Mining Correct Properties in Incomplete Databases

François Rioult and Bruno Crémilleux

GREYC, CNRS - UMR 6072, Université de Caen
F-14032 Caen Cédex France
{Francois.Rioult,Bruno.Cremilleux}@info.unicaen.fr

Abstract. Missing values issue in databases is an important problem because missing values bias the information provided by the usual data mining methods. In this paper, we are searching for mining patterns satisfying correct properties in presence of missing values (it means that these patterns must satisfy the properties in the corresponding complete database). We focus on k-free patterns. Thanks to a new definition of this property suitable for incomplete data and compatible with the usual one, we certify that the extracted k-free patterns in an incomplete database also satisfy this property in the corresponding complete database. Moreover, this approach enables to provide an anti-monotone criterion with respect to the pattern inclusion and thus design an efficient level-wise algorithm which extracts correct k-free patterns in presence of missing values.

1 Introduction

Missing values in databases is a problem as old as the origin of these storage structures. It is an important issue because information extracted by usual data mining or statistics methods in incomplete data are biased and do not reflect the sound knowledge on the domain. We show in Section 3.1 the damages due to missing values in the pattern mining area. The popular uses of (frequent) patterns (e.g., rules, classification, clustering) are no longer reliable. The basic idea of elementary techniques to cope with missing values is to guess them (e.g., use of the mean, the most common value, default value) and complete them. Unfortunately, these techniques are not satisfactory because they exaggerate correlations [1] and missing values completion remains a hard track.

On the contrary, in this paper, we are searching for mining patterns satisfying properties in presence of missing values which are also satisfied in the corresponding complete database. Our key idea is to highlight properties from an incomplete database, these properties must be consistent in the real database without missing values. We say that these properties are *correct*. This can be achieved because some characteristics are not removed by missing values. For instance, if a pattern is frequent in a database with missing values, it must be frequent in the corresponding complete database. In Section 4.1, we propose

S. Džeroski and J. Struyf (Eds.): KDID 2006, LNCS 4747, pp. 208–222, 2007.

an operator to define the relation between an incomplete database and every possible completion.

In this paper, we focus on the property of k-freeness [2]. This property is at the core of frequent pattern mining, association rules building, and more generally *condensed representations* of frequent patterns [3] which enable multiple uses of frequent patterns. Our main contribution is to propose a new definition of the k-freeness property in incomplete data which is fully compatible with the usual one in a database without missing values. This new definition certifies that the extracted patterns satisfying this definition in an incomplete database are k-free in the corresponding complete database and, in fact, in every completion of the incomplete database. Moreover, this approach leads to an anti-monotone criterion with respect to the pattern inclusion and thus allows to design an efficient level-wise algorithm which extracts k-free patterns in presence of missing values. This work is a first step towards classification in incomplete databases with generalized associations and its application to missing values imputation.

The presentation is organized as follows: Section 2 gives the background about the k-freeness of patterns and the generalized association rules. Section 3 briefly shows the damages caused by the missing values and presents our position statement. We define in Section 4 the computation of k-free patterns in presence of missing values and demonstrate that these patterns are correct in the corresponding complete database. Experiments on benchmark data confirm the effectiveness of our method (Section 5).

2 k-Free Patterns and Generalized Association Rules

In this section, we introduce the k-free patterns and the generalized association rules which stem from these patterns.

2.1 Preliminaries

Let us consider a database which gathers *objects* depicted by quantitative or qualitative *attributes* in an *attribute/values* format (see Table 1). Eight objects are described by three attributes X_1, X_2 and X_3. In the field of boolean pattern mining, qualitative attributes need to be discretized in order to get boolean contexts (this article does not discuss this stage).

Let r be a database and $(\mathcal{A}, \mathcal{O}, R)$ a *boolean context* where $\mathcal{O}$ is the set of objects, $\mathcal{A}$ is the set of attributes and R is a binary relation. An object is a subset of $\mathcal{A}$ (for example, $o_1 = \{a_1, a_3, a_5\}$) and it will be denoted as a string (i.e., $a_1a_3a_5$). $|r|$ is the number of objects in r, *i.e.* $|r| = |\mathcal{O}|$. Table 2 indicates the boolean context where X_3 is coded by the attributes a_5 to a_7.

A *pattern* X is a subset of $\mathcal{A}$, its *support* is the set of objects containing X (we denote $supp(X) = r_X = \{o \in \mathcal{O} \mid X \subseteq o\}$) and its *frequency* $\mathcal{F}(X) = |supp(X)|$ is the number of objects in the support. A *classical* association rule [4] is an expression $X \rightarrow Y$, where X and Y are two patterns. It is quantified by its frequency (i.e., $\mathcal{F}(X \cup Y)$) and its confidence: $conf(X \rightarrow Y) = \mathcal{F}(X \cup Y)/\mathcal{F}(X)$.

Table 1. Attribute/value format database

	attributes		
objects	X_1	X_2	X_3
o_1	+	→	0.2
o_2	−	→	0
o_3	+	→	0.1
o_4	+	←	0.4
o_5	−	→	0.6
o_6	−	→	0.5
o_7	+	←	1
o_8	−	←	0.8

Table 2. Boolean context r

	attributes						
objects	a_1	a_2	a_3	a_4	a_5	a_6	a_7
o_1	×		×		×		
o_2		×	×		×		
o_3	×		×		×		
o_4	×			×		×	
o_5		×	×			×	
o_6		×	×			×	
o_7	×			×			×
o_8		×		×			×

2.2 Generalized Association Rules and k-Freeness

We start by recalling generalized patterns [2] because they are at the core of the generalized association rules. A *generalized pattern* is made of boolean attributes and negations of boolean attributes. For example, the generalized pattern $Z = a_1\overline{a_2}a_3$ can be written as the union of a *positive* part $X = a_1a_3$ and a *negative* one $\overline{Y}$ where $Y = a_2$. An object o supports $Z = X \cup \overline{Y}$ if $X \subseteq o$ and $Y \cap o = \emptyset$. To alleviate the notations, we omit the union sign in the following and write $X\overline{Y}$ instead of $X \cup \overline{Y}$. $\mathcal{F}(X\overline{Y})$ is central: if it is null, one element of Y is always present with X and ensures a generalized association between X et Y. These associations lead to the *generalized association rules* introduced in [2] which are a generalized form of association rules. The originality of these rules (also called *disjunctive rules*) is to conclude on a disjunction of attributes as indicated by Definition 1, which comes from [2].

Definition 1. *A generalized association rule based on $Z = X \cup Y$ is an expression $X \rightarrow \vee Y$ where X and Y are two classical patterns. It is exact in a database r if every object of r containing the premise X also contains one attribute of the conclusion Y. We denote $\models_r X \rightarrow \vee Y \iff \mathcal{F}(X\overline{Y}, r) = 0$.*

We define the frequency of a generalized association rule as follows (this definition diverges from that of the classical association rules).

Definition 2. *The frequency $\mathcal{F}(X \rightarrow \vee Y)$ of $X \rightarrow \vee Y$ is the number of objects containing X and at least one attribute of Y. We get $\mathcal{F}(X \rightarrow \vee Y) = \mathcal{F}(X) - \mathcal{F}(X\overline{Y})$.*

Let us move now to k-free patterns. They have been proposed[1] by Calders and Goethals [2], and they are very useful to compute the generalized association rules. A k-free pattern expresses the absence of correlation between its attributes:

Definition 3 (k-free pattern). *A pattern Z is k-free in a complete database r (without missing values) and we denote $kFree(Z, r)$ if it does not exist any*

[1] With $k = 2$, these patterns have been introduced by [5] with the term of *disjunction-free sets*.

generalized association rules based on Z in r, or: $\forall X \cup Y = Z,\ |Y| \leq k \Rightarrow \mathcal{F}(X\overline{Y}) \neq 0$.

The k-free patterns have excellent properties to sum up the collections of frequent patterns. For example, in the `mushroom` dataset [6], there are $2.7 \cdot 10^9$ present patterns, but 426,134 1-free and 224,154 2-free patterns. With k higher than 5, the number of k-free patterns keeps under 214,530, and they are mined in two minutes. Until now, k-free patterns have mostly been employed to compute condensed representations of frequent patterns [3] but they get meaningful properties to produce rules. In particular, 1-free patterns are used to compute the non redundant classical association rules [7,8]. The premise of such a rule is a 1-free X and its conclusion is the GALOIS closure $h(X)$.

2.3 Generalized Association Rules Mining

The exhibition of non redundant generalized association rules is more complex. We indicate two techniques. The first one mines the 1-free patterns and then computes their *generalized closure* [9]. It gathers all minimal patterns Y sharing one attribute with every object containing X, it is obtained by computing the minimal transversals [10] of these objects [9]. The second technique takes benefit from the anti-monotonicity of the k-freeness and the border theory of this property [11]. The rules are built from the minimal non k-free patterns, which constitute the *negative border of the k-free patterns*. Generalized association rules stem from non k-free patterns (such a rule $X \rightarrow \vee Z \backslash X$ is built from a non k-free pattern Z where X is the smallest subset of Z such that $\mathcal{F}(X\overline{Z \backslash X}) = 0$).

Generalized association rules convey correlations with a richer formalism than the classical ones. They enable new uses such as supervised classification [12] based on positive and negative rules [13] (i.e., rules concluding on an attribute or its negation). For example, the rule $a_1 \rightarrow a_4 \vee a_5$ is exact in the Table 2 data and leads to the positive rule $a_1\overline{a_4} \rightarrow a_5$ and the negative one $\overline{a_4 a_5} \rightarrow \overline{a_1}$.

From the computation point of view, k-freeness is an anti-monotone property and these patterns can be efficiently mined thanks to the level-wise framework [11]. In order to check if a candidate pattern is k-free during the scan stage, the frequency of $X\overline{Y}$ is computed with the inclusion-exclusion principle [14], by using the frequencies of the subsets of XY: $\mathcal{F}(X\overline{Y}) = \sum_{\emptyset \subseteq J \subseteq Y} (-1)^{|J|} \mathcal{F}(XJ)$. As we have seen that in practice k remains low, the difficulty of computing the supports with the inclusion-exclusion principle is bearable.

3 Missing Values

We show here the damages due to the missing values and we give our position statement to solve this pattern mining problem.

3.1 Damages of Missing Values on k-Free Patterns

Assuming that some attributes of the dataset given in Table 1 are unknown, then missing values appear. We use the character `'?'` to denote that a value is neither

Table 3. Incomplete DB r'

	attributes						
objects	a_1	a_2	a_3	a_4	a_5	a_6	a_7
o_1	×		×		×		
o_2		×	×		×		
o_3	×		×		**?**	**?**	**?**
o_4	×			×		×	
o_5		×	×			×	
o_6	**?**	**?**	×			×	
o_7	×			×			×
o_8		×	**?**	**?**			×

Table 4. 1-free 2-frequent patterns and their closures

Complete DB r				Incomplete DB r'			
X	$h(X)$	X	$h(X)$	X	$h(X)$	X	$h(X)$
a_1		a_1a_3	a_5	a_1		a_1a_3	
a_2		a_1a_4		a_2		a_1a_5	
a_3		a_1a_5	a_3	a_3		a_2a_3	
a_4		a_2a_3		a_4	a_1	a_2a_5	a_3
a_5	a_3	a_2a_6	a_3	a_5	a_3	a_2a_6	a_3
a_6		a_3a_6	a_2	a_6		a_3a_6	
a_7	a_4			a_7		a_4a_7	

present nor absent for *every* boolean attribute coming from the corresponding attribute in the original database. We have introduced three missing values in our running example and the database r' resulting from this operation is indicated in Table 3.

The usual support computation for a pattern X in an incomplete database is realized as follows: an object belongs to the support of X if all of its attributes are present in X. If one of its attributes is missing or absent, the object does not belong to the support. How to compute the supports for generalized patterns in presence of missing values? Definition 3 does not plan this situation and the problem is particularly accurate for computing the frequency of $X\overline{Y}$. Without any recommendation, computations are performed by ignoring the missing values (i.e, they are not taken into account).

Table 4 depicts this problem. This table gives the 1-free patterns with a minimum support of two objects. The left part relates the results in the complete database, the right part in the incomplete one. For each pattern, the closure is indicated. The right part lists the 1-free patterns of r': a_1a_4 is 1-free in r and no longer in r'. Furthermore, the right part includes patterns, such as a_2a_5 and a_4a_7, which are not in r: we qualify them as *incorrect*.

Missing values lead to damages both on free patterns and their closures. Assuming that an attribute a belongs to X's closure in the complete database: it means that a is always present with X. If missing values appear on a, this association may break broken for some objects: a goes out from the closure (damage on the closure) and Xa can become free (damage on the free pattern). In our example, a_4 is in a_7's closure in r, while it goes out from this closure in r' because of the missing value in the object o_8. Thus, a_4a_7 is incorrectly declared 1-free.

Experiments on benchmarks from the UCI [6] emphasize these damages as well. Starting from a complete database, we artificially introduce missing values according to a uniform probability. Then we mine the 3-free patterns and measure the number of incorrect patterns relatively to the number of correct patterns in the original context (cf. Figure 1). The number of incorrect patterns differs according to the databases. It is less than 10% for the datasets `pima`, `wine`, `liver-disorders`, `servo` and `tic-tac-toe` (the corresponding chart is

not reported). For the datasets given on the left part of the figure, the number of incorrect patterns is between 10 and 90% of the number of exact patterns. In the right part, this quantity rises 300%, which means that for four computed patterns, three are incorrect.

In real conditions where the complete database is not known, it is impossible to differentiate good and bad patterns, and to foresee if a small or a big proportion of incorrect patterns will appear. Our work aims at avoiding the damages by correctly computing the k-free patterns in incomplete contexts.

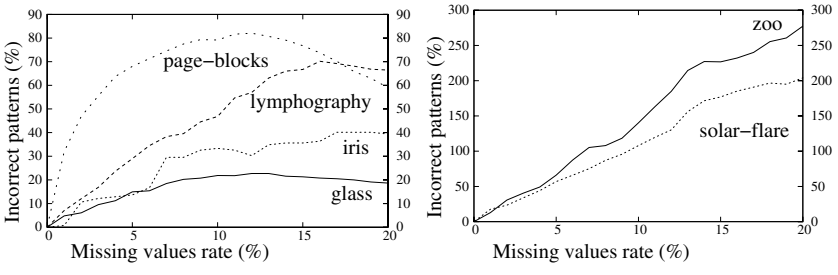

Fig. 1. Incorrect 3-free patterns in UCI datasets

3.2 Position of Our Work

There are several works which address the missing values issue in databases [15,16] but contributions in the field of data mining are few. Arnaud Ragel [17] studied association rules mining in presence of missing values by redefining the support and the confidence, these rules may be used to a *completion* (or *imputation*) goal. We will clearly state the differences with our work in Section 4.3). More recently, [18] gives a basic completion method, founded on the probability of the different attributes. The support of a pattern for an object is no longer boolean but probabilistic. [19] computes prediction rules in the complete part of a database. These rules provide intervals for continuous attributes.

Our work stems from the following principles:

- we do not want to impute the missing values before the knowledge discovery stage, because it is a difficult operation without any specific knowledge.
- we wish to mine the whole incomplete database without reducing it to its complete part. It means that we do not want to remove objects or attributes.

We do not assume any statistical hypothesis about the probability model of the missing values. In order to deal with missing values, the next section defines a *modeling operator* $mv()$. We will see that this formalization is useful because it allows to define an incomplete database as the result of an operation removing some values from the complete database. Then computations performed in an incomplete database can characterize properties which are common to every corresponding complete database.

The following shows that it is possible to discover valid knowledge for the complete database under these hypothesizes. As stated in introduction, this principle is not surprising: if we consider that missing values hide the true values of the data, the frequencies of some patterns will only decrease (we do not know for some objects if they are present). A frequent pattern in an incomplete database only can be *a fortiori* frequent in the complete dataset. We will use the same principle to compute correct k-free patterns in presence of missing values.

4 Mining k-Free Patterns in Incomplete Databases

We propose here a definition of the k-freeness property in an incomplete database. We show that it enables to compute patterns ensuring the property of freeness in every completion.

4.1 Missing Values Modeling Operator

As previously explained, our position for the missing value problem requires a modeling operator. It defines the relation between an incomplete database and every possible completion.

Definition 4 (Missing values modeling operator). *Let $r = (\mathcal{A}, \mathcal{O}, R)$ be a boolean context. An operator $mv()$ is named a missing values modeling operator if it transforms a complete database r in $mv(r) = (\mathcal{A}, \mathcal{O}, mv(R))$. The new binary relation $mv(R)$ takes its values in $\{present, absent, missing\}$ and satisfies the following properties, for every attribute a in $\mathcal{A}$, every object o in $\mathcal{O}$, and $value \in \{present, absent\}$:*

1. $mv(R)(a, o) = value \Rightarrow R(a, o) = value$;
2. $R(a, o) = value \Rightarrow mv(R)(a, o) \in \{value, absent\}$;

Section 3.1 showed that computing the k-free patterns without precaution leads to incorrect patterns. In our work, we *correctly* define the computation of the k-freeness property:

Definition 5 (k-correct pattern). *Let r' be an incomplete database and $mv()$ a modeling operator for the missing values. A pattern Z is k-correct in r' if for every complete database r, $(mv(r) = r') \Rightarrow kFree(Z, r)$.*

4.2 Temporarily Deactivating Objects

We introduce here the *deactivation* of objects in an incomplete database. It differentiates on the one hand the objects which support or not a given pattern, and on the other hand the incomplete objects where the decision of support can not be taken. The deactivation enables to quantify the frequency gap between the complete and the incomplete database. In presence of missing values, the frequencies can indeed only decrease. In our example (Table 2), $\mathcal{F}(a_3a_5, r) = 3$

but $\mathcal{F}(a_3a_5, mv(r)) = 2$ (Table 3 with $r' = mv(r)$). In order to correctly compute the frequency of a pattern X in $mv(r)$, it is necessary to differentiate the objects of $mv(r)$ having a missing value among the attributes of X. These objects will be temporarily deactivated in order to compute an estimation of $supp(X, r)$ with the help of $supp(X, mv(r))$, because it is impossible to decide if they do contain X or not.

Definition 6 (Deactivated object). *For a classical pattern $X \subseteq \mathcal{A}$, an object $o \in \mathcal{O}$ is deactivated if $\forall a \in X, mv(R)(a, o) \neq absent$ and $\exists a \in X$ s.t. $mv(R)(a, o) = missing$. We denote $\mathcal{DES}(X, mv(r))$ for the objects of $mv(r)$ deactivated for X.*

Figure 2 exemplifies the notion of deactivation, by simultaneously presenting the complete database r (on the left) and the incomplete one $mv(r)$ (on the right). We suppose that each object of the top part contains X and this part is named r_X. The down part is named $r_{\overline{X}}$.

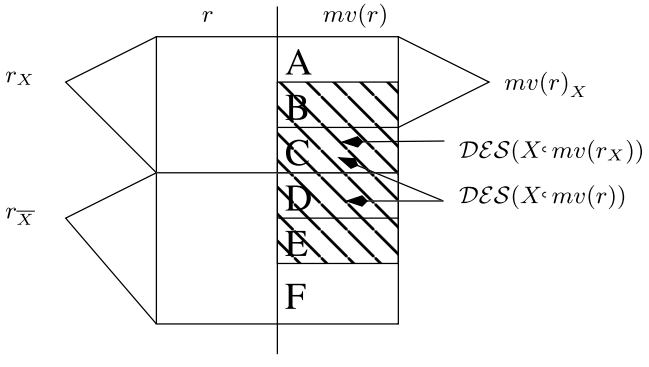

Fig. 2. Database $mv(r)$ and deactivated objects for X

On the right, the hatched zone shows the objects of $mv(r)$ which contain missing values. It is composed of six sets of objects, which are described below (their composition is indicated for our example of Table 3, with $X = a_2a_3$):

Region A: (o_2, o_5) the objects without missing value, containing X;
Region B: (no object in our example) the objects initially containing X, whose missing values do not obscure the presence of X. These objects belong to $mv(r)_X$;
Region C: (o_6) the objects initially containing X, whose missing values hide the presence of X and constitute $\mathcal{DES}(X, mv(r_X))$;
Region D: (o_8) the objects not containing X in the complete database, but which could contain it with a suitable imputation of the missing values. The object o_8 does not contain the pattern a_2a_3 in the complete database of our example, and it is preventively deactivated;

Region E: (o_3) the incomplete objects not containing X in the original dataset nor after any imputation of the missing values;
Region F: (o_1, o_4, o_7) the complete objects which do not contain X.

In the incomplete database $mv(r)$, each object is assigned in three different groups for deciding the support of X:

Regions A and B: $mv(r)_X$ the objects supporting X, in spite of the missing values of B;
Regions C and D: $\mathcal{DES}(X, mv(r))$ the objects where the support of X is undecidable;
Regions E and F: the objects not supporting X.

The deactivation allows to precisely characterize the support difference between the incomplete database and the complete one:

Proposition 1. *Let X be a classical pattern, r a database and mv a modeling operator.* $\mathcal{DES}(X, mv(r_X)) = r_X \setminus mv(r)_X$ *and* $|\mathcal{DES}(X, mv(r_X))| = \mathcal{F}(X, r) - \mathcal{F}(X, mv(r))$.

Let us detail this principle for our example and the pattern a_2a_3: $r_{a_2a_3} = \{o_2, o_5, o_6\}$ and its frequency is 3. In the incomplete database, its frequency is 2 and $\mathcal{DES}(a_2a_3, mv(r_{a_2a_3})) = \{o_6\}$: we have the equality of Proposition 1. If the complete dataset r is not known, r_X is neither known, nor $|\mathcal{DES}(X, mv(r_X))|$. But the support can be bounded with considering the deactivated objects in $mv(r)$ instead of $mv(r_X)$, because this database contains more objects than $mv(r_X)$. In our example $\mathcal{DES}(a_2a_3, mv(r)) = \{o_6, o_8\}$ because of the confusion induced in o_8 by the missing value on a_3 and a_4. $\mathcal{F}(a_2a_3, r)$ is then between $\mathcal{F}(a_2a_3, mv(r))$ and $\mathcal{F}(a_2a_3, mv(r)) + |\mathcal{DES}(a_2a_3, mv(r))|$, i.e. between 2 and 4.

In the following, it is necessary to define the deactivation for the generalized patterns. For that purpose, we use the inclusion-exclusion principle:

Definition 7 (Generalized deactivation)
$des(X\overline{Y}, mv(r_{X\overline{Y}})) = \sum_{\emptyset \subseteq J \subseteq Y} (-1)^{|J|} |\mathcal{DES}(XJ, mv(r_{XJ}))|$.

The set $\mathcal{DES}(X\overline{Y}, mv(r_{X\overline{Y}}))$ is not defined, so we denote the generalized deactivation with lower cases: $des(X\overline{Y}, mv(r_{X\overline{Y}}))$. It allows nevertheless to quantify the frequency difference between the complete and the incomplete database.

Proposition 2. $des(X\overline{Y}, mv(r_{X\overline{Y}})) = \mathcal{F}(X\overline{Y}, r) - \mathcal{F}(X\overline{Y}, mv(r))$.

This frequency gap can be negative. When the association between X and Y exists in the complete database ($\mathcal{F}(X\overline{Y}, r) = 0$), one missing value can delete it in the incomplete one ($\mathcal{F}(X\overline{Y}, mv(r)) > 0$). In this case, the difference is negative. In our example, $des(a_7\overline{a_4}) = 0 - 1 = -1$.

For the deactivated objects regarding an association $X \rightarrow \vee Y$, we define $|\mathcal{DES}(X \rightarrow \vee Y, mv(r_{X \rightarrow \vee Y}))| = |\mathcal{DES}(X, mv(r_X))| - des(X\overline{Y}, mv(r_{X\overline{Y}}))$. We then have a similar behavior as emphasized in Propositions 1 and 2:
$|\mathcal{DES}(X \rightarrow \vee Y, mv(r_{X \rightarrow \vee Y}))| = \mathcal{F}(X \rightarrow \vee Y, r) - \mathcal{F}(X \rightarrow \vee Y, mv(r))$.

Moreover, an object is deactivated for an association $X \rightarrow \vee Y$ if it is deactivated for X, or if it contains X but every attribute of Y is missing. Denoting $\mathcal{DES}(\wedge Y, mv(r_{X\rightarrow\vee Y})_X)$ for these objects, we have $|\mathcal{DES}(X \rightarrow \vee Y, mv(r_{X\rightarrow\vee Y})) = |\mathcal{DES}(X, mv(r_{X\rightarrow\vee Y}))| + |\mathcal{DES}(\wedge Y, mv(r_{X\rightarrow\vee Y})_X)|$.

4.3 Differences with Ragel's Approach

In this section, we set our deactivation principle with respect to A. Ragel's work [17,20]. To compute $\mathcal{F}(X, mv(r))$, A. Ragel deactivates all objects containing a missing value in X without regarding whether an object may support X. In our example, o_6 would be deactivated for a_2a_4 because a_2 is missing. Nevertheless, o_6 cannot support a_2a_4 because o_6 does not contain a_4. It means that there is no complete database where o_6 can support a_2a_4. This observation led M. Kryszkiewicz to propose a new definition [21]. In our paper, we use this definition (cf. Definition 6).

4.4 k-Freeness Definition and Correction in Incomplete Databases

With the help of the deactivation of the incomplete objects, the frequency of $X\overline{Y}$ in r can be bounded by two quantities which are computed in $mv(r)$:

Property 1. $\mathcal{F}(X\overline{Y}, mv(r)) - |\mathcal{DES}(\wedge Y, (mv(r))_X)| \leq \mathcal{F}(X\overline{Y}, r) \leq \mathcal{F}(X\overline{Y}, mv(r)) + |\mathcal{DES}(X, mv(r))|$.

Proof. Proposition 2 says that $\mathcal{F}(X\overline{Y}, r) = \mathcal{F}(X\overline{Y}, mv(r)) + des(X\overline{Y}, mv(r_{X\overline{Y}}))$. The deactivation of an association allows to write $des(X\overline{Y}, mv(r_{X\overline{Y}})) = |\mathcal{DES}(X, mv(r_X))| - |\mathcal{DES}(X\rightarrow\vee Y, mv(r_{X\rightarrow\vee Y}))|$. On one hand, we have the upper bound $des(X\overline{Y}, mv(r_{X\overline{Y}})) \leq |\mathcal{DES}(X, mv(r_X))|$, and when avoiding the restriction on the deactivation database, $des(X\overline{Y}, mv(r_{X\overline{Y}})) \leq |\mathcal{DES}(X, mv(r))|$. On the other hand, we break up $des(X\overline{Y}, mv(r_{X\overline{Y}})) = |\mathcal{DES}(X, mv(r_X))| - (|\mathcal{DES}(X, mv(r_{X\rightarrow\vee Y}))| + |\mathcal{DES}(\wedge Y, mv(r_{X\rightarrow\vee Y})_X)|) = (|\mathcal{DES}(X, mv(r_X))| - |\mathcal{DES}(X, mv(r_{X\rightarrow\vee Y}))|) - |\mathcal{DES}(\wedge Y, mv(r_{X\rightarrow\vee Y})_X)|$. The difference $|\mathcal{DES}(X, mv(r_X))| - |\mathcal{DES}(X, mv(r_{X\rightarrow\vee Y}))|$ is positive so we have the lower bound $des(X\overline{Y}, mv(r_{X\overline{Y}})) \geq |\mathcal{DES}(\wedge Y, mv(r_{X\rightarrow\vee Y})_X)|$. Without the restriction on the deactivation database, $des(X\overline{Y}, mv(r_{X\overline{Y}})) \geq |\mathcal{DES}(\wedge Y, mv(r)_X)|$.

The k-freeness property can be defined in incomplete databases with the bounds for the frequency of $X\overline{Y}$.

Definition 8 (k-freeness in incomplete databases)

- *A pattern Z is k-free in $mv(r)$ and we denote $kFree(Z, mv(r))$ if and only if* $\forall XY = Z,\ |Y| \leq k,\ \mathcal{F}(X\overline{Y}, mv(r)) - |\mathcal{DES}(\wedge Y, (mv(r))_X)| > 0$.
- *A pattern Z is k-dependent in $mv(r)$ and we denote $kDepdt(Z, r)$ if and only if* $\exists XY = Z,\ |Y| \leq k,\ \mathcal{F}(X\overline{Y}, mv(r)) + |\mathcal{DES}(X, mv(r))| = 0$.

k-freeness and k-dependence are independently introduced. Section 4.5 will justify this distinction because these definitions are not reverse, due to the missing values.

Let us first note that, in a complete database, our definition of the k-freeness is *compatible* with the classical Definition 3. In this case, the set of deactivated objects is empty when there is no missing values. It is an important point in order to design algorithms which work indifferently on complete or incomplete contexts.

The k-freeness in an incomplete database is linked to this in a complete database with the important following theorem:

Theorem 1 (k-freeness correction). *Let r' be an incomplete database and $mv()$ a missing values modeling operator. For every complete database r such that $mv(r) = r'$ and every pattern Z,*
- $kFree(Z, r') \Longrightarrow kFree(Z, r)$;
- $kDepdt(Z, r') \Longrightarrow \neg kFree(Z, r)$.
The k-free patterns of r' are k-correct.

Proof. Property 1 shows that $\mathcal{F}(X\overline{Y}, r)$ is bounded by $\mathcal{F}(X\overline{Y}, r') - |\mathcal{DES}(\wedge Y, r'_X)|$ and $\mathcal{F}(X\overline{Y}, r') + |\mathcal{DES}(X, r')|$. If the lower bound is strictly positive, $\mathcal{F}(X\overline{Y}, r)$ is also strictly positive then non null and the pattern is k-free in r. If the upper bound is null, $\mathcal{F}(X\overline{Y}, r)$ is null and the pattern is not k-free in r.

Computed with Definition 8, the k-free patterns are then k-correct, i.e. they are k-free in every database completion. In [22,23], this correction is shown for the particular case when $k = 1$. These definitions of the k-freeness and the k-dependence allow to compute properties which are true in every completion: our definitions are **correct**. They are also *complete* because they characterize all k-free patterns in every completion:

Theorem 2 (k-freeness completeness). *Let r' be an incomplete database. If Z is k-free in every complete database r such that there exists a modeling operator $mv()$ with $mv(r) = r'$, then Z is k-free in r': the k-correct patterns of r' are k-free in r'.*

Proof. Suppose the converse, i.e. let Z be k-free in every database r such that $mv(r) = r'$ but non k-free in r'. $\exists XY = Z \mid \mathcal{F}(X\overline{Y}, r') - \mathcal{DES}(\wedge Y, r'_X) \leq 0$. Let r_0 be the database stemming from r' with replacing each missing value by an absent value, then $mv(r_0) = r'$. In r_0, the deactivation is null because r_0 is complete, and the computation of $\mathcal{F}(X\overline{Y}, r_0)$ gives the same result as in r' where it is done with the frequencies of the present attributes. $\mathcal{F}(X\overline{Y}, r_0)$ is then null and Z is not k-free is r_0 : contradiction.

In an incomplete database, every computed k-free pattern is k-correct and every pattern which is k-free in every completion of the database is covered by this definition.

4.5 Properties of the k-Freeness in Incomplete Databases

The k-freeness and the k-dependency are not complementary: some patterns will be neither k-free nor k-dependent because it is sometimes impossible to decide

if they are present or not in an object. The table below details the computation of 1-freeness for the pattern a_4a_7:

X	Y	$\mathcal{F}(X\overline{Y}, mv(r))$	$\lvert\mathcal{DES}(\wedge Y, mv(r)_X)\rvert$	$\lvert\mathcal{DES}(X, mv(r))\rvert$	1-free?	1-dependent?
a_4	a_7	1	1	1	$1-1 \not> 0$: no	$1+1 \neq 0$: no
a_7	a_4	1	1	1	$1-1 \not> 0$: no	$1+1 \neq 0$: no

We now give a vital property for designing k-free patterns mining algorithms. It refers to the (anti)-monotonicity of the k-freeness or dependency. The k-freeness does not satisfy a property of (anti-)monotonicity, but Theorem 3 indicates that the k-dependency is monotone.

Theorem 3 (Monotonicity of the k-dependency property). *The k-dependency property is monotone, i.e. for all patterns Z and every database r', $Z \subseteq Z' \Rightarrow (kDepdt(Z, r')$*
$\Rightarrow kDepdt(Z', r'))$.

Proof. Let Z be a k-dependent pattern. $\exists XY = Z$, $\mathcal{F}(X\overline{Y}, mv(r)) + \lvert\mathcal{DES}(X, mv(r))\rvert = 0$ or $\mathcal{F}(X\overline{Y}, mv(r)) = 0$ and $\lvert\mathcal{DES}(X, mv(r))\rvert = 0$. $\mathcal{F}(X\overline{Y}, mv(r)) = 0$ means that for all object $o \in \mathcal{O}$, $X \subseteq o \Rightarrow Y \cap o \neq \emptyset$. *A fortiori*, $X \subseteq o \Rightarrow aY \cap o \neq \emptyset$ for all $a \in \mathcal{A}$, then $\mathcal{F}(X\overline{aY}, mv(r)) = 0$. By induction on all attributes of $Z' \backslash Z$, one deduces that Z' is also k-dependent.

With this result, the framework of the level-wise algorithms can be used with the negation of the k-dependency constraint, and we have written the `MV-k-miner` prototype.

4.6 Prototype `MV-k-miner`

We have designed the `MV-k-miner` prototype which returns the correct and complete collection of k-free patterns. It is based on the mining of k-dependent patterns. It runs on a two-step process: a classical level-wise scan of the search space (Algorithm 1) and the candidate process phase (Algorithm 2).

`MV-k-miner` stores $\mathcal{DES}(X, mv(r))$ and $\mathcal{DES}(\wedge Y, mv(r_X))$ for each pattern. This allows to compute two couples of bounds for $\mathcal{F}(X, mv(r))$ during the generation: the first couple is based on $\mathcal{DES}(X, mv(r))$ and stands for the k-dependency. The corresponding pruning criterion is only used during the generation phase, so these bounds are not stored. The second couple is related to $\mathcal{DES}(\wedge Y, mv(r_X))$ and stands for the k-freeness. It is used both during the generation phase and the scan phase, so these bounds are stored. The memory cost is finally higher than for a classical algorithm in complete databases, but the execution time is comparable. Our implementation should take benefit of recent development in k-free pattern mining [24].

5 Experiments on UCI Benchmarks

We show here the relevance of our missing values treatment by reproducing the experiments described in Section 3.1. We measure the number of 3-free

Data : an incomplete database $mv(r)$, a frequency minimum threshold γ, and $k > 0$ a rule depth
Result : the set $\mathcal{S}$ of the patterns satisfying $kFree$
$\mathcal{D}_l$ is the set of patterns of length l, k-dependent or non frequent ;
$l = 1$; initialize $\mathcal{C}and_1$ with the singletons ;
repeat
  /*`compute the disqualifiers` */
  $\mathcal{D}_l = \{X \in \mathcal{C}and_l \; s.t. \; kDepdt(X, mv(r)) \vee \neg frequent(X, mv(r))\}$;
  /*`discard disqualiers` */
  $\mathcal{S}_l = \{X \in \mathcal{C}and_l \backslash \mathcal{D}_l \mid kFree(X, mv(r))\}$;
  generate the candidates in $\mathcal{C}and_{l+1}$ (cf. algorithm 2);
  l = l + 1 ;
until $\mathcal{C}and_l = \emptyset$;
return $\mathcal{S} = \bigcup_l \mathcal{S}_l$;

Algorithm 1. `MV-k-miner`: k-free patterns miner in incomplete databases

Data : a set $\mathcal{S}_l$ of k-free patterns with length l
Result : the set $\mathcal{C}and_{l+1}$ of the pattern candidate to $kDepdt$
for *every candidate Z, generated by two patterns in $\mathcal{S}_l$ sharing the same $l-1$-prefix* **do**
  begin
    verify that all the $Z' \subsetneq Z$ of length $|Z| - 1$ are k-free ;
    /*`compute the frequency bounds` */
    build the tree of the patterns X and their frequencies such that $|Z \backslash X| \leq k$;
    for every X in the tree, compute the alternated sum of the frequency of its subsets, that constitutes a preliminary version of $\sigma(X, Y) = \sum_{\emptyset \subseteq J \subsetneq Y} (-1)^{|J|} \mathcal{F}(XJ)$ for bounding $\mathcal{F}(Z)$;
    compute $\sigma(X, Y) - |\mathcal{DES}(\wedge Y, mv(r_X))|$ and $\sigma(X, Y) + |\mathcal{DES}(X, mv(r))|$;
    store the bound $\sigma(X, Y) + |\mathcal{DES}(X, mv(r))|$ pour $\mathcal{F}(Z)$;
    if the bounds are equal, decline the candidate ;
  end
end

Algorithm 2. $l + 1$-candidate generation

patterns computed with `MV-k-miner` in $mv(r)$, compared to r. The results for `solar-flare` and `zoo` are reported in Figure 3. In the other datasets, the same trends appear. For each dataset, the running time of the whole experience is about 10 seconds.

As expected, the number of patterns recovered by our method decreases according to the number of missing values. Indeed, each pattern is k-correct or k-free in every complete dataset, whose number is exponential in the number of missing values. But `MV-k-miner` computes only k-correct patterns. While data mining is known to produce a huge number of patterns, their correctness is essential. Missing values damages are then avoided and this result opens the way for the uses of k-free patterns mentioned in Section 2.3. In particular, the future of this work has to address the interestingness of the correct k-free patterns

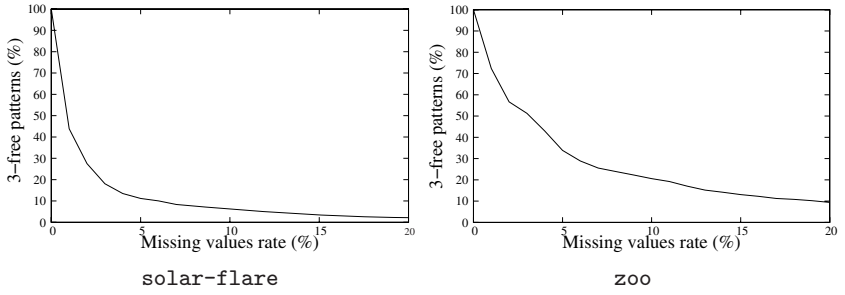

Fig. 3. Proportion of 3-free patterns in $mv(r)$ (r gives 100 %)

during a supervised classification process, according to the approach based on minimal k-free patterns, as outlined in Section 2.3.

6 Conclusion

Without any suitable treatment, missing values in incomplete databases lead the k-free pattern mining algorithms to produce incorrect patterns. With the help of a modeling operator, we have introduced the notion of k-correct patterns in an incomplete database. These patterns are k-free in every corresponding complete database. We have then proposed a new definition for the k-free property in an incomplete database. Thanks to this new definition, the mined patterns are k-correct, and all k-correct patterns are mined: this avoids damages due to missing values.

Our perspectives address now the classification with generalized associations and its application to missing values imputation. The first step of this project consists in studying how to compute generalized closures in incomplete databases.

References

1. Grzymala-Busse, J., Hu, M.: A comparison of several approaches to missing attribute values in data mining. In: Ziarko, W., Yao, Y. (eds.) RSCTC 2000. LNCS (LNAI), vol. 2005, pp. 378–385. Springer, Heidelberg (2001)
2. Calders, T., Goethals, B.: Minimal k-free representations of frequent sets. In: Lavrač, N., Gamberger, D., Todorovski, L., Blockeel, H. (eds.) PKDD 2003. LNCS (LNAI), vol. 2838, pp. 71–82. Springer, Heidelberg (2003)
3. Calders, T., Goethals, B.: Mining all non-derivable frequent itemsets. In: Elomaa, T., Mannila, H., Toivonen, H. (eds.) PKDD 2002. LNCS (LNAI), vol. 2431, Springer, Heidelberg (2002)
4. Agrawal, R., Srikant, R.: Fast algorithms for mining association rules. In: Intl. Conference on Very Large Data Bases (VLDB'94), Santiago de Chile, Chile, pp. 487–499 (1994)
5. Bykowski, A., Rigotti, C.: A condensed representation to find frequent patterns. In: ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems, Santa Barbara, USA, pp. 267–273. ACM Press, New York (2001)

6. Blake, C., Merz, C.: UCI repository of machine learning databases (1998)
7. Bastide, Y., Taouil, R., Pasquier, N., Stumme, G., Lakhal, L.: Mining minimal non-redundant association rules using frequent closed itemsets. In: International Conference on Deductive and Object Databases (DOOD'00), pp. 972–986 (2000)
8. Zaki, M.: Generating non-redundant association rules. In: ACM SIGKDD international conference on Knowledge discovery and data mining, Boston, USA, pp. 34–43. ACM Press, New York (2000)
9. Rioult, F.: Extraction de connaissances dans les bases de données comportant des valeurs manquantes ou un grand nombre d'attributs. PhD thesis, Université de Caen Basse-Normandie, France (2005)
10. Gunopulos, D., Mannila, H., Khardon, R., Toivonen, H.: Data mining, hypergraph transversals, and machine learning. In: ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS'97), Tucson, USA, ACM Press, New York (1997)
11. Mannila, H., Toivonen, H.: Levelwise search and borders of theories in knowledge discovery. Data Mining and Knowledge Discovery 1(3), 241–258 (1997)
12. Antonie, M.L., Zaïane, O.: An associative classifier based on positive and negative rules. In: ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery (DMKD'04), Paris, France, ACM Press, New York (2004)
13. Antonie, M.L., Zaïane, O.: Mining positive and negative association rules: An approach for confined rules. In: Boulicaut, J.-F., Esposito, F., Giannotti, F., Pedreschi, D. (eds.) PKDD 2004. LNCS (LNAI), vol. 3202, pp. 27–38. Springer, Heidelberg (2004)
14. Jaroszewicz, S., Simovici, D.: Support approximations using bonferroni-type inequalities. In: Elomaa, T., Mannila, H., Toivonen, H. (eds.) PKDD 2002. LNCS (LNAI), vol. 2431, pp. 212–224. Springer, Heidelberg (2002)
15. Dyreson, C.E.: A Bibliography on Uncertainty Management in Information Systems. In: Uncertainty Management in Information Systems, Kluwer Academic Publishers, Dordrecht (1997)
16. Levene, M., Loizou, G.: Database design for incomplete relations. ACM Transactions on Database Systems 24(1), 80–126 (1999)
17. Ragel, A., Crémilleux, B.: Mvc - a preprocessing method to deal with missing values. Knowledge-Based Systems 12(5-6), 285–291 (1999)
18. Nayak, J., Cook, D.: Approximate association rule mining. In: Florida Artificial Intelligence Research Symposium, Key West, Florida, USA, pp. 259–263 (2001)
19. Jami, S., Jen, T., Laurent, D., Loizou, G., Sy, O.: Extraction de régles d'association pour la prédiction de valeurs manquantes. In: Colloque Africain sur la Recherche en Informatique (CARI) (2004)
20. Ragel, A., Crémilleux, B.: Treatment of missing values for association rules. In: Wu, X., Kotagiri, R., Korb, K.B. (eds.) PAKDD 1998. LNCS, vol. 1394, pp. 258–270. Springer, Heidelberg (1998)
21. Kryszkiewicz., M.: Association rules in incomplete databases. In: Zhong, N., Zhou, L. (eds.) Methodologies for Knowledge Discovery and Data Mining. LNCS (LNAI), vol. 1574, pp. 84–93. Springer, Heidelberg (1999)
22. Rioult, F., Crémilleux, B.: Condensed representations in presence of missing values. In: Berthold, M.R., Lenz, H-J., Bradley, E., Kruse, R., Borgelt, C. (eds.) IDA 2003. LNCS, vol. 2810, pp. 578–588. Springer, Heidelberg (2003)
23. Rioult, F., Crémilleux, B.: Représentation condensée en présence de valeurs manquantes. In: XXIIé congrés Inforsid, Biarritz, France, pp. 301–317 (2004)
24. Calders, T., Goethals, B.: Quick inclusion-exclusion. In: Bonchi, F., Boulicaut, J.-F. (eds.) KDID 2005. LNCS, vol. 3933, Springer, Heidelberg (2006)