Sampling Large Databases for Association Rules

Hannu Toivonen
University of Helsinki, Department of Computer Science
FIN-00014 University of Helsinki, Finland
Hannu.Toivonen@Helsinki.FI

Abstract

Discovery of association rules is an important database mining problem. Current algorithms for finding association rules require several passes over the analyzed database, and obviously the role of I/O overhead is very significant for very large databases. We present new algorithms that reduce the database activity considerably. The idea is to pick a random sample, to find using this sample all association rules that probably hold in the whole database, and then to verify the results with the rest of the database. The algorithms thus produce exact association rules, not approximations based on a sample. The approach is, however, probabilistic, and in those rare cases where our sampling method does not produce all association rules, the missing rules can be found in a second pass. Our experiments show that the proposed algorithms can find association rules very efficiently in only one database pass.

1 Introduction

Database mining, or knowledge discovery in databases (KDD), has in the recent years attracted a lot of interest in the database community (for overviews, see [FPSSU96, PSF91]). The interest is motivated by the large amounts of computerized data that many organizations now have about their business. For instance, supermarkets store electronic copies of millions of receipts, and banks and credit card companies maintain extensive transaction histories. The goal in database mining is to analyze these large data sets and to discover patterns or regularities that are useful for decision support.

Discovery of association rules [AIS93] is an interesting subfield of database mining. The motivation for searching association rules has come from the desire to analyze large amounts of *supermarket basket data*. Association rules describe how often items are purchased together: for instance, an association rule "beer $\Rightarrow$ chips (87 %)" states that 87 % of the customers that have bought beer, also have bought chips. Such rules can be useful for decisions concerning, e.g., product pricing, promotions, or store layout. Association rules have been used to discover regularities in other databases as well: university course enrollment data has been analyzed to find combinations of courses taken by the same students, and alarms that occur close to each other in time have been searched in telecommunication alarm data [MTV94].

The size of the data collection has an essential role in database mining. Large data sets are necessary for reliable results; unfortunately, however, the efficiency of the mining algorithms depends heavily on the database. Association rule algorithms require multiple passes over the whole database, and subsequently the database size is by far the most influential factor of the execution time for very large databases. Recent research has managed to reduce the disk I/O activity to two full scans over the database [SON95].

We present algorithms that make only one full pass over the database. The idea is to pick a random sample, use it to determine all association rules that probably hold in the whole database, and then to verify the results with the rest of the database. The algorithms thus produce exact association rules in one full pass over the database. In those rare cases where our sampling method does not produce all association rules, the

Proceedings of the 22nd VLDB Conference
Mumbai(Bombay), India, 1996

missing rules can be found in a second pass. Our extensive experiments show that this method works very well in practice, making the approach attractive especially for very large databases.

This paper is organized as follows. Section 2 reviews the definition of association rules and the problem setting. An overview of previous work in association rule discovery is given in Section 3. In Section 4 we present our approach to discovering association rules using sampling; sampling is analyzed in Section 5. Experimental results are given in Section 6. Finally, Section 7 contains concluding remarks.

2 Association Rules

We shortly review the definitions related to association rules. Consider the supermarket domain and the so called basket data analysis: the goal is to discover associations between items that are often bought together.

Let $R = \{I_1, I_2, \ldots, I_m\}$ represent the set of items that are being sold in the supermarket. Each item I_i is an attribute over the binary domain $\{0, 1\}$, where value 1 indicates that a particular basket has contained the corresponding product. The baskets are represented by rows in a relation $r = \{t_1, \ldots, t_n\}$ over R. That is, each basket is represented by a binary vector of length m, which can be interpreted as the set of items that was bought in the basket.

Consider sets of items that are bought together. The relative *frequency* of an itemset $X \subseteq R$ in r is

$$fr(X, r) = \frac{|\{t \in r \mid t[i] = 1 \text{ for all } I_i \in X\}|}{|r|}.$$

In other words, the frequency of an itemset X is the fraction of the baskets that has contained all the items in X, i.e., the probability that a randomly chosen row from r contains the itemset X.

The goal is to find items that are bought often together. The user determines what is often enough by giving a *frequency threshold*. Given a frequency threshold *min_fr*, we say that a set X is *frequent*[1] if $fr(X, r) \geq min\_fr$. That is, an itemset is frequent if it is contained in at least a fraction *min_fr* of the baskets. For notational convenience, we introduce a shorthand notation $\mathcal{F}(r, min\_fr)$ for the collection of frequent sets in a given relation r with respect to a frequency threshold *min_fr*:

$$\mathcal{F}(r, min\_fr) = \{X \subseteq R \mid fr(X, r) \geq min\_fr\}.$$

Now, finally, we define association rules. Given disjoint and non-empty itemsets $X, Y \subseteq R$, the expression $X \Rightarrow Y$ is an *association rule* over r. The interpretation is that when all the items in set X are in a basket, also the items in Y tend to be in the basket. The strength or *confidence* of the rule is

$$\frac{fr(X \cup Y, r)}{fr(X, r)}.$$

The confidence can be seen as a conditional probability in the analyzed relation r. For instance, assume that the rule {beer} $\Rightarrow$ {chips} has a confidence of 87 %. This means that if a randomly chosen basket contains beer, then it also contains chips with probability 0.87.

A *confidence threshold* is used to exclude rules that are not strong enough to be interesting. Also, the given frequency threshold *min_fr* is used to remove rules that do not apply often enough, i.e., rules such that the frequency of $X \cup Y$ is below the frequency threshold.

The problem of finding association rules can now be stated in the following way. Given a set R of binary attributes, a corresponding relation r, a frequency threshold *min_fr*, and a confidence threshold *min_conf*, find all association rules in r that have confidence at least *min_conf* and frequency at least *min_fr*.

The discovery of association rules can be divided into two phases [AIS93]. First, discover all frequent itemsets $X \subseteq R$. Then, for each frequent X, test for all non-empty subsets $Y \subset X$ if the rule $X \setminus Y \Rightarrow Y$ holds with sufficient confidence. The second part of the problem can be solved in main memory in a straightforward manner once the frequent sets and their frequencies are known. Discovery of the frequent sets is the hard part of the problem: with m items, there are 2^m potentially frequent itemsets. For all but the smallest m, efficient methods for locating the frequent sets are needed. For the rest of this article, we consider the task of discovering all frequent sets.

3 Previous Work on Association Rules

Since the introduction of the problem of mining association rules [AIS93] several generate-and-test type of algorithms have been proposed for the task of discovering frequent sets. An efficient breadth-first or *level-wise* method for generating *candidate* sets, i.e., potentially frequent sets, has been presented in [AS94, MTV94, AMS+96]. This method—also called Apriori—is the core of all known algorithms except the original one [AIS93] and its variation for SQL [HS93], which have been shown to be inferior to the level-wise method [AS94, MTV94, AMS+96].

An alternative strategy for the database pass, using inverted structures and a general purpose DBMS, has been considered in [HKMT95]. The most efficient algorithm so far, in particular in terms of database operations, is Partition [SON95]. We review the level-wise method and the Partition algorithm below.

[1] In the literature, also the terms *large* and *covering* have been used for *frequent*, and the term *support* for *frequency*.

Other work related to association rules includes the problem of mining rules with generalizations [HF95, HKMT95, SA95], management of large amounts of discovered association rules [KMR+94, TKR+95], and a theoretical analysis of an algorithm for a class of KDD problems including the discovery of frequent sets [MT96]. A connectionist approach to mining rules is presented in [LSL95].

3.1 Level-Wise Algorithms

Algorithms for discovering frequent sets are based on the observation that if a set is not frequent then its supersets can not be frequent. All current algorithms [MTV94, AS94, HKMT95, HF95, PCY95, SA95, AMS+96] start on level 1 by evaluating the frequencies of singleton itemsets. On level k, candidate itemsets X of size k are generated such that all subsets of X are frequent. For instance, in the second pass, candidates are pairs of items such that both items are frequent. After the frequencies of the candidates on level k have been evaluated, new candidates for level $k+1$ are generated and evaluated. This is repeated until no new candidates can be generated. The efficiency of this approach is based on *not* generating and evaluating those candidate itemsets that cannot be frequent, given all smaller frequent sets.

The level-wise algorithms make one pass over the database for each level. There are thus K or $K+1$ passes over the database, where K is the size of the largest frequent set.[2] Sometimes, if there are only few candidates in the last iterations, candidates can be generated and tested for several levels at once.

3.2 Partition Algorithm

The Partition algorithm [SON95] reduces the database activity: it computes all frequent sets in two passes over the database. The algorithm works also in the level-wise manner, but the idea is to partition the database to sections small enough to be handled in main memory. That is, a part is read once from the disk, and level-wise generation and evaluation of candidates for that part are performed in main memory without further database activity.

The first database pass consists of identifying in each part the collection of all *locally frequent* sets. For the second pass, the union of the collections of locally frequent sets is used as the candidate set. The first pass is guaranteed to locate a superset of the collection of frequent itemsets; the second pass is needed to merely compute the frequencies of the sets.

[2] $K+1$ passes are needed if there are candidates of size $K+1$.

4 Sampling for Frequent Sets

A fairly obvious way of reducing the database activity of knowledge discovery is to use only a random sample of the relation and to find approximate regularities. In other words, one can trade off accuracy against efficiency. This can be very useful: samples small enough to be handled totally in main memory can give reasonably accurate results. Or, approximate results from a sample can be used to set the focus or to adjust parameters for a more complete discovery phase.

It is often important to know the frequencies and confidences of association rules exactly. In business applications, for example for large volumes of supermarket sales data, even very small differences can be significant. When relying on results from sampling alone, one also takes the risk of loosing valid association rules because their frequency in the sample is below the threshold.

Using a random sample to get approximate results is fairly straightforward. Below we give bounds for sample sizes, given the desired accuracy of the results. Our main contribution is, however, to show that exact frequencies can be found efficiently, by analyzing first a random sample and then the whole database as follows. Use a random sample to efficiently find a superset $\mathcal{S}$ of the collection of frequent sets. A superset can be determined efficiently by applying the level-wise method on the sample in main memory, and by using a lowered frequency threshold. In terms of the Partition algorithm, discover locally frequent sets from one part only, and with a lower threshold. Then use the rest of the database to compute the exact frequencies of the sets. This approach requires only one full pass over the database, and two passes in the worst case.

4.1 The Negative Border

As a tool for further analysis, consider the concept of *negative border*. Given a collection $\mathcal{S} \subseteq \mathcal{P}(R)$ of sets, closed with respect to the set inclusion relation, the negative border $\mathcal{B}d^-(\mathcal{S})$ of $\mathcal{S}$ consists of the minimal itemsets $X \subseteq R$ not in $\mathcal{S}$ [MT96].

The collection of all frequent sets is always closed with respect to set inclusion. For instance, let $R = \{A, \ldots, F\}$ and assume the collection $\mathcal{F}(r, \mathit{min\_fr})$ of frequent sets (with some r and $\mathit{min\_fr}$) is

$$\{A\}, \{B\}, \{C\}, \{F\}, \{A,B\}, \{A,C\}, \{A,F\}, \{C,F\}, \{A,C,F\}.$$

The negative border of this collection contains now, e.g., the set $\{B,C\}$: it is not in the collection, but all its subsets are. The whole negative border is

$$\mathcal{B}d^-(\mathcal{F}(r, \mathit{min\_fr})) = \{\{B,C\}, \{B,F\}, \{D\}, \{E\}\}.$$

Algorithm 1
Input: A relation r over a binary schema R, a frequency threshold *min_fr*, a sample size *ss*, and a lowered frequency threshold *low_fr*.
Output: The collection $\mathcal{F}(r, min\_fr)$ of frequent sets and their frequencies, or its subset and a failure report.
Method:

```
1.    draw a random sample $s$ of size $ss$ from $r$;
      // find frequent sets in the sample:
2.    compute $\mathcal{S} := \mathcal{F}(s, low\_fr)$ in main memory;
      // database pass:
3.    compute $\mathcal{F} :=$
          $\{X \in \mathcal{S} \cup Bd^-(\mathcal{S}) \mid fr(X,r) \geq min\_fr\}$;
4.    for all $X \in \mathcal{F}$ do output $X$ and $fr(X,r)$;
5.    report if there possibly was a failure;
```

The intuition behind the concept is that given a (closed) collection $\mathcal{S}$ of sets that are frequent, the negative border contains the "closest" itemsets that could be frequent, too. The candidate collections of the level-wise algorithms are, in effect, the negative borders of the collections of frequent sets found so far, and the collection of all candidates that were not frequent is the negative border of the collection of frequent sets. Of particular importance is the fact that the negative border needs to be evaluated, in order to be sure that no frequent sets are missed [MT96].

4.2 Frequent Set Discovery Using Sampling

We now apply the concept of negative border to using sampling for finding frequent sets. It is not sufficient to locate a superset $\mathcal{S}$ of $\mathcal{F}(r, min\_fr)$ using the sample and then to test $\mathcal{S}$ in r, because the negative border $Bd^-(\mathcal{F}(r, min\_fr))$ needs to be checked, too. If we have $\mathcal{F}(r, min\_fr) \subseteq \mathcal{S}$, then obviously $\mathcal{S} \cup Bd^-(\mathcal{S})$ is a sufficient collection to be checked. Determining $\mathcal{S} \cup Bd^-(\mathcal{S})$ is easy: it consists of all sets that were candidates of the level-wise method in the sample. Algorithm 1 presents the principle: search for frequent sets in the sample, but lower the frequency threshold so much that it is very unlikely that any frequent sets are missed. Then evaluate the sets and their border, i.e., all sets that were evaluated in the sample, also in the rest of the database.

Sometimes, unfortunately, it may happen that we find out that not all necessary sets have been evaluated. There has been a *failure* in the sampling if all frequent sets are not found in one pass, i.e., if there is a frequent set X in $\mathcal{F}(r, min\_fr)$ that is not in $\mathcal{S} \cup Bd^-(\mathcal{S})$. A *miss* is a frequent set Y in $\mathcal{F}(r, min\_fr)$ that is in $Bd^-(\mathcal{S})$.

If there are no misses, then the algorithm is guaranteed to have found all frequent sets. Misses themselves are not a problem. They are evaluated in the whole relation, and thus they are not actually missed by the algorithm. Misses, however, indicate a potential failure. Namely, if there is a miss Y, then some superset of Y might be frequent but not in $\mathcal{S} \cup Bd^-(\mathcal{S})$. A simple way to recognize a potential failure is thus to check if there are any misses.

Example. Assume that we have a relation r with 10 million rows over attributes $A, \ldots, F$, and that we want to find the frequent sets with the threshold 2 %. Algorithm 1 randomly picks a small fraction s of r, say 20,000 rows, and keeps this sample s in main memory. The algorithm can now, without any further database activity, discover very efficiently sets that are frequent in the sample.

To make (almost) sure that the collection of frequent sets in the sample includes all sets that really are frequent in r, the frequency threshold is lowered to, e.g., 1.5 %. Algorithm 1 now determines the collection $\mathcal{S} = \mathcal{F}(s, 1.5\ \%)$ from the sampled 20,000 rows.

Let the maximal sets of $\mathcal{S}$ be

$$\{A,B,C\}, \{A,C,F\}, \{A,D\}, \{B,D\}.$$

Since the threshold was lowered, $\mathcal{S}$ is likely to be a superset of the collection of frequent sets in r. In the pass over the rest of r, the frequency of all sets in $\mathcal{S}$ and $Bd^-(\mathcal{S})$ is evaluated. That is, in addition to the sets that are frequent in the sample, we evaluate also those candidates that were not frequent in the sample, i.e., the negative border

$$\{B,F\}, \{C,D\}, \{D,F\}, \{E\}.$$

The goal is to discover the collection $\mathcal{F}(r, 2\ \%)$ of frequent sets in r. Let sets

$$\{A,B\}, \{A,C,F\}.$$

and all their subsets be the frequent sets. All frequent sets are in $\mathcal{S}$, so they are evaluated and their exact frequencies are known after the full database pass. We also know that we have found all frequent sets since also the negative border

$$\{B,C\}, \{B,F\}, \{D\}, \{E\}$$

of $\mathcal{F}(r, 2\ \%)$ was evaluated and found not to be frequent.

Now assume a slightly different situation, where the set $\{B,F\}$ turns out to be frequent in r, that is, $\{B,F\}$ is a miss. What we have actually missed is the set $\{A,B,F\}$ which can be frequent in r, since all its subsets are. In this case Algorithm 1 reports that there possibly is a failure. $\square$

The problem formulation is now the following: given a database r and a frequency threshold *min_fr*, use a

Algorithm 2
Input: A relation r over a binary schema R, a frequency threshold *min_fr*, and a subset $\mathcal{S}$ of $\mathcal{F}(r, \mathit{min\_fr})$.
Output: The collection $\mathcal{F}(r, \mathit{min\_fr})$ of frequent sets and their frequencies.
Method:

```
1.    repeat
2.        compute $\mathcal{S} := \mathcal{S} \cup \mathcal{B}d^-(\mathcal{S})$;
3.    until $\mathcal{S}$ does not grow;
      // database pass:
4.    compute $\mathcal{F} := \{X \in \mathcal{S} \mid fr(X, r) \geq \mathit{min\_fr}\}$;
5.    for all $X \in \mathcal{F}$ do output $X$ and $fr(X, r)$;
```

random sample s to determine a collection $\mathcal{S}$ of sets such that $\mathcal{S}$ contains with a high probability the collection of frequent sets $\mathcal{F}(r, \mathit{min\_fr})$. For efficiency reasons, a secondary goal is that $\mathcal{S}$ does not contain unnecessarily many other sets. We present two variants of algorithms using sampling. One of them is guaranteed to find all frequent sets in one pass in fraction $1 - \Delta$ of the cases, where Δ is a parameter given by the user.

In the fraction of cases where a possible failure is reported, all frequent sets can be found by making a second pass over the database. Algorithm 2 can be used to extend Algorithm 1 with a second pass in such a case. The algorithm simply computes the collection of all sets that possibly could be frequent. This can be done in a similar way that candidates are generated in the algorithms for finding frequent sets.

5 Analysis of Sampling

Next we analyze the relation of sample size to the accuracy of results. We first consider how accurate the frequencies computed from a random sample are. As has been noted before, samples of reasonable size provide good approximations for frequent sets [MTV94, AMS+96]. Related work on using a sample for approximately verifying the truth of sentences of tuple calculus is considered in [KM94].

5.1 Accuracy and Sample Size

We consider the absolute error of the estimated frequency. Given an attribute set $X \subseteq R$ and a random sample s from a relation over binary attributes R, the *error* $e(X, s)$ is the difference of the frequencies:

$$e(X, s) = |fr(X) - fr(X, s)|,$$

where $fr(X)$ is the frequency of X in the relation from which s was drawn.

To analyze the error, we consider sampling with replacement. The reason is that we want to avoid making other assumptions of the database size except that it is large. For sampling with replacement the size of the database has no effect on the analysis, so the results apply, in principle, on infinitely large databases. To emphasize this, the relation from which a sample is drawn is not shown in the notation of the error. For very large databases there is practically no difference between sampling with and without replacement.

In the following we analyze the number of rows in the sample s that contain X, denoted $m(X, s)$. The random variable $m(X, s)$ has binomial distribution, i.e., the probability of $m(X, s) = c$, denoted $Pr[m(X, s) = c]$, is

$$\binom{|s|}{c} fr(X)^c (1 - fr(X))^{|s|-c}.$$

First consider the necessary size of a sample, given requirements on the size of the error. The following theorem gives a lower bound for the size of the sample, given an error bound ε and a maximum probability δ for an error that exceeds the bound.

Theorem 1 Given an attribute set X and a random sample s of size

$$|s| \geq \frac{1}{2\varepsilon^2} \ln \frac{2}{\delta}$$

the probability that $e(X, s) > \varepsilon$ is at most δ.

Proof. We have

$$Pr[e(X, s) > \varepsilon] = Pr[|fr(X, s) - fr(X)|\,|s| > \varepsilon\,|s|].$$

The Chernoff bounds [AS92] give an upper bound

$$2e^{-2(\varepsilon\,|s|)^2/|s|} = \delta$$

for the probability. □

Table 1 gives values for the sufficient sample size $|s|$, for $\varepsilon = 0.01, 0.001$ and $\delta = 0.01, 0.001, 0.0001$. With the tolerable error ε around 0.01, samples of a reasonable size suffice. E.g., if a chance of 0.0001 for an error of more than 0.01 is acceptable, then a sample of size 50,000 is sufficient. For many applications these parameter values are perfectly reasonable—errors in the input data may be more likely than 0.0001. In such cases, approximate rules can be produced based on a sample, i.e., in constant time independent of the size of r. With tighter error requirements the sample sizes can be quite large.

The result above is for a given set X. The following corollary gives a result for a more stringent case: given a collection $\mathcal{S}$ of sets, with probability $1 - \Delta$ there is no set in $\mathcal{S}$ with error more than ε.

Table 1: Sufficient sample sizes, given ε and δ.

ε	δ	$\|s\|$
0.01	0.01	27,000
0.01	0.001	38,000
0.01	0.0001	50,000
0.001	0.01	2,700,000
0.001	0.001	3,800,000
0.001	0.0001	5,000,000

Corollary 2 Given a collection $\mathcal{S}$ of sets and a random sample s of size

$$|s| \geq \frac{1}{2\varepsilon^2} \ln \frac{2|\mathcal{S}|}{\Delta},$$

the probability that there is a set $X \in \mathcal{S}$ such that $e(X, s) > \varepsilon$ is at most Δ.

Proof. By Theorem 1, the probability for $e(X, s) > \varepsilon$ for a given set X is at most $\frac{\Delta}{|\mathcal{S}|}$. Since there are $|\mathcal{S}|$ such sets, the probability in question is at most Δ. $\square$

The Chernoff bound is not always very tight, and in practice the exact probability from the binomial distribution or its normal approximation is much more useful.

5.2 Probability of a Failure

Consider now the proposed approach to finding all frequent sets exactly. The idea was to locate a superset of the collection of frequent sets by discovering frequent sets in a sample with a lower threshold.

Assume we have a sample s and a collection $\mathcal{S} = \mathcal{F}(s, \textit{low\_fr})$ of sets. What can we say about the probability of a failure? A failure is first of all possible if there is a set that should have been checked but was not. That is, sampling might have failed if there is such a collection $\mathcal{T}$ of misses that $\mathcal{B}d^-(\mathcal{S} \cup \mathcal{T})$ has sets that are not in $\mathcal{B}d^-(\mathcal{S})$. Given such a collection $\mathcal{T}$ of potential misses, estimating the probability that all sets in $\mathcal{T}$ actually are misses is difficult since the sets are not independent. To get an upper bound one can compute, as if the sets were independent, the probability of any of the sets in $\mathcal{T}$ being a miss. This, again, is bounded by considering the whole $\mathcal{B}d^-(\mathcal{S})$ instead of subcollections $\mathcal{T}$. In summary: to estimate the probability of a failure we can compute an upper bound for the probability of a miss.

An interesting aspect is that the upper bound can be computed on the fly when processing the sample. Thus, if an upper bound for the probability of a failure is desired, the frequency threshold *low_fr* can be adjusted dynamically to reduce the probability of a miss.

A variation of Theorem 1 gives the following result on how to set the lowered frequency threshold so that misses are avoided with a high probability.

Theorem 3 Given a frequent set X, a random sample s, and a probability parameter δ, the probability that X is a miss is at most δ when

$$\textit{low\_fr} < \textit{min\_fr} - \sqrt{\frac{1}{2|s|} \ln \frac{1}{\delta}}.$$

Proof. Using the Chernoff bounds again—this time for one-sided error—we have

$$Pr[\textit{fr}(X, s) < \textit{low\_fr}] \leq e^{-2(\sqrt{\frac{1}{2|s|}\ln\frac{1}{\delta}}|s|)^2/|s|} = \delta.$$

$\square$

Consider now the number of sets checked in the second pass by Algorithm 2, in the case of a failure. The collection $\mathcal{S}$ can, in principle, grow much. Each independent miss can in the worst case generate as many new candidates as there are frequent sets. Note, however, that if the probability that a given set is a miss is at most δ, then the probability of l independent misses can be at most δ^l. In a pathological case the misses are very much dependent: there is a very large frequent set X such that its subsets only occur with the whole X. If X is not frequent in a sample, then also all its subsets are missed. At least in the application domain of supermarket basket data this kind of sets are very unlikely to exist at all.

6 Experiments

We now describe the experiments we conducted in order to assess the practical feasibility of using samples for finding frequent sets. We present two new variants of algorithms using sampling and give experimental results.

6.1 Test Organization

We used three synthetic data sets from [AS94] in our tests. These databases model supermarket basket data, and they have been used as benchmarks for several association rule algorithms [AS94, HKMT95, PCY95, SON95, AMS+96]. The central properties of the data sets are the following. There are $|R| = 1,000$ attributes, and the average number T of attributes per row is 5, 10, and 20. The number $|r|$ of rows is approximately $100,000$. The average size I of maximal frequent sets is 2, 4, and 6. Table 2 summarizes the parameters for the data sets; see [AS94] for details of the data generation.

We assume that the real data sets from which association rules are discovered can be much larger than

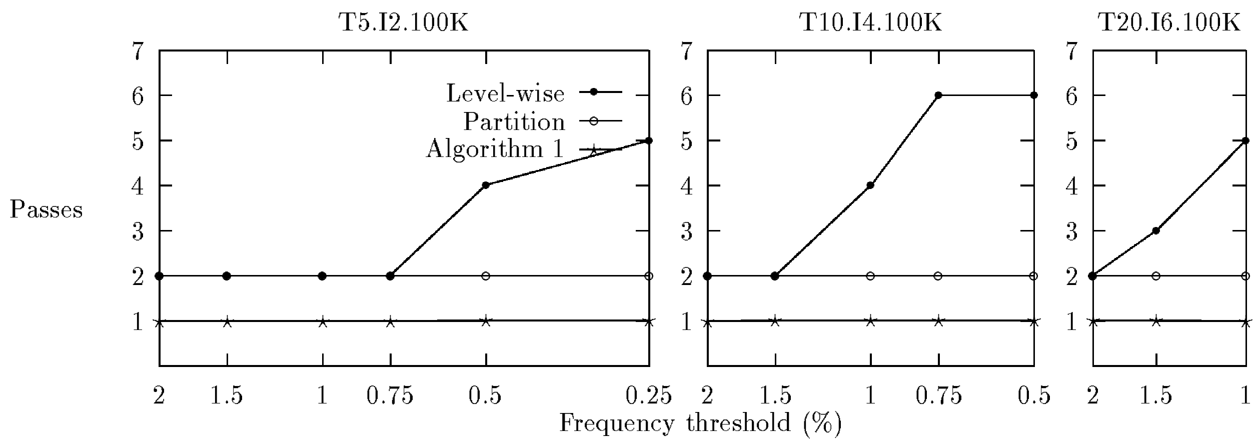

Figure 1: The number of database passes made by frequent set algorithms.

Table 2: Synthetic data set characteristics (T = row size on average, I = size of maximal frequent sets on average).

Data set name	$\|R\|$	T	I	$\|r\|$
T5.I2.D100K	1,000	5	2	97,233
T10.I4.D100K	1,000	10	4	98,827
T20.I6.D100K	1,000	20	6	99,941

the test data sets. To make the experiments fair we use sampling with replacement. This means that the real data sets could, in principle, have been arbitrary large data sets such that these data sets represent their distributional properties.

We considered sample sizes from 20,000 to 80,000. Samples of these sizes are large enough to give good approximations and small enough to be handled in main memory. Since our approach is probabilistic, we repeated every experiment 100 times for each parameter combination. Altogether, over 10,000 trials were run. We did not experiment with all the frequency thresholds used in the literature; the repeated trials would have taken too long. The tests were run on a PC with 90 MHz Pentium processor and 32 MB main memory under Linux operating system.

6.2 Number of Database Passes and Misses

We experimented with Algorithm 1 with the above mentioned sample sizes 20,000 to 80,000. We selected the lowered threshold so that the probability of missing any given frequent set X is less than $\delta = 0.001$, i.e., given any set X with $fr(X) \geq min\_fr$, we have

$$Pr[fr(X, s) < low\_fr] < 0.001.$$

The lowered threshold depends on the frequency threshold and the sample size. The lowered threshold

Table 3: Lowered frequency thresholds (%) for $\delta = 0.001$.

	Sample size			
min_fr (%)	20,000	40,000	60,000	80,000
0.25	0.13	0.17	0.18	0.19
0.50	0.34	0.38	0.40	0.41
0.75	0.55	0.61	0.63	0.65
1.00	0.77	0.83	0.86	0.88
1.50	1.22	1.30	1.33	1.35
2.00	1.67	1.77	1.81	1.84

values are given in Table 3; we used in the computations the exact probabilities from the binomial distribution, not the Chernoff bounds.

Figure 1 shows the number of database passes for the three different types of algorithms: the level-wise algorithm, Partition, and the sampling Algorithm 1. Each of the data points in the results shown for Algorithm 1 is the average value over 100 trials. Explaining the results is easy. The level-wise algorithm makes $K(+1)$ passes over the database, where K is the size of the largest frequent set. The Partition algorithm makes two passes over the database when there are any frequent sets. For Algorithm 1, the fraction of trials with misses is expected to be larger than $\delta = 0.001$, depending on how many frequent sets have a frequency relatively close to the threshold and are thus likely misses in a sample. The algorithm has succeeded in finding all frequent sets in one pass in almost all cases. The number of database passes made by Partition algorithm is practically twice that of Algorithm 1, and the number of passes of the level-wise algorithm is up to six times that of Algorithm 1.

Table 4 shows the number of trials with misses for each data set, sample size, and frequency threshold. In

Table 4: Number of trials with misses.

T5.I2.D100K

	Sample size			
min_fr (%)	20,000	40,000	60,000	80,000
0.25	0	1	0	0
0.50	0	1	0	1
0.75	0	0	0	0
1.00	0	0	0	0
1.50	0	0	0	0
2.00	0	0	0	0

T10.I4.D100K

	Sample size			
min_fr (%)	20,000	40,000	60,000	80,000
0.50	0	2	1	1
0.75	0	1	1	1
1.00	1	0	1	1
1.50	0	2	0	0
2.00	0	0	0	0

T20.I6.D100K

	Sample size			
min_fr (%)	20,000	40,000	60,000	80,000
1.00	0	0	0	0
1.50	1	1	1	0
2.00	0	1	0	2

each set of 100 trials, there have been zero to two trials with misses. The overall fraction of trials with misses was 0.0038. We repeated the experiment with $\delta = 0.01$, i.e., so that the miss probability of any given frequent set is at most 0.01. This experiment gave misses in fraction 0.041 of all the trials. In both cases the fraction of trials with misses was about four times δ.

The actual amount of reduction in the database activity depends very much on the storage structures. For instance, if the database has 10 million rows, a disk block contains on average 100 rows, and the sample size is 20,000, then the sampling phase could read up to 20 % of the database. For the design and analysis of sampling methods see, e.g, [OR89]. The related problem of sampling for query estimation is considered in more detail in [HS92]. An alternative for randomly drawing each row in separation is, of course, to draw whole blocks of rows to the sample. Depending on how randomly the rows have been assigned to the blocks, this method can give very good or very bad results.

The reduction in database activity is achieved at the cost of considering some attribute sets that the level-wise algorithm does not generate and check. Table 5 shows the average number of sets considered for data set T10.I4.D100K with different sample sizes, and the number of candidate sets of the level-wise algorithm. The largest absolute overhead occurs with low thresholds, where the number of itemsets considered has grown from 318,588 by 64,694 in the worst case. This growth is not significant for the total execution time since the itemsets are handled entirely in main memory. The relative overhead is larger with higher thresholds, but since the absolute overheads are very small the effect is negligible. Table 5 indicates that larger samples cause less overhead (with equally good results), but that for sample sizes from 20,000 to 80,000 the difference in the overhead is not significant.

To obtain a better picture of the relation of δ and the experimental number of trials with misses, we conducted the following test. We took 100 samples (for each frequency threshold and sample size) and determined the lowered frequency threshold that would have given misses in one out of the hundred trials. Figure 2 presents these results (as points), together with lines showing the lowered thresholds with $\delta = 0.01$ or 0.001, i.e., the thresholds corresponding to miss probabilities of 0.01 and 0.001 for a given frequent set. The frequency thresholds that would give misses in fraction 0.01 of cases approximate surprisingly closely the thresholds for $\delta = 0.01$. Experiments with a larger scale of sample sizes give comparable results. There are two explanations for the similarity of the values. One reason is that there are not necessarily many potential misses, i.e., not many frequent sets with frequency relatively close to the threshold. Another reason that contributes to the similarity is that the sets are not independent.

In the case of a possible failure, Algorithm 2 generates iteratively all new candidates and makes another pass over the database. In our experiments the number of frequent sets missed—when any were missed—was one or two for $\delta = 0.001$, and one to 16 for $\delta = 0.01$. The number of candidates checked on the second pass was very small compared to the total number of itemsets checked.

6.3 Guaranteed $1 - \Delta$ Success Probability

Setting the lowered threshold for Algorithm 1 is not trivial: how to select it so that the probability of a failure is low but there are not unnecessarily many sets to check? An automatic way of setting the parameter would be desirable. Consider, for instance, an interactive mining tool. It would be very useful to know in advance how long an operation will approximately take—or, in the case of mining association rules, how many database passes there will be.

We now present two algorithms that are guaranteed to find all frequent sets in a given fraction $1 - \Delta$ of

Table 5: Number of attribute sets considered for data set T10.I4.D100K.

	Sample size				
min_fr (%)	20,000	40,000	60,000	80,000	Level-wise
0.50	382,282	368,057	359,473	356,527	318,588
0.75	290,311	259,015	248,594	237,595	188,024
1.00	181,031	158,189	146,228	139,006	97,613
1.50	52,369	40,512	36,679	34,200	20,701
2.00	10,903	7,098	5,904	5,135	3,211

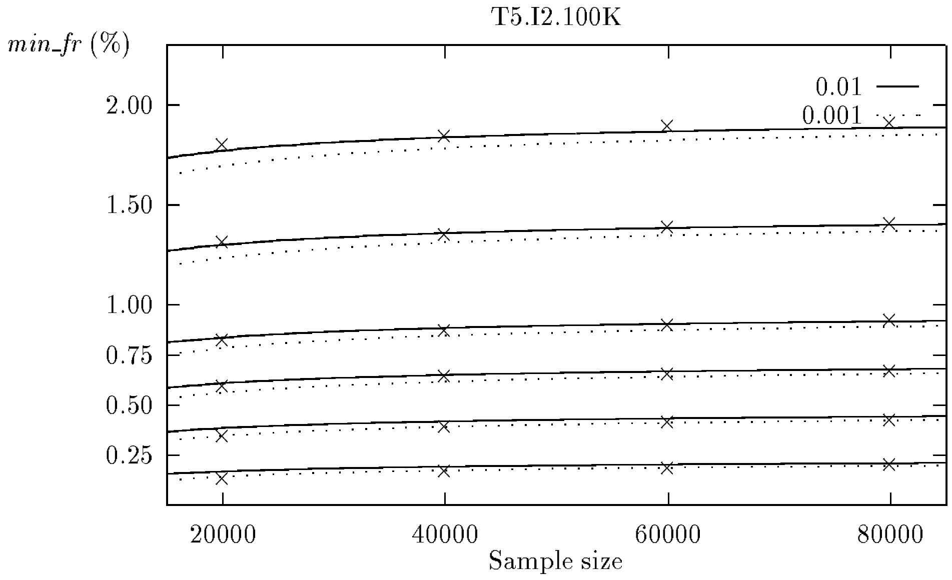

Figure 2: Frequency thresholds giving misses in 0.01 cases (points) and lowered thresholds with $\delta = 0.01$ and 0.001 (lines).

the cases. The first one uses a simple greedy principle with which an optimal lowered frequency threshold can be found. The other algorithm is not as optimal, but its central phase is almost identical to the level-wise method and it is therefore very easy to incorporate into existing software. We present experimental results with this latter algorithm.

The greedy Algorithm 3 starts with an empty set $\mathcal{S}$. It then greedily increases the probability of success by adding the most probable misses to $\mathcal{S}$ until the upper bound for the probability of a miss is at most Δ.

Algorithm 4 is a very simple variation of the level-wise algorithm. It utilizes also the upper bound approximation described in Section 5: it monitors the upper bound of the probability of a miss and keeps the probability small by lowering the frequency threshold *low_fr*, when necessary, for the rest of the algorithm.

To use the level-wise algorithm for the discovery of frequent sets in a sample with the dynamic adjustment of the lowered threshold, the only modification concerns the phase where candidates are either added to the collection of frequent sets or thrown away. In Algorithm 4, lines 8 – 13 have been added to implement this change. Every time there is a candidate X that is not frequent in the sample, compute the probability p that X is frequent. If the total probability P of a miss increases too much (see below) with such an X, then lower the frequency threshold *low_fr* to the frequency of X in the sample for the rest of the algorithm. Thus X is eventually considered frequent in the sample, and so are all following candidate sets that would increase the overall probability of a miss at least as much as X.

We use the following heuristic to decide whether the possibility of a miss increases too much. Given a parameter γ in $[0, 1]$, the frequency threshold is lowered if the probability p is larger than fraction γ of the "remaining error reserve" $\Delta - P$. More complex heuristics for changing the frequency threshold could be

Algorithm 3
Input: A relation r over a binary attributes R, a sample size ss, a frequency threshold min_fr, and a maximum miss probability Δ.
Output: The collection $\mathcal{F}(r, min\_fr)$ of frequent sets and their frequencies at least in fraction $1 - \Delta$ of the cases, and a subset of $\mathcal{F}(r, min\_fr)$ and a failure report in the rest of the cases.
Method:

```
1.  draw a random sample $s$ of size $ss$ from $r$;
2.  $\mathcal{S} := \emptyset$;
    // find frequent sets in the sample:
3.  while probability of a miss $> \Delta$ do begin
4.      select $X \in \mathcal{B}d^-(\mathcal{S})$ with the highest
            probability of being a miss;
5.      $\mathcal{S} := \mathcal{S} \cup \{X\}$;
6.  end;
    // database pass:
7.  compute $\mathcal{F} :=$
        $\{X \in \mathcal{S} \cup \mathcal{B}d^-(\mathcal{S}) \mid fr(X, r) \geq min\_fr\}$;
8.  for all $X \in \mathcal{F}$ do output $X$ and $fr(X, r)$;
9.  report if there possibly was a failure;
```

developed, e.g., by taking into account the number of candidates on the level and whether the number of frequent sets per level is growing or shrinking. The observations made from Figure 2 hint that the lowered threshold can be set in the start-up to roughly correspond to the desired probability of a miss, i.e., for $\Delta = 0.01$ the lowered threshold could be set as for $\delta = 0.01$.

We tested Algorithm 4 with maximum miss probability $\Delta = 0.1$ and dynamic adjustment parameter $\gamma = 0.01$ for two frequency thresholds for each data set. The fraction of trials with misses is shown in Table 6. The fraction succesfully remained below $\Delta = 0.1$ in each set of experiments.

As Table 6 shows, the fraction of cases with misses was actually less than half of Δ. The reason is that with a small γ the algorithm tends to be conservative and keeps a lot of space for the probability of a miss in reserve. This is useful when there can be very many candidates. The negligible trade-off is that the algorithm may consider unnecessarily many sets as frequent in the sample.

7 Concluding Remarks

We have discussed the use of sampling in the task of discovering association rules in large databases. We described algorithms that take a random sample from a database, identify those sets that probably are fre-

Algorithm 4
Input: A relation r over a binary attributes R, a sample size ss, a frequency threshold min_fr, and a maximum miss probability Δ.
Output: The collection $\mathcal{F}(r, min\_fr)$ of frequent sets and their frequencies at least in fraction $1 - \Delta$ of the cases, and a subset of $\mathcal{F}(r, min\_fr)$ and a failure report in the rest of the cases.
Method:

```
1.  draw a random sample $s$ of size $ss$ from $r$;
2.  $P := 0$;
3.  $low\_fr := min\_fr$;
    // find frequent sets in the sample:
4.  $\mathcal{C}_1 := \{\{A\} \mid A \in R\}$;
5.  $i := 1$;
6.  while $\mathcal{C}_i \neq \emptyset$ do begin
7.      for all $X \in \mathcal{C}_i$ do begin
8.          if $fr(X, s) < low\_fr$ then do
9.              $p := Pr[X \text{ is frequent}]$;
10.             if $p/(\Delta - P) > \gamma$ then
11.                 $low\_fr := fr(X, s)$
12.             else $P := 1 - (1 - P)(1 - p)$;
13.         end;
14.         if $fr(X, s) \geq low\_fr$ then
15.             $\mathcal{S}_i := \mathcal{S}_i \cup \{X\}$;
16.     end;
17.     $i := i + 1$;
18.     $\mathcal{C}_i :=$ ComputeCandidates($\mathcal{S}_{i-1}$);
19. end;
    // database pass:
20. compute $\mathcal{F} :=$
        $\{X \in \bigcup_{j<i} \mathcal{C}_j \mid fr(X, r) \geq min\_fr\}$;
21. for all $X \in \mathcal{F}$ do output $X$ and $fr(X, r)$;
22. report if there possibly was a failure;
```

quent in the database, and then compute the exact frequencies from the rest of the database.

The described algorithms are fairly simple. Two of the algorithms have the property that they discover all frequent sets in one pass in a fraction $1 - \Delta$ of the cases, where Δ is given by the user. Those cases where sampling possibly missed frequent sets can be recognized, and the missing sets can be found in a second pass.

Our experiments showed that the approach works: all frequent sets can actually be found in almost one pass over the database. For the efficiency of mining association rules in very large databases the reduction of disk I/O is significant.

This work raises two obvious open questions. The first one concerns the discovery of association rules: is there a method for discovering exact association rules

Table 6: Fraction of trials with misses with $\Delta = 0.10$.

T5.I2.D100K

	Sample size			
min_fr (%)	20,000	40,000	60,000	80,000
0.50	0.03	0.03	0.00	0.02
1.00	0.00	0.00	0.00	0.00

T10.I4.D100K

	Sample size			
min_fr (%)	20,000	40,000	60,000	80,000
0.75	0.01	0.04	0.02	0.01
1.50	0.00	0.02	0.04	0.01

T20.I6.D100K

	Sample size			
min_fr (%)	20,000	40,000	60,000	80,000
1.00	0.02	0.01	0.01	0.01
2.00	0.01	0.03	0.01	0.03

in at most one pass? The other question is, how much can database mining methods gain from sampling in general?

Acknowledgements

I would like to thank Heikki Mannila for his many helpful comments on an earlier draft. I also wish to thank the anonymous referees for their constructive criticism.

This research has been supported by the Academy of Finland.

References

[AIS93] Rakesh Agrawal, Tomasz Imielinski, and Arun Swami. Mining association rules between sets of items in large databases. In *Proceedings of ACM SIGMOD Conference on Management of Data (SIGMOD'93)*, pages 207 – 216, May 1993.

[AMS+96] Rakesh Agrawal, Heikki Mannila, Ramakrishnan Srikant, Hannu Toivonen, and A. Inkeri Verkamo. Fast discovery of association rules. In Usama M. Fayyad, Gregory Piatetsky-Shapiro, Padhraic Smyth, and Ramasamy Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining*, pages 307 – 328. AAAI Press, Menlo Park, CA, 1996.

[AS92] Noga Alon and Joel H. Spencer. *The Probabilistic Method.* John Wiley Inc., New York, NY, 1992.

[AS94] Rakesh Agrawal and Ramakrishnan Srikant. Fast algorithms for mining association rules in large databases. In *Proceedings of the Twentieth International Conference on Very Large Data Bases (VLDB'94)*, pages 487 – 499, September 1994.

[FPSSU96] Usama M. Fayyad, Gregory Piatetsky-Shapiro, Padhraic Smyth, and Ramasamy Uthurusamy, editors. *Advances in Knowledge Discovery and Data Mining.* AAAI Press, Menlo Park, CA, 1996.

[HF95] Jiawei Han and Yongjian Fu. Discovery of multiple-level association rules from large databases. In *Proceedings of the 21st International Conference on Very Large Data Bases (VLDB'95)*, pages 420 – 431, Zurich, Swizerland, 1995.

[HKMT95] Marcel Holsheimer, Martin Kersten, Heikki Mannila, and Hannu Toivonen. A perspective on databases and data mining. In *Proceedings of the First International Conference on Knowledge Discovery and Data Mining (KDD'95)*, pages 150 – 155, Montreal, Canada, August 1995.

[HS92] Peter J. Haas and Arun N. Swami. Sequential sampling procedures for query size estimation. In *Proceedings of ACM SIGMOD Conference on Management of Data (SIGMOD'92)*, pages 341 – 350, San Diego, CA, June 1992.

[HS93] Maurice Houtsma and Arun Swami. Set-oriented mining of association rules. Research Report RJ 9567, IBM Almaden Research Center, San Jose, California, October 1993.

[KM94] Jyrki Kivinen and Heikki Mannila. The power of sampling in knowledge discovery. In *Proceedings of the Thirteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS'94)*, pages 77 – 85, Minneapolis, MN, May 1994.

[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 *Proceedings of the Third International Conference on Information and Knowledge Management (CIKM'94)*, pages 401 – 407, Gaithersburg, MD, November 1994. ACM.

[LSL95] Hongjun Lu, Rudy Setiono, and Huan Liu. Neurorule: A connectionist approach to data mining. In *Proceedings of the 21st International Conference on Very Large Data Bases (VLDB'95)*, pages 478 – 489, Zurich, Swizerland, 1995.

[MT96] Heikki Mannila and Hannu Toivonen. On an algorithm for finding all interesting sentences. In *Cybernetics and Systems, Volume II, The Thirteenth European Meeting on Cybernetics and Systems Research*, pages 973 – 978, Vienna, Austria, April 1996.

[MTV94] Heikki Mannila, Hannu Toivonen, and A. Inkeri Verkamo. Efficient algorithms for discovering association rules. In *Knowledge Discovery in Databases, Papers from the 1994 AAAI Workshop (KDD'94)*, pages 181 – 192, Seattle, Washington, July 1994.

[OR89] Frank Olken and Doron Rotem. Random sampling from B^+ trees. In *Proceedings of the Fifteenth International Conference on Very Large Data Bases (VLDB'89)*, pages 269 – 277, Amsterdam, August 1989.

[PCY95] Jong Soo Park, Ming-Syan Chen, and Philip S. Yu. An effective hash-based algorithm for mining association rules. In *Proceedings of ACM SIGMOD Conference on Management of Data (SIGMOD'95)*, pages 175 – 186, San Jose, California, May 1995.

[PSF91] Gregory Piatetsky-Shapiro and William J. Frawley, editors. *Knowledge Discovery in Databases.* AAAI Press, Menlo Park, CA, 1991.

[SA95] Ramakrishnan Srikant and Rakesh Agrawal. Mining generalized association rules. In *Proceedings of the 21st International Conference on Very Large Data Bases (VLDB'95)*, pages 407 – 419, Zurich, Swizerland, 1995.

[SON95] Ashok Savasere, Edward Omiecinski, and Shamkant Navathe. An efficient algorithm for mining association rules in large databases. In *Proceedings of the 21st International Conference on Very Large Data Bases (VLDB'95)*, pages 432 – 444, Zurich, Swizerland, 1995.

[TKR$^+$95] Hannu Toivonen, Mika Klemettinen, Pirjo Ronkainen, Kimmo Hätönen, and Heikki Mannila. Pruning and grouping of discovered association rules. In *Workshop Notes of the ECML-95 Workshop on Statistics, Machine Learning, and Knowledge Discovery in Databases*, pages 47 – 52, Heraklion, Crete, Greece, April 1995.