Generating an informative cover for association rules

Laurentiu Cristofor
University of Massachusetts at Boston,
Department of Computer Science,
Boston, Massachusetts 02125, USA

Dan Simovici
University of Massachusetts at Boston,
Department of Computer Science,
Boston, Massachusetts 02125, USA

Abstract

Mining association rules may generate a large numbers of rules making the results hard to analyze manually. Pasquier et al. have discussed the generation of Guigues-Duquenne–Luxenburger basis (GD-L basis). Using a similar approach, we introduce a new rule of inference and define the notion of association rules cover as a minimal set of rules that are non-redundant with respect to this new rule of inference. Our experimental results (obtained using both synthetic and real data sets) show that our covers are smaller than the GD-L basis and they are computed in time that is comparable to the classic Apriori algorithm for generating rules.

1 Introduction

The number of association rules generated by data mining algorithms can easily overwhelm a human analyst. To address this problem several methods were proposed. Our paper continues the line of research from [7] by introducing a new rule of inference for association rules and by defining the concept of a cover of the association rules as a minimal set of rules that are non-redundant with respect to this new inference rule.

We use the terminology and notations of [9]. Let $\tau = (T, H, \rho)$ be a table, where T is the name of the table, $H = A_1 \dots A_n$ is the heading of the table, and $\rho \subseteq \{0,1\}^n$. We assume that each attribute have $\{0,1\}$ as its domain. The projection of a tuple on a set $L \subseteq H$ is denoted by $t[L]$. The tuple over I that consists of 1s is denoted by $\mathbf{1}_I$. An *itemset* I is a set of items $I \subseteq H$. The *support* of I, denoted by $\texttt{supp}(I)$, is given by $\texttt{supp}(I) = \frac{|\{t \in \rho | t[I] = \mathbf{1}_I\}|}{|\rho|}$. In addition, $\texttt{supp}(\emptyset)$ is defined to be 1. The *closure* of an itemset I is $\texttt{cl}(I) = \{A_i \in H | \text{if } t[I] = \mathbf{1}_I \text{ then } t[A_i] = 1\}$. An itemset is called *closed* if it is equal to its closure.

An *association rule* $X \to Y$ consists of two disjoint non-empty itemsets X and Y, called *antecedent* and *consequent* and denoted by $\texttt{antc}(r)$ and $\texttt{cons}(r)$, respectively. We refer to the items of a rule r by $\texttt{items}(r) = \texttt{antc}(r) \cup \texttt{cons}(r)$. The *support* of the association rule is the support of $\texttt{items}(r)$. The *confidence* of the association rule is the ratio:

$$\texttt{conf}(r) = \frac{\texttt{supp}(\texttt{items}(r))}{\texttt{supp}(\texttt{antc}(r))}.$$

If $\texttt{conf}(r) = 1$, then r is called an *exact* association rule and denoted by $X \Rightarrow Y$; otherwise, r is called an *approximative* association rule (see [7]). Given a table τ, a minimum support value $\texttt{minsupp}$, and a minimum confidence value $\texttt{minconf}$, we seek to generate all *valid association rules* (cf. [8]), that is, all rules with support greater or equal to $\texttt{minsupp}$ and confidence greater or equal to $\texttt{minconf}$.

To deal with the usual large number of association rules it is preferrable to generate only the association rules that cannot be inferred from other rules by using rules of inference. A minimal set of such association rules was called *basis* in [7]. To avoid confusion, we mention here that the single word "rule" will only be used in the sense of an association rule and will never be used to denote an inference rule. The *Guigues-Duquenne basis* for exact rules and the *Luxenburger basis* for approximative rules are introduced in [7]; together they form a basis for the valid rules.

The Guigues-Duquenne basis is a minimal set of exact rules from which the complete set of exact rules can be deduced using as following two inference rules: $X \Rightarrow Y, W \Rightarrow Z \vdash XW \Rightarrow YZ$, and $X \Rightarrow Y, Y \Rightarrow Z \vdash X \Rightarrow Z$. The Guigues-Duquenne basis does not allow us to infer the support of the rules and in fact, by ignoring the support values, the first inference rule can lead to rules that have inferior support compared to the rules used in its generation.

The Luxenburger basis is a minimal set of approximative rules from which the complete set of approximative rules can be deduced using the two properties introduced in [6]: (1) the association rule $X \to Y$ has the same support and confidence as the rule $\texttt{cl}(X) \to \texttt{cl}(Y)$, and (2) for any three closed itemsets X, Y, and Z, such that $X \subseteq Y \subseteq Z$, the confidence of the rule $X \to Z$ is equal to the product of the confidences of the rules $X \to Y$ and $Y \to Z$, and its support is equal to the support of the rule $Y \to Z$ Both these properties can be regarded as new inference rules and they permit the inference of both the support and confidence of the resulting rules. Together, the Guigues-Duquenne basis and the Luxenburger basis, provide a minimal basis for rules, which we will denote as the GD-L basis.

Next, we introduce a new rule of inference for rules.

Theorem 1.1 *Let r, r' be two rules such that $\mathtt{items}(r') \subseteq \mathtt{items}(r)$ and $\mathtt{supp}(\mathtt{antc}(r')) \leq \mathtt{supp}(\mathtt{antc}(r))$. Then, $\mathtt{supp}(r') \geq \mathtt{supp}(r)$ and $\mathtt{conf}(r') \geq \mathtt{conf}(r)$.*

This justifies the introduction of the inference rule:

$$\frac{r, \mathtt{items}(r') \subseteq \mathtt{items}(r), \\ \mathtt{supp}(\mathtt{antc}(r')) \leq \mathtt{supp}(\mathtt{antc}(r))}{r'}.$$

Definition 1.2 If for two rules, r_1 and r_2, it is possible to infer r_2 from r_1 using Theorem 1.1, then we say that rule r_2 is *covered* by rule r_1 (and that r_1 *covers* r_2), and we write $r_1 \prec r_2$. The *coverage relation* $\prec$ consists of all ordered pairs of rules (r_1, r_2), such that $r_1 \prec r_2$. If $r_1 \prec r_2$ and $r_2 \prec r_1$, then r_1, r_2 are said to be *equipotent*. □

Because of Definition 1.2, we will also refer to the property of Theorem 1.1 as the *coverage rule*.

Theorem 1.3 *The rules r_1 and r_2 are equipotent if and only if $\mathtt{items}(r_1) = \mathtt{items}(r_2)$ and $\mathtt{supp}(\mathtt{antc}(r_1)) = \mathtt{supp}(\mathtt{antc}(r_2))$.*

Equipotent rules are interchangeable from the point of view of the coverage relation, that is, if r_1 and r_2 are equipotent and $r_1 \prec r_3$ for some r_3, then $r_2 \prec r_3$.

2 Covers for association rules

Theorem 1.1 suggests the following definition:

Definition 2.1 Let $\mathcal{R}$ be the set of all valid rules extracted from a table τ. A *cover of* $\mathcal{R}$ is a minimal set $\mathcal{C} \subseteq \mathcal{R}$, such that any rule from $\mathcal{R} - \mathcal{C}$ is covered by a rule in $\mathcal{C}$. A rule belonging to $\mathcal{C}$ is called a $\mathcal{C}$-*cover rule*. □

Theorem 2.2 *Let $\mathcal{C}$ be a cover of a set of rules $\mathcal{R}$ extracted from a table τ. If r is a $\mathcal{C}$-cover rule, then for any $r' \in \mathcal{R}$ such that $\mathtt{items}(r) = \mathtt{items}(r')$, we have $\mathtt{supp}(r) \leq \mathtt{supp}(r')$ and $\mathtt{conf}(r) \leq \mathtt{conf}(r')$ and there is no $r_1 \in \mathcal{R}$ such that $\mathtt{antc}(r) = \mathtt{antc}(r_1)$ and $\mathtt{cons}(r) \subset \mathtt{cons}(r_1)$. Further, if $r_1, r_2 \in \mathcal{C}$, then $\mathtt{items}(r_1) \neq \mathtt{items}(r_2)$.*

Definition 2.3 An *informative cover* is a cover where for each cover rule r there is no equipotent rule r' such that $\mathtt{antc}(r') \subset \mathtt{antc}(r)$. □

Theorem 2.4 *Let $\mathcal{C}$ be an informative cover of a set of rules $\mathcal{R}$ extracted from a table τ. If r is a $\mathcal{C}$-cover rule, then there is no valid rule r' such that $\mathtt{items}(r') = \mathtt{items}(r)$ and $\mathtt{antc}(r') \subset \mathtt{antc}(r)$.*

Note that it is possible to have an informative cover rule r and a valid rule r', such that $\mathtt{items}(r) = \mathtt{items}(r')$ and $|\mathtt{antc}(r)| > |\mathtt{antc}(r')|$, as the next example shows.

A cover summarizes the set of valid rules in a similar way in which the large itemsets summarize the set of frequent itemsets [2]. A cover can also be used to simplify the presentation of rules to users: initially, only cover rules could be shown to a user, then the user could select a cover rule r and retrieve a subset of all rules covered by r, and then the process could be repeated. In this manner, the user could guide his search for rules without being overwhelmed by their number. A similar type of rule exploration has been proposed in [5], in the context of the so called *direction setting rules*.

The following pseudocode describes an algorithm for generating an informative cover for the set of valid rules.

Algorithm 2.5 (**CoverRules**) Algorithm for generating an informative cover for the valid rules.

Let R be a queue that will contain frequent itemsets and let C be the set of rules in which we will place the cover rules.

1 Initialize R by enqueuing into it all maximal frequent itemsets, in decreasing order of their size. C is $\emptyset$.

2 If R is empty, then output C and exit; else extract an itemset I from R.

3 For all strict non-empty subsets I_i of I, with $i = 1 \ldots 2^{|I|} - 2$, sorted primarily by their support values (decreasingly) and secondarily by their cardinality (increasingly), do:

    3.1 If the rule $I_i \rightarrow I - I_i$ is valid, then add it to C if it is not covered by a rule already in C. Go to step 2.

    3.2 If $i = 1$ and $|I| > 2$, then add to R each subset of I that has size $|I| - 1$ and that is not already included in an itemset from R. Continue step 3.

∎

Algorithm **CoverRules** starts from the set of maximal frequent itemsets and examines them in decreasing order of their cardinalities (steps 1–2). For each such itemset I, we search for a subset S having maximum support, such that $S \rightarrow I - S$ is a valid association rule (step 3). Such a rule is a candidate cover rule and, once found, the search stops and the rule is added to the set of cover rules C if it is not covered by one of the rules of C (step 3.1). During the examination of each subset S of I, we may encounter some subsets such that they cannot be used as an antecedent of a rule based on the items of I. For these subsets, we will have to verify whether they can be antecedents of rules based on subsets of I. This is why, in step 3.2 of the algorithm, we add to R all the subsets of I. Those subsets that are already included in an itemset of R, however, do not need to be added. Step 3.2 needs to be performed only once, so we perform it if the first subset examined in step 3 cannot be used as an antecedent. The collection R is a queue because we want to examine the maximal frequent itemsets in decreasing order of their size before we examine their subsets (added in step 3.2). We examine these itemsets in decreasing order of their size because a rule whose set of items is larger cannot be covered by a rule whose set of items is smaller. This ensures that a cover rule added to C cannot be covered by another cover rule that we may discover later. Each time that we intend to add a rule to C, however, we still need to check whether that rule can be covered by one of the rules already in C.

The strategy of examining first the maximal frequent itemsets and then their subsets, in decreasing order of their size, guarantees that the set of rules that we generate is minimal. Step 3.2 guarantees that all valid rules can be inferred from the rules in set C. Together, these ensure that the resulting set C is a minimal set of rules from which all valid rules can be inferred, and thus, C is a cover. The cover is informative because, in step 3, for subsets with same support, we examine first those with smaller cardinality.

3 Experimental results

The optimized version of **CoverRules** implemented in Java is available in ARtool [4]. We tested **CoverRules** on several databases. In a first experiment, presented in Table 1, we executed the algorithm on the Mushroom database from the UCI Repository of Machine Learning Databases [3]. Note that the UCI repository contains two versions of the Mushroom database. We have used the version containing fewer rows, which was used in the experiments of [7].

Mushroom database (`minsupp` = 30%)			
`minconf`	Valid rules	Cover	GD-L Basis
90%	20399	238	382
70%	45147	176	453
50%	64179	159	479
30%	78888	78	493

Table 1. Results for Mushroom database

The cover contains fewer rules than the GD-L basis and its size decreases as `minconf` is lowered (see also Fig. 1).

This may seem surprising at first, but is due to the fact that, as `minconf` is lowered, more valid rules exist, the re-

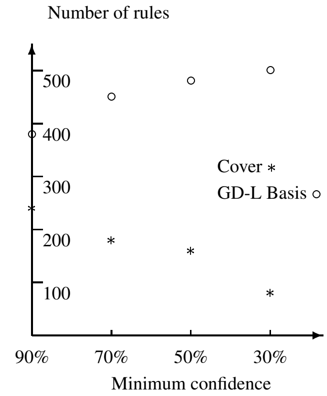

Figure 1. Comparative Results on Mushroom Database

dundancy of these rules is greater, and thus they can be summarized better. In fact, for `minconf` = 30%, the size of the cover is identical to the number of maximal frequent itemsets existing in the mushroom database (for `minsupp` = 30% there are 78 such maximal frequent itemsets), and this happens because all rules that can be generated using subsets of a maximal frequent itemset are valid. In this case, the cover size is one order of magnitude smaller than the size of the GD-L basis, and three orders of magnitude smaller than the total number of valid rules.

For `minconf` = 30%, all cover rules have the item `veil-type = partial` as antecedent. Interestingly, this item is common to all the mushrooms described in the database, so its support value is 1 — the maximum possible support value. By looking at a cover rule separately, the fact that the rule has the most frequent item as antecedent might make us think that the rule is trivial. Knowing that this is a cover rule, however, its antecedent being the most frequent item takes new meaning because it implies that any rule that we can build from the items of the cover rule will be a valid association rule. Usually, the most frequent items are known to the users of the database, so a cover rule having such an item as antecedent can be easily interpreted, even without the help of the computer.

In the case of the Mushroom database, the **CoverRules** algorithm is about as fast as the Apriori **ap-genrules** procedure for generating all valid rules, which was described in [1]. Both algorithms finished their processing in a couple of seconds, so we do not include their detailed time results here.

To test the on synthetic data we generated database SPARSE with 100,000 transactions of average size 10, having 100 items, and containing 300 patterns of average size 5. We mined SPARSE for `minsupp` = 5% and discovered 207 maximal frequent itemsets. For all our experiments on this database, the times taken by **CoverRules** and **ap-genrules** were well below 1 second so we omit them again. The number of rules discovered and the corresponding cover size are presented in Table 2.

SPARSE database (`minsupp` = 5%)		
`minconf`	Valid rules	Cover
90%	3	2
80%	19	13
70%	42	25
60%	87	55
50%	186	124
40%	321	196
30%	455	240
20%	658	257
10%	880	194
5%	880	194
1%	880	194

Table 2. Results for SPARSE database

Note that the cover size increases initially as `minconf` decreases. This happens because the database is sparse, so the redundancy is poor and rules that are discovered when the confidence threshold is lowered do not necessarily allow the inference of rules with higher confidence. For `minconf` = 10%, we obtain all valid rules and lowering the confidence threshold further does not bring any new rules. In fact, the 194 cover rules correspond to the maximal large itemsets that have cardinality greater than one, since there are 13 such maximal frequent itemsets of size one.

For our final experiment, we generated a dense synthetic database, which we will call DENSE, with 100,000 transactions of average size 15, having 100 items, and containing 100 patterns of average size 10. Our strategy for obtaining dense synthetic databases consists of choosing fewer and longer patterns. We mined this database for `minsupp` = 5% and we obtained 3,182 maximal frequent itemsets. For this experiment, the times taken by the **CoverRules** and **ap-genrules** algorithms became noticeable and we include them in Table 3.

Again, for this dense database, the cover size generally tends to decrease as we lower the confidence threshold. All valid rules are discovered for confidence 5%, so lowering `minconf` further does not result in more rules. There is only one maximum frequent itemset of size one, which accounts for the difference between the number of maximal frequent itemsets and the cover size obtained in this case. The time taken by the rule generation algorithms is more significative and allows us to notice that **CoverRules**'s performance tends to improve with the lowering of the confidence threshold, while **ap-genrules** tends to take more time as `minconf` is decreased. **ap-genrules** runs initially faster than **CoverRules**, which performs better for lower values of `minconf`. These results, however, do not include the time necessary to output the generated rules. The space requirements of **ap-genrules** are more significant than those of **CoverRules**, and in some experiments we had to increase the memory available to the Java Virtual Machine so that **ap-genrules** would not run out of memory.

DENSE database (`minsupp` = 5%)				
`minconf`	Valid rules	**ap-genrules** Time(seconds)	Cover	**CoverRules** Time(seconds)
90%	87722	9	8875	215
80%	344001	30	9375	236
70%	511191	46	9020	220
60%	574554	49	7878	178
50%	603861	50	6483	130
40%	630706	52	6506	133
30%	656724	51	5496	104
20%	682076	53	5674	99
10%	703373	52	3416	41
5%	703924	52	3181	37
1%	703924	52	3181	37

Table 3. Results for DENSE database

As expected, the performance of **CoverRules**, as well as that of **ap-genrules**, slows down when the databases are denser, and when the number of maximal frequent itemsets increases. The performance of the algorithms varies differently with the change of `minconf`. For dense databases, the size of the cover is one–two orders of magnitude smaller than the number of valid rules and shows the tendency of getting smaller as the redundancy in the generated rules increases.

References

[1] R. Agrawal and R. Srikant. Fast algorithms for mining association rules. RJ 9839, IBM Almaden Research Center, Almaden, California, 1994.

[2] R. J. Bayardo. Efficiently mining long patterns from databases. In *Proceedings of ACM-SIGMOD International Conference on Management of Data*, pages 85–93, 1998.

[3] C. L. Blake and C. J. Merz. University of California, Irvine: Repository of machine learning databases, 1998. http://www.ics.uci.edu/∼mlearn/MLRepository.html.

[4] L. Cristofor. ARtool: Association rule mining algorithms and tools, 2002. http://www.cs.umb.edu/∼laur/ARtool/.

[5] B. Liu, W. Hsu, and Y. Ma. Pruning and summarizing the discovered associations. In *Proceedings of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 125–134, 1999.

[6] M. Luxenburger. Implications partielles dans un contexte. *Mathématiques, Informatique et Sciences Humaines*, 29(113):35–55, 1991.

[7] N. Pasquier, Y. Bastide, R. Taouil, and L. Lakhal. Closed set based discovery of small covers for association rules. In *Proceedings of the 15th Conference on Advanced Databases*, pages 361–381, 1999.

[8] N. Pasquier, Y. Bastide, R. Taouil, and L. Lakhal. Efficient mining of association rules using closed itemset lattices. *Information Systems*, 24(1):25–46, 1999.

[9] D. A. Simovici and R. L. Tenney. *Relational Database Systems*. Academic Press, New York, 1995.