Pattern Comparison in Data Mining: a survey

Irene Ntoutsi Nikos Pelekis
Yannis Theodoridis
Database Group, Department of Informatics
University of Piraeus, Greece
{ntoutsi, npelekis, ytheod}@unipi.gr

Abstract

A huge amount of patterns is available nowadays due to the wide spreading of the Knowledge Discovery in Databases (KDD), as a result of the overwhelming amount of data. This "flood" of patterns imposes new challenges regarding their management. Pattern comparison, which aims at evaluating how close to each other two patterns are, is one of these challenges resulting in a variety of applications. In this work we investigate issues regarding the pattern comparison problem and present an overview of the work performed so far in this domain. Due to heterogeneity of data mining patterns, we focus on the most popular pattern types, namely frequent itemsets and association rules, clusters and clusterings, and decision trees.

1 Introduction

Nowadays a huge quantity of raw data is collected from different application domains (business, science, telecommunication, health care systems etc.). According to Lyman and Varian [37], "The world produces between 1 and 2 exabytes of unique information per year, which is roughly 250 megabytes for every man, woman, and child on earth". Due to their quantity and complexity, it is impossible for humans to thoroughly investigate these data collections directly. Knowledge Discovery in Databases (KDD) and Data Mining (DM) provides a solution to this problem by generating compact and rich in semantics representations of raw data, called *patterns* [47]. With roots in machine learning, statistics and pattern recognition, KKD aims at extracting valid, novel, potentially useful and ultimately understandable patterns from data [19]. Several pattern types exist in the literature mainly due to the wide heterogeneity of data and the different techniques for pattern extraction as a result of the different goals that a mining process tries to achieve (i.e. what data characteristics the mining process highlights). Frequent itemsets (and their extension, association rules), clusters (and their grouping, clusterings) and decision trees are among the most well known pattern types in data mining.

Due to the spreading of the DM technology nowadays, even the amount of patterns extracted from heterogeneous data sources is huge and hard to be managed by humans. Of course, patterns do not raise from the DM field only; signal processing, information retrieval and mathematics are among the fields that also "yield" patterns. The new reality imposes new challenges and requirements regarding the management of patterns in correspondence to the management of traditional raw data. These requirements have been recognized by both the academic and the industrial parts that try to deal with the problem of efficient and effective pattern management [10] including, among others, modeling, querying, indexing, and visualization issues.

Among the several interesting operations on patterns, one of the most important is that of *comparison*, i.e. evaluating how similar two patterns are. As an application example, consider a supermarket that is interested in discovering changes in its customers behavior over the last two months. For the supermarket owner, it is probably more important to discover what has changed over time in its customers behavior rather than to preview some more association rules on this topic. This is the case in general; the more familiar an expert becomes with data mining the more interesting becomes for her to discover changes rather than already known patterns. A similar example stands also in case of a distributed mining environment, where one might be interested in discovering what differentiates the distributed branches with respect to each other or in grouping together branches of similar patterns. From the latter, another application of similarity arises, that of exploiting similarity between patterns for meta-pattern management, i.e. applying data mining techniques over patterns instead of raw data like in [55].

So far, the importance of defining similarity operators between patterns has been justified. However, this definition is not so straightforward. At first, there are a lot of different pattern types like association rules, frequent itemsets, decision trees, clusters etc., so similarity operators should be defined for each pattern type. Secondly, except for patterns of the same pattern type an interesting extension would be the comparison between patterns of different pattern types, e.g. a cluster with a decision tree (extracted from the same raw data set). Furthermore, an important aspect is that of examining whether similarity between patterns reflects in some degree the similarity between the original raw data. From an efficiency point of view this is desirable, since the pattern space is usually of lower size and complexity.

In the next sections we overview the work performed so far in the area of data mining patterns comparison. Due to the wide spreading of the data mining pattern types we focus mainly on three basic pattern types that have been used extensively in KDD literature, namely frequent itemsets and association rules, clusters and clusterings, and decision trees.

The paper is organized as follows. In Section 2, we present the above mentioned basic pattern types in detail. In Section 3, we overview the work regarding the comparison of frequent itemsets and association rules. In Section 4, we focus on decision trees comparison. In Section 5, we present the related work regarding the comparison of clusters and clusterings. Next, in Section 6, we present the general frameworks for the comparison/monitoring of data mining patterns that have been appeared in the literature. Finally, we conclude our work in Section 7.

2 Data Mining Patterns

According to [47], patterns can be defined as compact and rich in semantics representations of raw data; *compact* by means that they summarize in some degree the amount of information contained in the original raw data and *rich in semantics* by means that they reveal new knowledge hidden in the huge amount of raw data.

A variety of pattern types exists in the literature due to the heterogeneity of the raw data from which patterns are extracted and the different goals that each mining task tries to accomplish. Different pattern types highlight different characteristics of the raw data; for example, *frequent itemsets* capture the correlations between attribute values, *clusters* reveal natural groups in the data whereas *decision trees* detect characteristics that predict (with respect to a given class attribute) the behavior of future records [24].

In [22] Ganti et al introduced the *2-component property* of patterns. The central idea of their work is that a broad class of pattern types (called models in authors' vocabulary) can be described in terms of a *structural component* and of a *measure component*. The structural component identifies "interesting regions", whereas the measure component summarizes the subset of the data that is mapped to each region. In other words, the structural component describes the pattern space, whereas the measure component quantifies, in some way, how well the pattern space describes the underlying raw data space.

The *2-component property* of patterns has been extended in [47], where authors introduced a general model for patterns including also a *source component* that describes the data set from which patterns have been extracted and an *expression component* that describes the relationship between the source data space and the pattern space. We refer to the 2-component property of patterns since, as will be shown from the related work, most of the similarity measures exploit these components.

In the following subsections we present three popular data mining pattern types that are relevant to this work, namely frequent itemsets (and their extensions, association rules), clusters (and their groupings, clusterings), and decision trees.

2.1 Frequent Itemsets and Association Rules

Frequent itemsets and association rules mining are strongly related to each other by means that frequent itemsets mining is the first step towards association rules mining. In this section we present more detail on both of them.

The **Frequent Itemset Mining (FIM)** problem is a core problem in many data mining tasks, although it was first introduced in the context of market basket analysis towards mining for relationships between sets of items in terms of customers purchases. To define the FIM problem we will follow the work by Agrawal et al [45]: Let I be a set of distinct items and D be a database of transactions where each transaction T contains a set of items $T \subseteq I$. A set $X \subseteq I$ with $|X| = k$ is called k-itemset or simply itemset. The frequency of X in D, equals to the number of transactions in D that contain X, i.e. $fr_D(X) = |\{T \in D : X \subseteq T\}|$. The percentage of transactions in D that contain X, is called *support* of X in D, i.e. $supp_D(X) = \frac{fr_D(X)}{D}$. An itemset X is called *frequent* if its support is greater than or equal to a user-specified minimum support threshold σ called *minSupport*, $supp_D(X) \geq \sigma$. The FIM problem is defined as finding all itemsets X in D that are frequent with respect to a given *minSupport* threshold σ. Let $F_\sigma(D)$ be the set of frequent itemsets extracted from D under minSupport threshold σ.

The set of frequent itemsets forms the itemset lattice L in which the lattice property holds: an itemset is frequent iff all of its subsets are frequent. The lattice property allows as enumerating all frequent itemsets using more compact representations like closed frequent and maximal frequent itemsets.

A frequent itemset X is called *closed* if there exists no frequent superset $Y \supseteq X$ with $supp_D(X) = supp_D(Y)$. Let $C_\sigma(D)$ be the set of closed frequent itemsets extracted from D under minSupport threshold σ. By definition, $C_\sigma(D)$ is a lossless representation of $F_\sigma(D)$ since both the lattice structure (i.e. frequent itemsets) and lattice measure (i.e. their supports) can be derived from $CFIs$. On the other hand, a frequent itemset is called *maximal* if it is not a subset of any other frequent itemset. Let $M_\sigma(D)$ be the set of maximal frequent itemsets extracted from D under minSupport threshold σ. Unlike $C_\sigma(D)$, $M_\sigma(D)$ is a lossy representation of $F_\sigma(D)$ since it is only the lattice structure (i.e. frequent itemsets) that can be determined from $MFIs$ whereas frequent itemsets supports are lost [56]. Practically, $CFIs$ can be orders of magnitude less than FIs, and $MFIs$ can be orders of magnitude less than $CFIs$ [56].

Recalling the *2-component property* of patterns we can say that in case of frequent itemsets, the *structure component* consists of the itemset itself, i.e. items that form it, whereas the *measure component* consists of itemset support.

The **Association Rules Mining (ARM)** problem was first introduced in [45] motivated mainly by the market basket analysis domain and could be defined as follows: Let D be a database of transactions, where each transaction consists of a set of distinct items I, called itemsets. An association rule is a implication of the form $X \rightarrow Y$, where $X \subseteq I$, $Y \subseteq I$ and $X \cap Y = \emptyset$ (X and Y are itemsets). The rule is associated with a *support* s and a *confidence* c. The rule $X \rightarrow Y$ is said to have support s, if $s\%$ of the transactions in D contain $X \cup Y$, whereas it is said to have confidence c, if $c\%$ of the transactions in D that contain X also contain Y.

The Association Rules Mining problem consists of two steps. In the first step the set of frequent itemsets is calculated which is then used as input to the second step where the association rules are finally extracted. So, association rules provide some additional information than frequent itemsets.

Recalling the *2-component property* of patterns we can say that in case of association rules, the *structure component* consists of the left hand side (also called head) and the right hand side (also called body) whereas the *measure component* consists of rule confidence and support.

2.2 Decision Trees

Decision trees (DT), first introduced by Hunt et al [18], are commonly used for classification due to their intuitive representation that makes them easy to understand by humans.

In this section, we provide some basic concepts on decision trees (DT) following the work of Mitchell [42]. DTs are used to classify instances by sorting them down to the tree from the root to some *leaf node*, which provides the classification of the instance. Each *internal node* of the tree specifies a test of some attribute of the instance w.r.t. to some of its values, and each branch descending from that node corresponds to one of the possible values for this attribute.

A *leaf node* corresponds to problem classes with some weight factor, which depends on the amount of instances that follow the path down to the tree and fall into this class. In the worst case, for each leaf node there is a weight associated with all classes. In the simple case, however, each leaf corresponds to only one class (actually this is the case of practical use).

More formally, let D be a set of problem instances to be classified. Let $A_1, A_2, \ldots, A_m$ be the attributes on which classification will be based (*predictor attributes*), where attribute A_i has domain $D(A_i)$. Let C be the *class attribute*, i.e. the attribute to be predicted with domain $D(C) = \{C_1, C_2, \ldots, C_k\}$, where k is the number of classes. A decision tree T over D provides a classification of D instances into the classes based on tests over the predictor attributes.

Predictor attributes might be either numeric, categorical or ordinal. Recall that in a *numerical attribute* the domain is ordered (e.g. age, income), in a *categorical or nominal attribute* the domain is a finite set without any natural ordering (e.g. colors, gender), whereas in a *ordinal attribute* the domain is ordered, but absolute differences between values is unknown (e.g. preference scale). Usually, numerical attributes are discretized and treated as categorical attributes.

According to FOCUS [22], a DT partitions the raw data space into a set of regions. Each leaf node of the tree corresponds to one region and furthermore each region is associated with a set of measures, each measure corresponding to the fraction of problem instances that result in this region for some of the problem classes.

Recalling the *2-component property* of patterns, we can say that in case of a decision tree, the *structure component* consists of a set of regions (one per leaf node of the tree), whereas the *measure component* consists of a set of measures associated with these regions (in the worst case a leaf node contains k measures, i.e. one per class).

2.3 Clusters and Clusterings

Clustering is the unsupervised classification of data into natural groups (called **clusters**) so that data points within a cluster are more similar to each other than to data points in other clusters [28]. The term *unsupervised* stands for the fact that there is no a priori knowledge about the partition of the data. In a more formal definition, we can state that a clustering C_l is the partition of a data set D into sets $C_1, C_2, , C_K$ called clusters such that $C_i \cap C_j = \emptyset$ and $\cup_{j=1}^{K}(C_j) = D$. This definition stands for *hard clustering*, where a data set instance is associated with only one cluster. A more "relaxed" definition is that of *soft clustering* where an instance is associated with every cluster to a certain extent (or probability) indicated by a weight.

Clustering algorithms are based on some *distance function* that evaluates in which cluster an object should be assigned. There is also an *evaluation function* that evaluates how good the achieved clustering is. For example, minimizing the distance of every data point from the mean of the cluster to which it belongs could be thought of as such a criterion.

Due to its broad application areas, the clustering problem has been studied extensively in many contexts and disciplines including data mining. As a result, a large number of clustering algorithms exists in the literature (see [28] for a survey). In fact, there is not only one correct answer in a clustering problem, but many answers can be found.

Different clustering algorithms proposed in the literature use a variety of cluster definitions. Han and Kamber [27] propose the following categorization for the major clustering methods:

1. *Partitioning methods* that create K partitions of the data (K is defined by the user) where each partition corresponds to a cluster. K-means and K-medoids [27] algorithms belong to this category.

2. *Hierarchical methods* that create a hierarchical decomposition of the data set. Depending on how the hierarchical decomposition is formed, i.e. in a bottom-up or top-down fashion, they are classified into agglomerative and divisive methods correspondingly. In both cases, a distance function between clusters is required and as such minimum, maximum, mean or average distance can be used [27].

3. *Density-based methods* that continue to grow a cluster as long as the density (i.e. number of data points) in its "neighbor" exceeds some threshold. DBScan [27] algorithm belongs to this category.

4. *Grid-based methods* that quantize the object space into a finite number of cells that form a grid structure. STING and CLIQUE [27] algorithms belong to this category.

5. *Model-based methods* that hypothesize a model for each of the clusters and finds the best fit of the data to the given model. Statistical approaches like COBWEB algorithm and neural network approaches are the two major approaches in this category [27].

Recalling the *2-component property* of patterns we can state that in case of clusters this property depends on the definition of the cluster itself. For example, in case of a "partitioning cluster", its structure could be defined by its center and radius as in case of K-means algorithm or by its centroid and radius as in case of K-medoids algorithm. In case of a "hierarchical cluster", its structure could be defined as the set of data points that fall into it. In case of a "density-based cluster", its structure could be defined by the cluster distribution function (i.e. the mean and the standard deviation of the distribution). Regarding the measure component, a possible measure is cluster support (i.e. the percentage of data set records that fall into this cluster). Other measures like the intra-cluster distance within the cluster or the average distance of cluster records from the center or the centroid of the cluster could be used as well.

3 Comparing FIM or ARM results

In this section we present the work performed so far regarding the comparison of FIM results, i.e. sets of frequent itemsets, and ARM results, i.e. association rules. In case of FIM results, related work demonstrates methods that utilize the comparison between sets of frequent itemsets in order to compare the underlying raw data sets. In case of association rules, related work gives emphasis on the temporal aspects of rules, namely the necessary operations to maintain and monitor rules evolution over time. However, these are ad hoc approaches to measure the distance between association rules, as well as incremental techniques to update previously discovered rules.

3.1 Comparing sets of frequent itemsets

In order to define the similarity between frequent itemsets, let us consider two sets of itemsets (or itemsets lattices) A and B, like the ones illustrated in Fig. 1. Each itemset is described through the 2-component property as a pair $< structure, measure >$. Suppose also that both A and B were generated under the same *minSupport* threshold from the data sets D and E respectively which are defined over the same set of items I. The problem we try to deal with is how similar to each other A and B are.

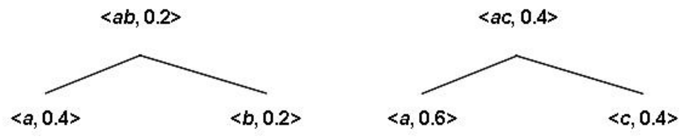

Figure 1: Two lattices of frequent itemsets

Parthasarathy and Ogihara [44] propose the following metric for the comparison of sets of frequent itemsets:

$$dis(A,B) = 1 - \frac{\sum_{X \in A \cap B}(max\{0, 1 - \theta * (supp_D(X) - supp_E(X))\})}{|A \cup B|} \quad (1)$$

where θ is a scaling parameter that is specified by the user and reflects how significant the variations in support are for the user. For $\theta = 0$, the measure component (i.e. support) carries no significance. For $\theta = 1$, the measure component is of equal importance with the structure component.

Recalling the example of Fig. 1, the intersection of the two sets is: $A \cap B = \{a\}$. Assuming $\theta = 1$, $dis(A,B) = 1 - max\{0, 1 - \theta * |0.4 - 0.6|\}/5 = 0.84$, according to Eq. 1.

In fact, the measure of Parthasarathy and Ogihara utilize frequent itemsets comparison for data set comparison. Authors consider that $dis(D,E) = dis(A,B)$ based on the intuition that itemsets indicate the correlations within the data sets to be compared.

Ganti et al [22] propose the FOCUS framework for quantifying the deviation between two data sets D and E in terms of the pattern sets A and B, respectively, they induce. In order to compare the pattern sets authors introduce the notion of *Greatest Common Refinement (GCR)*. The GCR is a kind of refinement of the structure components (recall the *2-component property* of patterns) of the models to be compared (see Section 6 for more detail on FOCUS).

In case of frequent itemsets the GCR of the two sets to be compared is their union. Using absolute difference as the difference function and sum as the aggregation function, an instantiation of the FOCUS framework is as follows:

$$dis_{abs}(A,B) = \sum_{X \in A \cup B}(|supp_D(X) - supp_E(X)|) \quad (2)$$

Since maximum distance occurs when A and B are totally different, the normalized (in $[0 \ldots 1]$) distance can be defined as:

$$dis(A,B) = \frac{\sum_{X \in A \cup B}(|supp_D(X) - supp_E(X)|)}{\sum_{X \in A \cup B}(supp_D(X) + supp_E(X))} \quad (3)$$

Especially for the frequent itemsets case, authors provide an upper bound regarding the distance between the two pattern sets which does not require re-querying the original raw data space. In this case, if an itemset X appears in one of the pattern sets, e.g. in A, FOCUS considers that it also appears in B with support equal to 0 without re-querying the original data set E from which B has been extracted.

Recalling the example of Fig. 1, $GCR(A,B)$ is given below in the form: $< X, supp_D(X), supp_E(X) >$ for each itemset X belonging to $A \cup B$: $GCR(A,B) = \{< a : 0.4, 0.6 >, < b : 0.2, 0 >, < c : 0, 0.4 >, < ab : 0.2, 0 >, < ac : 0, 0.4 >\}$. Hence, $dis(A,B) = (|0.4-0.6| + |0.2-0| + |0-0.4| + |0.2-0| + |0-0.4|)/(|0.4-0| + |0.2-0| + |0.2-0| + |0.6-0| + |0.4-0| + |0.4-0|) = 1.4/2.2 = 0.64$, according to Eq. 3.

As with the measure of Parthasarathy and Ogihara presented above, FOCUS also utilizes frequent itemsets comparison for data comparison. Authors justify that using pattern comparison for data comparison is meaningful since the interesting characteristics of the data sets are captured by the induced pattern models.

Li et al [32] exploit the dissimilarity between two sets of MFIs in order to compare the data sets from which MFIs have been extracted. If $A = \{X_i, supp_D(X_i)\}$ and $B = \{X_i, supp_E(Y_j)\}$, X_i, Y_j are the MFIs in D, E respectively, then their metric is defined as follows:

$$dis(A, B) = 1 - \frac{2 * I_3}{I_1 + I_2} \tag{4}$$

where
$I_1 = \sum_{i,j} \frac{|X_i \cap X_j|}{|X_i \cup X_j|} * \log(1 + \frac{|X_i \cap X_j|}{|X_i \cup X_j|}) * \min(supp_D(X_i), supp_D(X_j))$,
$I_2 = \sum_{i,j} \frac{|Y_i \cap Y_j|}{|Y_i \cup Y_j|} * \log(1 + \frac{|Y_i \cap Y_j|}{|Y_i \cup Y_j|}) * \min(supp_E(Y_i), supp_E(Y_j))$,
$I_3 = \sum_{i,j} \frac{|X_i \cap Y_j|}{|X_i \cup Y_j|} * \log(1 + \frac{|X_i \cap Y_j|}{|X_i \cup Y_j|}) * \min(supp_D(X_i), supp_E(Y_j))$

I_3 can be considered as a measure of "mutual information" between A and B; the term $|X_i \cup Y_j|/|X_i \cap Y_j|$ represents the fraction of itemsets in common, whereas the fraction $2/|I_1 + I_2|$ serves as a normalization factor.

Recalling the example of Fig. 1 and applying Eq. 4 it arises that $dis(A, B) = 0.58$.

Once more, the pattern space, MFIs in this case, has been exploited towards evaluating similarity in raw data space. Authors support that using MFIs is meaningful since MFIs encapsulate the information regarding the associations among the data sets.

Concluding, we can state that the similarity measures between sets of itemsets have been introduced in order to evaluate the similarity between the underlying raw data sets and not per se. All of these measures, make use of the structure (i.e. items that form an itemset) and of the measure components (i.e. the support of an itemset) of frequent itemsets. The measures of Parthasarathy - Ogihara and FOCUS are based on some kind of $1-1$ matching between itemsets (i.e. an itemset of the first set is matched to only one itemset of the second set and an "optimal", according to some criteria, score is calculated) whereas the measure of Li et al utilizes $N-M$ matching (all itemsets of the first set are matched to all itemsets of the second set and an "average" score is calculated). Comparing the Parthasarathy - Ogihara and FOCUS measures we can state that the measure of Parthasarathy - Ogihara is based on the intersection of the two sets to be compared ($A \cap B$), whereas the measure of FOCUS framework also make use of the itemsets that appear in the difference sets ($A - B, B - A$).

All of these measures could be expressed as instantiations of the PANDA framework [8] (see Section 6 for more details on PANDA). Following the PANDA terminology, a frequent itemsets could be considered as a simple pattern, whereas a set of frequent itemsets could be considered as a complex pattern. Several similarity measure configurations might arise within the PANDA framework for the FI comparison case, by just instantiating the following parameters: (a) how the similarity between two frequent itemsets (simple patterns) is evaluated (b) how the simple patterns of the two sets (complex patterns) are matched and (c) how the scores of the matched patterns are aggregated into an overal similarity score.

3.2 Comparing association rules

Comparison of association rules can be defined on rules features such as support, confidence or their bit-vector representations. These direct features are very limited in capturing the interaction of rules on the data and characterize only a single rule.

Toivonen et al [51] proposed a first approach of defining distance between rules based on the overlap of their market-baskets. More specifically, the authors define the distance between two rules $X \Rightarrow Z$ and $Y \Rightarrow Z$ as the amount of rows where the rules differ:

$$\begin{aligned} dis(X \Rightarrow Z, Y \Rightarrow Z) &= |\frac{m(XZ) \cup m(YZ)}{m(XYZ)}| \\ &= |m(XZ)| + |m(YZ)| - 2|m(XYZ)| \end{aligned} \tag{5}$$

where $m(X)$ is the number of matching rows for attribute set X. The problem with this metric is that it grows as the number of market-baskets in the database increases.

In [26] authors argue that this can be corrected by normalization (i.e. dividing the measure by the size of the database). However, the measure is still strongly correlated with support, as high support rules will on average tend to have higher distances to everybody else. For example, two pairs of rules, both pairs consisting of non-overlapping rules, may have different distances. This is an undesired property. Furthermore, high support pairs have a higher distance than low support pairs.

As an improvement to this metric, Gupta at al [26] proposed a new distance measure based on a conditional probability estimate:

$$dis_{i,j} = P(\bar{BS_i} \vee \bar{BS_j} | BS_i \vee BS_j) = 1 - \frac{|m(BS_i, BS_j)|}{|mBS_i||mBS_j| - |mBS_i, BS_j|} \tag{6}$$

where the set BS_i is the union of items in the left and right hand sides of rule i, and $m(X)$ is the set of all transactions containing itemset X. This distance function called the *Conditional Market-Basket Probability (CMPB)*, results to a distance of 1 for rules having no common MBs, while rules valid for an identical set of baskets are at a distance of 0.

A specialized approach regarding similarity between association rules was proposed in [31] where the authors consider the problem of clustering association rules of the form $A \cap B \Rightarrow C$ where the left-hand side attributes (A and B) are quantitative while the right-hand side attribute (C) is categorical. A segmentation is defined as the collection of all clustered association rules for a specific value C. Having as input a set of two-attribute association rules over binned data, the methodology forms a two-dimensional grid where each axis corresponds to one of the left-hand side attributes. All the corresponding association rules for a specific value of the right-hand side attribute are plotted on this grid. The goal is to find the minimum number of clusters that cover the association rules within this grid. The authors introduce a series of algorithms to form clusters of adjacent association rules in the grid.

A similar line of research concentrates on the statistical properties of rules by considering their lifetime, meaning the time in which they are sufficiently supported by the data. When data is continuously collected over a long period, the concepts reflected in the data change over time. This requires the user to monitor the discovered rules continuously. Except the well-known but application dependent solution of an appropriate partitioning scheme, formal methods have been proposed for application independent partitioning of data. In [11], the authors focus on the identification of valid time intervals for previously discovered association rules. They propose a methodology that finds all adjacent time intervals during which a specific association holds, and furthermore all interesting periodicities that a specific association has.

Such temporal aspects of rules are also taken into account in the rule monitor of [2, 34, 33]. In [2], upward and downward trends in the statistics of rules are identified using an SQL-like query mechanism. In [34], the authors count the significant rule changes across the temporal axis. They pay particular attention on rules that are "stable" over the whole time period, i.e. do not exhibit significant changes, and contradict them with rules that show trends of noteworthy increase or decrease. In [33], Liu et al study the discovery of "fundamental rule changes". More analytically, they detect changes on support or confidence between two successive timepoints by applying a X^2-test.

In [4, 5, 7, 6] Byron and Spiliopoulou introduced Pattern Monitor (PAM), a framework for efficiently maintaining data mining results. PAM builds on a temporal rule model. More specifically, the temporal representation of the patterns follow the Generic Rule Model (GRM) presented in [4, 5], where a rule R is a temporal object with signature:

$$R = ((ID, query, body, head), \{(timestamp, statistics)\}) \quad (7)$$

ID is an identifier, ensuring that rules with the same body (antecedent) and head (consequent) have the same ID. The query is the data mining query, while the statistics depend on the rule type. Support, confidence and certainty factor of association rules are statistics considered in this work. Based on this representation scheme the authors introduce a *change detector mechanism*, a mechanism for identifying changes to a rules statistic which exhibit a particular strength. Statistical significance is used to assess the strength of pattern changes. (see more detail on PAM in Section 6.)

Recently, a series of methods having intrinsic the notion of similarity have emerged and focus on maintaining and updating previously discovered knowledge, thus being able to deal with dynamic data sets. Such a popular approach is that of incremental mining in which the knowledge about already extracted association rules is re-used. Updating association rules was first introduced in

[12, 15]. These approaches as well as subsequent ones are based on the abstract framework of that the problem of updating association rules can be solved by maintaining the large itemsets. In this initial approach the authors proposed the Fast Update (FUP) algorithm (the framework of which is similar to that of Apriori and DHP) for computing the large itemsets in the updated database. Furthermore, optimization techniques for reducing the size of the database as well as the pool of candidate itemsets during the update process are discussed. In [13] the FUP algorithm was generalized for handling insertions to and deletions from an existing set of transactions.

Subsequently, Ayan et al [3] proposed the Update With Early Pruning (UWEP) algorithm, which employs a dynamic look-ahead pruning strategy in updating the existing large itemsets by detecting and removing those that will no longer remain large after the contribution of the new set of transactions. In [48, 49], the concept of negative border [50], is used to compute the new set of large itemsets when new transactions are added to or deleted from the database.

As inferred from the previous discussion, there are three main lines of research regarding the similarity issue between association rules: (a) ad hoc solutions which can only be applied to special forms of association rules (b) time-oriented rule monitoring and (c) incremental approaches which focus on identifying change instead of measuring it. Naturally, an approach combining the above characteristics is an emerging research issue in the domain.

4 Comparing Decision Trees

Ganti et al [22] argue that the difference between different decision tree models is quantified as the amount of work required to transform one model decision tree models is quantified as the amount of work required to transform one model into the other, which is small if the two models are "similar" to each other, and high if they are "different". According to FOCUS [22], the decision tree induced by a data set identifies a set of regions. Each region is described through a set of attributes (structure component) and corresponds to a set of raw data (measure component). If the structures extracted from the data sets to be compared are identical, then the deviation between the data sets equals to their measures deviation.

More specifically, the authors define the 2-component property of a decision tree model M as $\langle \Gamma_M, \Sigma(\Gamma_M, D) \rangle$ where $\Gamma_M = \{1 \leq i \leq l\}$ is the set of regions defined as a subset of the attribute space, $\Sigma(\Gamma_M, D) = \{\sigma(\gamma_M^i, D) : \gamma_M^i \in \Gamma_M\}$ and $\sigma(\gamma_M^i, D)$ is the selectivity of the region γ_M^i (the fraction of tuples in data set D that correspond to this region). So, when the structural components of the two models M_1 and M_2 are identical ($\Gamma_{M1} = \Gamma_{M2}$) then, the amount of work for transforming $\Sigma(\Gamma_{M1}, D_1)$ into $\Sigma(\Gamma_{M2}, D_2)$ is the aggregate of the differences between $\sigma(\gamma_{M1}^i, D_1)$ and $\sigma(\gamma_{M2}^i, D_2)$, $i = 1, ..., |\Gamma_{M1}|$. The difference, at a region, between the measures of the first and the second models is given by a *difference function* (not necessarily the usual difference operator "-"), and that the aggregate of the differences is given by an aggregate function. If f is a difference function and g is an aggregate function, the formal definition of the deviation when the structural components of the two models are identical is:

$$\delta^1_{f,g}(M_1, M_2) = g(\{f(\kappa^1_{D1}, \kappa^1_{D2}, |D_1|, |D_2|), ..., f(\kappa^l_{D1}, \kappa^l_{D2}, |D_1|, |D_2|)\}) \quad (8)$$

where l denotes the number of regions and (for $j \in 1, 2$), $\kappa^i_{Dj} = \sigma(\gamma_i, D_j) * |D_j|$ denotes the absolute number of tuples in D_j that are mapped into $\gamma^i_{Mj} \in \Gamma Mj$.

In the general case, however structures differ and thus a first step is required to make them identical by "extending" them to their GCR. This extension involves splitting regions until they became identical. Then the measures components for each region are computed either directly or by querying back the raw data space. In this case the deviation of the two models is defined as:

$$\begin{aligned}\delta_{f,g}(M_1, M_2) = \delta^1_{f,g}(&\langle \Gamma_{GCR(M1,M2)}, \Sigma(\Gamma_{GCR(M1,M2),D_1})\rangle, \\ &\langle \Gamma_{GCR(M1,M2)}, \Sigma(\Gamma_{GCR(M1,M2),D_2})\rangle)\end{aligned} \quad (9)$$

Another approach to quantify the deviation between two data sets D_1 and D_2 is to find how well does a decision tree model M induced by the first data set represent the second data set. To estimate this deviation, in [9, 36, 35] the authors utilize the notion of *misclassification error*, which in the case of decision trees corresponds to the fraction of tuples in a data set that a decision tree misclassifies. In this particular case, let C be the class label predicted by M for tuple $t \in D_2$. If the true class of t is different from C then t is said to be misclassified by M.

Thus the overall misclassification error $ME^M(D_2)$ of M with respect to D_2 is given by the following equation:

$$ME^M(D_2) = \frac{|\{t \in D_2 \wedge M misclassifies \quad t\}|}{|D_2|} \quad (10)$$

An additional methodology that can be employed to measure differences between two data sets is that of the *chi-squared metric*. A prerequisite for the utilization of this metric is the ability to partition the attribute space into a grid consisting of disjoint regions. This requirement, which is the base of the FOCUS framework [22], is met by decision trees and has been utilized in [14] as a means to indicate how the chi-squared metric describes whether two data sets have the same characteristics. More specifically, the chi-squared metric for D_1 and D_2 is given by the subsequent equation:

$$X^2(D_1, D_2) = \sum i = 1^n |D_2| \frac{(m_{\gamma i}(D_1) - m_{\gamma i}(D_2))^2}{m_{\gamma i}(D_1)} \quad (11)$$

where $r_1, \ldots, r_n$ is the grid of disjoint regions and $m_r(D_i)$ is the measure of a region r with respect to D_i.

Similarly with the incremental approaches for maintaining and updating association rules, techniques that adjust decision trees inducted by dynamic data sets have been also proposed. Such an approach introducing a series of tree-restructuring methods was firstly presented in [52, 53]. The disadvantage of these techniques is that it is assumed that the entire database fits in main memory and as such the subject of scalability with respect to data size is not addressed.

This issue is handled in [25] where BOAT, an incremental algorithm for the maintenance decision trees is introduced. In case of an ad hoc change, BOAT adjusts a tree in a two-step process. Initially, it classifies the amount of change at a node as *drastic* or *moderate* and depending on this categorization it adjusts the corresponding splitting criterion following different tactics on numerical and categorical criterions. The adjustment of a tree node is based on additional information deliberately kept in the node. Such annotated information concerns a confidence interval of around the point where the splitting occurs, the set of tuples that fall within this interval, a histogram where each bin contains the class distribution of the previously tuples in the range implied by the splitting criterion and a list of the best splitting criterions at that node for the remaining attributes.

The above description shows that the issue of similarity between decision trees is an open research topic as some of the special features of decision trees, basically emanating of their complicated structure, e.g. order of the splitting attributes, are not handled in current efforts.

5 Comparing Clusters and Clusterings

The notion of similarity between two clusters is fundamental in clustering algorithms. In fact the division of a data set into clusters is based on the similarity between clusters, since the goal of clustering is grouping together the most similar data points and assigning dissimilar data points into different clusters.

Several distance measures have been proposed in the literature in order to compare two clusters as a step of cluster generation algorithms. The definition of these measures assumes that we are able to quantify how similar two data points are. In case of numerical data, this distance is usually expressed by some $p - norm$ based distance measure like the well known Manhattan distance (1-norm distance), Euclidean distance (2-norm distance) etc. In case of categorical data, alternative approaches have been exploited like the Jaccard coefficient used by the ROCK algorithm [17]. In this case the similarity between two data points (like for example two customers' transactions) equals to the number of common items of the two transactions divided by all items appearing in both transactions.

Regarding the distance between two clusters, several measures, based on set theory, are utilized by cluster generation algorithms [17]. The *single linkage* distance calculates the smallest distance between an element in one cluster and an element in the other. The *complete linkage* distance calculates the largest distance between an element in one cluster and an element in the other. The *average* distance calculates the average distance between the elements of the two clusters. The *centroid* distance calculates the distance between the centroids of the two clusters (recall here that the centroid of a cluster is the mean of its elements and it is not required to be an actual point in the cluster). The *medoid* distance calculates the distance between the medoids of the two clusters (recall here that the medoid of a cluster is a centrally located element in the cluster).

There are also many paradigms in the literature the exploit the notion of similarity between two clusters in order to perform clusters monitoring or spatial clustering. Among others, we mention the work by Neill et al [43] on a new class of space-time clusters for the rapid detection of emerging clusters demonstrated for indicating emerging disease outbreaks. Their method consists of two parts: time series analysis for computing the expected number of cases for each spatial region on each day and space-time scan statistics for determining whether the actual numbers of cases in some region are significantly higher than expected on the last W days (W is the window size). Towards the same direction is the work by Aggarwal [1] for modeling and detecting spatiotemporal changes in clusters. Clusters are modeled through a kernel function and at each spatial location X the kernel density is computed. For each time point t, two estimates of density change are computed, the backward $t - h_t$ and the forward estimate $t + h_t$ upon a sliding time window. Their difference is the velocity density (or evolutiondensity) of the location X. This model also identifies the data properties that mostly contribute to change.

Furthermore, some of the measures regarding clusterings comparison (to be discussed in the sequel) could also be utilized in this case by considering that a cluster is a clustering of size one, i.e. it contains only one cluster.

Concluding, we can state that several measures for cluster comparison has been proposed in the literature either in the context of clustering algorithms or in the context of cluster monitoring across the time axis. These measures exploit both the data points that belong into the clusters and the features regarding clusters structure like centroids, medoids or density functions. The different measures, however, depend on the clustering algorithm used for the generation of clusters and thus they cannot be applied for the comparison of patterns of arbitrary type.

In the following section we discuss in more detail the comparison of clustering results, a problem that appears quite often in the literature due to its broad range of applications.

5.1 Comparing sets of Clusters

Defining similarity between sets of clusters (i.e. clusterings) results in a variety of applications. Some approaches compare clusterings extracted from the same data set but under different algorithms, thus evaluating the quality of the different clustering algorithms. Other approaches compare clusterings extracted from the same data set under the same algorithm but with different parameters (like for example different K values in case of K-means algorithm) thus evaluating the impact of the parameters on the resulting clusterings.

There are also more generic approaches that compare clusterings extracted from different but homogeneous data sets (i.e. data sets defined over the same attribute space). In this case, the applications are broader. Consider for example, the distributed data mining domain, where it is often required to group together similar clusterings (e.g. group together branches of a supermarket with similar customers profiles so as to apply common marketing strategies to each group). In this case, grouping requires using some similarity function to compare clusterings. Alternatively, consider monitoring of clusterings results over time; in this case the notion of similarity between two clusterings is crucial by means that monitoring is based on being able to quantify how similar two snapshots of the pattern base are (e.g. detect changes in customers profiles of a branch over time).

Generally speaking, we can think of clustering similarity as a way of accessing the "agreement" between two clustering results [38].

5.1.1 Comparing clustering results from the same data set

In order to define the similarity between two clusterings, let us consider a data set D of n data points and two clusterings over D, namely Cl_1, Cl_2, of K_1, K_2 clusters respectively.

Meila [38] provides an overview of the related work on comparing different clustering results produced from the same data set D under different mining parameters (different algorithms or different parameters over the same algorithm).

According to this work, the comparison between two clusterings Cl_1, Cl_2 can be virtually represented through a contingency matrix M of size $K_1 X K_2$, where the $[Cl_{1i}, Cl_{2j}]$ cell contains the number of data points that belong to both clusters C_i (of clustering Cl_1) and C_j (of clustering Cl_2). The different clustering comparison criteria are categorized into three types:

1. Criteria based on counting pairs
2. Criteria based on cluster matching
3. Criteria based on variation of information (VI)

In the next paragraphs we present more detail on each of these types (see a thorough presentation and analysis in [38]).

The first category of criteria, **criteria based on counting pairs**, are based on counting the pair of points on which the two clusterings agree/disagree. Let us define the following quantities:

N_{11} : the number of data point pairs $(x_i, x_j \in D)$ that are clustered in the same cluster under both Cl_1 and Cl_2 clusterings.

N_{00} : the number of data point pairs $(x_i, x_j \in D)$ that are clustered in different clusters under Cl_1 and Cl_2 clusterings.

N_{10} : the number of data point pairs $(x_i, x_j \in D)$ that are clustered in the same cluster under Cl_1 clustering but not under Cl_2 clustering.

N_{01} : the number of data point pairs $(x_i, x_j \in D)$ that are clustered in the same cluster under Cl_2 clustering but not under Cl_1 clustering.

It holds that $N_{11} + N_{00} + N_{10} + N_{01} = n$. Let also n_{k1}, n_{k2} be the number of data points belonging to clusters C_{k1}, C_{k2} respectively and let n_{k1k2} be the number of data points belonging to both C_{k1} and C_{k2} clusters.

The different measures proposed in the literature for this category make use of the above mentioned parameters. We present some characteristic examples (see [38] for a detailed representation):

Wallace [54] proposed the two following asymmetric criteria:

$$W_I(Cl_1, Cl_2) = \frac{N_{11}}{\sum_{C_{k_1} \in Cl_1} (n_{k1} * (n_{k1} - 1)/2)},$$
$$W_{II}(Cl_1, Cl_2) = \frac{N_{11}}{\sum_{Ck_2 \in Cl_2} (n_{k2} * (n_{k2} - 1)/2)} \quad (12)$$

Where W_I (respectively W_{II}) represents the probability that a pair of data points which are in the same cluster under Cl_1 (respectively Cl_2) are also in the same cluster under Cl_2 (respectively Cl_1).

Fowlkes and Mallows [20] introduced the following symmetric criterion which is based on the number of pair of data points clustered together:

$$F(Cl_1, Cl_2) = \sqrt{W_I(Cl_1, Cl_2) * W_{II}(Cl_1, Cl_2)} \quad (13)$$

In [46], the Rand index criterion has been proposed which is defined as the fraction of pair of data points for which there is an agreement in both clusterings:

$$R(Cl_1, Cl_2) = \frac{N_{11} + N_{00}}{n * (n-1)/2} \quad (14)$$

By just ignoring the "negative" agreements, the well known Jaccard coefficient arises:

$$J(Cl_1, Cl_2) = \frac{N_{11}}{n * (n-1)/2} \quad (15)$$

Another criterion is the Mirkin distance metric [41] defined as:

$$M(Cl_1, Cl_2) = \sum_{C_{k_1} \in Cl_1} (n_{k1}^2) + \sum_{C_{k_2} \in Cl_2} (n_{k2}^2) - 2 * \sum_{C_{k_1} \in Cl_1} \sum_{C_{k_2} \in Cl_2} (n_{k1k2}^2) \quad (16)$$

The second category of criteria, **criteria based on set matching** [38], are based on finding for each cluster $C_1 \in Cl_1$ a cluster $C_2 \in Cl_1$ that consist "best match" of C_1. The notion of "best matching" is implemented as follows: The contingency matrix M is scanned in decreasing order and the cluster C_2 with which C_1 shares the larger number of data points (with respect to the other clusters of Cl_2) is considered to be its match. Ignoring the row and the column of the contingency matrix for which the "best matching" has been achieved and repeating the above procedure the "second matching" is found and so on until $min(K_1, K_2)$ matches are found.

Based on this logic, several measures has been proposed. For example, Larsen et al [30] proposed the following measure:

$$L(Cl_1, Cl_2) = \frac{1}{K_1} * \sum_{k_1}(max_{k_2}(\frac{2 * n_{k1k2}}{n_{k1} + n_{k2}})) \tag{17}$$

whereas Meila and Heckerman [40] proposed the following measure:

$$H(Cl_1, Cl_2) = \frac{1}{K_1} * \sum_{k_2=match(k_1)} n_{k1k2} \tag{18}$$

where $k_1 \in Cl_1, k_2 \in Cl_2$.

Van Dongen [16] proposed the following measure:

$$H(Cl_1, Cl_2) = \frac{1}{K_1} * \sum_{k_2=match(k_1)} n_{k1k2} where k_1 \in Cl_1, k_2 \in Cl_2 \tag{19}$$

In this category also belongs the "classification error" (CE) clustering distance discussed in [39], which expresses the minimum error of classifying Cl_2 results according to Cl_1 results. CE distance measure is given by:

$$d_{CE}(Cl_1, Cl_2) = 1 - \frac{1}{n} \max_{\sigma} \sum_{k_1=1}^{K_1} n_{k_1,\sigma(k_1)} \tag{20}$$

In the above formula it is assumed that $K_1 \leq K_2$ and σ is an injective mapping of clusters $\{1, \ldots K_1\}$ of Cl_1 to clusters $\{1 \ldots K_2\}$ of Cl_2. For each σ, a partial correspondence between clusters of Cl_1 and those of Cl_2 is created and then the classification error of Cl_2 w.r.t Cl_1 is computed (think of clustering as a classification task). Among all the possible correspondences the one with the minimum classification error is d_{CE}. To find the "best correspondence" a naive solution is to examine all possible correspondences and then choose the one with the minimum CE, however this is a solution of high complexity. A polynomial time complexity solution is also available by reducing the problem to the maximum bipartite graph matching problem of the graph theory domain and use either the exact solution provided by the Hungarian algorithm [29] or some approximate solution like in [21].

Meila [38] introduced the variation of information method which belongs to the third category of clusterings comparison criteria, **criteria based on variation of information**. This measure is based on the notion of the entropy associated with a clustering which is defined as:

$$H(Cl) = -\sum_{k=1}^{K} P(k) * logP(k) \tag{21}$$

and $P(k)$ is the probability of occurrence of cluster k, defined as $P(k) = n_k/n$.

The variation of information (VI) metric computes the amount of information that is lost or gained in changing from one clustering to the other and is expressed as:

$$VI(Cl_1, Cl_2) = H(Cl_1) + H(Cl_2) - 2 * I(Cl_1, Cl_2) \tag{22}$$

where $I(Cl_1, Cl_2)$ is the mutual information between the two clusterings defined as:

$$I(Cl_1, Cl_2) = \sum_{C_{k_1} \in K_1} \sum_{C_{k_2} \in K_2} P(k_1, k_2) * log \frac{P(k_1, k_2)}{P(k_1) * P(k_2)} \tag{23}$$

and $P(k_1, k_2)$ is the joint probability that a point belongs to C_{k1} in Cl_1 and to C_{k2} in Cl_2 and is defined as: $P(k_1, k_2) = \frac{|C_{k1} \cap C_{k2}|}{n}$

Alternatively, the VI measure can be expressed as:

$$VI(Cl_1, Cl_2) = H(Cl_1|Cl_2) + H(Cl_2|Cl_1) \tag{24}$$

where the first term expresses the loss in information about Cl_1, whereas the second term expresses the gain in information about Cl_2 when going from Cl_1 to Cl_2. Although, as presented above, this measure stands for hard clustering it can be extended to apply to soft clustering as well (in this case, however, it is not a metric).

Summarizing, we can say that the first category of criteria, i.e. criteria based on counting pairs, evaluates two clusterings by examining how likely it is for them to group a pair of data points in the same cluster or separate it in different clusters. This category is mainly based on the relationships between data points and is restricted to hard clustering only [57].

The second category of criteria, i.e. criteria based on set matching, is based on the notion of "matching" between a cluster of the first clustering and one among the clusters of the second clustering. This "matching" is evaluated in terms of the data points that are grouped together (i.e. in the same cluster) under both clusterings. The proposed measures in this category are based on some greedy technique so as to find the clusters matches. At each step the matching with the greatest score is selected and finally the scores of the matched clusters are aggregated to produce a total score. However this technique does not lead to the global optimal matching score, but it can fall into some local optima. An exception is the CE error measure, which finds the global optimal solution. Furthermore, this category does not work well in cases of clusterings that contain different number of clusters, since in this case some clusters might be ignored during the comparison process [57].

The third category of criteria, i.e. criteria based on the variation of information, is based on the amount of information loss/ gain as going from one clustering to the other. The VI metric can be applied to soft clustering as well but in this case it is not a metric.

All three categories are based on the membership of data points to clusters. However, clusters are usually described not only by their members (i.e. data points) but also by other properties like cluster centroid/ medoid or some cluster density function. The description of a cluster depends of course on the algorithm used for its generation as mentioned earlier in this paper.

Towards this direction, Zhou et al [57] proposed a new measure which takes also into account the distance between cluster representatives. The proposed measure is a metric and stands for both hard and soft clustering. In fact, Zhou et al [57] proposed two measures. The first one is based on the membership of data points to clusters and comes from the Mallows distance - a metric between probability distributions in statistics. The formula that stands for soft clustering (the hard clustering is a special case of this formula) is as follows:

$$D(Cl_1, Cl_2) = \min_{w_{k,j}} \sum_{k=1}^{K} \sum_{j=1}^{J} w_{k,j} \sum_{i=1}^{N} |p_{i,k} - q_{i,j}| \tag{25}$$

where $p_{i,k}$ ($q_{i,j}$ respectively) is the probability that the point i belongs to cluster k (j respectively) of Cl_1 (Cl_2 respectively) and $w_{k,j}$ is the "weight" of matching cluster k with cluster j. This weight depends on the importance of a cluster within a clustering (a possible solution is to consider all clusters of equal importance or to define a weight based on the percentage of data points assigned to the cluster with respect to the total number of data points). The Mallows distance can be interpreted as a global optimal cluster matching schema between the two clusterings.

The second measure proposed by Zhou et al [57] takes also into account the distance between clusters centroids yielding thus more intuitive results. This measure is given by the following formula:

$$D(Cl_1, Cl_2) = \min_{w_{k,j}} \sum_{k=1}^{K} \sum_{j=1}^{J} (1 - \frac{2}{a_k + b_j} * w_{k,j}) * \sum_{i=1}^{N} p_{i,k} * q_{i,j} * L(k,j) \tag{26}$$

where $L(k, j)$ is the distance between the centroids of cluster k (belonging to Cl_1) and of cluster j (belonging to Cl_2).

If we would like to place the measures of Zhou et al [57] into the categorization of Meila [38], we could say that they belong to the second category of clustering comparison criteria, those based on set matching. In contrast to the majority of measures in this category, and together with the CE error measure, the Zhou et al measures provide a global optimal solution. Furthermore, their second measure exploits also information regarding clusters structures, like the distance between the centroids of the clusters, whereas previous measures exploit only the data points participating in each cluster.

To conclude with measures for comparing clusterings results over the same data set, we can say that a variety of such measures have been proposed in the literature with applications in evaluating different clustering algorithms, different clusterings criteria etc. The majority of these measures are based on the membership of data points to clusters, whereas there are also approaches that take also into account parameters concerning clusters structure, like the distance between cluster centroids.

5.1.2 Comparing clustering results from different data sets

The measures presented in the previous section have been introduced towards comparison of clustering results over the same data set. In the general case however, what is required is the comparison between clusterings resulted from different homogeneous data sets, i.e data sets defined over the same attribute space. Consider, for example, comparing clusterings of customers behavior produced by two different branches of a super market. Several measures have been proposed towards this aim and we will present them in more detail. It is obvious that these measures could also be applied towards comparison of clusterings results over the same data set presented in the previous subsection.

The FOCUS framework [22] proposes a way for clustering comparison based on the GCR. A cluster model is a special case of the decision tree model mentioned above, however we will give some details on this correspondence for better understanding. Each cluster corresponds to a region which is described through a set of constraints over the attributes (structure component) and is associated with the fraction of tuples in the raw data set that is assigned to this cluster (measure component). The GCR of two clusterings resulted from data set D and E is formed up by splitting clusters regions of the two clusterings until they become identical. Then the deviation between the data sets equals to the deviation between them over the set of all regions in the GCR (see 6 for more detail on FOCUS).

The PANDA framework [8] could be exploited towards clusterings comparison (see Section 6 for more detail on PANDA). According to PANDA terminology, a cluster could be considered as a simple pattern, whereas a clustering could be considered as a complex pattern. Defining how two clusters should be compared, how simple clusters of the two clusterings should be matched and how the scores of the matched clusters should be aggregated, a total score could be calculated that expresses the similarity between the two clusterings to be compared. Different instantiations of the PANDA framework result in several different similarity configurations.

Moreover, some of the measures presented in the previous subsection could also be utilized towards comparing clusterings results over different data sets.

To conclude with measures for comparing clustering results over different data set, we can say that only a few attempts exist in the literature - most of the work is towards comparing clusterings over the same data set. However, performing comparison over different data sets clusterings results in a variety of useful applications and could also be exploited towards clusterings monitoring or towards second generation clustering (i.e. clustering over clusterings).

6 General Frameworks

Despite the individual approaches towards comparing specific pattern types, more general approaches (frameworks) also have been proposed in the literature (in fact there exist only a few attempts towards this direction). In this section, we will present these approaches in more detail.

Ganti et al [22] presented the FOCUS framework for measuring the deviation between two data sets D and E. Although designed for data set comparison, the FOCUS framework actually utilizes the comparison between the corresponding pattern sets. The intuition behind this hypothesis is that interesting data characteristics in a data set are captured by a model induced from the data set. In that work authors introduced the *2-component property of patterns*, already discussed in Section 2.

If the induced models have different structure components, a first step is required to make them identical by extending them to their *Greatest Common Refinement* (GCR). For better understanding, we can think of GCR as a way of refining the structure of the models to be compared, which is achieved by splitting down the models to be compared until identical regions are formed. Then, the deviation between the data sets equals to the deviation between them over the set of all regions in the GCR. A difference function that calculates the deviation of two regions and an aggregate function that aggregates all these differences are required; depending on the choice of these functions, several instantiations of the FOCUS framework might arise. Deviation computation requires the measures of all regions in GCR to be computed with respect to both data sets D and E, so the comparison in pattern space (*patterns*) also involves the data space (*raw data*).

The FOCUS framework works for three well known data mining pattern types, namely frequent itemsets, clusters and decision trees. The further details on each case have already been presented in the corresponding sections of this work.

Bartolini et al [8] proposed the PANDA framework for the comparison of both simple and complex patterns. *Simple* patterns are those defined over raw data (e.g. a cluster of data points) whereas *complex* patterns are those defined over other patterns (e.g. a set of clusters, i.e. clustering). Authors adopt the 2-component property of patterns introduced by FOCUS, thus patterns are expressed through a structure and a measure component and their similarity is evaluated in terms of both of these components.

The similarity between two simple patterns p_1, p_2 is evaluated by combining, by means of an aggregation function, f_{aggr}, the similarities between both the structure ($p_1.s, p_2.s$ respectively) and the measure components ($p_1.m, p_2.m$ respectively):

$$sim(p_1, p_2) = f_{aggr}(sim_{struct}(p_1.s, p_2.s), sim_{meas}(p_1.m, p_2.m)) \qquad (27)$$

Regarding the structural similarity, if the two patterns have the same structure, then their measures similarity is only considered. In the general case, however, the patterns to be compared have different structural components, thus a preliminary step is needed to "reconcile" the two structures so as to make them comparable (by calculating, for example, the amount of work required to turn one structure to the other). Regarding the measure components similarity several measures could be used like absolute difference or relative difference. Regarding the aggregation function, either sum or weighted sum could be exploited.

The PANDA idea is depicted in Figure 2:

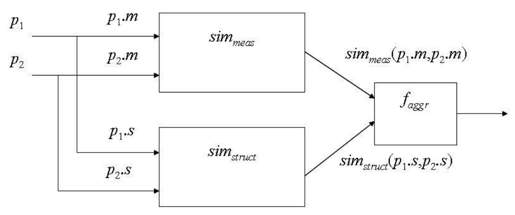

Figure 2: Assessment of similarity between two patterns

Evaluation of similarity between complex patterns follows the same logic depicted in Figure 2. However, the structure of complex patterns now consists of several other patterns. Within PANDA framework, the similarity between structures of complex patterns depends in turn on the similarity between component patterns. Similarity is conceptually evaluated in a bottom-up fashion, and can be adapted to specific needs/constraints by acting on two fundamental abstractions:

- the *coupling type*, which is used to establish how component patterns can be matched;

- the *aggregation logic*, which is used to combine the similarity scores obtained for coupled component patterns into a single overall score representing the similarity between the complex patterns.

Depending on the instantiations of the different blocks of the framework (coupling type, aggregation logic, structure similarity and measure similarity) various similarity functions configurations can be defined within the PANDA framework.

Comparing the above frameworks, we can state that both FOCUS and PANDA frameworks try to give a general solution to the problem of pattern comparison between patterns of the same pattern type. PANDA framework could be thought of as an extension of the FOCUS framework by means that it provides a wide variety of matching criteria (coupling type) in contrast to the specific type of GCR matching provided by FOCUS. Furthermore, it is more generic since it can be applied to arbitrarily complex patterns, like Web site structures and not only to patterns for which their GCR can be defined (like frequent itemsets, decision trees and clusters). Also, it works exclusively in the pattern space and does not involve the raw data space as well, thus it is more efficient.

Except for frameworks for pattern comparison, frameworks for pattern monitoring over time and efficient pattern updating have been also proposed in the literature. Their goal is monitoring and understanding pattern changes over time (recall the dynamic nature of data); in some cases, however, monitoring concerns raw data (instead of patterns) and utilizes corresponding patterns towards this aim.

Ganti et al [23] proposed the DEMON framework for mining systematically evolving data across the temporal dimension, where "systematically" means that data changes through additions or deletions of blocks of records (a block is a set of records added simultaneously to the database). DEMON mainly focuses on efficient updating of models (i.e. pattern base) by detecting changes in raw data. To find data changes authors build on their prior work, i.e. FOCUS framework. After detecting (across the time dimension) the data blocks that have been changed, these blocks are processed in order to maintain the corresponding models. Also, authors describe efficient model maintenance algorithms for frequent itemsets and clusters.

In [4, 5] Baron and Spiliopoulou introduced Pattern Monitor (PAM), a framework for efficiently maintaining data mining results. In contrast to other approaches which consider only part of a pattern, either its content, i.e., the relationship in the data the pattern reflects, or the statistical properties of the pattern, the authors model rules as integrated temporal objects, which may exhibit changes in either of these two aspects of a rule.

PAM builds on the temporal rule model already presented in Section 3.2. The core of PAM implements the change detector and a series of heuristics to identify not only significant but also interesting rule changes which take different aspects of pattern reliability into account.

The *Occurrence-based Grouping Heuristic* reveals not patterns observed within a specific time interval but patterns that are present in each period and, as such, reflect (part of) the invariant properties of the underlying data set. The *Corridor-based Heuristic* defines a corridor (around the time series of a pattern) as an interval of values, which is dynamically adjusted at each time point to reflect the range of values encountered so far. In *Interval-Based Heuristic* the range of values of the time series is partitioned into intervals of equal width, so an alert for a pattern R is raised for each time point t_i at which the value of the time series is in a different interval than for t_{i-1}.

Furthermore, due to the fact that the change detector returns at each time point t_i the set of all patterns, whose observed statistic measure has changed with respect to the previous period and the fact that this set is usually large as patterns overlap in content, the authors introduce the term *atomic change* to identify a minimal set of patterns whose characteristic is that there are no components (i.e. the body and the head) that have themselves experienced a change.

In [7], authors distinguish rules into *permanent rules* that are always present (though they may undergo significant changes) and *temporarily rules* that appear only temporarily and indicate periodic trends.

In [6], PAM was applied in a different case study following an evaluation procedure intending to use the monitor not only to identify rule changes of a particular strength, but also to check whether old rules still hold.

Concluding the presentation of frameworks for pattern monitoring, we can state that they are very important by means that they give insights on how raw data and corresponding patterns evolve over time (recall that most of the data collected nowadays are dynamic). These frameworks utilize methods for pattern comparison in order to detect significant changes across the time axis. Furthermore they can also be applied towards efficient maintenance of pattern bases (i.e. data mining results). However, the DEMON framework gives more emphasis on frequent itemsets and clusters, while the PAM framework focuses mainly on association rules, thus there is not some generic approach supporting all data mining pattern types monitoring and maintenance.

In this section we have presented the work performed so far towards developing generic frameworks for pattern comparison. The importance of such frameworks is straightforward since there are several different pattern types in the literature, so being able to deal with issues like comparison under a common framework forms an elegant solution. We also presented the work regarding frameworks for pattern monitoring and maintenance, which are based on comparing consecutive snapshots of the database or pattern base.

Although some steps have already been done towards a generic, universal way of comparison, the current methods concern specific pattern types and their extension to other pattern types is not so straightforward.

7 Conclusions

In this work, we presented an overview of the research performed so far in the area of data mining patterns comparison focusing on popular data mining pattern types, namely frequent itemsets and association rules, clusters and clusterings, and decision trees. Despite individual approaches towards comparing specific types of pattens, more general approaches (frameworks) have been also presented.

As demonstrated by the related work, several ad-hoc approaches for similarity assessment of particular pattern types have been proposed in the literature. All of these measures utilize information regarding the pattern space that a pattern describes (i.e. structure component) as well as information about how well the pattern represents the underlying raw data space (measure component).

All of these measures are pattern type specific, and if we follow this rationale, new similarity measures should be defined each time a new pattern type emerges. Its obvious that this solution is not so efficient, especially nowadays where new types of data or patterns arise continuously from various domains, e.g. bio-informatics, telecommunications, audio/visual entertainment, etc. A nice solution to this problem would be some pattern type independent similarity measure(s).

Furthermore, related work is currently limited to similarity assessment between patterns of the same pattern type. In the general case, however, even patterns from different pattern types should be able to be compared. As a motivating example, consider the problem of comparing a decision tree and a clustering both extracted from different branches of the same supermarket.

As demonstrated by the related work, similarity in pattern space has been utilized towards similarity assessment in underlying raw data space, based on the intuition that patterns encompass most of the information lying in the corresponding raw data. However, the effect of mining parameters on this correspondence usually is ignored. In case of FIM problem, for example, minSupport threshold used for the generation of patterns and the adopted lattice representation, i.e. frequent, closed frequent or maximal frequent itemsets, could be considered as such parameters. Thus, a future direction would be associating distance in pattern space with distance in the original raw data space from which patterns have been extracted through some data mining process. This direction would result in a wide variety of applications, since pattern space is usually of lower complexity than raw data space.

Acknowledgement This research is partially supported by the Greek Ministry of Education and the European Union under a grant of the "Heracletos" EPEAEK II Programme (2003-06).

References

[1] C. C. Aggarwal. On change diagnosis in evolving data streams. *IEEE Transactions on Knowledge and Data Engineering*, 17:587–600, 2005.

[2] R. Agrawal, G. Psaila, E. L. Wimmers, and M. Zaki. Querying shapes of histories. In *Proceedings of VLDB'95*, 1995.

[3] N. F. Ayan, A. U. Tansel, and E. Arkun. An efficient algorithm to update large itemsets with early pruning. In *Proceedings of KDD'99*, 1999.

[4] S. Baron and M. Spiliopoulou. Monitoring change in mining results. In *Proceedings of DaWaK'01*, 2001.

[5] S. Baron and M. Spiliopoulou. *Monitoring the results of the KDD process: An overview of pattern evolution*, chapter 5, pages 845–863. Dealing with the Data Flood: Mining data, text and multimedia. STT Netherlands Study Center for Technology Trends, 2002.

[6] S. Baron and M. Spiliopoulou. Monitoring the evolution of web usage patterns. In *Proceedings of EWMF'03*, 2003.

[7] S. Baron, M. Spiliopoulou, and O. Günther. Efficient monitoring of patterns in data mining environments. In *Proceedings of ADBIS'03*, 2003.

[8] I. Bartolini, P. Ciaccia, I. Ntoutsi, M. Patella, and Y. Theodoridis. A unified and flexible framework for comparing simple and complex patterns. In *Proceedings of PKDD'04*, 2004.

[9] L. Breiman, J. Friedman, R. Olshen, and C. Stone. *Classification and regression trees.* Belmont CA. Wadsworth International Group, 1984.

[10] B. Catania and A. Maddalena. *Pattern Management: Practice and Challenges*, chapter 10, pages 280–317. Processing and Managing Complex Data for Decision Support. Idea Group Publishing, 2006.

[11] X. Chen and I. Petrounias. Mining temporal features in association rules. In *Proceedings of PKDD'99*, 1999.

[12] D. W.-L. Cheung, J. Han, V. Ng, and C. Y. Wong. Maintenance of discovered association rules in large databases: An incremental updating technique. In *Proceedings of ICDE'96*, 1996.

[13] D. W.-L. Cheung, S. D. Lee, and B. Kao. A general incremental technique for maintaining discovered association rules. In *Proceedings of DASFAA'97*, 1997.

[14] R. B. D'Agostino and M. A. Stephens, editors. *Goodness-of-fit techniques.* Marcel Dekker, Inc., 1986.

[15] V. N. David Cheung and B. Tam. Maintenance of discovered knowledge: A case in multi-level association rules. In *Proceedings of KDD'96*, 1996.

[16] S. V. Dongen. Performance criteria for graph clustering and markov cluster experiments. Technical report, Center for Mathematics and Computer Science (CWI), Amsterdam, 2000.

[17] M. H. Dunham. *Data Mining: Introductory and Advanced Topics.* Prentice Hall, 2003.

[18] J. M. E. B. Hunt and P. T. Stone. *Experiments in Induction.* Academic Press, 1966.

[19] U. M. Fayyad, G. Piatetsky-Shapiro, and P. Smyth. From data mining to knowledge discovery: An overview. In *Advances in Knowledge Discovery and Data Mining*, pages 1–34. 1996.

[20] E. B. Fowlkes and C. L. Mallows. A method for comparing two hierarchical clusterings. *Journal of the American Statistical Association*, 78:553 – 569, 1983.

[21] H. N. Gabow and R. E. Tarjan. Faster scaling algorithms for network problems. *SIAM J. Comput.*, 18:1013–1036, 1989.

[22] V. Ganti, J. Gehrke, and R. Ramakrishnan. A framework for measuring changes in data characteristics. In *Proceedings of PODS'99*, 1999.

[23] V. Ganti, J. Gehrke, and R. Ramakrishnan. Demon: Mining and monitoring evolving data. In *Proceedings of ICDE'00*, 2000.

[24] V. Ganti and R. Ramakrishnan. *Mining and Monitoring Evolving Data*, chapter 17, pages 593–642. Handbook of Massive Data Sets. Kluwer Academic Publishers, 2002.

[25] J. Gehrke, V. Ganti, R. Ramakrishnan, and W.-Y. Loh. Boat optimistic decision tree construction. *SIGMOD Records*, 28:169–180, 1999.

[26] G. K. Gupta, A. Strehl, and J. Ghosh. Distance based clustering of association rules. In *Proceedings of ANNIE'99*, 1999.

[27] J. Han and M. Kamber. *Data mining: concepts and techniques.* Morgan Kaufmann Publishers Inc., 2000.

[28] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: a review. *ACM Computer Surveys*, 31:264–323, 1999.

[29] H. Kuhn. Hungarian method for the assignment problem. *Nay. Res. Logist. Quart.*, 2:83–97, 1955.

[30] B. Larsen and C. Aone. Fast and effective text mining using linear-time document clustering. In *Proceedings of KDD'99*, 1999.

[31] B. Lent, A. N. Swami, and J. Widom. Clustering association rules. In *Proceedings of ICDE'97*, 1997.

[32] T. Li, M. Ogihara, and S. Zhu. Association-based similarity testing and its applications. *Intelligent Data Analysis*, 7:209–232, 2003.

[33] B. Liu, W. Hsu, and Y. Ma. Discovering the set of fundamental rule changes. In *Proceedings of KDD'01*, 2001.

[34] B. Liu, Y. Ma, and R. Lee. Analyzing the interestingness of association rules from the temporal dimension. In *Proceedings of ICDM'01*, 2001.

[35] W.-Y. Loh and Y.-S. Shih. Split selection methods for classification trees. *Statistica Sinica*, 7, 1997.

[36] W.-Y. Loh and N. Vanichsetakul. Tree-structured classification via generalized discriminant analysis (with discussion). *Journal of the American Statistical Association*, 83:715–728, 1988.

[37] P. Lyman and H. R. Varian. How much information. Retrieved from http://www.sims.berkeley.edu/research/projects/how-much-info (valid as on January 2005), 2003.

[38] M. Meila. Comparing clusterings by the variation of information. In *Proceedings of COLT'03*, 2003.

[39] M. Meila. Comparing clusterings: an axiomatic view. In *Proceedings of ICML'05*, 2005.

[40] M. Meila and D. Heckerman. An experimental comparison of model-based clustering methods. *Machine Learning*, 42:9–29, 2001.

[41] B. Mirkin. *Mathematical classification and clustering.* Kluwer Academic Press, 1996.

[42] T. Mitchell. *Machine Learning.* Kluwer Academic Publishers, 1997.

[43] D. B. Neill, A. W. Moore, M. Sabhnani, and K. Daniel. Detection of emerging space-time clusters. In *Proceedings of KDD'05*, 2005.

[44] S. Parthasarathy and M. Ogihara. Clustering distributed homogeneous datasets. In *Proceedings of PKDD'00*, 2000.

[45] T. I. Rakesh Agrawal and A. Swami. Mining association rules between sets of items ins large databases. In *Proceedings of ACM SIGMOD'93*, 1993.

[46] W. M. Rand. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical Association*, 66:846 – 850, 1971.

[47] S. Rizzi, E. Bertino, B. Catania, M. Golfarelli, M. Halkidi, M. Terrovitis, P. Vassiliadis, M. Vazirgiannis, and E. Vrachnos. Towards a logical model for patterns. In *Proceedings of ER'03*, 2003.

[48] K. A. S. Thomas, S. Bodagala and S. Ranka. An efficient algorithm for the incremental updation of association rules in large databases. In *Proceedings of KDD'97*, 1997.

[49] N. L. Sarda and N. V. Srinivas. An adaptive algorithm for incremental mining of association rules. In *Proceedings of DEXA'98*, 1998.

[50] H. Toivonen. Sampling large databases for association rules. In *Proceedings of VLDB'96*, 1996.

[51] H. Toivonen, M. Klemettinen, P. Ronkainen, K. Hatonen, and H. Mannila. Pruning and grouping of discovered association rules. In *Mlnet Workshop on Statistics, Machine Learning, and Discovery in Databases*, 1995.

[52] P. E. Utgoff. Incremental induction of decision trees. *Machine Learning*, 4:161–186, 1989.

[53] P. E. Utgoff, N. C. Berkman, and J. A. Clouse. Decision tree induction based on efficient tree restructuring. *Machine Learning*, 29:5–44, 1997.

[54] D. L. Wallace. Comment. *Journal of the American Statistical Association*, 78:569 – 576, 1983.

[55] D. Xin, J. Han, X. Yan, and H. Cheng. Mining compressed frequent-pattern sets. In *Proceedings of VLDB'05*, 2005.

[56] M. Zaki and C.-J. Hsiao. Efficient algorithms for mining closed itemsets and their lattice structure. *IEEE Transactions on Knowledge and Data Engineering*, 17:462–478, 2005.

[57] D. Zhou, J. Li, and H. Zha. A new mallows distance based metric for comparing clusterings. In *Proceedings of ICML'05*, 2005.