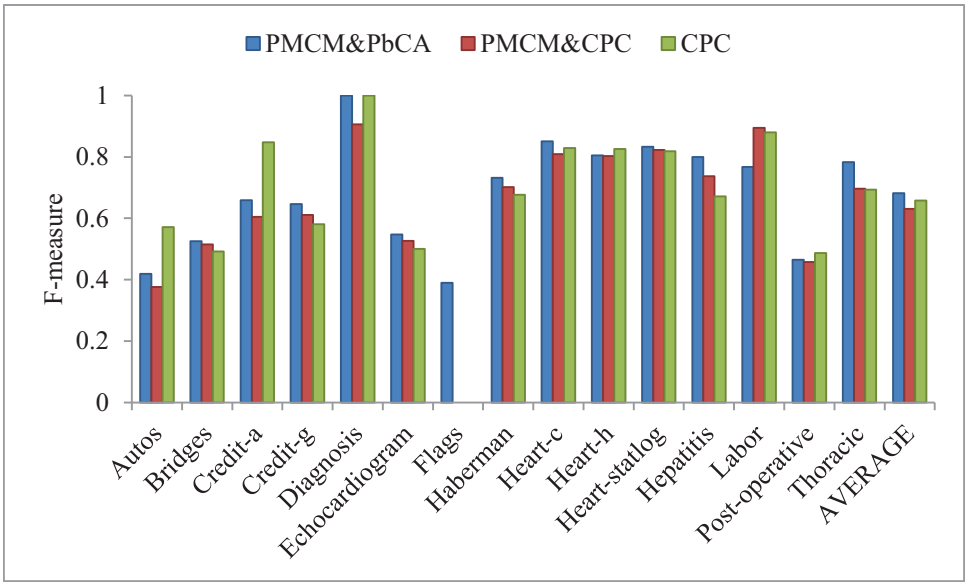

Figure 5.13: Results of F-measure for PMCM&PbCA, PMCM&CPC and CPC on mixed datasets.

statistically significant. Therefore, the Bergmann-Hommel test is performed with significance level of 0.1, to determine which clustering results are significantly different. The results of the Bergmann-Hommel test are shown in Table 5.16. As we can see in Table 5.16, PMCM&PbCA is statistical significantly better than PMCM&CPC and CPC in mixed datasets.

Algorithm	Ranking
PMCM&PbCA	1.4
CPC	2.2
PMCM&CPC	2.4

Table 5.15: Average rankings of PMCM&PbCA, PMCM&CPC and CPC on mixed datasets.

Fig. 5.14 shows the runtime (in seconds) in logarithmic scale for PMCM& PbCA, PMCM&CPC and CPC. In this experiment, as already we mentioned, CPC cannot cluster the Flags dataset. Due to the average runtime of PMC&PbCA is 11 seconds while the average runtime of CPC is 1100 seconds, PMCM&PbCA is, in average, 100 times faster than CPC. PMCM&PbCA is faster than PMCM&CPC and CPC in 11 of the 15 datasets, and sometimes

Algorithms	Adjusted P-value
PMCM&PbCA vs PMCM&CPC	0.014
PMCM&PbCA vs CPC	0.027
PMCM&CPC vs CPC	0.536

Table 5.16: Adjusted P-values of the Bergmann-Hommel test for PMCM&PbCA, PMCM&CPC and CPC on mixed datasets.

(Thoracic, Hepatitis and Credit-g) PMCM&PbCA is up to 1000 times faster than CPC.

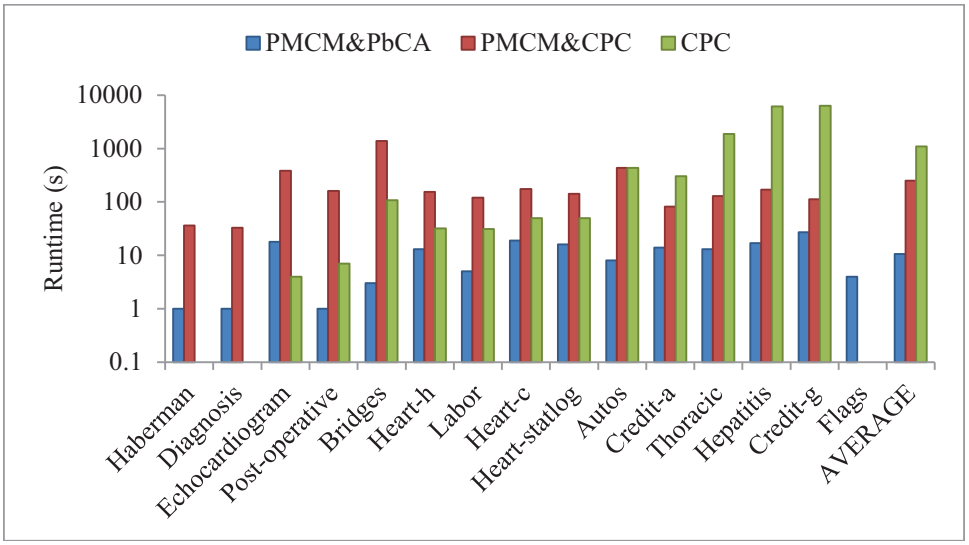

Figure 5.14: Runtime (in seconds) in logarithmic scale for PMCM&PbCA, PMCM&CPC and CPC on mixed datasets.

Fig. 5.15 shows the number of patterns in logarithmic scale extracted by PMCM and FP-growth from mixed datasets. In these datasets, PMCM extracts a higher number of patterns than FP-growth on Diagnosis, Echocardiogram and Haberman. In the remaining datasets, PMCM extracts less patterns than FP-growth with 634 patterns in average, while FP-growth extracts 70076 patterns in average.

From these experiments, we can conclude that our proposed pattern-based clustering algorithm obtains better clustering results than CPC in a shorter time. In addition, PbCA returns a small set of patterns than CPC, which is desirable in situations when an explanation of the results is needed.

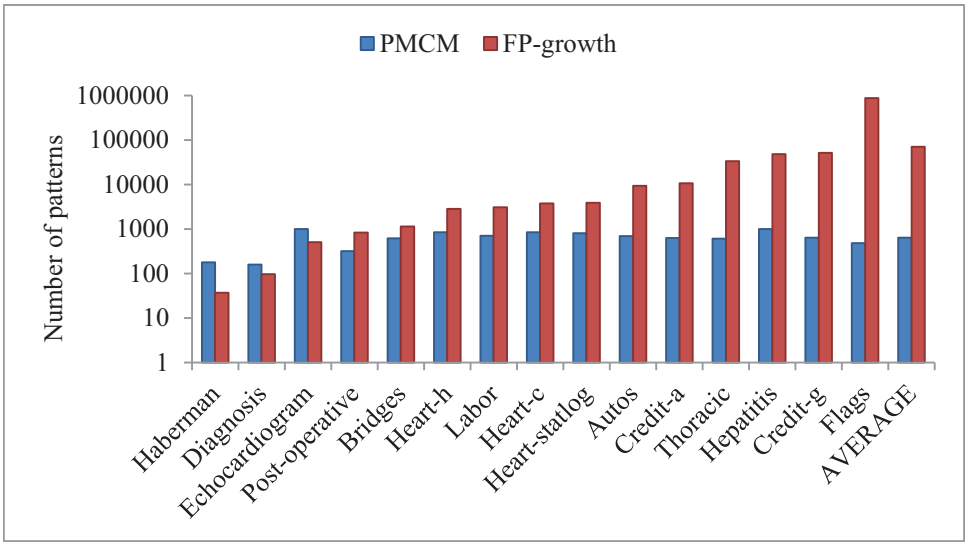

Figure 5.15: Number of patterns in logarithmic scale for PMCM&PbCA, PMCM&CPC and CPC on mixed datasets.

5.4 Comparing PbCA against other clustering algorithms

This section compares the results of our proposed pattern-based clustering algorithms against other state-of-the-art pattern-based clustering algorithms (including CPC), and against traditional (non pattern-based) clustering algorithms. In our experiments, the results of PbCA using PMCM for extracting patterns, against CPC (Fore and Dong, 2012), COBWEB (Fisher, 1987), CLUS (Blockeel H., 1998), K-means (MacQueen, 1967) and EM (Moon, 1996) are compared. This section is divided in three experiments, each one for each type of datasets (categorical, numerical and mixed). In order to evaluate the quality of the results, a comparison between all the algorithms in terms of F-measure is performed.

5.4.1 Comparison over categorical datasets

We compare the clustering results of PbCA against CPC, COBWEB, CLUS, K-means and EM over the 15 categorical datasets shown in Table 5.7. Fig. 5.16 shows the F-measure results of all algorithms.

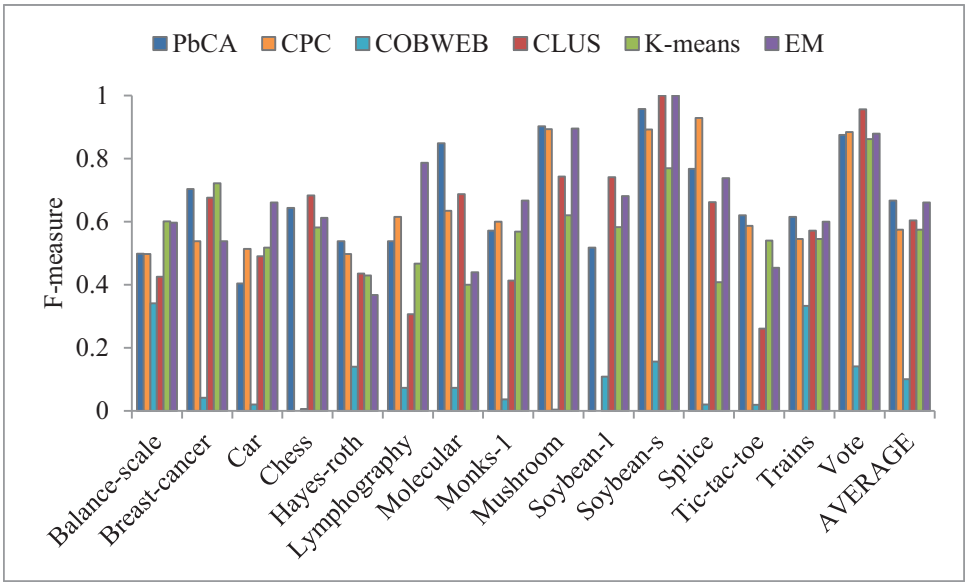

Figure 5.16: Results of F-measure for PbCA, CPC, COBWEB, CLUS, K-means and EM on categorical datasets.

Table 5.17 shows the average ranking of F-measure according to the Friedman test. The P-value computed by the test is 0.000003, this value is less than 0.05. It means that the differences between the results are statistically significant. Therefore, the Bergmann-Hommel test is performed; the results appear in Table 5.18. From this test, we can conclude that COBWEB is significantly the worst algorithm in comparison with the remaining clustering algorithms on categorical datasets, while among the rest of the algorithms there are not statistical significant differences. In this experiment, PbCA obtains the best ranking according to the Friedman test.

5.4.2 Comparison over numerical datasets

For comparing PbCA against CPC, COBWEB, CLUS, K-means and EM over numerical datasets, the 15 numerical datasets shown in Table 5.9 are selected. Fig. 5.17 shows the F-measure results for all the algorithms.

Table 5.19 shows the average ranking of clustering results according to the Friedman test. The P-value computed by the test is 0.0000006, this value is less than 0.05. The differences between the results of the algorithms are

Algorithm	Ranking
PbCA	2.40
EM	2.63
CLUS	3.17
CPC	3.23
K-means	3.70
COBWEB	5.87

Table 5.17: Average rankings of PbCA, CPC, COBWEB, CLUS, K-means and EM on categorical datasets.

Algorithms	Adjusted P-value
PbCA vs COBWEB	0.000006
COBWEB vs EM	0.000022
COBWEB vs CLUS	0.000542
CPC vs COBWEB	0.000811
COBWEB vs K-means	0.010609
PbCA vs K-means	0.570399
K-means vs EM	0.710520
PbCA vs CPC	1.335073
PbCA vs CLUS	1.335073
CPC vs EM	1.335073
CLUS vs EM	1.335073
CLUS vs K-means	1.739869
CPC vs K-means	1.739869
PbCA vs EM	1.739869
CPC vs CLUS	1.739869

Table 5.18: Adjusted P-values of the Bergmann-Hommel test for PbCA, CPC, COBWEB, CLUS, K-means and EM on categorical datasets.

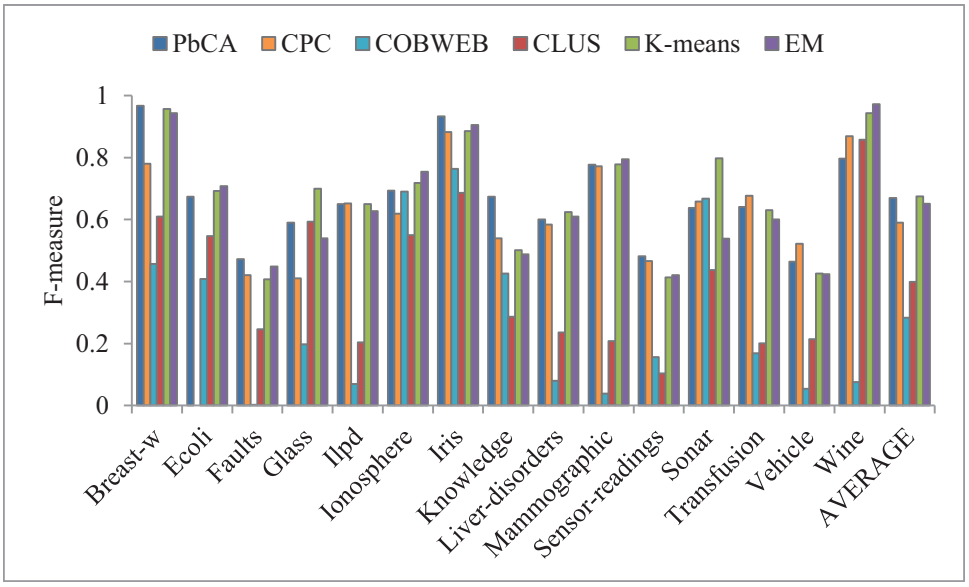

Figure 5.17: Results of F-measure for PbCA, CPC, COBWEB, CLUS, K-means and EM on numerical datasets.

statistically significant. The results of the Bergmann-Hommel test appears in Table 5.20. From this test, we can conclude that COBWEB and CLUS are significantly the worst algorithms in numerical datasets; while there are not statistical significant differences between the remaining algorithms. However, PbCA and K-means obtained the best ranking value.

Algorithm	Ranking
PbCA	2.37
K-means	2.37
EM	2.73
CPC	3.27
CLUS	4.99
COBWEB	5.27

Table 5.19: Average rankings of PbCA, CPC, COBWEB, CLUS, K-means and EM on numerical datasets.

Algorithms	Adjusted P-value
PbCA vs COBWEB	0.0003
COBWEB vs K-means	0.0003
PbCA vs CLUS	0.0012
CLUS vs K-means	0.0012
COBWEB vs EM	0.0015
CLUS vs EM	0.0036
CPC vs COBWEB	0.0205
CPC vs CLUS	0.0447
PbCA vs CPC	1.3138
CPC vs K-means	1.3138
CPC vs EM	1.3138
PbCA vs EM	2.3658
K-means vs EM	2.3658
COBWEB vs CLUS	2.3658
PbCA vs K-means	2.3658

Table 5.20: Adjusted P-values of the Bergmann-Hommel test for PbCA, CPC, COBWEB, CLUS, K-means and EM on categorical datasets.

5.4.3 Comparison over mixed datasets

This experiment compares the clustering results of PbCA against CPC, COBWEB, CLUS, K-means and EM over the 15 mixed datasets shown in Table 5.11. Fig. 5.18 shows the F-measure results for all the algorithms.

Table 5.21 shows the average ranking of the clustering results according to the Friedman test. The P-value computed by the test is 0.000000005, this value is less than 0.05. It means that the differences between the results are statistically significant. Therefore, the Bergmann-Hommel test is performed; these results appear in Table 5.22. From this test, we can conclude that COBWEB and CLUS are significantly the worst algorithms for mixed datasets. Among the rest of the algorithms there are not statistical significant differences. In this experiment, PbCA obtained the best ranking value.

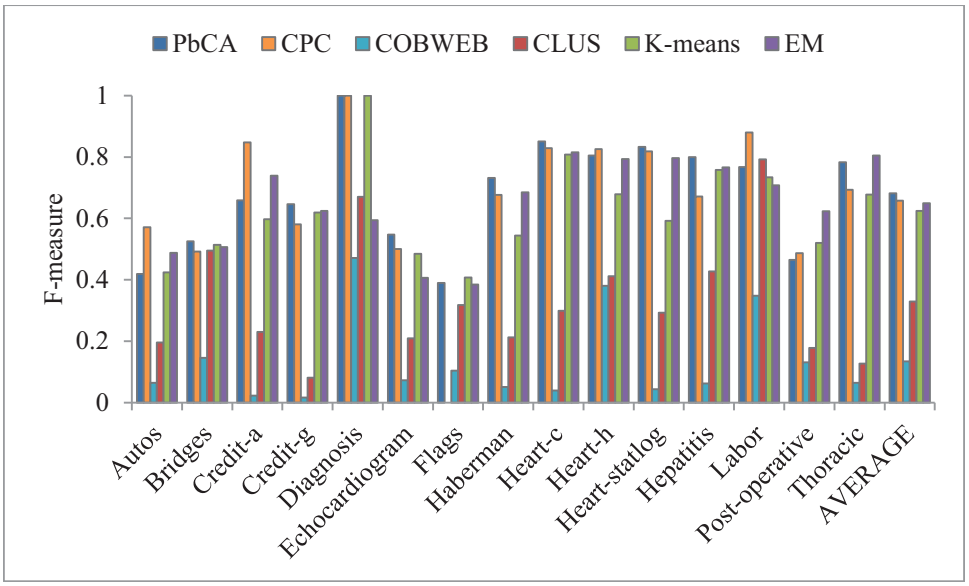

Figure 5.18: Results of F-measure for PbCA, CPC, COBWEB, CLUS, K-means and EM on mixed datasets.

Algorithm	Ranking
PbCA	1.93
CPC	2.67
EM	2.73
K-means	3.13
CLUS	4.60
COBWEB	5.93

Table 5.21: Average rankings of PbCA, CPC, COBWEB, CLUS, K-means and EM on mixed datasets.

Algorithms	Adjusted P-value
PbCA vs COBWEB	0.00000007
CPC vs COBWEB	0.00001736
COBWEB vs EM	0.00001966
COBWEB vs K-means	0.00029073
PbCA vs CLUS	0.00094772
CPC vs CLUS	0.02791918
CLUS vs EM	0.02791918
CLUS vs K-means	0.12717944
COBWEB vs CLUS	0.35673355
PbCA vs K-means	0.47389547
PbCA vs EM	0.72469976
PbCA vs CPC	0.72469976
CPC vs K-means	1.48357400
K-means vs EM	1.48357400
CPC vs EM	1.48357400

Table 5.22: Adjusted P-values of the Bergmann-Hommel test for PbCA, CPC, COBWEB, CLUS, K-means and EM on mixed datasets.

5.5 Concluding remarks

From our experiments, we can conclude that the proposed miners obtain good results with 20, 40, 60, 80 or 100 trees, but the best results are obtained with 40 trees. Generating less trees decrease the clustering results (in terms of F-measure), while more trees do not improve the F-measure values but increase the runtime. Our proposed pattern-based clustering algorithm, PbCA, obtains its best results using the patterns extracted by our miner PMCM. In our experiments, PbCA obtains significantly better clustering results than other pattern-based clustering algorithms like CPC, COBWEB and CLUS. Our PbCA, in terms of F-measure, is in average 11% better than CPC, 76% better than COBWEB, and 34% better than CLUS. In addition, PbCA is around 100 times faster than CPC, the closest algorithm to our proposal. Moreover, in average, PMCM extracts 640 patterns, while FP-growth extracts 409922. Thus, the clustering results of PMCM&PbCA uses, in average, 186 patterns for describing each cluster; while CPC requires, in average, 119010 patterns per cluster. Finally, PbCA obtains competitive results in comparison to traditional clustering algorithms like K-means and EM.

Chapter 6

Conclusions

Pattern-based clustering algorithms aim to build a set of clusters together with an explanation of the clustering results in terms of the features used to describe the data. This is very useful in applications where users need some explanation about clustering results together with the list of objects for each cluster. In these applications, it is also desirable that the algorithms return accurate results in a short time, with just a few patterns for describing the clusters. However, most state-of-the-art pattern-based clustering algorithms have a high computational cost for both extracting patterns and building clusters. To avoid this, some pattern-based clustering algorithms, like COBWEB, build only one pattern for each cluster, discarding several patterns that could be useful for obtaining better clustering results. In contrast, other pattern-based clustering algorithms, as CPC, use pattern mining algorithms that produces a huge amount of patterns, which leads to a high computational cost at the clustering stage. Moreover, in most pattern-based clustering algorithms, numerical features must be a priori discretized, since they are designed for working with categorical features.

In this thesis, we introduced three pattern mining algorithms for categorical, numerical and mixed datasets respectively. These miners extract patterns from a collection of diverse unsupervised decision trees created through new induction procedures combined with a diversity generation strategy based on RandomForest. For the induction procedure, we introduce a new candidate split quality measure for categorical and numerical features. From the induced trees, we extract only a small number of patterns. Additionally, we introduce a new pattern-based clustering algorithm. Our pattern-based clustering algorithm evaluates the relationship between patterns, extracted

by the proposed miners, in terms of the objects covered by each pattern. Then, a clustering of the patterns is built by using a special modification of the K-means algorithm. Finally, each object is assigned to the pattern cluster having the highest fraction of patterns covering it. Therefore, in addition to the list of objects belonging to each cluster, the proposed algorithm returns a set of patterns that describes each cluster.

6.1 Conclusions

Regarding to our proposed miners, based on our experimental results, we can conclude that:

- From our experiments, we can conclude that the proposed miners obtain similar (good) results with 20, 40, 60, 80 or 100 trees, but the best results are obtained with 40 trees. Generating less than 20 trees (for example 10 or 5) highly decrease the quality of the clusters (evaluated through F-measure). On the other hand, values larger than 100 do not improve the quality of the clusters obtained with 40 trees, but increase the runtime.

- Transforming categorical features into numerical features, to mine patterns and grouping them through pattern-based clustering algorithms designed for numerical features, obtains worse clustering results than using pattern-based clustering algorithms designed for categorical data.

- Applying an a priori discretization of the numerical features, in order to mine patterns and grouping them with pattern-based clustering algorithms designed for categorical features, obtains worse clustering results than using pattern-based clustering algorithms designed for numerical data.

- The patterns extracted by PMCM allow CPC (the closest pattern-based clustering algorithm to our proposal) to obtain better clustering results than using those patterns extracted by FP-growth (the miner used by CPC).

- PMCM commonly obtains a small subset of patterns useful for clustering, which produces an easier explanation of the results. The clustering results of PMCM&PbCA involves, in average, 186 patterns for each

cluster. On the other hand, CPC provides, in average, 119010 patterns per cluster.

Regarding to our proposed pattern-based clustering algorithm, we can conclude that:

- By using the patterns extracted by PMCM, in average PbCA is around 100 times faster than CPC.
- The clustering results obtained by PbCA, in terms of F-measure, are better than the results obtained by state-of-the-art pattern-based clustering algorithms like CPC, COBWEB and CLUS. Specifically, PbCA in average is 11% better than CPC, 76% better than COBWEB, and 34% better than CLUS.
- PbCA obtains competitive results, in terms of F-measure, when compared against traditional (non pattern-based) clustering algorithms like K-means and EM, but our algorithm additionally produces a set of patterns for each cluster as an explanation of the clustering results.

6.2 Contributions

The contributions of this PhD research are:

1. We introduce three pattern mining algorithms which extract patterns useful for clustering:
    - PMCC (Pattern Mining algorithm for Clustering Categorical datasets using unsupervised decision trees), which mines, from categorical datasets, only a subset of high-quality patterns useful for clustering; instead of extracting all patterns from the dataset as FP-growth does.
    - PMCN (Pattern Mining algorithm for Clustering Numerical datasets), which extracts a subset of patterns (instead of extracting all patterns) from numerical datasets without applying an a priori discretization of numerical features. The extracted patterns are useful for clustering.

- PMCM (Pattern Mining algorithm for Clustering Mixed datasets), which combines the advantages of the first two proposed miners, in order to extract a subset of patterns useful for clustering mixed (categorical and numerical) datasets, without transforming a priori the dataset.

2. We also propose a new Pattern-based Clustering Algorithm (PbCA), which allows clustering datasets faster and more accurately than other state-of-the-art pattern-based clustering algorithms, by using the subset of patterns obtained by the proposed miners.

6.3 Future work

As future work, we will focus on adapting the proposed pattern mining algorithms to deal with datasets containing missing values, large datasets, or noisy datasets, among others. In addition, we will work on using our proposed algorithms on real-world applications like text processing, web mining and bioinformatics. These applications are a challenge for pattern-based clustering algorithms, since these applications commonly have a lot of features.

On the other hand, for our pattern-based clustering algorithm, we will focus on building clusters of patterns by using a different clustering algorithm, not based on K-means, that can build an undetermined number of clusters. This could be very useful in those problems where the number of clusters is not known a priori.

Finally, from our experiments, we can notice that there are a lot of objects partially covered by a great number of patterns, but in our proposal we did not take this issue into account. By using inexact matching between objects and patterns, we will explore how the clustering results could be improved.

6.4 Publications

From this research, the following publications were generated:

- A.E. Gutierrez-Rodríguez, et. al. Mining Patterns for Clustering using Unsupervised Decision Trees. *Intelligent Data Analysis*, Volume 19(6), 2015.

In this paper, we report the pattern mining algorithm for categorical features presented in Section 4.1.1. The patterns extracted by our miner were evaluated by building clusters from them through the clustering stage of the CPC algorithm.

- A.E. Gutierrez-Rodríguez, et. al. Mining Patterns for Clustering on Numerical Datasets using Unsupervised Decision Trees. *Knowledge-Based Systems* 82 (2015) 70−79.

  In this paper, we report the pattern mining algorithm for numerical features presented in Section 4.1.2. The patterns extracted by our miner were also evaluated by building clusters through the clustering stage of the CPC algorithm.

- A.E. Gutierrez-Rodríguez, et. al. Mining Patterns for Clustering on Mixed Datasets using Unsupervised Decision Trees. *In preparation.*

  In this paper, we will report the pattern mining algorithm for mixed datasets presented in Section 4.1.3. Again, we will evaluate the patterns extracted by our miner by using the clustering stage of the CPC algorithm.

- A.E. Gutierrez-Rodríguez, et. al. A new Pattern-based Clustering Algorithm for Mixed Datasets. *In preparation.*

  In this paper, we will report the pattern-based clustering algorithm proposed in Section 4.2.

Bibliography

Adler, A., Elad, M., and Hel-Or, Y. (2013). Probabilistic subspace clustering via sparse representations. *Signal Processing Letters, IEEE*, 20(1):63–66.

Aggarwal, C. C. and Reddy, C. K. (2013). *Data clustering: algorithms and applications*. CRC Press.

Agrawal, R., Gehrke, J., Gunopulos, D., and Raghavan, P. (2005). Automatic subspace clustering of high dimensional data. *Data Mining and Knowledge Discovery*, 11(1):5–33.

Agrawal, R., Mannila, H., Srikant, R., Toivonen, H., Verkamo, A. I., et al. (1996). Fast discovery of association rules. *Advances in Knowledge Discovery and Data Dining*, 12(1):307–328.

Amigó, E., Gonzalo, J., Artiles, J., and Verdejo, F. (2009). A comparison of extrinsic clustering evaluation metrics based on formal constraints. *Information Retrieval*, 12(4):461–486.

Ayaquica-Martínez, I. O., Martínez-Trinidad, J. F., and Carrasco-Ochoa, J. A. (2007). Restricted conceptual clustering algorithms based on seeds. *Computación y Sistemas*, 11(2).

Bache, K. and Lichman, M. (2013). {UCI} Machine Learning Repository.

Bandyopadhyay, S. and Saha, S. (2013). Similarity measures. In *Unsupervised Classification*, pages 59–73. Springer.

Baridam, B. B. and Owolabi, O. (2010). Conceptual clustering of rna sequences with codon usage model. *Global Journal of Computer Science and Technology*, 10(8):41–45.

Basak, J. (2008). Online adaptive clustering in a decision tree framework. In *Pattern Recognition, 2008. ICPR 2008. 19th International Conference on*, pages 1–4. IEEE.

Basak, J. and Krishnapuram, R. (2005). Interpretable hierarchical clustering by constructing an unsupervised decision tree. *IEEE Transactions on Knowledge and Data Engineering*, 17(1):121–132.

Beil, F., Ester, M., and Xu, X. (2002). Frequent term-based text clustering. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 436–442. ACM.

Bisson, G. (1992). Conceptual clustering in a first order logic representation. In *Proceedings of the 10th European conference on Artificial intelligence*, pages 458–462. Citeseer.

Blockeel H., De Raedt L., R. J. (1998). Top-down induction of clustering trees. In *Proceedings of the 15th International Conference on Machine Learning*, pages 55–63.

Boongoen, T., Shang, C., Iam-On, N., and Shen, Q. (2011). Extending data reliability measure to a filter approach for soft subspace clustering. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 41(6):1705–1714.

Breaban, M. and Luchian, H. (2011). A unifying criterion for unsupervised clustering and feature selection. *Pattern Recognition*, 44(4):854–865.

Breiman, L. (2001). Random forests. *Machine learning*, 45(1):5–32.

Carpineto, C. and Romano, G. (1996). A lattice conceptual clustering system and its application to browsing retrieval. *Machine Learning*, 24(2):95–122.

Cha, S.-H. (2012). On integer sequences derived from balanced k-ary trees. In *Proceedings of American Conference on Applied Mathematics*, pages 377–381.

Dash, R., Paramguru, R. L., and Dash, R. (2011). Comparative analysis of supervised and unsupervised discretization techniques. *International Journal of Advances in Science and Technology*, 2(3):1.

Demsar, J. (2006). Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7:1–30.

Dimitrovski, I., Kocev, D., Loskovska, S., and Džeroski, S. (2012). Hierarchical classification of diatom images using ensembles of predictive clustering trees. *Ecological Informatics*, 7(1):19–29.

Duda, R. O., Hart, P. E., and Stork, D. G. (2012). *Pattern classification*. Wiley-interscience.

Ester, M., Kriegel, H.-P., Sander, J., and Xu, X. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. In *Kdd*, volume 96 (34), pages 226–231.

Färber, I., Günnemann, S., Kriegel, H.-P., Kröger, P., Müller, E., Schubert, E., Seidl, T., and Zimek, A. (2010). On using class-labels in evaluation of clusterings. In *MultiClust: 1st International Workshop on Discovering, Summarizing and Using Multiple Clusterings Held in Conjunction with KDD*.

Fisher, D. H. (1987). Knowledge acquisition via incremental conceptual clustering. *Machine Learning*, 2(2):139–172.

Fore, N. and Dong, G. (2012). CPC: A Contrast Pattern Based Clustering Algorithm. In Dong, G. and Bailey, J., editors, *Contrast Data Mining: Concepts, Algorithms, and Applications*, Data Mining and Knowledge Discovery Series, chapter 14, pages 197–216. Chapman & Hall/CRC, United States of America.

Frank, A. and Asuncion, A. (2010). Uci machine learning repository [http://archive. ics. uci. edu/ml]. irvine, ca: University of california. *School of Information and Computer Science*, 213.

Funes, A., Ferri, C., Hernández-Orallo, J., and Ramírez-Quintana, M. J. (2009). An instantiation of hierarchical distance-based conceptual clustering for propositional learning. In *Advances in Knowledge Discovery and Data Mining*, pages 637–646. Springer.

Fung, B. C., Wang, K., and Ester, M. (2003). Hierarchical document clustering using frequent itemsets. In *Proceedings of the SIAM international conference on data mining*, pages 59–70.

García, S. and Herrera, F. (2008). An extension on statistical comparisons of classifiers over multiple data sets for all pairwise comparisons. *Journal of Machine Learning Research*, 9(2677-2694):66.

Garcia, S., Luengo, J., Sáez, J. A., López, V., and Herrera, F. (2013). A survey of discretization techniques: Taxonomy and empirical analysis in supervised learning. *Knowledge and Data Engineering, IEEE Transactions on*, 25(4):734–750.

Gennari, J. H., Langley, P., and Fisher, D. (1989). Models of incremental concept formation. *Artificial intelligence*, 40(1):11–61.

Günnemann, S., Boden, B., and Seidl, T. (2011). Db-csc: a density-based approach for subspace clustering in graphs with feature vectors. In *Machine Learning and Knowledge Discovery in Databases*, pages 565–580. Springer.

Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., and Witten, I. H. (2009). The weka data mining software: an update. *ACM SIGKDD Explorations Newsletter*, 11(1):10–18.

Han, J., Pei, J., Yin, Y., and Mao, R. (2004). Mining frequent patterns without candidate generation: A frequent-pattern tree approach. *Data Mining and Knowledge Discovery*, 8(1):53–87.

Hanson, S. J. and Bauer, M. (1989). Conceptual clustering, categorization, and polymorphy. *Machine Learning*, 3(4):343–372.

Hinneburg, A. and Keim, D. A. (1998). An efficient approach to clustering in large multimedia databases with noise. In *KDD*, volume 98, pages 58–65.

Hofmann, T. and Buhmann, J. M. (1997). Pairwise data clustering by deterministic annealing. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 19(1):1–14.

Hotho, A., Staab, S., and Stumme, G. (2003). Explaining text clustering results using semantic structures. In *Knowledge Discovery in Databases: PKDD 2003*, pages 217–228. Springer.

Jain, A. K. (2010). Data clustering: 50 years beyond k-means. *Pattern Recognition Letters*, 31(8):651–666.

Jänichen, S. and Perner, P. (2005). Acquisition of concept descriptions by conceptual clustering. In *Machine Learning and Data Mining in Pattern Recognition*, pages 153–162. Springer.

Jonyer, I., Cook, D. J., and Holder, L. B. (2002). Graph-based hierarchical conceptual clustering. *The Journal of Machine Learning Research*, 2:19–43.

Karaboga, D. and Ozturk, C. (2011). A novel clustering approach: Artificial bee colony (abc) algorithm. *Applied Soft Computing*, 11(1):652–657.

Kaufman, L. and Rousseeuw, P. J. (2005). Finding groups in data: an introduction to cluster analysis.

Kosters, W. A., Pijls, W., and Popova, V. (2003). Complexity analysis of depth first and fp-growth implementations of apriori. In *Machine Learning and Data Mining in Pattern Recognition*, pages 284–292. Springer.

Kriegel, H.-P., Kröger, P., and Zimek, A. (2009). Clustering high-dimensional data: A survey on subspace clustering, pattern-based clustering, and correlation clustering. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 3(1):1.

Kryszkiewicz, M. and Skonieczny, Ł. (2006). Hierarchical document clustering using frequent closed sets. In *Intelligent Information Processing and Web Mining*, pages 489–498. Springer.

Kuncheva, L. I. (2004). *Combining pattern classifiers: methods and algorithms*. John Wiley & Sons.

Li, Y., Chung, S. M., and Holt, J. D. (2008). Text document clustering based on frequent word meaning sequences. *Data & Knowledge Engineering*, 64(1):381–404.

Liang, C. and Forbus, K. D. (2014). Constructing hierarchical concepts via analogical generalization. In *Proceedings of the Cognitive Science Society*.

Lisi, F. A. (2006). A pattern-based approach to conceptual clustering in fol. In *Conceptual Structures: Inspiration and Application*, pages 346–359. Springer.

Liu, B., Xia, Y., and Yu, P. S. (2000). Clustering through decision tree construction. In *Proceedings of the ninth international conference on Information and knowledge management*, pages 20–29. ACM.

Liu, Q. and Dong, G. (2012). Cpcq: Contrast pattern based clustering quality index for categorical data. *Pattern Recognition*, 45(4):1739–1748.

MacQueen, J. (1967). Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1(14), pages 281–297. Oakland, CA, USA.

Makhoul, J., Kubala, F., Schwartz, R., Weischedel, R., et al. (1999). Performance measures for information extraction. In *Proceedings of DARPA broadcast news workshop*, pages 249–252.

Malik, H. H. and Kender, J. R. (2006). High quality, efficient hierarchical document clustering using closed interesting itemsets. In *Data Mining, 2006. ICDM'06. Sixth International Conference on*, pages 991–996. IEEE.

Malik, H. H. and Kender, J. R. (2008). Instance driven hierarchical clustering of document collections. In *Proceedings of local patterns to global models workshop (ECML/PKDD)*.

Michalski, R. S., Kaufman, K. A., Pietrzykowski, J., Wojtusiak, J., Mitchell, S., and Seeman, D. (2006). Natural induction and conceptual clustering: a review of applications. *Reports of the Machine Learning and Inference Laboratory*, 1051:06–3.

Michalski, R. S. and Stepp, R. E. (1983). Automated construction of classifications: Conceptual clustering versus numerical taxonomy. *IEEE Transactions on Pattern Analysis and Machine Learning*, 5(4):396–410.

Mineau, G. W. and Godin, R. (1995). Automatic structuring of knowledge bases by conceptual clustering. *Knowledge and Data Engineering, IEEE Transactions on*, 7(5):824–829.

Mishra, N., Ron, D., and Swaminathan, R. (2004). A new conceptual clustering framework. *Machine Learning*, 56(1-3):115–151.

Moon, T. K. (1996). The expectation-maximization algorithm. *IEEE Signal Processing Magazine*, 13(6):47–60.

Morik, K., Kaspari, A., Wurst, M., and Skirzynski, M. (2012). Multi-objective frequent termset clustering. *Knowledge and Information Systems*, 30(3):715–738.

Ozdal, M. M. and Aykanat, C. (2004). Hypergraph models and algorithms for data-pattern-based clustering. *Data Mining and Knowledge Discovery*, 9(1):29–57.

Pandit, S., Gupta, S., et al. (2011). A comparative study on distance measuring approaches for clustering. *International Journal of Research in Computer Science*, 2(1):29–31.

Patel, V. M., Nguyen, H. V., and Vidal, R. (2013). Latent space sparse subspace clustering. In *Computer Vision (ICCV), 2013 IEEE International Conference on*, pages 225–232. IEEE.

Peeters, R. (2003). The maximum edge biclique problem is np-complete. *Discrete Applied Mathematics*, 131(3):651–654.

Pei, J., Zhang, X., Cho, M., Wang, H., and Yu, P. S. (2003). Maple: A fast algorithm for maximal pattern-based clustering. In *Third IEEE International Conference on Data Mining, 2003. ICDM 2003.*, pages 259–266. IEEE.

Pelleg, D., Moore, A. W., et al. (2000). X-means: Extending k-means with efficient estimation of the number of clusters. In *ICML*, pages 727–734.

Peng, T. and Liu, L. (2015). A novel incremental conceptual hierarchical text clustering method using cfu-tree. *Applied Soft Computing*, 27:269–278.

Perkowitz, M. and Etzioni, O. (1999). Adaptive web sites: Conceptual cluster mining. In *IJCAI*, volume 99, pages 264–269. Citeseer.

Perner, P. and Attig, A. (2010). Fuzzy conceptual clustering. *Quality and Reliability Engineering International*, 26(8):909–922.

Quinlan, J. R. (1993). *C4. 5: programs for machine learning*, volume 1. Morgan kaufmann.

Ralambondrainy, H. (1995). A conceptual version of the k-means algorithm. *Pattern Recognition Letters*, 16(11):1147–1157.

Robardet, C. and Feschet, F. (2001). Effcient local search in conceptual clustering. In *Discovery Science*, pages 323–335. Springer.

Roberts, S. J., Holmes, C., and Denison, D. (2001). Minimum-entropy data clustering using reversible jump markov chain monte carlo. In *Artificial Neural NetworksICANN 2001*, pages 103–110. Springer.

Romero-Zaliz, R., Rubio-Escudero, C., Cordón, O., Harari, O., del Val, C., and Zwir, I. (2006). Mining structural databases: an evolutionary multiobjetive conceptual clustering methodology. In *Applications of Evolutionary Computing*, pages 159–171. Springer.

Rosenberg, A. and Hirschberg, J. (2007). V-measure: A conditional entropy-based external cluster evaluation measure. In *EMNLP-CoNLL*, volume 7, pages 410–420. Citeseer.

Schmidt, J., Ansorge, S., and Kramer, S. (2012). Scalable induction of probabilistic real-time automata using maximum frequent pattern based clustering. In *SDM*, pages 272–283. SIAM.

Schölkopf, B., Smola, A., and Müller, K.-R. (1998). Nonlinear component analysis as a kernel eigenvalue problem. *Neural computation*, 10(5):1299–1319.

Scott, D. W. (2009). *Multivariate density estimation: theory, practice, and visualization*, volume 383. John Wiley & Sons.

Shi, J. and Malik, J. (2000). Normalized cuts and image segmentation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(8):888–905.

Soltanolkotabi, M., Elhamifar, E., Candes, E. J., et al. (2014). Robust subspace clustering. *The Annals of Statistics*, 42(2):669–699.

Struyf, J. (May 5, 2015). *http://dtai.cs.kuleuven.be/clus/*.

Tiddi, I., dAquin, M., and Motta, E. (2014). Dedalo: Looking for clusters explanations in a labyrinth of linked data. In *The Semantic Web: Trends and Challenges*, pages 333–348. Springer.

Voggenreiter, O., Bleuler, S., Gruissem, W., et al. (2012). Exact biclustering algorithm for the analysis of large gene expression data sets. *BMC Bioinformatics*, 13(S18):A10.

Wang, H., Wang, W., Yang, J., and Yu, P. S. (2002). Clustering by pattern similarity in large data sets. In *Proceedings of the 2002 ACM SIGMOD international conference on Management of data*, pages 394–405. ACM.

Webb, A. R. (2003). *Statistical pattern recognition*. John Wiley & Sons.

Winters-Hilt, S. and Merat, S. (2007). Svm clustering. *BMC bioinformatics*, 8(Suppl 7):S18.

Wong, A. K. and Li, G. C. (2008). Simultaneous pattern and data clustering for pattern cluster analysis. *IEEE Transactions on Knowledge and Data Engineering*, 20(7):911–923.

Xia, Y. and Xi, B. (2007). Conceptual clustering categorical data with uncertainty. In *19th IEEE International Conference on Tools with Artificial Intelligence, 2007. ICTAI 2007.*, volume 1, pages 329–336. IEEE.

Yang, Y. and Padmanabhan, B. (2005). Ghic: A hierarchical pattern-based clustering algorithm for grouping web transactions. *IEEE Transactions on Knowledge and Data Engineering*, 17(9):1300–1304.

Yu, H., Searsmith, D., Li, X., and Han, J. (2004). Scalable construction of topic directory with nonparametric closed termset mining. In *Data Mining, 2004. ICDM'04. Fourth IEEE International Conference on*, pages 563–566. IEEE.

Zhang, W., Yoshida, T., Tang, X., and Wang, Q. (2010). Text clustering using frequent itemsets. *Knowledge-Based Systems*, 23(5):379–388.

Zhao, Q. and Fränti, P. (2014). Wb-index: A sum-of-squares based index for cluster validity. *Data & Knowledge Engineering*.

Zheng, H.-T., Chen, H., and Gong, S.-Q. (2014). A frequent term-based multiple clustering approach for text documents. In *Web Technologies and Applications*, pages 602–609. Springer.