In *Proceedings of the European Conference of Principles and Practice of Knowledge Discovery in Databases (PKDD-01).* Springer-Verlag, 2001.

Finding Association Rules that Trade Support Optimally Against Confidence

Tobias Scheffer[1,2]

[1] University of Magdeburg, FIN/IWS, PO Box 4120, 39016 Magdeburg, Germany
[2] SemanticEdge, Kaiserin-Augusta-Allee 10-11, 10553 Berlin, Germany
`scheffer@iws.cs.uni-magdeburg.de`

Abstract. When evaluating association rules, rules that differ in both support and confidence have to 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) on realistic knowledge discovery problems.

1 Introduction

Association rules (*e.g.,* [1, 5, 2]), express regularities between sets of data items in a database. [Beer and TV magazine ⇒ chips] is an example of an association rule and expresses that, in a particular store, all customers who buy beer and a TV magazine are also likely to buy chips. In contrast to classifiers, association rules do not make a prediction for *all* database records. When a customer does not buy beer and a magazine, then our example rule does *not* conjecture that he will not buy chips 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 [2] possesses 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. He or she 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 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 is in a way optimal by maximizing the chance of correct predictions on unseen data. We concretize 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. Section 6 concludes.

2 Preliminaries

Let D be a database consisting of one table over binary attributes $a_1, \ldots, a_k$, called items. In general, D has been generated by discretizing the attributes of a relation of an original database D'. For instance, when D' contains an attribute *income*, then D may contain binary attributes $0 \leq \mathit{income} \leq 20k$, $20k < \mathit{income} \leq 40k$, 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 will assume that the resulting association rules provide information on the process that generated the database which will be valid in 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.

Definition 1. *Let D be a database the records r of which are generated by a static process P, let $[x \Rightarrow y]$ be an association rule. The predictive accuracy $c([x \Rightarrow y]) = Pr[r$ satisfies $y|r$ 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.*

Definition 2. *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 the utility of which 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. The intuitive idea is that we "mistrust" the confidence a little. 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 and "corrects" it by returning 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 application of Bayes' rule implies "P(predictive accuracy given confidence and support) = P(confidence given predictive accuracy and support)P(predictive accuracy)/ normalization constant". Note that 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 histogram over the space of all association rules. This histogram 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 histogram. 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.

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

In Equation 4, we apply Equation 2. Since, over all c, the distribution $p(c([x \Rightarrow y]) = c|\hat{c}([x \Rightarrow y]), s(x))$ has to integrate to one (Equation 3), 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.

$$\begin{aligned}
&\int p(c([x \Rightarrow y]) = c|\hat{c}([x \Rightarrow y]), s(x))dc = 1 \qquad (3) \\
&\Leftrightarrow \int \frac{P(\hat{c}([x \Rightarrow y])|c([x \Rightarrow y]) = c, s(x))\pi(c)}{P(\hat{c}([x \Rightarrow y])|s(x))}dc = 1 \qquad (4) \\
&\Leftrightarrow P(\hat{c}([x \Rightarrow y])|s(x)) = \int P(\hat{c}([x \Rightarrow y])|c([x \Rightarrow y]) = c, s(x))\pi(c)dc \qquad (5)
\end{aligned}$$

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.

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

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 body support $s(x)$. 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 shown to be governed by a certain binomial distribution [9]. However, empirical studies (see Section 5 and Figure 2a) show that the shape of the prior can deviate strongly from this binomial distributions. Reasonably accurate estimates can be obtained by following a Markov Chain Monte Carlo [4] approach to estimating the prior, using the available database (see Section 4). For an extended discussion of the complexity of estimating this distributions, see [9, 6].

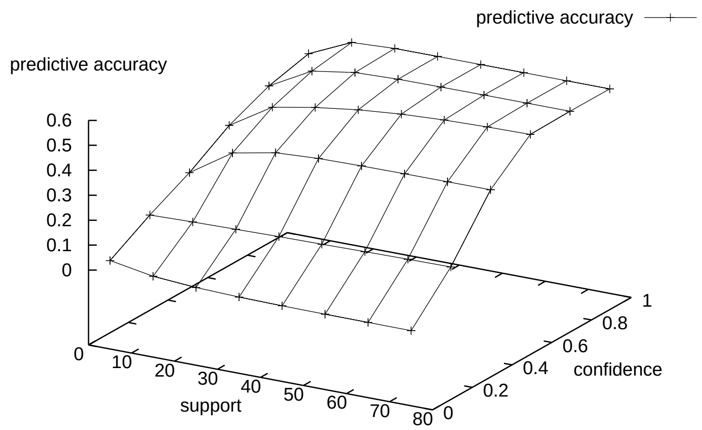

Fig. 1. Contributions of support $s(x)$ and confidence $\hat{c}([x \Rightarrow y])$ to predictive accuracy $c([x \Rightarrow y])$ of rule $[x \Rightarrow y]$

Example curve. Figure 1 shows how expected predictive accuracy, confidence, and body support 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 [1] 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 support thresholds. Instead, we want to find the n best rules.

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.

Algorithm RuleGen(x) (generate all rules with body x)

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** $i = 1 \ldots k$ **With** $a_i \notin x$ **Do** (for all items not in x)
   (a) **If** $i = 1$ **Then Let** $Y_1 = \{\{a_i\}|a_i \notin x\}$ (item sets with one element not in x).
   (b) **Else Let** $Y_i = \{y \cup y' | y, y' \in Y_{i-1}, |y \cup y'| = i\}$ analogous to the generation of candidates in step 6a.
   (c) **For** all $y \in Y_i$ **Do**
      i. Measure the support $s(x \cup y)$. **If** $s(x \cup y) \leq \gamma$, **Then** eliminate y from Y_i and **Continue** the for loop with the next y.
      ii. Equation 6 gives the predictive accuracy $E(c([x \Rightarrow y])|s(x \cup y)/s(x), s(x))$.
      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 rules, 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 step 11(c)iii, **Then** recur from step 10.

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.

Estimating $\pi(c)$. We can estimate π by drawing many hypotheses at random under uniform distribution, measuring their confidence, and recording the resulting histogram. Algorithmically, 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 as there are many more long rules than there are short ones. 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 7 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.

$$P[i \text{ items}] = \frac{\binom{k}{i}(2^i - 1)}{\sum_{j=1}^{k} \binom{k}{j}(2^j - 1)} \tag{7}$$

In step 4 we apply a Markov Chain Monte Carlo style correction to the prior by weighting each prior for rule length i by the probability of a rule length of i.

Enumerating item sets with dynamic minsup threshold. Similarly to the Apriori algorithm, the PredictiveApriori algorithm generates frequent item sets, but using 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 call 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 $best[1 \ldots n]$ which saves the best rules found so far. In step 6d, we refer to $best[n]$, meaning the nth 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 nth 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 accuracy 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 rules 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 2 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.

Theorem 1. *We can decide whether a rule subsumes another rule by two simple subset tests:* $[x \Rightarrow y] \models [x' \Rightarrow y'] \Leftrightarrow x \subseteq x' \wedge y \supseteq y'$. *Moreover, if* $[x \Rightarrow y]$ *is supported by a database* D, *and* $[x \Rightarrow y] \models [x' \Rightarrow y']$ *then this database also supports* $[x' \Rightarrow y']$.

Proofs of Theorems 1 and 2 are left for the full paper. 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 that imply subsumption according to Theorem 1.

Theorem 2. *The PredictiveApriori algorithm (Table 1) returns* n *association rules* $[x_i \Rightarrow y_i]$ *with the following properties. (i) For all returned solutions* $[x \Rightarrow y]$, $[x' \Rightarrow y']$*:* $[x \Rightarrow y] \not\models [x' \Rightarrow y']$. *(ii) Subject to constraint (i), the returned rules maximize* $E(c[x_i \Rightarrow y_i] | \hat{c}([x_i \Rightarrow y_i]), s(x))$ *according to Equation 6.*

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

Table 2. (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]	$E(c\|\hat{c} = 1, s = 1410) = 1$

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 can be obtained by using sampling techniques (*e.g.,* [11]).

5 Experiments

For our experiments, we used 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 dtabase 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 varied 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 2 (top) shows the five 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). Apparently, markets 1, 4, and 6 only sell non-reuseable 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 need to find a uniform measure that is independent of implementation details.

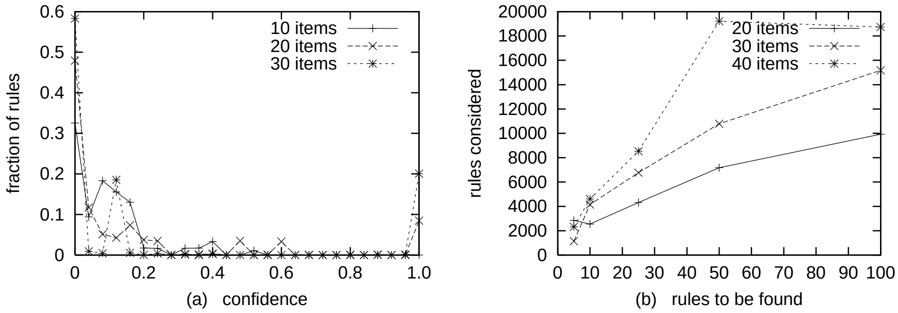

Fig. 2. (a) Confidence prior π for various numbers of items. (b) Number of rules that PredictiveApriori has to consider dependent on the number n of desired solutions.

For Apriori, we count how many association rules have to be compared against the minconf threshold (this number is independent of the actual minconf threshold). We can determine this number from the item sets without actually enumerating all rules. For PredictiveApriori, we measure for how many rules we need to determine the predictive accuracy by evaluating Equation 6.

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 body support and confidence 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 Discussion and Related Results

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

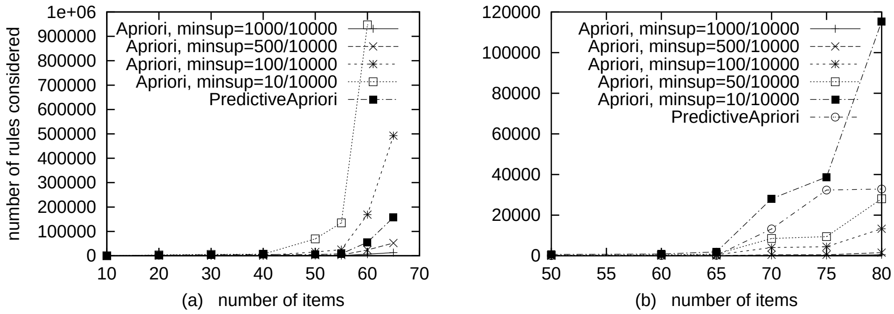

Fig. 3. Time complexity of PredictiveApriori and Apriori, depending on the number of items and (in case of Apriori) of minsup (a) fruit juice problem, (b) KDD cup 1998

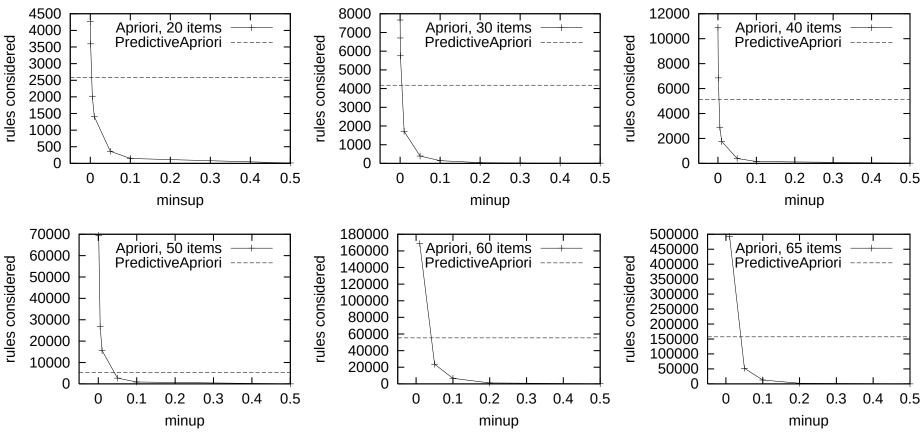

Fig. 4. Number of rules that PredictiveApriori and Apriori and need to consider, depending on the number of items (in case of Apriori also depending on minsup).

the assumption that the database records are independent and identically distributed and requires us to estimate the confidence prior. The PredictiveApriori algorithm does this using a MCMC approach [4].

The Bayesian frequency correction approach that eliminates the optimistical bias of high confidences relates to an analysis of classification algorithms [8] that yields a parameter-free regularization criterion for decision tree algorithms [10]. The PredictiveApriori algorithm returns the n rules which maximize the expected 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 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. Guided by similar ideas, the Midos algorithm [12] performs a similarity test for hypotheses. In [13], a rule discovery algorithm is discussed that selects from classes of redundant rules the most simple, rather than the most general ones. For example, given two equally accurate rules $[a \Rightarrow b]$ and $[a \Rightarrow b, c]$ PredictiveApriori would prefer the latter which predicts more values whereas [13] would prefer the shorter first one.

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 [12].

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 [2], and sampling approaches for estimating the support of item sets [2, 11]. In particular, efficient search for frequent itemsets has been addressed intensely and successfully [7, 3, 14]. Many of these improvements can, and should be, applied to the PredictiveApriori algorithm as well.

References

1. R. Agrawal, T. Imielinski, and A. Swami. Mining association rules between sets of items in large databases. In *ACM SIGMOD Conference on Management of Data*, pages 207–216, 1993.
2. R. Agrawal, H. Mannila, R. Srikant, H. Toivonen, and A. Verkamo. Fast discovery of association rules. In *Advances in Knowledge Discovery and Data Mining*, 1996.
3. S. Brin, R. Motwani, J. Ullmann, and S. Tsur. Dynamic itemset counting and implication rules for market basket data. In *Proceedings of the ACM SIGMOD Conference on Managament of Data*, 1997.
4. W. Gilks, S. Richardson, and D. Spiegelhalter, editors. *Markov Chain Monte Carlo in Practice*. Chapman & Hall, 1995.
5. M. Klemettinen, H. Mannila, P. Ronkainen, H.Toivonen, and A. I. Verkamo. Finding interesting rules from large sets of discovered associacion rules. *Proc. Third International Conference on Information and Knowledge Management*, 1994.
6. J. Langford and D. McAllester. Computable shell decomposition bounds. In *Proceedings of the International Conference on Computational Learning Theory*, 2000.

7. D. Lin and Z. Kedem. Pincer search: a new algorithm for discovering the maximum frequent set. In *Proceedings of the International Conference on Extending Database Technology*, 1998.
8. T. Scheffer. *Error Estimation and Model Selection.* Infix Publisher, Sankt Augustin, 1999.
9. T. Scheffer. Average-case analysis of classification algorithms for boolean functions and decision trees. In *Proceedings of the International Conference on Algorithmic Learning Theory*, 2000.
10. T. Scheffer. Nonparametric regularization of decision trees. In *Proceedings of the European Conference on Machine Learning*, 2000.
11. T. Scheffer and S. Wrobel. A sequential sampling algorithm for a general class of utility functions. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining*, 2000.
12. S. Wrobel. Inductive logic programming for knowledge discovery in databases. In Sašo Džeroski and Nada Lavrač, editors, *Relational Data Mining*, 2001.
13. M. Zaki. Generating non-redundant association rules. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining*, 2000.
14. M. Zaki and C. Hiao. Charm: an efficient algorithm for closed association rule mining. Technical Report 99-10, Rensselaer Polytechnic Institute, 1999.