The notion of patterns in Data Mining

Irene Ntoutsi, Yannis Theodoridis

Information Systems Lab
Department of Informatics
University of Piraeus
Hellas

Technical Report Series
UNIPI-ISL-TR-2007-02

March 2007

The Notion of Patterns in Data Mining*

Irene Ntoutsi and Yannis Theodoridis
Department of Informatics,
University of Piraeus, Greece
{ntoutsi,ythed}@unipi.gr

Abstract

Several heterogeneous pattern types are available nowadays as a result of the different goals that a mining task tries to accomplish and of the application of Data Mining techniques over different domains. In this chapter we overview three popular pattern types, namely frequent itemsets and association rules, clusters and clusterings, and decision trees. We also adopt a pattern representation schema based on both extensional and intensional description of patterns and describe how the different pattern types are expressed with respect to this schema.

1 Introduction

Knowledge Discovery in Databases (KDD) and Data Mining (DM) provide a solution to the information flood problem, by extracting valid, novel, potentially useful, and ultimately understandable patterns from data [1]. According to Rizzi et al. [8], patterns constitute 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.

Several pattern types exist in the literature mainly due to the wide heterogeneity of data and data mining applications, as well as due to the different techniques for pattern extraction as a result of the different goals that a mining process tries to accomplish (i.e. what data characteristics the mining task tries to highlight). For example, frequent itemsets capture the correlations between attribute values, clusters reveal natural groups in data, whereas decision trees detect characteristics that predict (with respect to a given class attribute) the behavior of future records.

In the following subsections we overview 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

*This research is partially supported by the Greek Ministry of Education and the European Union under a grant of the "Heracletos" EPEAEK II Programme (2003–07).

trees. At the end of the chapter, we present our pattern representation schema which is based on both extensional and intensional description of patterns and show how the different pattern types presented through this chapter are adapted into this schema.

2 Pattern representation schema

Patterns summarize the raw data in a compact and semantically reach way. As such, the description of a pattern might be either *extensional*, i.e. in terms of the data members participated in its generation, or *intensional*, i.e. in terms of the meaning/ concept represented by the pattern. The extensional description of a pattern is just an enumeration of its data members, thus it is common for all pattern types. The intensional description, however, reveals information about the "shape" and the semantics of the pattern, thus it depends on the pattern type.

Regarding the intensional description of patterns, Ganti et al. [2] introduced the 2-component property of patterns. According to this property, a broad class of pattern types can be described in terms of a structure and a measure component. The *structure component* identifies "interesting regions" within the pattern space, 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 the settings of the PANDA project, where Rizzi et al. [8] 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.

Similar ideas appear in [5], where authors introduce the 3W model towards unified Data Mining. Here patterns are described as constraints over the attribute space (I-World) and are related (E-World) to the raw data (D-World) from which they have been extracted through some data mining task.

Throughout this work, we adopt the extensional – intensional description of patterns, and for the intensional part we adopt the 2–component property of patterns.

3 Decision Trees

Decision Trees (DTs) are very popular *classification methods* due to their intuitive representation that render them easily understandable by humans. In this section, we provide some basic concepts on DTs following the work by Mitchell [6].

Let $A = \{A_1, A_2, ..., A_m\}$ be the set of attributes on which classification will be based (*predictive attributes*), where attribute A_i has domain $dom(A_i)$.

Let C be the *class attribute*, i.e. the attribute to be predicted, with domain $dom(C) = \{C_1, C_2, ..., C_k\}$, k is the number of classes. P is a probability distribution on $dom(A_1) \times dom(A_2) \times \cdots \times dom(A_m) \times dom(C)$, i.e. P is the joint probability distribution of the predictive and the class attributes; P is called the *problem distribution*. The probability distribution of the predictive attributes, i.e. $D(A_1) \times D(A_2) \times \cdots \times D(A_m))$ is called the *attribute space distribution*.

The goal of a decision tree is to learn a predictor function $f : dom(A_1) \times dom(A_2) \times \cdots \times dom(A_m) \to dom(C)$. Towards this goal, a set of problem instances drawn from P is utilized; this is known as the *training set* D. A decision tree T constructed from D provides a classification of D's instances into the C_j, $j = 1 \ldots k$, classes based on the values of the predictive attributes A_i, $i = 1 \ldots m$.

Predictive attributes might be either numerical, categorical or ordinal. The domain of a *numerical attribute* is an ordered set (e.g. age, income), the domain of a *categorical or nominal attribute* is a finite set without any natural ordering (e.g. colors, gender, marital status), whereas the domain of an *ordinal attribute* is a set of discrete values with an imposed order, but without any knowledge regarding the absolute differences between values (e.g. preference scale, severity of an injury). Usually, the predictive attributes are numeric.

Regarding its structure, a DT consists of internal and leaf nodes. An *internal node* has an associated test condition (*splitting predicate*) which specifies a test over some predictive attribute (e.g. "$Age \geq 20$"). Each branch descending from that node corresponds to one of the possible values for this attribute. Most common are binary predicates, i.e. predicates of the form "Yes" or "No". A *leaf node* provides the class label of the instances that follow the path from the root to this node. In case the instances belong to more than one classes, this label might be the label of the majority class. In the general case, a leaf node might be associated with some weight to all problem classes; this weight depends on the amount of instances that fall into the leaf and belong to the specific class.

In Figure 1 an example of a DT is depicted, which refers to to the bank loan allowance problem. It is determined over two predictive attributes, Age and $Income$, and a class attribute $C = \{C_1, C_2\}$.

A small part of the dataset used for its generation is depicted in Table 1.

Instance	Age	Salary	Class
1	30	30K	C_1
2	35	10K	C_2
3	50	100K	C_1

Table 1: A sample of the training set used for the DT of Figure 1

Evaluation As already stated, a DT is build upon a training set D of problem instances drawn from the joint probability distribution of the predictive attributes and of the class attribute$P$. A "fully developed tree" will perfectly fit the training set. However, a DT should not only fit the training data well, but

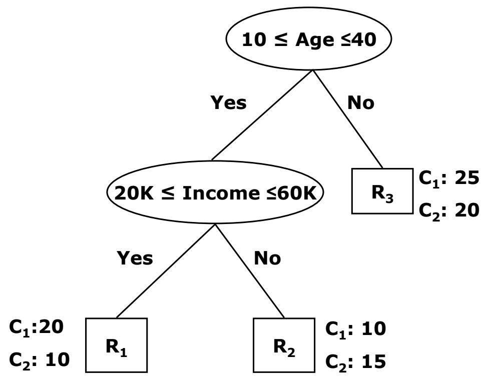

Figure 1: An example of a decision tree

it should also predict correctly the class labels of future, previously unseen problem instances. Over–fitting the training set is a wrong property for a DT, since it might follow every idiosyncrasy of the training set, much of which are unlikely to occur in future seen problem instances. This desirable property is known as *generalization accuracy* and is measured through the *miss–classification error(ME)*. ME is based on counting the number of instances predicted wrongly by the DT model. Ideally we would like to know the ME of the classifier f on the problem distribution P. However, since P is unknown (the only we know are some instances drawn from it, i.e. the training set), several techniques have been developed for estimating $ME(f, P)$.

The most common technique is the *holdout test estimate*: the initial set of problem instances is split into two disjoint sets: train and test. The *train set* is used for building the classifier, whereas the *test set* for evaluating its performance. Usually, 1/3 of instances is used for testing and 2/3 is used for training. Other popular techniques in this category are *re–substitution estimation* and *V–fold cross validation*.

A solution to the generalization problem is *tree pruning*. Beginning at the last DT level, the child nodes are pruned away if the resulting change in the tree accuracy is less than a times the change in tree complexity. Due to pruning, the resulting tree might not perfectly predict its training set; this error is called *re–substitution error*. Thus, a good DT should minimize both the re–substitution error (with respect to the training set) and the miss–classification error (with respect to an independent test set).

Partitioning The DT growing process can be viewed as the process of partitioning the *attribute space* (i.e. the space defined by the predictive attributes, $D(A_1) \times D(A_2) \times \cdots \times D(A_m)$) into disjoint regions until each region contains instances of the same class (this holds in the maximal case, since after pruning a region might contain instances from more than one problem classes). The border line between two neighboring regions of different classes is knows as the *decision boundary*. Decision boundaries are parallel to the attribute axis since each test condition involves only a single attribute. Thus, decision regions are axis parallel hyper–rectangles, also called *isothetic* [5].

Each leaf node of the tree corresponds to a region R. A region can be described through the set of instances that are mapped to the corresponding leaf node, this is called the *extension of the region*. A semantic/ implicit description of the region is also available: a region can be described through the path starting from the root of the tree and resulting to the corresponding leaf node, this is called the *structure of the region*. For example, the structure component of the most left leaf node in the tree of Figure 1 is: $(10 \leq Age \leq 40) \cap (20K \leq Income \leq 60K)$. In case the actual instances are not available, a summarized description of them might exist, like for the example the fraction of problem instances that fall into this region for each of the problem classes, this is called the *measure of the region*. For example, the measure component of the most left leaf node in the tree of Figure 1 is: 20% for C_1 and 10% for C_2. The structure and the measure component of a region, comprise the *intension of the region*.

The partitioning of the previously presented DT (Figure 1) is depicted in Figure 2. Note here that the attribute space is defined by the predictive at-

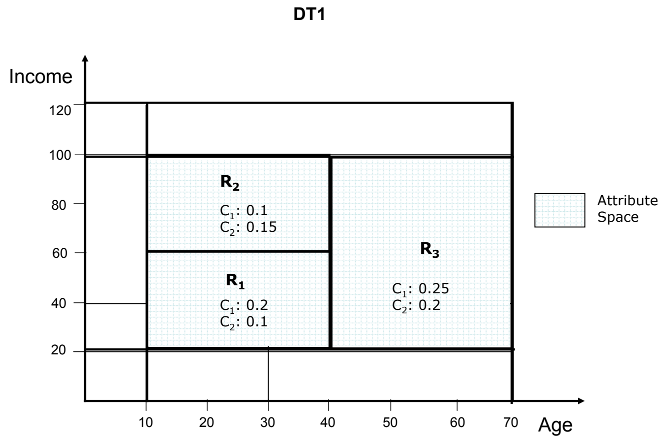

Figure 2: The attribute space partitioning achieved by the DT of Figure 1

tributes of the problem, and as such it is common for all DTs referring to the specific classification problem. What actually differentiates the different DTs is the partitioning they perform over the attribute space, i.e. what are the resulting regions. Further detail on the partitioning can be found on the DT comparison chapter.

4 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 [4]. 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 cluster $C_1, C_2, \ldots, C_K$ such that $C_i \cap C_j = \emptyset$ and $\cup_{j=1}^{K}(C_j) = D$. This definition stands for *hard clustering*, where an instance is assigned to exactly one cluster, forming thus a crisp partition of the data set. A more "relaxed" definition is that of *soft clustering* which allows for degrees of membership, to which an instance belongs to different clusters.

Clustering algorithms are based on some *distance function* that evaluates in which cluster an object should be assigned. For example, Euclidean distance is usually utilized for numeric instances. There is also an *evaluation function* that evaluates how good the achieved clustering is. Minimizing the distance of every data point from the mean of the cluster to which it is assigned could be considered 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 [4] for a survey).

Different clustering algorithms proposed in the literature use a variety of cluster definitions. Han and Kamber [3] 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 and is "represented" through some centroid like in the K-means algorithm or through some medoid like in the K-medoids algorithm [3]. An example of the $K-$means algorithm (generated through MATLAB) is depicted in Figure 3.

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; different distance functions can be used like single link, complete link, average link or centroids distance [3]. In this case, clusters discovered during the clustering process are organized in a dendrogram structure, like the one depicted in Figure 4.

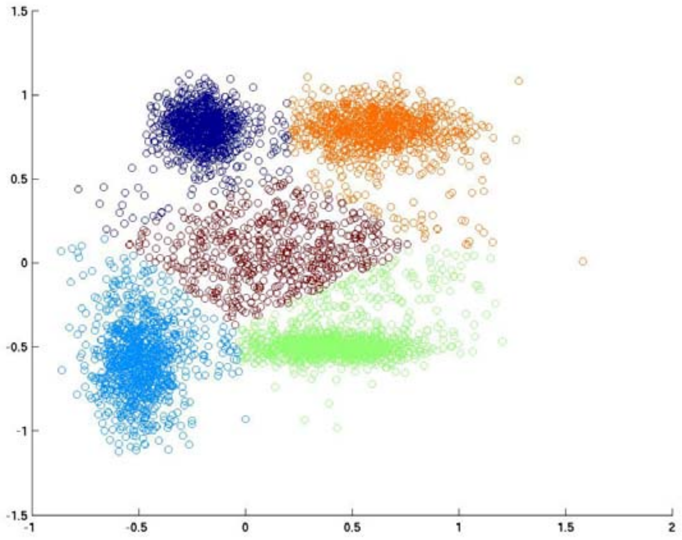

Figure 3: An example of K–means clustering

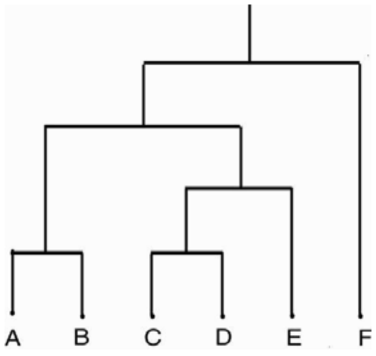

Figure 4: A dendrogram generated through some hierarchical clusterer

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. In this category belongs the DBScan algorithm [3], some examples of which are depicted in Figure 5.

4. *Grid–based methods* that quantize the object space into a finite number of cells that form a grid structure. In this category belong STING and

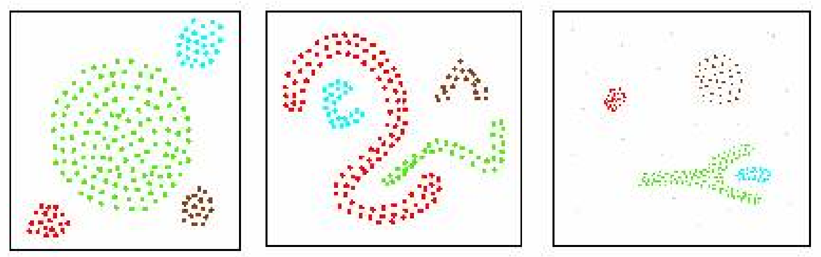

Figure 5: An example of DBScan

CLIQUE algorithms [3].

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 the COBWEB algorithm, and neural network approaches are the two major approaches in this category [3].

Lets return to the extensional–intensional description of clusters: The *extensional description* of a cluster is straightforward and can be extracted by an explicit enumeration of the data that fall into the cluster boundary, thus this description is common for the different cluster types.

The *intensional description*, however, depends on the specific cluster type, even on the specific characteristics of the clustering algorithm: for example, in case of a "partitioning cluster", the *structure component* can be defined by its center as in K-means or by its centroid as in K-medoids. Note however, that there are algorithms like the hierarchical ones, where some intensional description of the structure of the generated clusters is not available. In this case, only the extensional description of the clusters in terms of their data members is available.

Regarding the *measure component* of the intensional description, there are several alternative possibilities like the cluster support (i.e. the percentage of data set records that fall into this cluster) or the intra-cluster distance (i.e. average distance between cluster members) or the average distance of the cluster members from its centroid/ medoid.

We demonstrate the notion of explicit, implicit representation of a cluster through an example. For display purposes, we consider numerical 2D data, like those in Figure 6. Clusters are also shown in this figure; they are the result of the K-means execution over the data. Consider the left cluster of Figure 6. Its extensional description is an enumeration of its members, thus it is {(3,4), (3,8), (4,5), (4,7), (2,6)}, where each pair (x, y) represents a point (the members are painted in navy blue in the figure). The intensional description consists, in this case, of the cluster centroid, thus it is $(3.2, 6)$ (the centroid is painted in pink in the figure).

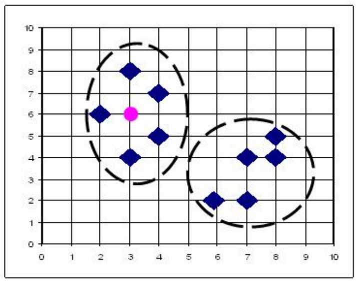

Figure 6: An example of the extensional/ intensional description of a cluster

5 Frequent Itemsets and Association Rules

Frequent itemsets and association rules mining are strongly related to each other since frequent itemsets mining is the first step towards association rules mining.

The **Frequent Itemset Mining (FIM)** problem is a core problem in many data mining tasks, such as association rules, correlations, sequences, episodes etc, although the original motivation for frequent itemsets mining came from the need to analyze supermarket transaction data in order to find items that are frequently purchased together.

For the definition of the FIM problem, we follow the work by Agrawal et al [7]: Let I be a set of distinct items and D be a database of transactions where each transaction T contains a set of items $T \subseteq I$. An example of a transaction database is depicted in Table 2.

Transaction ID	**Transaction Items**
100	1 3 4
200	2 3 5
300	1 2 3 5
400	2 5

Table 2: An example transaction database D

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 (FI) forms the itemset lattice L in which the lattice property holds: an itemset is frequent iff all of its subsets are frequent. Considering $minSupport = 2$ and the database example of Table 2, the corresponding frequent itemsets lattice is depicted in Figure 7.

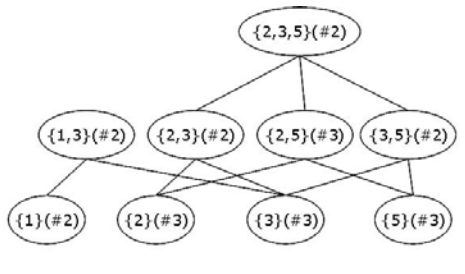

Figure 7: An example of a frequent itemsets lattice

The lattice property allows to enumerate all frequent itemsets using more compact representations like closed frequent itemsets(CFI) and maximal frequent itemsets (MFI). 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 $CFIs$. The set of closed frequent itemsets for the example of Figure 7 is depicted in Figure 8.

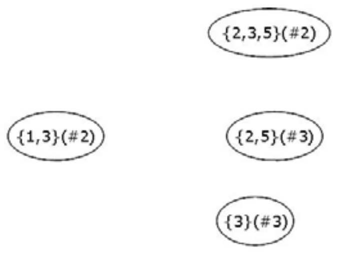

Figure 8: The closed frequent itemsets for the lattice of Figure 7

Closed frequent itemsets provide some compression over the FI lattice, however this compression is small since it is based on both the structure and the measure of the items in the lattice. To provide further compression capabilities, the notion of maximal frequent itemsets has been proposed. 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 [9]. Practically, $CFIs$ can be orders of magnitude less than FIs, and $MFIs$ can be orders of magnitude less than $CFIs$ [9]. The set of maximal frequent itemsets for the example of Figure 7 is depicted in Figure 9.

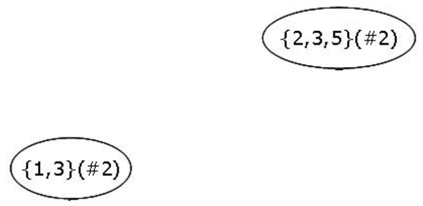

Figure 9: The maximal frequent itemsets for the lattice of Figure 7

Lets return to the extensional – intensional description of patterns: the extensional description of an itemset consists of an explicit enumeration of the data instances that support this itemset. For example, the extensional description of the itemset $(\{1,3\},2)$ is the set of instances $\{100,300\}$ from Table 2. Regarding the intensional description, the *structure component* consists of the itemset itself, i.e. items that form it like $\{1,3\}$, whereas the *measure component* consists of itemset support, e.g. 2.

The **Association Rules Mining (ARM)** problem was first introduced by Agrawal and Swami [7] and was mainly motivated by the market basket analysis domain. It is 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. A rule is *interesting* or *strong* if its support and threshold exceed some user specified thresholds.

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.

Lets return to the extensional – intensional description of patterns: the extensional description of a rule comprises of all those instances that contribute to its generation. For example, the extensional description of the rule $2 \rightarrow 5$ is the set of instances $\{200,400\}$ from Table 2. The structure component of a rule consists of its head (item 2 in our example) and body (item 5 in our example), whereas the measure component consists of its confidence (100% in our example) and support (50% in our example).

References

[1] 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.

[2] V. Ganti, J. Gehrke, and R. Ramakrishnan. A framework for measuring changes in data characteristics. In *PODS*, pages 126–137. ACM Press, 1999.

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

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

[5] T. Johnson, L. Lashmanan, and T. Ravmond. The 3W Model and Algebra for Unified Data Mining. In *VLDB*, 2000.

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

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

[8] 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 *ER*, pages 77–90, 2003.

[9] 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.