Finding Association Rules that Trade Support Optimally Against Confidence

Tobias Scheffer
Humboldt-Universität zu Berlin, Department of Computer Science
Unter den Linden 6, 10099 Berlin, Germany
scheffer@informatik.hu-berlin.de

August 9, 2004

Abstract

When evaluating association rules, rules that differ in both support and confidence have to be compared; a larger support has to be traded against a higher confidence. The solution which we propose for this problem is to maximize the expected accuracy that the association rule will have for future data. In a Bayesian framework, we determine the contributions of confidence and support to the expected accuracy on future data. We present a fast algorithm that finds the n best rules which maximize the resulting criterion. The algorithm dynamically prunes redundant rules and parts of the hypothesis space that cannot contain better solutions than the best ones found so far. We evaluate the performance of the algorithm relative to the Apriori algorithm.

1 Introduction

Association rules (*e.g.,* Agrawal et al., 1993; Klemettinen et al., 1994; Agrawal et al., 1996), express regularities between sets of data items in a database. [Caçasa and limes ⇒ brown sugar] is an example of an association rule and expresses that, in a particular store, all customers who buy caçasa and limes are also likely to buy brown sugar. In contrast to classifiers, association rules do not make a prediction for *all* database records. When a customer does not buy caçasa and limes, then our example rule does *not* conjecture that he will not buy brown sugar either. The number of database records for which a rule does predict the proper value of an attribute is called the *support* of that rule.

Associations rules may not be perfectly accurate. The fraction of database records for which the rules conjectures a correct attribute value, relative to the fraction of records for which it makes any prediction, is called the *confidence*. Note that the confidence is the relative frequency of a correct prediction on the data that is used for training. We expect the confidence (or accuracy) on unseen data to lie below that on average, in particular, when the support is small.

When deciding which rules to return, association rule algorithms need to take both confidence and support into account. Of course, we can find any number of rules with perfectly high confidence but support of only one or very few records. On the other hand, we can construct very general rules with large support but low confidence. The Apriori algorithm (Agrawal et al., 1996) employs confidence and support thresholds and returns all rules which lie above these bounds. However, a knowledge discovery system has to evaluate the interestingness of these rules and provide the user with a reasonable number of interesting rules.

Which rules are interesting to the user depends on the problem which the user wants to solve and hopes the rules to be helpful for. In many cases, the user will be interested in finding items that do not only happen to co-occur in the available data. They will rather be interested in finding items between which there is a connection in the underlying reality. Items that truly correlate, will most likely also correlate in future data. In statistics, confidence intervals (which bound the difference between relative frequencies and their probabilities) can be used to derive guarantees that empirical observations reflect existing regularities in the underlying reality, rather than occurring just by chance. The number of observation plays a crucial role; when a rule has a large support, then we can be much more certain that the observed confidence is close to the confidence that we can expect to see in the future. This is one reason why association rules with very small support are considered less interesting.

In this paper, we propose a trade-off between confidence and support which maximizes the chance of correct predictions on unseen data. We detail the problem setting in Section 2, and in Section 3 we present our resulting utility criterion. In Section 4, we present a fast algorithm that finds the n best association rules with respect to this criterion. We discuss the algorithm's mechanism for pruning regions of the hypothesis space that cannot contain solutions that are better than the ones found so far, as well as the technique used to delete redundant rules which are already implied by other rules. In Section 5, we evaluate our algorithm empirically. We discuss related work in Section 6; Section 7 concludes.

2 Preliminaries

Let D be a database consisting of one table over binary attributes $a_1, \ldots, a_k$, called items. D may have been generated by binning the attributes of a relation of an original database D'. For instance, when D' contains an attribute *income*, then D may contain binary attributes $0 \leq income \leq 20k$, $20k < income \leq 40k$, and so on. A *database record* $r \subseteq \{a_1, \ldots, a_k\}$ is the set of attributes that take value one in a focused row of the table D.

A database record r satisfies an item set $x \subseteq \{a_1, \ldots, a_k\}$ if $x \subseteq r$. The support $s(x)$ of an item set x is the number of records in D which satisfy x. Often, the *fraction* $\frac{s(x)}{|D|}$ of records in D that satisfy x is called the support of x. But since the database D is constant, these terms are equivalent.

An association rule $[x \Rightarrow y]$ with $x, y \subseteq \{a_1, \ldots, a_k\}$, $y \neq \emptyset$, and $x \cap y = \emptyset$ expresses a relationship between an item set x and a nonempty item set y. The intuitive semantic of the rule is that all records which satisfy x are predicted to also satisfy y. The confidence of the rule with respect to the (training) database D is $\hat{c}([x \Rightarrow y]) = \frac{s(x \cup y)}{s(x)}$ – that is, the ratio of correct predictions over all records for which a prediction is made.

The confidence is measured with respect to the database D that is used for training. Often, a user assumes that the resulting association rules provide information on the process that generated the database, and that they will be valid in the future, too. But the confidence on the training data is only an estimate of the rules' accuracy in the future, and since we search the space of association rules to maximize the confidence, the estimate is optimistically biased. We define the *predictive accuracy* $c([x \Rightarrow y])$ of a rule as the probability of a correct prediction with respect to the process underlying the database (we sometimes refer to the predictive accuracy as accuracy).

Definition 1 *Let D be a database whose individual records r are generated by a static process P, let $[x \Rightarrow y]$ be an association rule. The predictive accuracy*

$c([x \Rightarrow y]) = Pr[r \text{ satisfies } y | r \text{ satisfies } x]$ *is the conditional probability of* $y \subseteq r$ *given that* $x \subseteq r$ *when the distribution of* r *is governed by* P.

The confidence $\hat{c}([x \Rightarrow y])$ is the relative frequency of probability $c([x \Rightarrow y])$ for given database D. We now pose the *n most accurate association rules problem.*

Problem setting 1 *Given a database* D *(defined like above) and a set of database items* a_1 *through* a_k*, find* n *rules* $h_1, \ldots, h_n \in \{[x \Rightarrow y] | x, y \subseteq \{a_1, \ldots, a_k\}; y \neq \emptyset; x \cap y = \emptyset\}$ *which maximize the expected predictive accuracy* $c([x \Rightarrow y])$.

We formulate the problem such that the algorithm needs to return a fixed number of best association rules rather than all rules whose utility exceeds a given threshold. We think that this setting is more appropriate in many situation because a threshold may not be easy to specify and a user may not be satisfied with either an empty or an outrageously large set of rules.

3 Bayesian Frequency Correction

In this section, we analyze how confidence and support contribute to the predictive accuracy. Intuitively, confidence estimates for rules with low support are less accurate; high confidence estimates for rules with low support are optimistic on average, and we want to quantify and correct this optimistic bias. How strongly we have to discount the confidence depends on the support – the greater the support, the more closely does the confidence relate to the predictive accuracy. In the Bayesian framework that we adopt, there is an exact solution as to how much we have to discount the confidence. We call this approach *Bayesian frequency correction* since the resulting formula (Equation 6) takes a confidence, "corrects" it by applying prior knowledge about confidences, and returns a somewhat lower predictive accuracy.

Suppose that we have a given association rule $[x \Rightarrow y]$ with observed confidence $\hat{c}([x \Rightarrow y])$. We can read $P(c([x \Rightarrow y]) | \hat{c}([x \Rightarrow y]), s(x))$ as "P(predictive accuracy of $[x \Rightarrow y]$ given confidence of $[x \Rightarrow y]$ and support of x)". The intuition of our analysis is that the application of Bayes' rule implies "P(predictive accuracy given confidence and support) = P(confidence given predictive accuracy and support)P(predictive accuracy)/ normalization constant". The likelihood $P(\hat{c}|c, s)$ is simply the binomial distribution: the target attributes of each record that is satisfied by x can be classified correctly or erroneously; the chance of a correct prediction is just the predictive accuracy c; this leads to a binomial distribution. "P(predictive accuracy)", the prior in our equation, is the accuracy prior over the space of all association rules. This prior counts, for every accuracy c, the fraction of rules which possess that accuracy.

In Equation 1, we decompose the expectation by integrating over all possible values of $c([x \Rightarrow y])$. In Equation 2, we apply Bayes' rule. $\pi(c) = \frac{|\{[x \Rightarrow y] | c([x \Rightarrow y]) = c\}|}{|\{[x \Rightarrow y]\}|}$ is the accuracy prior. It specifies the probability of drawing an association rule with accuracy c when drawing at random under uniform distribution from the space of association rules of length up to k.

$$
\begin{aligned}
& E(c([x \Rightarrow y]) | \hat{c}([x \Rightarrow y]), s(x)) \\
&= \int c P(c([x \Rightarrow y]) = c | \hat{c}([x \Rightarrow y]), s(x)) dc && (1) \\
&= \int c \frac{P(\hat{c}([x \Rightarrow y]) | c([x \Rightarrow y]) = c, s(x)) \pi(c)}{P(\hat{c}([x \Rightarrow y]) | s(x))} dc && (2)
\end{aligned}
$$

Over all values c, the distribution $P(c([x \Rightarrow y]) = c | \hat{c}([x \Rightarrow y]), s(x))$ has to integrate to one (Equation 3). Applying Equation 2 takes us to Equation 4. Hence, we

can treat $P(\hat{c}([x \Rightarrow y])|c([x \Rightarrow y]), s(x))$ as a normalizing constant which we can determine uniquely in Equation 5.

$$\int P(c([x \Rightarrow y]) = c|\hat{c}([x \Rightarrow y]), s(x))dc = 1 \tag{3}$$

$$\Leftrightarrow \quad \int \frac{P(\hat{c}([x \Rightarrow y])|c([x \Rightarrow y]) = c, s(x))\pi(c)}{P(\hat{c}([x \Rightarrow y])|s(x))}dc = 1 \tag{4}$$

$$\Leftrightarrow \quad P(\hat{c}([x \Rightarrow y])|s(x)) = \int P(\hat{c}([x \Rightarrow y])|c([x \Rightarrow y]) = c, s(x))\pi(c)dc \tag{5}$$

Combining Equations 2 and 5 we obtain Equation 6. In this equation, we also state that, when the accuracy c is given, the confidence $\hat{c}$ is governed by the binomial distribution which we write as $B[c, s](\hat{c})$. This requires us make the standard assumption of independent and identically distributed instances.

$$E(c([x \Rightarrow y])|\hat{c}([x \Rightarrow y]), s(x)) \quad = \quad \frac{\int cB[c, s(x)](\hat{c}([x \Rightarrow y]))\pi(c)dc}{\int B[c, s(x)](\hat{c}([x \Rightarrow y]))\pi(c)dc} \tag{6}$$

Equation 6 quantifies the expected confidence of a rule with confidence $\hat{c}$ whose body (the left hand side of the rule) x has support $s(x)$. The expected value is taken over two random variables: database D (governed by P), and all rules with $s(x)$ and $\hat{c}$. Association rule algorithms do not draw rules at random, but rather choose them in a way that is biased towards rules with large support and confidence. Let us clarify whether Equation 6 also holds when the rule is chosen *because* it has a maximally large $\hat{c}$.

Let $\hat{c}_{max}$ be the largest empirical confidence with respect to D and the available attributes; $\hat{c}_{max}$ is a new random variable. Equations 7 through 10 show that $E(c([x \Rightarrow y])|\hat{c}([x \Rightarrow y]), \hat{c}([x \Rightarrow y]) = c_{max}, s(x)) = P(c([x \Rightarrow y])|\hat{c}([x \Rightarrow y]), s(x))$. That is, the expectation of $c([x \Rightarrow y])$ is *conditionally independent* of $\hat{c}_{max}$, *given* $\hat{c}([x \Rightarrow y])$.

In Equation 7, we reduce the conditional to the joint probability and condition only on $s(x)$. We factorize the joint probability in Equation 8 and condition on $\hat{c}([x \Rightarrow y])$ in Equation 9. Equation 10 holds because $\hat{c}([x \Rightarrow y]) = \hat{c}_{max}$ depends on $\hat{c}([x \Rightarrow y])$; it is *conditionally* independent of $c([x \Rightarrow y])$ *given* $\hat{c}([x \Rightarrow y])$.

$$\begin{aligned}
&P(c([x \Rightarrow y])|\hat{c}([x \Rightarrow y]), \hat{c}([x \Rightarrow y]) = \hat{c}_{max}, s(x)) \\
&= \frac{P(c([x \Rightarrow y]), \hat{c}([x \Rightarrow y]), \hat{c}([x \Rightarrow y]) = \hat{c}_{max}|s(x))}{P(\hat{c}([x \Rightarrow y]), \hat{c}([x \Rightarrow y]) = \hat{c}_{max}|s(x))} && (7)\\
&= \frac{P(c([x \Rightarrow y], \hat{c}([x \Rightarrow y])|s(x))P(\hat{c}([x \Rightarrow y]) = \hat{c}_{max}|c([x \Rightarrow y]), \hat{c}([x \Rightarrow y]), s(x))}{P(\hat{c}([x \Rightarrow y]), \hat{c}([x \Rightarrow y]) = \hat{c}_{max}|s(x))} && (8)\\
&= P(c([x \Rightarrow y])|\hat{c}([x \Rightarrow y]), s(x))\frac{P(\hat{c}([x \Rightarrow y]) = \hat{c}_{max}|c([x \Rightarrow y]), \hat{c}([x \Rightarrow y]), s(x))}{P(\hat{c}([x \Rightarrow y]) = c_{max}|\hat{c}([x \Rightarrow y]), s(x))} && (9)\\
&= P(c([x \Rightarrow y])|\hat{c}([x \Rightarrow y]), s(x)) && (10)
\end{aligned}$$

We have now found a solution that quantifies $E(c([x \Rightarrow y])|\hat{c}([x \Rightarrow y]), s(x))$, the *exact* expected predictive accuracy of an association rule $[x \Rightarrow y]$ with given confidence $\hat{c}$ and support of the rule's body (the left hand side of the rule) of $s(x)$. Equation 6 thus quantifies just how strongly the confidence of a rule has to be corrected, given the support of that rule. Note that the solution depends on the prior $\pi(c)$ which is the histogram of accuracies of all association rules over the given items for the given database.

One way of treating such priors is to assume a certain standard distribution. Under a set of assumptions on the process that generated the database, $\pi(c)$ can be

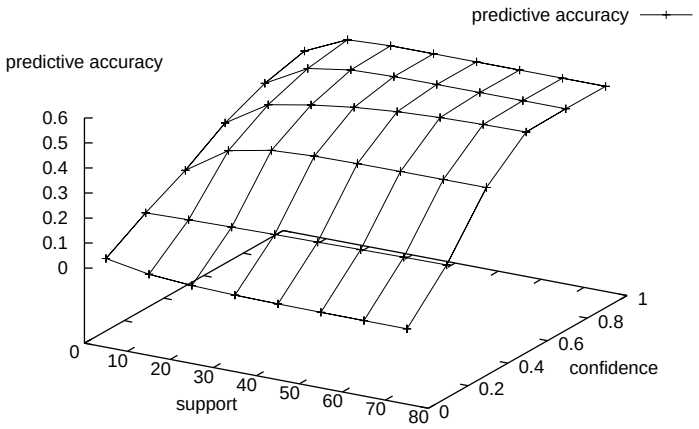

Figure 1: Contributions of support $s(x)$ and confidence $\hat{c}([x \Rightarrow y])$ to predictive accuracy $c([x \Rightarrow y])$ of rule $[x \Rightarrow y]$

shown to be governed by a certain binomial distribution (Scheffer, 2000). However, empirical studies (see Section 5 and Figure 2a) show that the shape of the prior can deviate strongly from this binomial distribution. Better empirical results can be obtained by estimating the prior from the available database, applying a Markov Chain Monte Carlo (Gilks et al., 1995) correction (see Section 4).

A theoretical result on the complexity of estimating $\pi(c)$ has been obtained by Langford and McAllester (2000). They have been able to show that estimating the histogram $\pi(c)$ is much easier than estimating the predictive accuracy of all association rules by proving the following theorem. In order to estimate the predictive accuracy of *all* association rules, it follows from PAC theory that $\frac{\log|H|}{|D|}$ has to be small. Now consider a process in which both the database $|D_i|$ and the number of association rules H_i grow in parallel when $i \to \infty$, such that $\frac{\log|H_i|}{|D_i|}$ stays constantly large. Over this process, we are unable to estimate all predictive accuracies in H_i but $\hat{\pi}(\hat{c})$ converges to $\pi(c)$ as i grows (Langford & McAllester, 2000).

Example curve. Figure 1 shows how expected predictive accuracy, confidence, and support of the rule body relate for the database that we also use for our experiments in Section 5, using 10 items. The predictive accuracy grows with both confidence and body support of the rule. When the confidence exceeds 0.5, then the predictive accuracy is lower than the confidence, depending on the support and on the histogram π of accuracies of association rules for this database.

4 Discovery of Association Rules

The Apriori algorithm (Agrawal et al., 1993) finds association rules in two steps. First, all item sets x with support of more then the fixed threshold "minsup" are found. Then, all item sets are split into left and right hand side x and y (in all possible ways) and the confidence of the rules $[x \Rightarrow y]$ is calculated as $\frac{s(x \cup y)}{s(x)}$. All rules with a confidence above the confidence threshold "minconf" are returned. Our algorithm differs from that scheme since we do not have fixed confidence and

Table 1: Algorithm PredictiveApriori: discovery of n most predictive association rules.

1. **Input:** n (desired number of association rules), database with items $a_1, \ldots, a_k$.
2. **Let** $\tau = 1$.
3. **For** $i = 1 \ldots k$ **Do:** Draw a number of association rules $[x \Rightarrow y]$ with i items at random. Measure their confidence (provided $s(x) > 0$). Let $\pi_i(c)$ be the distribution of confidences.
4. **For** all c, **Let** $\pi(c) = \frac{\sum_{i=1}^{k} \pi_i(c)\binom{k}{i}(2^i - 1)}{\sum_{i=1}^{k} \binom{k}{i}(2^i - 1)}$.
5. **Let** $X_0 = \{\emptyset\}$; **Let** $X_1 = \{\{a_1\}, \ldots, \{a_k\}\}$ be all item sets with one single element.
6. **For** $i = 1 \ldots k - 1$ **While** ($i = 1$ or $X_{i-1} \neq \emptyset$).
   (a) **If** $i > 1$ **Then** determine the set of candidate item sets of length i as $X_i = \{x \cup x' | x, x' \in X_{i-1}, |x \cup x'| = i\}$. Generation of X_i can be optimized by considering only item sets x and $x' \in X_{i-1}$ that differ only in the element with highest item index. Eliminate double occurrences of item sets in X_i.
   (b) Run a database pass and determine the support of the generated item sets. Eliminate item sets with support less than τ from X_i.
   (c) **For** all $x \in X_i$ **Call RuleGen**(x).
   (d) **If** *best* has been changed, **Then Increase** τ to be the smallest number such that $E(c|1, \tau) > E(c(best[n])|\hat{c}(best[n]), s(best[n]))$ (refer to Equation 6). **If** $\tau >$ database size, **Then Exit**.
   (e) **If** τ has been increased in the last step, **Then** eliminate all item sets from X_i which have support below τ.
7. **Output** $best[1] \ldots best[n]$, the list of the n best association rules.

Table 2: Algorithm RuleGen: Searching efficiently for all rules with body x.

Algorithm RuleGen(x) (find the best rules with body x efficiently)

10. **Let** γ be the smallest number such that $E(c|\gamma/s(x), s(x)) > E(c(best[n])|\hat{c}(best[n]), s(best[n]))$.
11. **For** $j = 1 \ldots k - |x|$ (number of items not in x)
    (a) **If** $j = 1$ **Then Let** $Y_1 = \{a_1, \ldots a_k\} \setminus x$.
    (b) **Else Let** $Y_j = \{y \cup y' | y, y' \in Y_{j-1}, |y \cup y'| = j\}$ analogous to the generation of candidates in step 6a.
    (c) **For** all $y \in Y_j$ **Do**
        i. Measure the support $s(x \cup y)$. **If** $s(x \cup y) \leq \gamma$, **Then** eliminate y from Y_j and **Continue** the for loop with the next y.
        ii. Calculate predictive accuracy $E(c([x \Rightarrow y])|s(x \cup y)/s(x), s(x))$ according to Equation 6.
        iii. **If** the predictive accuracy is among the n best found so far (recorded in *best*), **Then** update *best*, remove rules in *best* that are subsumed by other, at least equally accurate rules (utilize Theorem 1 and test for $x \subseteq x' \wedge y \supseteq y'$), and **Increase** γ to be the smallest number such that $E(c|\gamma/s(x), s(x)) \geq E(c(best[n])|\hat{c}(best[n]), s(best[n]))$.
12. **If** any subsumed rule has been erased in 11(c)iii, **Then** recur from step 10.

support thresholds. Instead, we want to find the n best rules.

In the first step, our algorithm estimates the prior $\pi(c)$. Then generation of frequent item sets, pruning the hypothesis space by dynamically adjusting the minsup threshold, generating association rules, and removing redundant association rules interleave. The algorithm is displayed in Table 1; the procedure for generation of all rules with fixed body x is presented in Table 2.

Estimating $\pi(c)$. We can estimate π by drawing many hypotheses at random under uniform distribution, measuring their confidence, and recording the resulting histogram. However, there are many more long rules than there are short ones (the number of distinct item sets grows exponentially in the length). If we drew rules at random, we would almost never get to see short rules, our estimate of $\pi(c)$ for short rules would be poor. In order to avoid this problem, we run a loop over the length of the rule and, given that length, draw a fixed number of rules. We determine the items and the split into body and head by drawing at random (Step 3).

We have now drawn equally many rules for each size while the uniform distribution requires us to prefer long rules. There are $\binom{k}{i}$ item sets of size i over k database items, and given i items, there are $2^i - 1$ distinct association rules (each item can be located on the left or right hand side of the rule but the right hand side must be nonempty). Hence, Equation 11 gives the probability that exactly i items occur in a rule which is drawn at random under uniform distribution from the space of all association rules over k items.

$$P[i \text{ items}] = \frac{\binom{k}{i}(2^i - 1)}{\sum_{j=1}^{k} \binom{k}{j}(2^j - 1)} \tag{11}$$

In Step 4 we estimate the prior over all association rules in a way that accounts for the number of rules with a specific length that exist by weighting each prior for

rule length i by the probability of a rule length of i. This can be seen as a Markov Chain Monte Carlo style correction to the prior

Enumerating item sets with dynamic minsup threshold. Similarly to the Apriori algorithm, the PredictiveApriori algorithm generates frequent item sets, but it uses a dynamically increasing minsup threshold τ. Note that we start with size zero (only the empty item set is contained in X_0). X_1 contains all item sets with one element. Given X_{i-1}, the algorithm computes X_i in Step 6a just like Apriori does. An item set can only be frequent when all its subsets are frequent, too. We can thus generate X_i by only joining those elements of X_{i-1} which differ exactly in the last element (where last refers to the highest item index). Since all subsets of an element of X_i must be in X_{i-1}, the subsets that result from removing the last, or the last but one element must be in X_{i-1}, too. After running a database pass and measuring the support of each element of X_i, we can delete all those candidates that do not achieve the required support of τ.

We then invoke the RuleGen procedure in Step 6c that generates all rules over body x, for each $x \in X_i$. The RuleGen procedure alters our array $best[1 \dots n]$ which saves the best rules found so far. In Step 6d, we refer to $best[n]$, meaning the n-th best rule found so far. We now refer to Equation 6 again to determine the least support that the body of an association rule with perfect confidence must possess in order to exceed the predictive accuracy of the currently n-th best rule. If that required support exceeds the database size we can exit because no such rule can exist. We delete all item sets in Step 6e which lie below that new τ. Finally, we output the n best rules in Step 7.

Generating all rules over given body x. In Step 10, we introduce a new threshold γ which quantifies the confidence that a rule with support $s(x)$ needs in order to be among the n best ones. We then start enumerating all possible heads y, taking into account in Step 11 that body and head must be disjoint and generating candidates in Step 11(b) analogous to Step 6a. In Step 11(c)i we calculate the support of $x \cup y$ for all heads y. When a rule lies among the best ones so far, we update *best*. We will not bother with rules that have a predictive accuracy below the accuracy of $best[n]$, so we increase γ. In Step 11(c)iii, we delete rules from *best* which are subsumed by other rules. This may result in the unfortunate fact that rules which we dropped from *best* earlier, now belong to the n best rules again. So in Step 11(c)iii we have to check this and recur from Step 10 if necessary.

Removing redundant rules. Consider an association rule $[a \Rightarrow c, d]$. When this rule is satisfied by a database, then that database must also satisfy $[a, b \Rightarrow c, d]$, $[a \Rightarrow c]$, $[a \Rightarrow d]$, and many other rules. We write $[x \Rightarrow y] \models [x' \Rightarrow y']$ to express that any database that satisfies $[x \Rightarrow y]$ must also satisfy $[x' \Rightarrow y']$. Since we can generate exponentially many redundant rules that can be inferred from a more general rule, it is not desirable to present all these redundant rules to the user. Consider the example in Table 3 which shows the five most interesting rules generated by PredictiveApriori for the purchase database that we study in Section 5. The first and second rule in the bottom part are special cases of the third rule; the fourth and fifth rules are subsumed by the second rule of the top part. The top part shows the best rules with redundant variants removed.

Definition 2 (Subsumption) *Let $[x \Rightarrow y]$ and $[x' \Rightarrow y']$ be association rules. $[x \Rightarrow y] \models [x' \Rightarrow y'] \Leftrightarrow$ (for all databases $D : \forall r \in D : (x \in r \rightarrow y \in r) \Rightarrow (x' \in r \rightarrow y' \in r)$). Informally, $[x \Rightarrow y]$ subsumes $[x' \Rightarrow y']$ if every database that supports $[x \Rightarrow y]$ also supports $[x' \Rightarrow y']$.*

Theorem 1 *We can decide whether a rule subsumes another rule by two subset tests: $[x \Rightarrow y] \models [x' \Rightarrow y'] \Leftrightarrow x \subseteq x' \wedge y \supseteq y'$.*

Proof. We have to prove both directions. "⇒": By contradiction: Assume there is a database D such that $\forall r : (*)\ (x \in r \rightarrow y \in r) \Rightarrow (x' \in r \rightarrow y' \in r)$ but $x \not\subseteq x' \vee y \not\supseteq y'$. $(*)$ implies that when $(x \in r \rightarrow y \in r)$, then $(x' \in r \rightarrow y' \in r)$ have to be true as well. But when $x \not\subseteq x' \vee y \not\supseteq y'$ we can easily construct a contradictory example: $1 \in \{2,3\} \rightarrow 2 \in \{2,3\}$ is true but $2 \in \{2,3\} \rightarrow 1 \in \{2,3\}$ is false. "⇐": When $x \subseteq x' \wedge y \supseteq y'$, then $(*)$ becomes $(x \in r \rightarrow y \cup y^+ \in r) \Rightarrow (x \cup x^+ \rightarrow y \in r)$. This is always true because $x \cup x^+ \in r \Rightarrow x \in r$ and $y \cup y^+ \in r \Rightarrow y \in r$. ∎

Theorem 1 says that $[x \Rightarrow y]$ subsumes $[x' \Rightarrow y']$ if and only if x is a *subset* of x' (weaker precondition) and y is a *superset* of y' (y predicts more attribute values than y'). We can then delete $[x' \Rightarrow y']$ because Theorem 1 says that from a more general rule we can infer that all subsumed rules must be satisfied, too. In order to assure that the n rules which the user is provided are not redundant specializations of each other, we test for subsumption in Step 11(c)iii by performing the two subset tests.

Table 3: (Top) five best association rules when subsumed rules are removed; (bottom) five best rules when subsumed rules are not removed.

[⇒ PanelID=9 ProductGroup=84]
$E(c\|\hat{c}=1, s=10000)=1$
[Location=market_4 ⇒ PanelID=9, ProductGroup=84, Container=nonreuseable]
$E(c\|\hat{c}=1, s=1410)=1$
[Location=market_6 ⇒ PanelID=9, ProductGroup=84, Container=nonreuseable]
$E(c\|\hat{c}=1, s=1193)=1$
[Location=market_1 ⇒ PanelID=9, ProductGroup=84, Container=nonreuseable]
$E(c\|\hat{c}=1, s=1025)=1$
[Manufacturer=producer_18 ⇒ PanelID=9, ProductGroup=84, Type=0, Container=nonreuseable]
$E(c\|\hat{c}=1, s=1804)=1$

[⇒ PanelID=9]	$E(c\|\hat{c}=1, s=10000)=1$
[⇒ ProductGroup=84]	$E(c\|\hat{c}=1, s=10000)=1$
[⇒ PanelID=9, ProductGroup=84]	$E(c\|\hat{c}=1, s=10000)=1$
[Location=market_4 ⇒ PanelID=9]	$E(c\|\hat{c}=1, s=1410)=1$
[Location=market_4 ⇒ ProductGroup=84]	$E(c\|\hat{c}=1, s=1410)=1$

We are now ready to state our main theorem on the correctness of PredictiveApriori. Theorem 2 claims that the rules returned by PredictiveApriori are indeed the most accurate ones, and do not include rules which are subsumed by other returned rules.

Theorem 2 *The PredictiveApriori algorithm (Table 1) returns an array* $best[1 \ldots n]$ *of association rules* $[x_i \Rightarrow y_i]$ $(x_i, y_i \subseteq \{a_1, \ldots, a_k\},\ x_i \cap y_i = \emptyset)$ *with the following property: For all rules* $[x \Rightarrow y] \notin best$ *and for all* i *with* $1 \leq i \leq n$*:* $E(c([x \Rightarrow y])|\hat{c}([x \Rightarrow y]), s(x)) \leq E(c(best[i])|\hat{c}(best[i]), s(best[i]))$ *or* $\exists [x' \Rightarrow y'] \in best$ *such that* $E(c([x' \Rightarrow y'])|\hat{c}([x' \Rightarrow y']), s(x')) \geq E(c([x \Rightarrow y])|\hat{c}([x \Rightarrow y]), s(x))$ *and* $[x' \Rightarrow y'] \models [x \Rightarrow y]$.

Theorem 2 says that any rule *not* included in *best* is either less accurate than the n-th best rule $best[n]$, or it is subsumed by an at least equally accurate rule that is contained in *best*.

Proof. The proof of Theorem 2 can be found in Appendix A.

Improvements. Several improvements of the Apriori algorithm have been suggested that improve on the PredictiveApriori algorithm as well. The AprioriTid algorithm requires much fewer database passes by storing, for each database record, a list of item sets of length i which this record supports. From these lists, the support of each item set can easily be computed. In the next iteration, the list of item sets of length $i+1$ that each transaction supports can be computed without accessing the database. We can expect this modification to enhance the overall performance when the database is very large but sparse. Further improvements for large databases can be obtained by using sampling techniques (*e.g.,* Mannila et al., 1994; Scheffer & Wrobel, 2002).

5 Experiments

For our experiments, we use a database of 14,000 fruit juice purchase transactions, and the mailing campaign data used for the KDD cup 1998. Each transaction of the fruit juice database is described by 29 real valued and string valued attributes which specify properties of the purchased juice as well as attributes of the customer (*e.g.,* age and job). By binarizing the attributes and considering only a subset of the binary attributes, we vary the number of items during the experiments. For instance, we transformed the attribute "ContainerSize" into five binary attributes, "ContainerSize ≤ 0.3", "$0.3 <$ ContainerSize ≤ 0.5", etc.

Figure 2a shows the prior $\pi(c)$ as estimated by the algorithm in Step 3 for several numbers of items. Figure 1 shows the predictive accuracy for this prior, depending on the confidence and the body support. Table 3 (top) shows the five

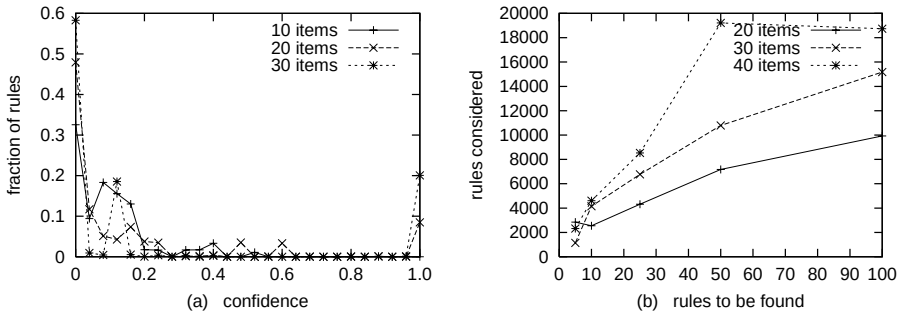

Figure 2: (a) Confidence prior π for various numbers of items. (b) Number of rules that PredictiveApriori has to consider dependent on the number n of desired solutions.

best association rules found for the fruit juice problem by the PredictiveApriori algorithm. The rules say that all transactions are performed under PanelID 9 and refer to product group 84 (fruit juice purchases). The second through fourth rules say that markets 1, 4, and 6 only sell non-reusable bottles (in contrast to the refillable bottles sold by most German supermarkets). Producer 18 does not sell refillable bottles either.

In order to compare the performance of Apriori and PredictiveApriori, we use a uniform measure that is independent of implementation details. For Apriori, we count how many association rules have to be compared against the minconf threshold. This comparison in the innermost loop is the bottleneck of Apriori; the number is independent of the actual minconf threshold. We can determine the number of comparisons from the item sets without actually enumerating all rules.

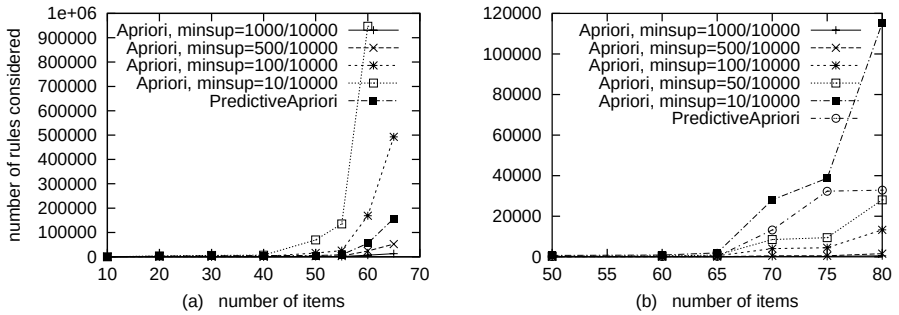

Figure 3: Time complexity of PredictiveApriori and Apriori, depending on the number of items and (in case of Apriori) of minsup (a) fruit juice problem, (b) KDD cup 1998

This enables us to draw performance diagrams for large sets of items where it would be infeasible to actually run Apriori and measure execution time.

For PredictiveApriori, we measure for how many rules we need to determine the predictive accuracy by evaluating Equation 6. This calculation takes place in the innermost loop and is the bottleneck of PredictiveApriori.

The performance of Apriori depends crucially on the choice of the support threshold "minsup". In Figure 3, we compare the computational expenses imposed by PredictiveApriori (10 best solutions) to the complexity of Apriori for several different minsup thresholds and numbers of items for both the fruit juice and the KDD cup database. The time required by Apriori grows rapidly with decreasing minsup values. Among the 25 best solutions for the juice problem found by PredictiveApriori we can see rules with confidence 92 whose body also has support of 92. In order to find such special but accurate rules, Apriori would run many times as long as PredictiveApriori. Figure 2b shows how the complexity increases with the number of desired solutions. The increase is only sub-linear. Figure 4 shows extended comparisons of the Apriori and PredictiveApriori performance for the fruit juice problem. The horizontal lines show the time required by PredictiveApriori for the given number of database items ($n = 10$ best solutions). The curves show how the time required by Apriori depends on the minsup support threshold. Apriori is faster for large thresholds since it then searches only a small fraction of the space of association rules.

6 Related Work

It has early been observed that confidence and support are not in every case equivalent to the interestingness of a rule to the user (Brin et al., 1997; Aggarwal & Yu, 1998). Alternative performance metrics include *lift* ($P(x,y)/P(x)/P(y)$) and *conviction* ($P(x)P(\neg y)/P(x,\neg y)$); conviction takes into account that association rules are not symmetric (Bayardo et al., 2000; Brin et al., 1997; Aggarwal & Yu, 1998).

The χ^2 test is only loosely related to the approach discussed here. The χ^2 statistic quantifies the likelihood of the data under the assumption that x and y uncorrelated. By contrast, our Bayesian perspective allows us to focus on the *a posteriori* probability of the rule giving correct predictions in the future. Our Bayesian analysis of predictive accuracy is based on an earlier analysis of expected error rates of classifiers (Scheffer & Joachims, 1999). Many additional performance metrics have been proposed in the area of subgroup mining (Klösgen, 1992; Klösgen,

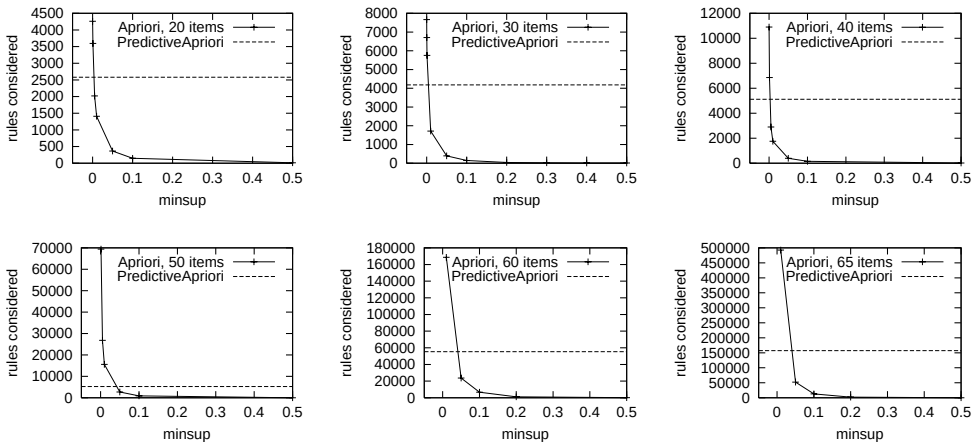

Figure 4: Number of rules that PredictiveApriori and Apriori and need to consider, depending on the number of items (in case of Apriori also depending on minsup).

1995; Scheffer & Wrobel, 2002).

It has frequently been observed that association rules contain many redundant variations. Zaki (2000) proposes the closed item set framework which leads to substantially smaller item sets and hence simpler rules. Rather than focusing on the length of item sets, we are interested in finding the most general rules. When every database that supports rule $[x \Rightarrow y]$ also supports $[x' \Rightarrow y']$, then it is not necessary to present both of these rules to the user. Note that the most general rule is not identical to the most compact rule: The most general rule $[\ \Rightarrow a_1, \ldots, a_k]$ subsumes (*i.e.,* implies) all simpler rules such as $[\ \Rightarrow a_1]$. Guided by similar ideas, the Midos algorithm (Wrobel, 2001) performs a similarity test for hypotheses.

Theorem 1 has first been proposed by Scheffer (2001). Independently, Li et al. (2001) have shown that $[x \Rightarrow y] \models [x' \Rightarrow y']$ if $x \subseteq x'$ and $y = y'$; Theorem 1 furthermore says that the same is true when $y \supseteq y'$. Further results on minimum redundancy rule mining have been obtained by Brijs et al. (2000).

7 Conclusion

We discussed the problem of trading confidence of an association rule against support. When the goal is to maximize the expected accuracy on future database records that are generated by the same underlying process, then Equation 6 gives us the optimal trade-off between confidence and support of the rule's body. Equation 6 results from a Bayesian analysis of the predictive accuracy; it is based on the assumption that the database records are independent and identically distributed and requires us knowledge about the confidence prior. The PredictiveApriori algorithm estimates the confidence prior from the available database.

The Bayesian frequency correction approach eliminates the optimistic bias of high confidences. The PredictiveApriori algorithm returns the n rules which maximize the expected predictive accuracy; the user only has to specify how many rules he or she wants to be presented. This is perhaps a more natural parameter than minsup and minconf, required by the Apriori algorithm.

The favorable computational performance of the PredictiveApriori algorithm can be credited to the dynamic pruning technique that uses an upper bound on the accuracy of all rules over supersets of a given item set. Very large parts of the search

space can thus be excluded. A similar idea is realized in Midos (Wrobel, 2001). The algorithm also checks the rules for redundancies. It has a bias towards returning general rules and eliminating all rules which are entailed by equally accurate, more general ones.

Many optimizations of the Apriori algorithm have been proposed which have helped this algorithm gain its huge practical relevance. These include the AprioriTid approach for minimizing the number of database passes (Agrawal et al., 1996), and sampling approaches for estimating the support of item sets (Mannila et al., 1994; Toivonen, 1996; Scheffer & Wrobel, 2002). In particular, efficient search for frequent item sets has been addressed intensely and successfully (Lin & Kedem, 1998; Brin et al., 1997; Zaki & Hiao, 1999). Many of these improvements can, and should be, applied to the PredictiveApriori algorithm as well.

A Appendix: Proof of Theorem 2

In order to prove Theorem 2, we will prove the following list of assertions.

Naming conventions. Let $best_i[1 \ldots n]$ be the assignment of the array $best[1 \ldots n]$ after the i-th iteration of loop 6. Let $best_i^j[1 \ldots n]$ be the assignment of the array $best[1 \ldots n]$ *in* the i-th iteration of loop 6, *after* the j-th iteration of loop 11. Let $E(c([x \Rightarrow y]))$ abbreviate $E(c([x \Rightarrow y])|\hat{c}([x \Rightarrow y]), s(x))$ according to Equation 6.

After every step. For all i, j with $i < j \leq n$: $E(c(best[i])) \geq E(c(best[j]))$.

After Step 6a. For all item sets c with $|c| = i$: $c \notin X_i \Rightarrow$ for all item sets $head$, $E(c([c \Rightarrow head])) \leq E(c(best_{i-1}[n]))$.

After Step 6b. For all item sets c with $|c| = i$: $c \notin X_i \Rightarrow$ for all item sets $head$, $E(c([c \Rightarrow head])) \leq E(c(best_{i-1}[n]))$.

After Step 10. There is no rule $[x \Rightarrow y]$ with observed confidence $\hat{c}([x \Rightarrow y]) \leq \gamma/s(x)$, such that $E(c([x \Rightarrow y])|\hat{c}([x \Rightarrow y]), s(x)) \geq E(c(best[n]))$.

After Step 11a. The condition $a_i \notin x$ assures that for all rules $[x \Rightarrow y]$ that are considered throughout the algorithm, $x \cap y = \emptyset$. Hence, this also holds for the returned hypotheses $best[1 \ldots n]$. In order to keep the proof simple, we do not explicitly mention this assertion elsewhere in the proof.

After Step 11b. For all item sets $c \notin Y_i$ with $|c| = j$: $E(c([x \Rightarrow c])) \leq E(c(best_i^{j-1}[n]))$ *or* a rule has been deleted in Step 11(c)iii since Step 10 was last visited.

After Step 11(c)i. For all item sets $c \notin Y_i$ with $|c| = j$: $E(c([x \Rightarrow c])) \leq E(c(best_i^{j-1}[n]))$ *or* a rule has been deleted in Step 11(c)iii since Step 10 was last visited.

After 11(c)iii. For all y with $|y| \leq j$ and $[x \Rightarrow y] \notin best$: $E(c([x \Rightarrow y])) \leq E(c(best_i[n]))$ or there exists $[x' \Rightarrow y'] \in best$ with $E(c([x' \Rightarrow y'])) \geq E(c([x \Rightarrow y]))$ and $[x' \Rightarrow y'] \models [x \Rightarrow y]$ *or* a rule has been deleted in Step 11(c)iii since Step 10 was last visited. Furthermore, there is no rule $[x \Rightarrow y]$ with observed confidence $\hat{c}([x \Rightarrow y]) \leq \gamma/s(x)$, such that $E(c([x \Rightarrow y])|\hat{c}([x \Rightarrow y]), s(x)) \geq E(c(best[n]))$.

After termination of loop 11. For all y with $[x \Rightarrow y] \notin best$: $E(c([x \Rightarrow y])) \leq E(c(best_i[n]))$ or there exists $[x' \Rightarrow y'] \in best$ with $E(c([x' \Rightarrow y'])) \geq E(c([x \Rightarrow y]))$ and $[x' \Rightarrow y'] \models [x \Rightarrow y]$ *or* a rule has been deleted in Step 11(c)iii since Step 10 was last visited.

After Step 12. For all y with $[x \Rightarrow y] \notin best$: $E(c([x \Rightarrow y])) \leq E(c(best_i[n]))$ or there exists $[x' \Rightarrow y'] \in best$ with $E(c([x' \Rightarrow y'])) \geq E(c([x \Rightarrow y]))$ and $[x' \Rightarrow y'] \models [x \Rightarrow y]$.

After iteration x of loop 6c. For all y with $[x \Rightarrow y] \notin best : E(c([x \Rightarrow y])) \leq best[n]$ or there exists $[x' \Rightarrow y'] \in best$ with $E(c([x' \Rightarrow y'])) \geq E(c([x \Rightarrow y]))$ and $[x' \Rightarrow y'] \models [x \Rightarrow y]$.

After termination of loop 6c. For all $[x \Rightarrow y]$ with $|x| = i$ and $[x \Rightarrow y] \notin best : E(c([x \Rightarrow y])) \leq E(c(best[n]))$ or there exists $[x' \Rightarrow y'] \in best$ with $E(c([x' \Rightarrow y'])) \geq E(c([x \Rightarrow y]))$ and $[x' \Rightarrow y'] \models [x \Rightarrow y]$.

After Step 6d. There is no rule $[x \Rightarrow y]$ with $s(x) < \tau$ and $E(c([x \Rightarrow y])) > E(c(best[n]))$.

After Step 6e. For all item sets $c \notin X_i$ with $|c| = i$: For all item sets $head$, $E(c([c \Rightarrow head])) \leq E(c(best_i[n]))$ or there exists $[x \Rightarrow y] \in best$ with $E(c([x \Rightarrow y])) \geq E(c([c \Rightarrow head]))$ and $[x \Rightarrow y] \models [c \Rightarrow head]$.

After the i-th iteration of loop 6. For all rules $[x \Rightarrow y] \notin best$ with $|x| \leq i$: $E(c([x \Rightarrow y])) \leq E(c(best[n]))$ or there exists $[x' \Rightarrow y'] \in best$ with $E(c([x' \Rightarrow y'])) \geq E(c([x \Rightarrow y]))$ and $[x' \Rightarrow y'] \models [x \Rightarrow y]$.

After termination of loop 6. For all rules $[x \Rightarrow y] \notin best$: $E(c([x \Rightarrow y])) \leq E(c(best[n]))$ or there exists $[x' \Rightarrow y'] \in best$ with $E(c([x' \Rightarrow y'])) \geq E(c([x \Rightarrow y]))$ and $[x' \Rightarrow y'] \models [x \Rightarrow y]$.

After Step 7, termination of PredictiveApriori. Up to n non-empty rules $best[1 \ldots n]$ have been output. For all rules $[x \Rightarrow y] \notin best$: $E(c([x \Rightarrow y])) \leq E(c(best[n]))$ or there exists $[x' \Rightarrow y'] \in best$ with $E(c([x' \Rightarrow y'])) \geq E(c([x \Rightarrow y]))$ and $[x' \Rightarrow y'] \models [x \Rightarrow y]$.

Correctness. The first assertion says that the array *best* stays properly ordered. This assertion is upheld provided that Step "update *best*", 11(c)iii, inserts newly found solutions correctly. Deletion of an element from a sorted list leaves the list sorted, so removing elements in Step 11(c)iii does not harm this assertion.

In Step 6(a), the candidates are generated. The assertion after Step 6(a) claims that no rule with a body that is not in X_i can be better than the n-th best rule found after iteration $i-1$. In the initial iteration, X_i contains all item sets of size 1, so the assertion is trivial. In consecutive iterations, we argue by induction; the assertions after 6(d) and 6(e) for $i-1$ form our inductive assumption. The assertion after 6(e) guarantees that only candidate item sets c are eliminated from X_{i-1} which have a support below τ. The assertion after 6(d) guarantees that no rule with body support below τ can outperform the n-th best rule after iteration $i-1$. All supersets of c have support *at most* $s(c)$; hence, $|x \cup x'|$ can only have the required support of τ if both, x and x' have support τ. This guarantees that the assertions after 6(a) and 6(b) are never violated.

Now we have to look at Table 2. From the assignment of γ in Step 10, the assertion after Step 10 follows immediately. The construction of Y_1 in Step 11(a) assures that the side conditions $x, y \subseteq \{a_1, \ldots a_k\}$ and $x \cap y = \emptyset$ are satisfied by all considered candidate rules.

The assertion after 11(b) claims that no item set *not* included in Y_i can lead to a more accurate rule than $best_i^{j-1}[n]$. In the initial iteration, this claim is trivial. For consecutive iterations, we argue by induction with the assertions after 11(c)i and 11(c)iii for iteration $j-1$ as inductive assumption. The second part of the assertion after 11(c)iii says that there is no rule with body x and confidence below $\gamma / s(x)$

which has predicted accuracy above $best_i^{j-1}[n]$. The assertion after 11(c)i says that Y_{j-1} contains all candidate heads which constitute rules that may outperform $best_i^{j-1}$. In Step 11(b) we construct the candidate heads of size j as supersets of candidate heads of size $j-1$; since the support of a superset of items is at most that of all its subsets, we have that Y_j contains all item sets which may possibly constitute rules with support $\gamma/s(x)$.

This assertion is maintained by the filtering operation in Step 11(c)i. In Step 11(c)ii we refer to Equation 6 in order to measure the predictive accuracy.

In Step 11(c)iii, we update the *best* array. When $[x \Rightarrow y]$ subsumes one or several rules in *best* that are at most as accurate, then these rules are eliminated. When at least one rule is deleted, then a new rule not in *best* becomes the new n-th best hypothesis. The assertion that *best* contains the best rules enumerated so far is violated. However, we can show that only rules with body x that is equal to the current parameter to RuleGen can be deleted. This is because $[x \Rightarrow y]$ cannot subsume a rule with distinct body x' that has been constructed in an earlier invocation of RuleGen: Distinct bodies enumerated in the same iteration of i (loop 6) are never subsets of each other as they have the same cardinality but are not equal; a body x' constructed in an earlier iteration $i' < i$ cannot be a superset of x because $|x'| < |x|$. Hence, whenever a deletion has occurred, we have to restart RuleGen by recurring from Step 10.

The rule for setting γ immediately implies the second part of the assertion after 11(c)iii. After termination of loop 11, the assertion after 11(c)iii holds for all j (note that $x \cap y = \emptyset$ and so the largest possible value for j is $k - |x|$). When no deletion has occurred, then the termination assertion for loop 11 implies the assertion after Step 12.

Here, RuleGen terminates and we are taken back to Step 6(c); the assertion after Step 12 implies the assertion after iteration x of loop 6(c). When loop 6(c) has iterated all $x \in X_i$, the assertion after iteration x of loop 6(c) becomes the assertion after termination of loop 6(c).

In Step 6(d), τ is set such that the assertion after 6(d) is guaranteed. After 6(e), *best* is now $best_i$ (the i-th iteration is completed). So the assertion after the termination of loop 6(c) (together with the assertions after earlier iterations of loop 6) implies the assertion after the i-th iteration of loop 6. Together with the termination criterion of loop 6, we have the assertion after termination of loop 6.

Now all rules in *best* are returned. *best* may contain less than n rules if it contains rules that subsume all other rules. For example, the most general rule $[\ \Rightarrow a_1, \ldots, a_k]$ subsumes all other rules; it would be the result of a database that consists only of items with constant value of 1. The assertion after the termination of loop 6 together with the monotonicity assumption (asserted after every step) immediately imply the assertion of Theorem 2.

Termination. The interesting point is the restarting operation of RuleGen by branching from Step 12 to Step 10. We have to prove that this restarting can only take place finitely often. i and j ($|x|$ and $|y|$) are incremented in the for loops, so no rule is considered twice in Step 11(c)iii. A deletion occurs when a new rule $[x \Rightarrow y]$ subsumes an existing rule $[x' \Rightarrow y']$, in which case the new rule $[x \Rightarrow y]$ is added to *best*. For every body x, there are only finitely many heads y with $y \subseteq \{a_1, \ldots, a_k\}$ and $x \cap y = \emptyset$. So after finitely many replacement steps, the most general rule $[x \Rightarrow y_m]$ with $y_m = \{a_1, \ldots, a_k\} \setminus x$ cannot be replaced by a more general rule with the same body x. $[x \Rightarrow y_m]$ can be replaced by a rule $[x' \Rightarrow y']$ with $x' \subset x$. But again, after finitely many replacements, at the latest the most general rule $[\ \Rightarrow a_1, \ldots, a_k]$ cannot be replaced and RuleGen terminates. This completes the proof of Theorem 2 ∎

ACKNOWLEDGMENT

The author is supported by grant SCHE540/10-1 of the German Science Foundation DFG.

References

Aggarwal, C., & Yu, P. (1998). A new framework for itemset generation. *Proceedings of the Symposium on Principles of Database Systems*.

Agrawal, R., Imielinski, T., & Swami, A. (1993). Mining association rules between sets of items in large databases. *ACM SIGMOD Conference on Management of Data* (pp. 207–216).

Agrawal, R., Mannila, H., Srikant, R., Toivonen, H., & Verkamo, A. (1996). Fast discovery of association rules. *Advances in Knowledge Discovery and Data Mining*.

Bayardo, R., Agrawal, R., & Gunopulos, D. (2000). Constraint based rule mining in large dense databases. *Data Mining and Knowledge Discovery*, *4*, 217–240.

Brijs, T., Vanhoof, K., & Wets, G. (2000). Reducing redundancy in characteristic rule discovery by using integer programming techniques. *Intelligent Data Analysis*, *4*, 229–240.

Brin, S., Motwani, R., Ullmann, J., & Tsur, S. (1997). Dynamic itemset counting and implication rules for market basket data. *Proceedings of the ACM SIGMOD Conference on Managament of Data*.

Gilks, W., Richardson, S., & Spiegelhalter, D. (Eds.). (1995). *Markov chain monte carlo in practice*. Chapman & Hall.

Klemettinen, M., Mannila, H., Ronkainen, P., H.Toivonen, & Verkamo, A. I. (1994). Finding interesting rules from large sets of discovered associacion rules. *Proceedings of the Third International Conference on Information and Knowledge Management*.

Klösgen, W. (1992). Problems in knowledge discovery in databases and their treatment in the statistics interpreter explora. *Journal of Intelligent Systems*, *7*, 649–673.

Klösgen, W. (1995). Assistant for knowledge discovery in data. *Assisting Computer: A New Generation of Support Systems*.

Langford, J., & McAllester, D. (2000). Computable shell decomposition bounds. *Proceedings of the International Conference on Computational Learning Theory*.

Li, J., Shen, H., & Topor, R. (2001). Mining the smallest association rule set for predictions. *Proceedings of the IEEE International Conference on Data Mining*.

Lin, D., & Kedem, Z. (1998). Pincer search: a new algorithm for discovering the maximum frequent set. *Proceedings of the International Conference on Extending Database Technology*.

Mannila, H., Toivonen, H., & Verkano, A. I. (1994). Efficient algorithms for discovering association rules. *Proc. AAAI Workshop on Knowledge Discovery in Databases*, 181–192.

Scheffer, T. (2000). Average-case analysis of classification algorithms for boolean functions and decision trees. *Proceedings of the International Conference on Algorithmic Learning Theory*.

Scheffer, T. (2001). Finding association rules that trade support optimally against confidence. *Proceedings of the European Conference on Principles and Practice of Knowledge Discovery in Databases*.

Scheffer, T., & Joachims, T. (1999). Expected error analysis for model selection. *Proceedings of the Sixteenth International Conference on Machine Learning*.

Scheffer, T., & Wrobel, S. (2002). Finding the most interesting patterns in a database quickly by using sequential sampling. *Journal of Machine Learning Research*, *3*, 833–862.

Toivonen, H. (1996). Sampling large databases for association rules. *Proceedings of the International Conference on Very Large Databases*.

Wrobel, S. (2001). Inductive logic programming for knowledge discovery in databases. *Relational Data Mining*.

Zaki, M. (2000). Generating non-redundant association rules. *Proceedings of the International Conference on Knowledge Discovery and Data Mining*.

Zaki, M., & Hiao, C. (1999). *Charm: an efficient algorithm for closed association rule mining*Technical Report 99-10). Rensselaer Polytechnic Institute.