Pruning and Grouping Discovered Association Rules

H Toivonen M Klemettinen P Ronkainen K Hätönen H Mannila

Department of Computer Science, P.O. Box 26
FIN-00014 University of Helsinki, Finland
email: Hannu.Toivonen@Helsinki.FI

Abstract

Association rules are statements of the form "for 90 % of the rows of the relation, if the row has value 1 in the columns in set X, then it has 1 also in the columns in set Y". Efficient methods exist for discovering association rules from large collections of data. The number of discovered rules can, however, be so large that the rules cannot be presented to the user. We show how the set of rules can be pruned by forming *rule covers*. A rule cover is a subset of the original set of rules such that for each row in the relation there is an applicable rule in the cover if and only if there is an applicable rule in the original set. We also discuss grouping of association rules by clustering, and present some experimental results of both pruning and grouping.

Keywords: data mining, association rules, covers, clustering.

1 Introduction

Association rules are an interesting class of database regularities, introduced by Agrawal, Imielinski, and Swami [AIS93]. An association rule is an expression $X \Rightarrow Y$, where X and Y are sets of attributes. The intuitive meaning of such a rule is that in the rows of the database where the attributes in X have value true, also the attributes in Y tend to have value true. Efficient methods exist for the discovery of association rules [MTV94, AS94].

Paradoxically, data mining itself can produce such great amounts of data that there is a new knowledge management problem: there can easily be thousands or even more association rules holding in a data set. For instance, we discovered 2000 rules from a course enrollment database, and almost 30000 rules from a telecommunications network alarm database. In this paper we discuss pruning and grouping of association rules, in order to improve the understandability of the collection of discovered association rules.

The rest of this paper is organized as follows. Association rules and there properties are discussed in Section 2. In Section 3 we present how rule sets can be pruned by forming rule covers. Grouping of association rules is discussed in Section 4, while Section 5 is a conclusion.

2 Association rules and their properties

Let $R = \{I_1, I_2, \ldots, I_m\}$ be a set of attributes, also called items, over the binary domain $\{0, 1\}$. The input $r = \{t_1, \ldots, t_n\}$ for the data mining method is a relation over the relation schema $\{I_1, I_2, \ldots, I_m\}$, i.e., a set of binary vectors of size m. Each row can be considered as a set of properties or items (that is, $t[i] = 1 \Leftrightarrow I_i \in t$).

Let $X \subseteq R$ be a set of attributes and $t \in r$ a row of the relation. If $t[A] = 1$ for all $A \in X$, we write $t[X] = \bar{1}$. The set of rows matched by X is $m(X) = \{t \in r \mid t[X] = \bar{1}\}$. An *association rule* over r is an expression $X \Rightarrow Y$, where $X \subseteq R$ and $Y \subseteq R \setminus X$. Given real numbers γ (confidence threshold) and σ (support threshold), we say that r satisfies $X \Rightarrow Y$ with respect to γ and σ, if

$$|m(XY)| \geq \sigma n$$

and

$$\frac{|m(XY)|}{|m(X)|} \geq \gamma.$$

That is, at least a fraction σ of the rows of r have 1's in all attributes of X and Y, and at least a fraction γ of the rows having a 1 in all attributes of X also have a 1 in all attributes of Y. Given a set of attributes X, we say that X is *large* (with respect to the database and the given support threshold σ), if

$$|m(X)| \geq \sigma n.$$

That is, at least a fraction σ of the rows in the relation have 1's in all the attributes of X.

Example 1 Our case study is an enrollment database of courses in computer science. There is a row per student, containing the courses the student has registered for. The discovered association rule

> Distributed Operating Systems, Introduction to Unix $\Rightarrow$ Data Communications (0.96, 0.02)

states that 96 % of the students that have taken courses Distributed Operating Systems and Introduction to Unix, also have taken the course Data Communications, and that 2 % of all the students actually have taken all three courses. □

Given support threshold σ and confidence threshold γ, the set of all association rules that hold in a database can be computed efficiently [MTV94, AS94]. Additionally, one can obtain good approximations to the collection of association rules by using reasonably small samples [MTV94]. The data sets of our case study is representative of reasonable sample size.

One of the fundamental problems in data mining is to know what is useful to the user. The thresholds γ and σ ensure that the discovered rules have enough positive evidence. However, a given relation may satisfy a large number of such association rules. To be useful, a data mining system must manage the large amount of generated information by offering facilities for further rule pruning.

Example 2 The course enrollment database consists of registration information of 1066 students who had registered for at least two courses. On average, a row has 5 courses. The total number of courses is 112.

In the data set we discovered 2010 association rules with support threshold $\sigma = 0.01$ (corresponding to 11 students) and with no confidence threshold. The rule set contains 420 rules with a confidence of 0.7 or more, and 99 rules with a confidence of 0.9 or more. Raising the support threshold has a more dramatic effect: of those 99 rules with a confidence of 0.9 or more, only 4 rules have a support of at least 0.1.

With support threshold $\sigma = 0.005$, there are 6715 rules, 592 of which have confidence of 0.9 or more, and 459 of which have confidence of exactly 1. Thus, raising the confidence threshold to a high value prunes the set considerably, but many rules still may remain. □

Not all of the rules with high confidence and support are interesting. Some of these rules can correspond to prior knowledge or expectations, refer to uninteresting attributes or attribute combinations, or they can present redundant information.

Prior knowledge and expectations are background information that only the user can provide, and so are of course any special interests of the user. One way of specifying such information is to provide structural information about the rule expressions that are interesting. With *templates*, for instance, the user can explicitly specify both what is interesting and what is not [KMR+94].

In the rest of this paper we present a way for user and domain independent pruning of redundant rules, and discuss grouping of rules.

3 Association rule covers

In this section we present rule covers, a domain independent method of reducing the number of rules by elimination of redundancy.

Consider a collection Γ of rules with the same attribute set Y as the consequent:

$$\Gamma = \{X_i \Rightarrow Y \mid i = 1, \ldots, n\}.$$

Every rule in Γ gives a description of the presence of the attributes Y. Discovered rule sets often are heavily redundant, i.e. several rules describe the same database rows.

Given a rule set Γ, its subset $\Delta \subseteq \Gamma$ is a *rule cover*[1], if

$$\bigcup_{(X \Rightarrow Y) \in \Gamma} m(XY) \;=\; \bigcup_{(X \Rightarrow Y) \in \Delta} m(XY).$$

I.e., the rules in a rule cover Δ describe Y in all the cases in the database that the original rule set Γ does.

It is useful to constrain the rule sets and their covers to contain rules with high confidence only: for justification and prediction of the consequent Y, mostly rules of a high confidence are of interest. If one wants to be sure that the pruning methods do not lose any information, one has to assume the following. The data set should be monotonic in the sense that if there is a matching rule for Y with a high confidence, there should not exist a more special rule with a lower confidence. This is to ensure that we do not miss any contradictory information by limiting ourselves to the rules with high confidence only. These restrictions seem to hold in many real world data sets.

3.1 Structural rule cover

We first describe a conservative but fast way of pruning, based on the structure of the rules. The method is based on the observation that for all attribute sets X, Y, and Z we have

$$m(XYZ) \subseteq m(XZ),$$

i.e., a rule $X \Rightarrow Z$ matches a superset of the database rows matched by the rule $XY \Rightarrow Z$. If such rules $XY \Rightarrow Z$ are removed from a rule cover, the remaining set is also a rule cover.

A set of rules $\Delta \subseteq \Gamma$ is a *structural rule cover* for Γ, if for all rules $(X \Rightarrow Y) \in \Delta$ there is no rule $(X' \Rightarrow Y) \in \Gamma$ such that $X' \subset X$. A structural cover thus contains the most general rules of the original rule set. Clearly, a structural cover is a cover. Note that a structural cover can be computed completely without looking at the rows of the relation.

Example 3 The latter of the following rules is more special than the first one, and has no additional predictive power over it:

[1] The usage of the term *cover* is borrowed from database theory, where it is extensively used (see [Ull88]); there is also a close similarity to the set cover problem. Similar notions have also been used in the machine learning literature [MMHL86, CN89].

Programming in C, Object Data Bases $\Rightarrow$ Data Communications (0.90, 0.02)

Programming in C, Object Data Bases, Computer-Supported Cooperative Work $\Rightarrow$ Data Communications (0.90, 0.01)

The second rule is pruned from the structural rule cover as redundant. □

Structural covers are an efficient way of pruning rule sets. Next we describe an algorithm that finds more optimal rule covers. The algorithm can be used to improve structural covers.

3.2 Rule cover algorithm

A close to optimal rule cover can be found with the following greedy algorithm.

```
Algorithm RuleCover
Input:    Set of rules Γ = {X_i ⇒ Y | i = 1, ..., n}.
          Sets of matched rows m(X_iY) for all i ∈ {1, ..., n}.
Output:   Rule cover Δ.
Method:
    Δ := ∅;                          // rule cover
    s' := ⋃_{i=1}^{n} m(X_iY);       // rows unmatched by cover
    for all i ∈ {1, ..., n} do
        s_i = m(X_iY);               // rows of s' matched by rule i
    end;
    while s' ≠ ∅ do
        choose i ∈ {1, ..., n} so that (X_i ⇒ Y) ∈ Γ and |s_i| is largest;
        Δ := Δ ∪ {X_i ⇒ Y};          // add the rule to the cover
        Γ := Γ \ (X_i ⇒ Y);          // remove the rule from the original set
        for all (X_j ⇒ Y) ∈ Γ do
            s_j = s_j \ m(X_iY);     // remove matched rows
        end;
        s' := s' \ m(X_iY);          // remove matched rows
    end;
```

This algorithm gets as input the original rule set Γ and the sets of rows matched by each of these rules. Rule cover Δ is initialized to an empty set. Variable s' is used to store those database rows that are not matched by rules in Δ; the sets s_i contain those rows in s' that are matched by the rule $X_i \Rightarrow Y$. Iteratively, the rule in Γ that matches most of the rows in s' is moved from the rule set Γ to the rule cover. The rows matched by this rule are removed from s'. This is repeated until all the rows matched by the original rule set are matched by the rule cover, i.e., until the set s' is empty.

This greedy algorithm presumes that the rule cover Δ must match exactly the same rows as the original rule set Γ. It can be useful to loosen this restriction by allowing that some small portion ε of the rows is not matched by the rule cover Δ.

The next lemmas follow immediately from the properties of the greedy heuristics for set cover problems (see, e.g., [CLR90]).

Lemma 1 Algorithm RuleCover works correctly.

Lemma 2 Denote by *opt* the size of the smallest cover for the matched database rows $m(X_iY)$. Then for the size of the output Δ of Algorithm RuleCover we have

$$|\Delta| \leq \log(|\bigcup_{i\in\{1,\ldots,n\}} m(X_iY)|) \cdot opt,$$

i.e., the output Δ contains at most a logarithmic factor of extra rules.

Lemma 3 The time complexity of Algorithm RuleCover is polynomial with respect to $|\cup_i m(X_iY)|$.

The covers computed by the above algorithm seem to produce useful short descriptions of large sets of rules.

Example 4 A real telecommunications alarm database induces 1461 rules with the same attribute as a consequent and with a confidence of at least 0.9 (and with support of at least 2 %). The structural cover of the subset consists of 20 rules, which is 98 % improvement. A further rule cover computed by Algorithm RuleCover from the 20 rules only contains 5 rules. □

4 Grouping rules

Given an attribute set Y, the cover for the set of rules of the form $X \Rightarrow Y$ can still be quite large. The set of rules in the cover can be made more understandable by ordering and grouping the rules.

Rules can be ordered based on their interestingness. Obvious measures of interestingness are the confidence and support factors of rules. More complex measures can also be computed to reflect the statistical significance or financial value of rules [PSM94]. Some interestingness measures can be derived from user-specified template information. Such measures are useful for spotting the most interesting rules, but they do not help much in presenting a large collection of rules.

A useful method for structuring a set of rules is clustering. In particular, we claim that grouping those rules together that make statements about the same database rows is useful, as then also the way the grouping is done describes the database.

For clustering, the distance between two association rules can be defined in numerous ways. We defined the distance between two rules $X \Rightarrow Z$ and $Y \Rightarrow Z$ as the amount of rows where the rules differ:

$$\begin{aligned} d(X \Rightarrow Z, Y \Rightarrow Z) &= |(m(XZ) \cup m(YZ)) \setminus m(XYZ)| \\ &= |m(XZ)| + |m(YZ)| - 2|m(XYZ)|. \end{aligned}$$

Note that the number of matching rows $|m(X)|$ has already been computed for all large attribute sets X while forming association rules, and the distances are thus fast to compute.

Example 5 We experimented with the structural cover for the course Data Communications. The rule set in the cover consists of 29 rules with confidence of at least 0.9 and with altogether 15 different attributes. We clustered the 29 rules with SAS using a nonparametric density estimation method.

A clustering with 3 clusters turned out to break the rule set into clearly separate clusters. For instance, only cluster 1 contains courses VAX/VMS and Distributed Operating Systems. Students matched by rules in the cluster clearly are those specializing in operating systems. Cluster 2, in turn, contains rules that match to the students specializing in information systems: frequent or distinctive attributes are Object Data Bases, Database Systems II, and Computer Graphics. Cluster 3 is more diverse: the distinctive courses are Artificial Intelligence, Database Systems II, and Programming in C. Studends matched by rules in this cluster are specializing in different areas.

This experiment demonstrates that intuitive and "correct" rule clusters can be found. An interesting detail is that although the courses VAX/VMS and Distributed Operating Systems are often taken by the same students there actually is no rule in the cover with both of these courses — still rules with either of them ended in the same cluster. □

5 Concluding remarks

Association rules are a simple and natural class of database regularities, useful in various analysis and prediction tasks. The problem with this data mining technique is that the collection of all rules can be very large. We have considered the problem of pruning and grouping rules, in order to improve the understandability of the collection of discovered association rules.

A rule cover is a subset of the original set of rules such that the cover matches all the rows that the original set matches. In our experiments, rule covers turned out to produce useful short descriptions of large sets of rules. Particularly useful are structural rule covers which can be computed very efficiently.

In order to structure the rule set, we considered clustering of rules. In the experiments we were able to find intuitive rule clusters from rule covers. The distance measures of rules as well as the clustering methods are subject to further investigation.

The methods we presented for forming rule covers and rule clusters are based on how the rules match the data set. The results indicate that the data set is itself a very good source for additional information about the discovered rules. The presented methods are efficient. As the techniques are automatic they should be augmented with user-guided methods, such as templates, to take domain knowledge and user's interests into account.

An open problem is pruning within those association rules that are not very strong — the problem is actually to find a satisfactory general definition for redundancy. A closely related problem is how to combine association rules with the same consequent.

References

[AIS93] Rakesh Agrawal, Tomasz Imielinski, and Arun Swami. Mining association rules between sets of items in large databases. In *SIGMOD-93*, 207 – 216, May 1993.

[AS94] Rakesh Agrawal and Ramakrishnan Srikant. Fast algorithms for mining association rules in large databases. In *VLDB-94*, September 1994.

[CLR90] Thomas H. Cormen, Charles E. Leiserson, and Ronald L. Rivest. *Introduction to Algorithms.* MIT Press, Cambridge, MA, 1990.

[CN89] Peter Clark and Tim Niblett. The CN2 induction algorithm. *Machine Learning*, 3:261 – 283, 1989.

[KMR+94] Mika Klemettinen, Heikki Mannila, Pirjo Ronkainen, Hannu Toivonen, and A. Inkeri Verkamo. Finding interesting rules from large sets of discovered association rules. In *CIKM-94*, 401 – 407, November 1994.

[MMHL86] Ryszard S. Michalski, Igor Mozetic, Jiarong Hong, and Nada Lavrac. The multipurpose incremental learning system AQ15 and its testing application to three medical domains. In *AAAI-86*, 1041 – 1045, 1986.

[MTV94] Heikki Mannila, Hannu Toivonen, and A. Inkeri Verkamo. Efficient algorithms for discovering association rules. In *KDD-94*, 181 – 192, July 1994.

[PSM94] Gregory Piatetsky-Shapiro and Christopher J. Matheus. The interestingness of deviations. In *KDD-94*, 25 – 36, July 1994.

[Ull88] Jeffrey D. Ullman. *Principles of Database and Knowledge-Base Systems*, volume I. Computer Science Press, Rockville, MD, 1988.