On-Line Sampling Methods for Discovering Association Rules

Carlos Domingo and Ricard Gavaldà [*]

Department de LSI, Universitat Politècnica de Catalunya
Campus Nord, Mòdul C5, 08034-Barcelona, Spain
{carlos, gavalda}@lsi.upc.es

Osamu Watanabe

Dept. of Mathematical and Computing Sciences
Tokyo Institute of Technology, Tokyo 152-8552, Japan
watanabe@is.titech.ac.jp

December 4, 1998

Abstract

Association rule discovery is one of the prototypical problems in data mining. In this problem, the input database is assumed to be very large and most of the algorithms are designed to minimize the number of scans of the database. Enumerating association rules is usually an expensive task due to the size of the input database. A proposed approach for reducing the running time of this process is random sampling. Of course, any implementation of an algorithm that uses sampling must solve the problem of determining which sample size is appropriate. Previous research of sampling for association rule mining has approached this problem concluding that, in general, the theoretically obtained sample size bounds are far from what is observed in practice. In this paper, we try to reduce this gap between theory and practice. We propose two on-line sampling algorithms for association rule mining. Our algorithms maintain the same theoretical guarantees of previous approaches while using a much smaller number

[*]Supported by EC ESPRIT Project no. 20244 (ALCOM-IT), CICYT Project TIC97-1475-CE and EC ESPRIT Working Group NeuroCOLT2 (nr. EP27150).

of transactions in most of the cases. In the experiments we report, this improvement is often by an order of magnitude.

1 Introduction

The problem of mining association rules was first proposed in [3] and since then it has become one of the central tasks in knowledge discovery and data mining. The prototypical application of association rules is what is known as basket data analysis. Basket data consists of a list of the items bought by every customer in, for instance, a large department store, together with a transaction identifier. Given this large amount of data the goal is to infer statements of the form "98% of the customers that buy beer also purchase peanuts". Such statements are called association rules (see a more formal definition in the following section) and are typically used for cross-marketing, store layout or customer segmentation. Some other successful applications of association rules have been shown in different domains like decision support, telecommunications alarm diagnosis, prediction, university enrollments, etc.

Since its introduction, efficient enumeration of association rules and several related problems have become an important topic in data mining with too many papers to refer here (see [1] or [9] for further references). Typically, when mining association rules, a heuristic approach is taken wherein one first enumerates all of the *frequent sets* in the database. A frequent set is any set of items such that the fraction of transactions in the database where all those items appears is at least σ, where σ is some minimum frequency threshold provided by the user.

Some algorithms like Apriori [2, 10] or Partition [13] work with a very large database stored in a remote system and were designed originally to minimize the number of passes over the database. The computational bottleneck of these algorithms is the size of the database. An obvious approach for reducing the complexity of these algorithms is random sampling. This paper is devoted to developing efficient sampling algorithms for association rule discovery.

The importance of sampling for association rule mining has been recognized by several researchers [10, 1, 15, 16]. The usual approach is to take a portion of the database randomly of a previously determined size and then calculate the frequency of the itemsets over the sample using a lower minimum support threshold σ' that is slightly smaller than the user-specified minimum support σ.

Obviously, this approach may produce a wrong set of frequent itemsets since some itemsets with frequency bigger than σ' but smaller than σ might be wrongly declared frequent. Some techniques for removing these errors by performing a posterior check using the whole database have been proposed by Toivonen in [15]. Another problem might arise from obtaining a random sample not representative at all of the database regularities. In general, the possible errors might be reduced by obtaining a sufficiently large sample. Therefore, the most important problem with sampling is how to determine the sample size appropriately so that we have certain guarantees that the sample is representative and that the number of itemsets that are wrongly declared frequent or non-frequent is small. This need for a "formula" that determines the appropriate sample size for every possible input problem is particularly important if an algorithm that uses sampling is going to be implemented in automatic tool.

The main statistical tool used in the literature for deciding appropriate sample sizes is the Chernoff bound, in any of its forms. The relevance of this bound has been recognized in the context of sampling for association rules. All the theoretical analyses that we are aware of are obtained through the Chernoff bound [10, 1, 15, 16]. It is a common criticism to this bound that it overestimates the necessary sample size. Experimental results [15, 16] showed that, compared with the sample sizes obtained through the Chernoff bound, much smaller sample sizes suffice for deciding whether an itemset is frequent or not with high confidence. An immediate improvement can be obtained by computing the exact probabilities from the binomial distribution but this number still overestimates. It is a worst case upper bound on a probability while we expect in practice an average case behavior.

While it is true that the Chernoff bound, being a very general bound as it is, overestimates, there is another source of inefficiency in the analysis of the sample size needed performed by previous researchers. This source of inefficiency comes from the fact that, in order to obtain the appropriate sample size, the minimum support σ is used as a lower bound of the real frequency of a given itemset. Obviously, this is always done in this way because the real frequency of the itemset is not known, it is precisely what we are trying to determine. However, if the frequency of the itemset is much bigger or much smaller [1] than the minimum

[1]If the minimum support is very small, as it happens in some applications like basket data analysis, the case where the real support is much smaller than the minimum one can be omitted. We have also considered it here for completeness.

support σ, a much smaller sample should be enough to notice whether the itemset is frequent or not. This factor crucially affects the sample size obtained from the Chernoff bound as we can see in the following numerical example. Suppose that the we want to test through sampling whether an itemset X is frequent using a minimum support $\sigma = 2\%$. For this, we set the lowered support threshold to $\sigma' = 1.5\%$. Using the Chernoff bound we want to obtain a sufficiently big sample so that the probability that the frequency of X is less than σ' in the sample when X is frequent in the whole database is less than a confidence value δ. A straightforward application of the Chernoff bound tells us that the sample should be of size $1/\gamma^2$ (ignoring constant factors and the dependency on the confidence value δ) where $\gamma = \sigma - \sigma'$. For our choice of parameters this value is 40000. Now, suppose that X was in fact frequent and that its frequency p is slightly larger than σ, for instance $p = 4\%$. If we know the value of p, we can use the Chernoff bound in the same way as before to determine the sample size obtaining this time $1/\gamma_0^2$ where $\gamma_0 = p - \sigma'$. Substituting the numerical values we obtain a sufficient sample size of 1600, a significant improvement on the bound above.

In this paper, we try to reduce the gap between theory and practice proposing a new sampling method that achieves a worst case upper bound on the sample size of $\mathcal{O}(\ln(1/\gamma)/\gamma\gamma_0)$ for one algorithm and $\mathcal{O}(\ln(1/\gamma)/\gamma_0^2)$ for the other instead of the commonly used bound of $\mathcal{O}(1/\gamma^2)$ (ignoring the dependency on δ). This should be advantageous when p and σ' are sufficiently apart. Furtermore, we propose a third improvement that, although also achieving a worst case bound of $\mathcal{O}(\ln(1/\gamma)/\gamma_0^2)$ it will perform much better for support thresholds that are below 16.66%, a very reasonable assumption for the problem we are studying.

The main feature of our method is that it does not need to know p in advance. The algorithm obtains the transactions randomly one by one in an on-line fashion. It is adaptively changing so that its performance converges to the real frequency of the itemset without knowing it. This contrasts with the usual batch approach where the sample size is calculated a priori using just γ and then the whole sample is required at once. Moreover, since our algorithms work in an on-line fashion, we expect that in practice the number of transactions required will be even smaller than the worst case bound that we provide.

We have performed some experiments with our algorithms to confirm that, in many situations, they can use much fewer transactions than the usual batch approach. In our experiments we have ignored the added technological cost of performing sampling in an on-line fashion. Our experimental results show that, for support values slightly apart from above σ and below σ' the number of transactions used by our algorithms is much smaller than the usual batch approach, in some cases of more than one order of magnitude.

Some of the ideas described in this paper were developed in [4] for the problem of model or hypothesis selection.

This paper is organized as follows. In the next section we formally describe the problem of mining association rules and state the usual batch sampling algorithm. In Section 3 we give the first solution, an on-line sampling algorithm that achieves a worst case bound of $\mathcal{O}(\ln(1/\gamma)/\gamma\gamma_0)$. In Sections 4 and 5 we give two improvements, both of them with a worst case bound of $\mathcal{O}(\ln(1/\gamma)/\gamma_0^2)$, where the second one will perform better for small supports. For all these algorithms we provide precise statements of their reliability and efficiency. Section 6 reports on some of the experimental results obtained with the algorithms. Finally, in Section 7 we briefly summarize the results and discuss future work.

2 Preliminaries

The association rule mining task was introduced in [3] and it can be stated as follows. Let $\mathcal{I} = \{i_1, \ldots, i_n\}$ be a set of n items, and $\mathcal{T}$ a database of transactions, where each transaction has a unique identifier and contains a set of items. A set of items is also called an *itemset*. The *support* or *frequency* of an itemset X, denoted by $\mathrm{fr}(X)$, is the number of transactions on which it occurs as a subset. Given a user-specified *minimum support* value (denoted by σ), we say that an itemset X is *σ-frequent* if its support is more than the minimum support, i.e. $\mathrm{fr}(X) \geq \sigma$. When σ is clear from the context we will just say that an itemset X is frequent. An *association rule* is an expression of the form $A \to B$ where A and B are itemsets. The *support of a rule* is given by $\mathrm{fr}(A \cup B)$ and its *confidence* as $\mathrm{fr}(A \cup B)/\mathrm{fr}(A)$, i.e. the conditional probability that a transaction contains B given that it contains A. We will say that, given a user-specified *minimum confidence* value (denoted by δ), a rule $A \to B$ has *high confidence* if its confidence is more than the minimum value, i.e. $\mathrm{fr}(A \cup B)/\mathrm{fr}(A) \geq \delta$.

Given σ and δ, the association rule mining task, as described in [1], consists of two steps:

1. Find all frequent itemsets.

2. Generate high confidence rules.

The second step is relatively straightforward. Rules of the form $X\backslash Y \to Y$, where $X \subset Y$, are generated for all frequent itemsets X, provided the rules have at least confidence δ. On the other hand, the first step is computationally and I/O intensive and the algorithms studied in this paper are devoted to solve it efficiently.

In order to distinguish between the real frequency in the whole database and the frequency in a sample S of a given itemset X, we will extend the $\mathrm{fr}(X)$ notation in the following way. Given a database $\mathcal{T}$ and a sample S obtained from it, the frequency of an itemset X in the database will be denoted by $\mathrm{fr}(X, \mathcal{T})$ and the frequency of X in the sample will be denoted by $\mathrm{fr}(X, S)$. Through all the paper we will assume that σ is the user specified minimum support and σ' is the lowered threshold level where $\gamma = \sigma - \sigma'$.

We start by describing more precisely the problem we want to solve. Let C be a collection of itemsets, σ the minimum support threshold, σ' a lowered support threshold and δ a confidence value. Our goal is to design an algorithm that, given as input C, σ, σ' and δ, does the following. It randomly selects a sample S of sufficient size such that, based on it, the following two conditions hold with probability more than $1 - \delta$:

(C1) If $X \in C$ is σ-frequent, that is, $\mathrm{fr}(X, \mathcal{T}) \geq \sigma$, the algorithm declares X frequent.

(C2) If $X \in C$ is not σ'-frequent, that is $\mathrm{fr}(X, \mathcal{T}) < \sigma'$, then the algorithm declares X non-frequent.

Therefore, we do not specify the behavior of the algorithm for the itemsets in C that are non-frequent but close to being frequent. That is, the algorithm might wrongly declare frequent all the itemsets $X \in C$ such that $\sigma' < \mathrm{fr}(X, \mathcal{T}) < \sigma$. On the other hand, for the other itemsets we want the algorithm to classify then correctly with high probability. Obviously, the closer we select σ' to σ the smaller number of itemsets will be wrongly classified by the algorithm but this higher precision will translate on a bigger sample. An algorithm like the described above is particularly suitable to be combined with an algorithm like Apriori to calculate the frequency of an itemset without using the whole database. Since our sampling algorithm will (with high probability) detect all the frequent itemsets plus some others, we can remove all these errors by doing a posteriori check with the whole database. See the paper by Toivonen for discussion about this [15]. In this paper, we just concentrate on the problem of how to efficiently calculate the value of $\text{fr}(C, \mathcal{T})$ for a collection of itemsets C using random sampling.

Once our problem is formally stated, we go on describing more precisely the usual approach for solving it and the number of transactions required to achieve the performance mentioned above. We call this algorithm BSAR from *Batch Sampling for Association Rules* since the sample size needed is calculated in advance and all the sample can be obtained in batch in contrast with our algorithms that work in an on-line manner. Algorithm BSAR is shown in Figure 1.

```
Algorithm  BSAR ($C, \sigma, \sigma', \delta$)
1   $\gamma \leftarrow \sigma - \sigma'$;
2   $S \leftarrow$ obtain randomly $2\ln(|C|/\delta)/\gamma^2$ transactions from $\mathcal{T}$;
3   for all $X \in C$ do
4     if $\text{fr}(X, S) \geq \sigma' + \gamma/2$ then declare $X$ frequent;
5                                 else declare $X$ non-frequent;
6   end
```

Figure 1: Pseudo-code of algorithm BSAR.

The main tool for proving the correctness of BSAR and the other algorithm in the paper is the Chernoff bound [7, 11]. More precisely, we are using the additive form of the Chernoff bound, that is usually referred to in the computer science literature as the Hoeffding bound, that we state in the following theorem.

Theorem 2.1. (Hoeffding bound) For any $t \geq 1$ and p, $0 \leq p \leq 1$, consider t independent random variables $X_1, \ldots, X_t$ each of which takes values 0 and 1 with probabilities $1-p$ and p. Then for any $\epsilon > 0$, we have

$$\Pr\{\sum_{i=1}^{t} X_i > pt + \epsilon t\} < \exp(-2\epsilon^2 t), \qquad \Pr\{\sum_{i=1}^{t} X_i < pt - \epsilon t\} < \exp(-2\epsilon^2 t).$$

An straightforward application of this bound yields the following theorem.

Theorem 2.2. With probability at least $1-\delta$, algorithm BSAR on input C, σ, σ', δ satisfies conditions (C1) and (C2) for all itemsets $X \in C$.

The lowered threshold σ' can be set to any value. For obtaining a very high precision, we might want to have a lowered threshold that is an ϵ fraction away from σ, that is, $\sigma' = \sigma(1-\epsilon)$. In that case, the number of transactions used by the algorithm will be proportional to $1/(\epsilon\sigma)^2$.

Notice that the number of transactions used by algorithm BSAR is totally independent of the real frequency of itemset X, it depends mostly on $\sigma - \sigma'$. The rest of the paper is devoted to solving this problem by introducing algorithms that reduce the dependence on $\gamma = \sigma - \sigma'$ as much as possible and introduce a dependence on the real frequency of itemset X.

3 Variance Based Sampling Algorithm

Here we present our first on-line sampling algorithm together with its proof of reliability and efficiency. Figure 2 shows the algorithm that we have called VSAR from *Variance Based Sampling for Association Rules.* Here for given γ and δ, we denote by $\tau(\gamma,\delta)$ the smallest integer τ that satisfies the following inequality.

$$\tau \;\geq\; \frac{2}{\gamma^2}\ln\left(\frac{|C|(\tau+1)}{\delta}\right).$$

We start showing a theorem about the performance of algorithm VSAR for the frequent itemsets X in C.

Theorem 3.1. Let $\sigma > 0$ be a minimum support threshold, σ' a lowered support threshold, where $\gamma = \sigma - \sigma' > 0$ and let C be a collection of itemsets. Then for any itemset $X \in C$ such that $\mathrm{fr}(X, \mathcal{T}) = p \geq \sigma$, algorithm VSAR on input C, σ, σ' and $\delta < 1$ declares X frequent with probability more than $1-\delta$. Moreover, algorithm VSAR uses at most $\tau_0 = \tau\gamma/\gamma_0$ transactions before classifying X with probability at least $1-\delta$ where $\gamma_0 = p - \sigma'$.

Proof. First we concentrate on a single frequent itemset $X \in C$ and we bound the probability that X is wrongly declared non-frequent by algorithm VSAR. For this case, it suffices to consider the sum of the following two probabilities.

$$\begin{aligned} P_1(X) &= \Pr\{\, \forall t \leq \tau\,[\mathrm{fr}(X,S_t) < \sigma' + (\gamma/2)\tau/t\,]\,\}, \quad \text{and} \\ P_2(X) &= \Pr\{\, \exists t \leq \tau\,[\mathrm{fr}(X,S_t) < \sigma' - (\gamma/2)\tau/t\,]\,\}. \end{aligned}$$

```
Algorithm VSAR ($C, \sigma, \sigma', \delta$)
 1  $\gamma \leftarrow \sigma - \sigma'$;   $\tau \leftarrow \tau(\delta, \gamma)$;
 2  $t \leftarrow 0$;  $S_t \leftarrow \emptyset$;
 3  while $C \neq \emptyset$ and  $t \leq \tau$ do
 4     obtain $r$ from $\mathcal{T}$ randomly;
 5     $S_{t+1} \leftarrow S_t \cup \{r\}$;   $t \leftarrow t + 1$;
 6     for all $X \in C$ do
 7        if $\mathrm{fr}(X, S_t) > \sigma' + (\gamma/2)\tau/t$ then
 8           declare $X$ frequent and remove it from $C$;
 9        if $\mathrm{fr}(X, S_t) < \sigma' - (\gamma/2)\tau/t$ then
10           declare $X$ non-frequent and remove it from $C$;
11     end
12  end
13  declare all remaining $X \in C$ non-frequent;
```

Figure 2: Pseudo-code of algorithm VSAR.

Let us first consider probability $P_1(X)$. Clearly, it is less than $P_1'(X) = \Pr\{\, \mathrm{fr}(X, S_\tau) < \sigma' + \gamma/2 \,\}$, which we bound as follows.

$$\begin{aligned}
P_1'(X) &= \Pr\{\, \mathrm{fr}(X, S_\tau) + \mathrm{fr}(X, \mathcal{T}) < \sigma' + (\gamma/2)\tau/t + \mathrm{fr}(X, \mathcal{T}) \,\} \\
&\leq \Pr\{\, \sigma - \sigma' - (\gamma/2)\tau/t < \mathrm{fr}(X, \mathcal{T}) - \mathrm{fr}(X, S_\tau) \,\} \\
&= \Pr\{\, (\gamma/2)\tau/t < \mathrm{fr}(X, \mathcal{T}) - \mathrm{fr}(X, S_\tau) \,\} \\
&\leq \exp\left(-2 \left(\frac{\gamma}{2} \frac{\tau}{t} \right)^2 \tau \right) \;\leq\; \left(\frac{\delta}{|C|(\tau + 1)} \right).
\end{aligned}$$

where the last inequalities follow from the Hoeffding bound (Theorem 2.1) and our choice of τ. On the other hand, we bound $P_2(X)$ as follows:

$$
\begin{aligned}
P_2(X) &= \Pr\{\, \exists t \le \tau \, [\mathrm{fr}(X, S_t) < \sigma' - (\gamma/2)\tau/t \,] \,\} \\
&\le \sum_{t \le \tau} \Pr\{\, \mathrm{fr}(X, S_t) < \mathrm{fr}(X, \mathcal{T}) - (\gamma/2)\tau/t, \} \\
&= \sum_{t \le \tau} \Pr\{\, (\gamma/2)\tau/t < \mathrm{fr}(X, \mathcal{T}) - \mathrm{fr}(X, S_t), \} \\
&\le \sum_{t \le \tau} \exp\left(-2\left(\frac{\gamma}{2}\frac{\tau}{t}\right)^2 t\right) \\
&\le \sum_{t \le \tau} \left(\frac{\delta}{|C|(\tau+1)}\right) \le \tau \frac{\delta}{|C|(\tau+1)}.
\end{aligned}
$$

where again, in the last line we have used the Hoeffding bound and the definition of τ.

Thus, the probability that VSAR makes an error on any frequent itemset in C can be bounded as follows.

$$
\begin{aligned}
P_{error} &= \Pr\{\, \exists X \in C,\, \mathrm{fr}(X, \mathcal{T}) \ge \sigma \text{ and VSAR declares } X \text{ non-frequent } \} \\
&\le \sum_{X \in C} \Pr\{\, \mathrm{fr}(X, \mathcal{T}) \ge \sigma \text{ and VSAR declares } X \text{ non-frequent } \} \\
&\le \sum_{X \in C} P_1(X) + P_2(X) \le |C| \left(\frac{\delta}{|C|(\tau+1)} + \tau \frac{\delta}{|C|(\tau+1)}\right) = \delta.
\end{aligned}
$$

Now we discuss the second part of the theorem concerning the efficiency of the algorithm. Here we count the number of transactions used by every frequent itemset $X \in C$, i.e. the case where $\mathrm{fr}(X, \mathcal{T}) = p \ge \sigma$, before it is removed from C (and, by the above proof, declared frequent with high probability). We claim that the probability that a frequent itemset is not removed from C after τ_0 steps is very small. More precisely, we want to bound this probability:

$$P_3(X) = \Pr\{\forall t \le \tau_0 \, [\mathrm{fr}(X, S_t) \le \sigma' + (\gamma/2)\tau/t]\}$$

which can be bounded by probability P_3' as follows:

$$
\begin{aligned}
P_3' &= \Pr\{\mathrm{fr}(X, S_{\tau_0}) \le \sigma' + (\gamma/2)\tau/\tau_0 \,\} \\
&= \Pr\{\mathrm{fr}(X, S_{\tau_0}) + \mathrm{fr}(X, \mathcal{T}) \le \sigma' + (\gamma/2)\gamma_0/\gamma + \mathrm{fr}(X, \mathcal{T}) \,\} \\
&= \Pr\{\mathrm{fr}(X, \mathcal{T}) - \sigma' - (\gamma/2)\gamma_0/\gamma \le \mathrm{fr}(X, \mathcal{T}) - \mathrm{fr}(X, S_{\tau_0}) \,\} \\
&= \Pr\{\gamma_0 - (\gamma/2)\gamma_0/\gamma \le \mathrm{fr}(X, \mathcal{T}) - \mathrm{fr}(X, S_{\tau_0}) \,\} \\
&\le \Pr\{\gamma_0/2 \le \mathrm{fr}(X, \mathcal{T}) - \mathrm{fr}(X, S_{\tau_0}) \,\} \\
&\le \exp\left(-2\left(\frac{\gamma_0}{2}\right)^2 \tau_0\right) \le \exp\left(-2\left(\frac{\gamma}{2}\right)^2 \tau\right) \le \frac{\delta}{|C|}.
\end{aligned}
$$

Thus, the probability that there is a frequent itemset X that it is not classified before τ_0 iterations is the following.

$$\begin{aligned} P_{time} &= \Pr\{\exists X \in C, \mathrm{fr}(X,\mathcal{T}) \geq \sigma \text{ such that } \forall t \leq \tau_0\, [\mathrm{fr}(X,S_t) \leq \sigma' + (\gamma/2)\tau/t]\} \\ &\leq \sum_{X \in C} \Pr\{\forall t \leq \tau_0\, [\,\mathrm{fr}(X,\mathcal{T}) \geq \sigma \text{ and } \mathrm{fr}(X,S_t) \leq \sigma' + (\gamma/2)\tau/t\,]\,\} \\ &\leq \sum_{X \in C} \Pr\{\,\mathrm{fr}(X,\mathcal{T}) \geq \sigma \text{ and } \mathrm{fr}(X,S_{\tau_0}) \leq \sigma' + (\gamma/2)\tau/\tau_0\,\} \\ &\leq \sum_{X \in C} \Pr\{\,\mathrm{fr}(X,\mathcal{T}) - \sigma' - (\gamma/2)\gamma_0/\gamma < \mathrm{fr}(X,\mathcal{T}) - \mathrm{fr}(X,S_{\tau_0})\,\} \\ &\leq \sum_{X \in C} \Pr\{\,\gamma_0/2 < \mathrm{fr}(X,\mathcal{T}) - \mathrm{fr}(X,S_{\tau_0})\,\} \leq |C|\frac{\delta}{|C|} = \delta. \end{aligned}$$

□

Next we consider the case of the non-frequent itemset $X \in C$ such that $\mathrm{fr}(X,\mathcal{T}) \leq \sigma'$. For this case, we can prove the following theorem symmetric to Theorem 3.1.

Theorem 3.2. Let $\sigma > 0$ be a minimum support threshold, σ' a lowered support threshold, where $\gamma = \sigma - \sigma' > 0$ and let C be a collection of itemsets. Then, for any itemset $X \in C$ such that $\mathrm{fr}(X,\mathcal{T}) = p \leq \sigma'$, algorithm VSAR on input C, σ, σ' and $\delta < 1$ declares X non-frequent with probability at least $1-\delta$. Moreover, algorithm VSAR uses at most $\min\{\tau, \tau_0\}$ transactions before classifying X with probability at least $1-\delta$ where $\tau_0 = \tau\gamma/\gamma_0$ and $\gamma_0 = \sigma' - p$.

Proof. We start with the reliability, that is, the probability that VSAR declares frequent a non-frequent itemset $X \in C$. For this, we need to bound the following probability:

$$\Pr\{\;\exists t \leq \tau\,[\,\mathrm{fr}(X,S_t) > \sigma' + (\gamma/2)\tau/t\,]\;\}.$$

Using the same analysis as for probability $P_2(X)$ in the proof of Theorem 3.1 we can show that this probability is smaller than $\delta/|C|$. Moreover, considering all the non-frequent itemsets, the probability that any of them is wrongly declared frequent is smaller than δ as claimed.

Now we focus on the second part of the theorem, the number of steps before an itemset is found to be non-frequent. Obviously, the algorithm always terminates within τ steps and τ is smaller than τ_0 for any $\gamma_0 \leq \gamma$. Hence, we consider the other case, that is, $\gamma_0 > \gamma$ or equivalently $\mathrm{fr}(X,\mathcal{T}) = p < \sigma' - \gamma$. For this case, we can show the probability that any itemset $X \in C$ with $\mathrm{fr}(X, \mathcal{T}) = p < \sigma' - p$ is not declared non-frequent within $\tau_0 = \tau\gamma/\gamma_0$ steps is smaller than δ. The rest of the proof mimics the proof of complexity in Theorem 3.1 and is omitted. $\square$

4 First Refinement: the Adaptive Sampling Algorithm

We now show how to improve previous algorithm to obtain a bound on the number of transactions that has less dependency on γ. Recall that algorithm VSAR had a bound on the number of transactions of $\tau = \mathcal{O}((1/\gamma^2)\ln(1/\gamma))$ (for this discussion we are assuming that δ and $|C|$ are fixed and we are ignoring $\ln\ln(1/\gamma)$ factors) in the worst case and of $\tau_0 = \mathcal{O}((1/\gamma\gamma_0)\ln(1/\gamma))$ in case that either $\mathrm{fr}(X, \mathcal{T}) = p \geq \sigma$ or $\mathrm{fr}(X, \mathcal{T}) = p < \sigma'$ where $\gamma_0 = |p - \sigma'|$. Let us concentrate on this latter case. While we are in the loop of algorithm VSAR, if at any time step t an itemset X falsifies the condition $|\mathrm{fr}(X, S_t) - \sigma'| \leq (\gamma/2)\sqrt{\tau/t}$ [2] then the algorithm correctly classifies such an itemset with high probability as shown in Theorems 3.1 and 3.2. In fact, the $1/\gamma$ factor in our time bound comes from the choice of such a condition. So suppose that instead of $\gamma/2$ we could use $\gamma_0/2$ in the condition. In that case, we should be able to obtain a bound of $\mathcal{O}((1/\gamma_0^2)\ln(1/\gamma))$ instead of previous one, a significant improvement. Obviously, this cannot be done in a straightforward way since the value of γ_0 is not known in advance. One way to get around this difficulty is the following. We use an estimate of γ_0 that it is calculated adaptively as we collect transactions in an on-line fashion. As soon as we realize that this estimate is close enough to $\gamma_0/2$ for a certain itemset, the algorithm classifies that itemset. The algorithm in Figure 3, that we call ASAR from Adaptive Sampling for Association Rules, implements this idea.

Notice that by the choice of τ and the choice of ϵ_t in algorithm ASAR, when $t \geq \tau$ (and thus, the condition of the while-loop is falsified) then $\epsilon_t \leq \gamma/2$ and we are in the same situation as algorithm VSAR. So for the worst case situation, both algorithms are identical. However, in the situations where γ_0 is bigger than γ, we will see that as soon as ϵ_t becomes smaller than $\gamma_0/2$ we will correctly

[2] In algorithm VSAR, this condition is tested in two separate if-then statements.

```
Algorithm  ASAR $(X, \sigma, \sigma', \delta)$
1   $\gamma \leftarrow \sigma - \sigma'$;  $\tau \leftarrow \tau(\delta, \gamma)$;
2   $t \leftarrow 0$;  $S_t \leftarrow \emptyset$;
3   $\epsilon_t \leftarrow 1$;
4   while $(C \neq \emptyset)$ and $(t \leq \tau)$ do
5      obtain $r$ from $\mathcal{T}$ randomly;
6      $S_{t+1} \leftarrow S_t \cup \{r\}$;  $t \leftarrow t + 1$;
7      $\epsilon_t \leftarrow \sqrt{\ln(|C|(\tau+1)/\delta)/2t}$;
8      for all $X \in C$ do
9         if $\mathrm{fr}(X, S_t) > \sigma' + \epsilon_t$ then
10           declare $X$ frequent and remove it from $C$;
11        if $\mathrm{fr}(X, S_t) < \sigma' - \epsilon_t$ then
12           declare $X$ non-frequent and remove it from $C$;
13     end
14  end
15  declare all remaining $X \in C$ non-frequent;
```

Figure 3: Pseudo-code of algorithm ASAR.

classify the itemset with high probability using much fewer transactions. This is what it is proved in the following two theorems.

We start showing the performance of the algorithm for the frequent itemsets in C. In this case, as shown in the following theorem, the algorithm declares those itemsets frequents with high probability. Moreover, the number of transactions that the algorithm uses to classify those itemsets is proportional to the square of the inverse of the distance of their real frequency to the lowered support threshold (that is, γ_0). The dependency on the inverse of γ is now only logarithmic.

Theorem 4.1. Let $\sigma > 0$ be a minimum support threshold, σ' a lowered support threshold where $\gamma = \sigma - \sigma'$ and C a collection of itemsets. Let $X \in C$ be a frequent itemset such that $\mathrm{fr}(X, \mathcal{T}) = p \geq \sigma$ and let $\gamma_0 = p - \sigma'$. Then, algorithm ASAR on input C, σ, σ' and $\delta < 1$ correctly declares any frequent itemset $X \in C$ as frequent and it uses $\tau_0 = \tau \cdot (\gamma/\gamma_0)^2$ transactions to classify X with probability at least $1 - \delta$.

Proof. The proof of this theorem follows the same lines of the proof of Theorem 3.1. We first show the reliability of the algorithm for a single itemset. In order to do this, we need to bound the probability that the algorithm declares a frequent $X \in C$ as non-frequent at some step t. That is, we need to bound the sum of the following probabilities.

$$\begin{aligned} P_1(X) &= \Pr\{\, \forall t \leq \tau \, [\, \mathrm{fr}(X, S_t) < \sigma' + \epsilon_t \,] \,\}, \quad \text{and} \\ P_2(X) &= \Pr\{\, \exists t \leq \tau \, [\, \mathrm{fr}(X, S_t) < \sigma' - \epsilon_t \,] \,\}. \end{aligned}$$

We start bounding probability $P_1(X)$. Clearly, $P_1(X) \leq P_1'(X) = \Pr\{\, \mathrm{fr}(X, S_\tau) < \sigma' + \epsilon_\tau \,\}$ which we bound as follows:

$$\begin{aligned} P_1'(X) &= \Pr\{\, \mathrm{fr}(X, S_\tau) + \mathrm{fr}(X, \mathcal{T}) \leq \sigma' + \epsilon_\tau + \mathrm{fr}(X, \mathcal{T}) \,\} \\ &\leq \Pr\{\, \sigma - \sigma' - \epsilon_\tau \leq \mathrm{fr}(X, \mathcal{T}) - \mathrm{fr}(X, S_\tau) \,\} \\ &= \Pr\{\, \gamma - \epsilon_\tau \leq \mathrm{fr}(X, \mathcal{T}) - \mathrm{fr}(X, S_\tau) \,\} \\ &\leq \Pr\{\, \epsilon_\tau \leq \mathrm{fr}(X, \mathcal{T}) - \mathrm{fr}(X, S_\tau) \,\} \\ &\leq \exp(-2\epsilon_\tau^2 \tau) \leq \frac{\delta}{|C|(\tau + 1)} \end{aligned}$$

where the inequalities in the last line hold by the Hoeffding bound and our choice of τ and ϵ_τ.

Now, we bound $P_2(X)$ as follows:

$$\begin{aligned} P_2(X) &= \Pr\{\, \exists t \leq \tau \, [\, \mathrm{fr}(X, S_t) < \sigma' - \epsilon_t] \,\} \\ &\leq \sum_{t \leq \tau} \Pr\{\, \mathrm{fr}(X, S_t) < \sigma' - \epsilon_t \,\} \\ &\leq \sum_{t \leq \tau} \Pr\{\, \epsilon_t < \mathrm{fr}(X, \mathcal{T}) - \mathrm{fr}(X, S_t) \,\} \\ &\leq \sum_{t \leq \tau} \exp(-2\epsilon_t^2 t) \leq \sum_{t \leq \tau} \frac{\delta}{|C|(\tau + 1)} \leq \tau \frac{\delta}{|C|(\tau + 1)} \end{aligned}$$

where again, the inequalities in the last line hold by applying the Hoeffding bound and by our choice of ϵ_t. Therefore, since $P_1(X) + P_2(X) \leq \delta / |C|$ the probability that any frequent itemset is misclassified by ASAR can be shown to be at most δ by using the union bound as in the proof of Theorem 3.1.

This proves the first part of the theorem that deals with the reliability of algorithm ASAR.

Next, we study the efficiency of ASAR, namely, the number of transactions that need to be sampled from the database. Notice that while we are in the loop, the value of ϵ_t is always strictly decreasing. Consider the first step t where the value of ϵ_t has became small enough so that $2\epsilon_t < p - \sigma'$, that is, the smallest t such that $t \geq \tau_0$. We will show that at that time step the probability that the algorithm does not remove a frequent itemset X from C is very small. That is, we will show that the following probability $P_3(X)$ is smaller than $\delta/|C|$.

$$\begin{aligned} P_3(X) &= \Pr\{\, 2\epsilon_t < p - \sigma' \text{ and } (|\mathrm{fr}(X, S_t) - \sigma'| \leq \epsilon_t \text{ and } t < \tau)\,\} \\ &\leq \Pr\{\, 2\epsilon_t < p - \sigma' \text{ and } \mathrm{fr}(X, S_t) \leq \sigma' + \epsilon_t \,\} \\ &\leq \Pr\{\, \epsilon_t < p - \mathrm{fr}(X, S_t)\,\} \\ &= \Pr\{\, \epsilon_t < \mathrm{fr}(X, S_t) - \mathrm{fr}(X, \mathcal{T})\,\} \\ &\leq \exp(-2\epsilon_t^2 t) \;\leq\; \frac{\delta}{|C|} \end{aligned}$$

From the above bound, we can conclude that for any non-frequent itemset $X \in C$ such that $\mathrm{fr}(X, \mathcal{T}) = p < \sigma'$, as soon as $t \geq \tau_0$ (and thus, $2\epsilon_t$ becomes smaller than $\gamma_0 = p - \sigma'$) algorithm ASAR removes X from C with probability at least $1 - \delta$ as claimed. □

Now we analyze the case where the itemset input to the algorithm is non frequent. The proof is symmetric to the proof of previous theorem and Theorem 3.2.

Theorem 4.2. Let $\sigma > 0$ be a minimum support threshold, σ' a lowered support threshold where $\gamma = \sigma - \sigma'$ and let C be a collection of itemsets. Then, algorithm ASAR on input C, σ, σ' and $\delta < 1$ declares any X non-frequent with $\mathrm{fr}(X, \mathcal{T}) = p \leq \sigma'$ as non-frequent and it uses at most $\min\{\, \tau, \tau_0, \}$ steps to remove X from C with probability at least $1 - \delta$ where $\gamma_0 = \sigma' - p$ and $\tau_0 = \tau \cdot (\gamma/\gamma_0)^2$.

Proof. For showing the reliability, we need to show that the probability that X is declared frequent by ASAR is smaller than δ. That is, we need to bound the following probability:

$$\Pr\{\; \exists t \leq \tau \,[\, \mathrm{fr}(X, S_t) > \sigma' + \epsilon_t \,]\}$$

Using the same analysis as for probability P_2 in Theorem 4.1 we can show that this probability is smaller than δ. Now we prove the complexity of the algorithm. Obviously, the algorithm finishes in at most τ steps and τ is smaller than $2\ln((\tau + 1)/\delta)/\gamma_0^2$ for any $\gamma_0 < \gamma$. So consider the other case where $\gamma_0 \geq \gamma$, that is $\mathrm{fr}(S, \mathcal{T}) = p < \sigma' - \gamma$. We claim that in this case, when $2\epsilon_t$ becomes smaller than $\sigma' - p$, the algorithm stops with probability at least $1 - \delta$. This claim is shown by the same analysis used in Theorem 4.1. By our choice of ϵ_t this happens for $t \geq \tau_0 = 2\ln((\tau+1)/\delta)/\gamma_0^2$ and the theorem follows. $\Box$

To summarize, we can see from the statements of Theorems 4.1 and 4.2 that algorithm ASAR uses $\mathcal{O}(\ln(1/\gamma)\max\{1/\gamma^2, 1/\gamma_0^2\})$ transactions (ignoring the dependency on δ and $|C|$) to classify an itemset.

5 Further Improvement of the Bounds: Algorithm ImpASAR

We can improve the previous algorithm with the following observation. One of the central ideas of algorithm ASAR is that at any time step t we are always calculating a value for ϵ_t such that our estimate is at most ϵ_t away of the real frequency of the itemset with very high probability. In order to guarantee this property we are using the Hoeffding bound stated in Theorem 2.1 and we derived the value of ϵ_t from there. The advantage of the Hoeffding bound is that it provides us an upper bound on the probability of having a bad estimate that depends only on the values of ϵ_t and t. Thus, it can be applied at any time without any other knowledge.

For our application, however, we have one more bit of knowledge: the real frequencies of most itemsets in the database are fairly small; it is very unlikely in practice that we have itemsets with, say 50% or more frequency. As we will see below, it may be advantageous to use the multiplicative version of the Chernoff bound stated in Theorem 5.1 instead of the Hoeffding bound, the one normally called the Chernoff bound. We first state this bound in the following theorem.

Theorem 5.1. (The Chernoff bound) [3] For any $t \geq 1$ and p, $0 \leq p \leq 1$, consider t independent random variables $X_1, \ldots, X_t$ each of which takes values 0 and 1 with probabilities $1-p$ and p. Then for any $\epsilon > 0$, we have

$$\Pr\{\sum_{i=1}^{t} X_i > pt+\epsilon t\} < \exp(-\epsilon^2 t/3p), \quad \Pr\{\sum_{i=1}^{t} X_i < pt-\epsilon t\} < \exp(-\epsilon^2 t/2p).$$

[3]This bound is usually stated in the literature using a multiplicative error. We have reformulated here for an additive error since it is what we need for our application.

The disadvantage of this bound is that it depends on the value of p and this is precisely what we want to estimate. It is easy to verify from the statements of Hoeffding and Chernoff bounds that the second is better for any $p < 1/6$. However, we can still use it in the following way. The main idea is again to use $\mathrm{fr}(X, S_t)$ as an estimate of p, and to keep a different value of $\epsilon_t(X)$ for every itemset X. As soon as $\mathrm{fr}(X, S_t) + \epsilon_t(X)$ becomes smaller than $1/6$, we start using Chernoff instead of Hoeffding to compute the values of $\epsilon_{t'}(X)$ for later steps t'. Since this decision itself may be wrong with a small probability, the value of τ must be slightly increased to keep the theoretical guarantees. In particular, we will calculate the value for ϵ_t using this formula

$$\epsilon_t(X) = \sqrt{\ln\left(\frac{2|C|(\tau+1)}{\delta}\right)\frac{3(\mathrm{fr}(X, S_{t-1}) + \epsilon_{t-1}(X))}{t}}$$

in case $\mathrm{fr}(X, S_{t-1}) + \epsilon_{t-1} \leq 1/6$ where the 2 factor that appears inside the logarithm comes from the fact that now we are using $\delta/2$ to cover the probability that we are not using the Chernoff bound (in case we use it) with an appropriate estimate and the other $\delta/2$ is to guarantee that the algorithm works well. The choice of τ also gets affected by this, it will also have an extra 2 factor inside the logarithm. The rest of the algorithm will be the same as algorithm ASAR shown in Figure 3 except in line 7 where an if-the-else statement will control which value of ϵ_t to use at every time. We call ImpASAR, for *Improved ASAR*, the algorithm that incorporates this improvement.

The worst case bound of the algorithm that implements this improvement is, asymptotically, the same as algorithm ASAR. However, for the range of values of the problem of mining association rules, the support is usually assumed to be small, something below 5%. Thus, the cases where our algorithms consume a big number of transactions are the itemsets with frequencies around the minimun support threshold and the lowered support threshold. Those values are usually quite far from 1/6. Thus, we expect this improvement to be a significant one. In fact, the experiments that we have performed in the following section confirm this intuition.

6 Experimental Evaluation

This section is devoted to the experimental evaluation of the algorithms presented in the previous sections. The worst case number of transactions used by the algorithms are summarized in Table 1 where we have ignored the dependency on δ and $|C|$. We have denoted by ImpASAR the algorithm that implements the improvement described in the previous section. Notice that for the case $\sigma' \leq \mathrm{fr}(X, \mathcal{T}) < \sigma$ we do not have any theoretical guarantee that the algorithm will correctly classify the input itemsets while for the other cases we proved that the algorithms correctly classify all those itemsets with probability at least $1 - \delta$ for any $\delta < 1$.

$\mathrm{fr}(X, \mathcal{T}) = p$	BSAR	VSAR	ASAR/ImpASAR
$\sigma \leq p$	$\mathcal{O}(1/\gamma^2)$	$\mathcal{O}((1/\gamma\gamma_0)\ln(1/\gamma))$	$\mathcal{O}((1/\gamma_0^2)\ln(1/\gamma))$
$\sigma' \leq p < \sigma$	$\mathcal{O}(1/\gamma^2)$	$\mathcal{O}((1/\gamma^2)\ln(1/\gamma))$	$\mathcal{O}((1/\gamma^2)\ln(1/\gamma))$
$p < \sigma'$	$\mathcal{O}(1/\gamma^2)$	$\mathcal{O}((1/\gamma\gamma_0)\ln(1/\gamma))$	$\mathcal{O}((1/\gamma_0^2)\ln(1/\gamma))$

Table 1: Summary of the number of transactions used by the algorithms where $\gamma = \sigma - \sigma'$ and $\gamma_0 = |p - \sigma'|$.

The purpose of this section is two folded. On the one hand, we want to determine for which combinations of γ and γ_0 our algorithms outperform the commonly used algorithm BSAR. We have just compared BSAR with ASAR and ImpASAR since, from our bounds, we can already see that algorithm ASAR will always perform better than algorithm VSAR. On the other hand, we want to determine experimentally how tight our theoretical analysis is.

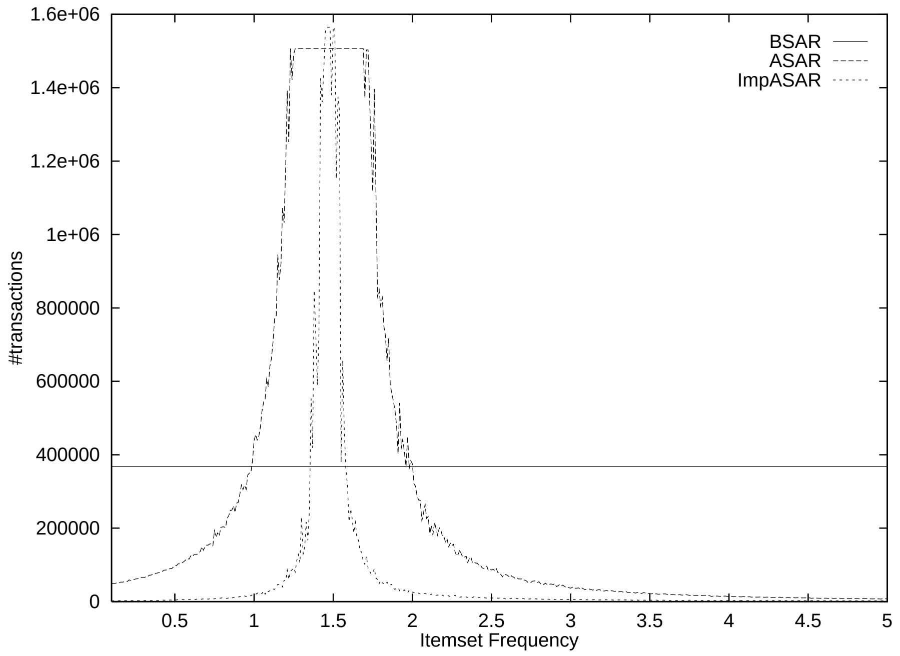
6.1 Description of the Experiments

In order to investigate the behavior of the algorithms in a wide range of values, so far we have used synthetic data. The experiment setup is similar to the one used in [4]. The common set of parameters have been fixed to the following values. For the remaining of the discussion and for all the experiments performed, we have fixed δ to 0.01. That it is, we expect a confidence of 99%. In order to simulate the frequency of an itemset we have generated a *success pattern*. A success pattern is a 0/1 string of 10000 bits that are used to determine whether an itemset appears in a transaction or not. That is, to simulate the behavior of an itemset we just draw a random number i between 1 and 10000, and decide whether the itemset is in the transaction or not if the ith bit of the success pattern is 1/0. Finally, for every fixed setting of all the parameters, we run these experiments 30 times, i.e., run each algorithm 30 times, and average the results.

Since the number of transactions used by the algorithms does not depend on σ itself, but only on the distance of σ to the lowered threshold and to the real frequency, we have fixed the minimum support to $\sigma = 2\%$ and we have performed several experiments with different values of σ'. For every combination of σ and σ' we have run all the algorithms with a single itemset as input where the frequency of this itemset ranges from 0.1% up to 10% and with a 0.01 increment.

6.2 Comparison of the Algorithms

Figure 4 shows the number of transactions used by the three algorithms for $\sigma' = 2\%$ and $\sigma' = 1.5\%$ and thus, $\gamma = 0.5\%$, and for different frequencies of the input itemset. For this combination of values, algorithm BSAR uses always 368413 transactions no matter what the frequency of the input itemset is; this is represented by the horizontal line in the graph. Algorithms ASAR and ImpASAR use 1506435 and 1564934 transactions respectively in the worst case, which is around σ'. This number is much bigger than the bound for algorithm BSAR. However, as soon as the frequency of the itemset is a bit far from σ', our algorithms outperform BSAR by several orders of magnitude. In particular, algorithm ASAR performs better than BSAR for frequencies smaller than 1% or bigger than 2% while algorithm ImpASAR outperforms BSAR and ASAR for frequencies smaller than 1.35% and bigger than 1.58%. In the bottom graph of Figure 4 we have magnified the range of frequencies going from 2% up to 3%, that is, when the input itemset is frequent. For those values we can see that algorithm ASAR is already better than algorithm BSAR. For instance, for a frequency of 2.5% ASAR uses 86226 transactions an improvement on the 368413 bound of BSAR. The results of algorithm ImpASAR are much more impressive. For a frequency of 2% it only uses 25723 transactions while for a frequency of 2.5% requires just 9637 transactions to classify the itemset. We can also see in that graph that ImpASAR, although performing marginally worse than ASAR for some values very close to σ', it greatly outperforms ASAR everywhere else. The figures only show the performance of the algorithms up to a frequency of 5% since after that the number of examples used by algorithms ASAR and ImpASAR is extremely small. So our goal of showing the superiority of our algorithms against



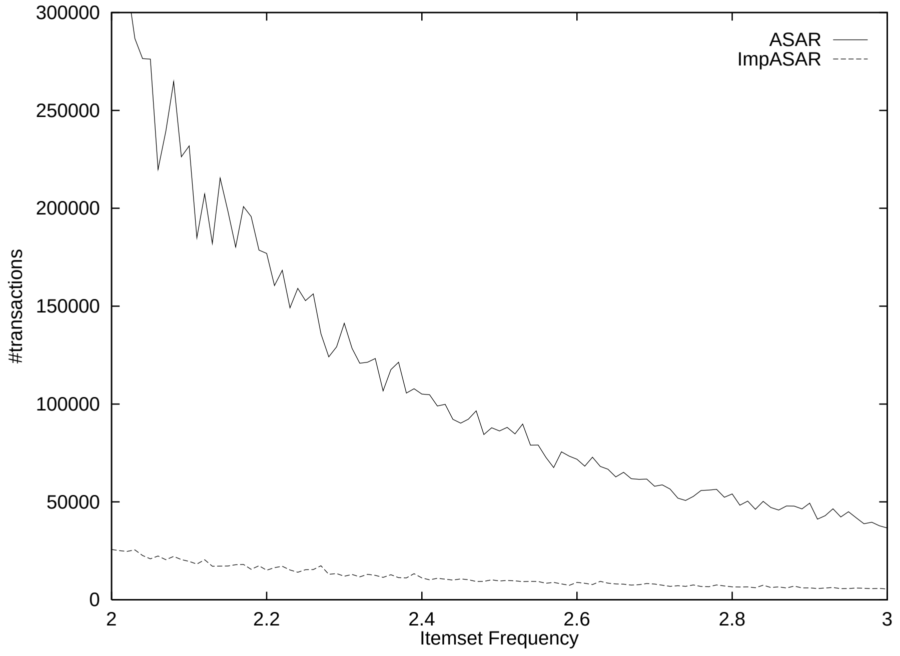


Figure 4: Number of transactions used by BSAR, ASAR, and ImpASAR for $\sigma = 2\%$ and $\sigma' = 1.5\%$.

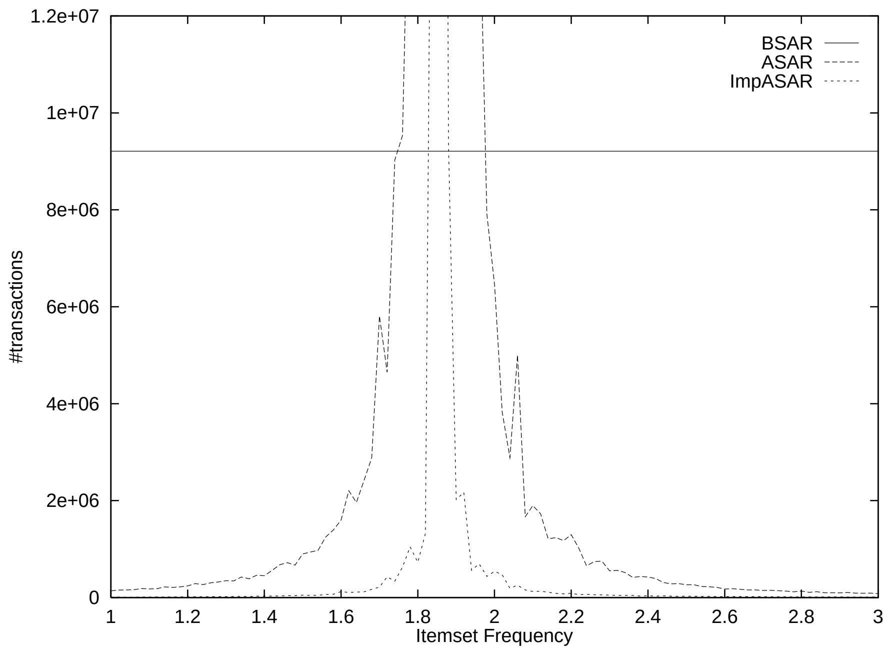
algorithm BSAR is already achieved in that range.

Figure 5 shows the results for the three algorithms using $\sigma' = 1.9\%$. For this value, the number of transactions required by algorithm BSAR is huge, more than 9 millions of transactions. For most databases, this number rules out any advantage that we might gain by using sampling. On the other hand, we can see in Figure 5 that algorithm ImpASAR for any itemset with frequency more than 2.07% uses less than 200000 transactions, a number that is very reasonable.

6.3 The Tightness of the Theoretical Analysis

With respect to the number of transactions, the bounds on the number of examples provided by the theory are overestimating compared to the experimental results. This is not that surprising since our theorems provide worst case bounds while the experiments are reflecting an average case situation. One of source of inefficiency in our theoretical bounds might come from the following fact. Recall that in the proof of Theorem 4.1 and Theorem 4.2 concerning the number of transactions used, we showed that, with high probability, when ϵ_t becomes smaller than $\gamma_0/2$ the itemset is removed from C and classified. Figure 6 shows the value of ϵ_t for the last iteration that we have obtained experimentally when running the algorithm with a single itemset as input for $\sigma' = 1.5\%$ and $\sigma' = 1.9\%$. It can be seen that as soon as ϵ_t takes a value very close to γ_0 the algorithm stops. It never reaches $\gamma_0/2$ as assumed by the theory. This is one of the reasons why our experiments reflect a better bound than the one predicted by the theory.

Now we discuss the tightness of our analysis of the confidence of the algorithms. Recall that we do not have any theoretical guarantee of success for the case where the frequency of the itemset is between σ' and σ. Thus, for the experiments shown in Figure 4 we might expect that some of the itemsets with frequency between 1.5% and 2% are wrongly classified as frequent. The situation that we have found experimentally is the following (similar results have been obtained for other values of σ and σ'). Algorithm BSAR only misclassifies some itemsets that have a frequency between 1.65% and 2% and algorithm ASAR misclassifies itemsets between 1.7% and 2%. On the other hand, algorithm ImpASAR misclassifies itemsets that are between 1.51% and 2%, much closer to our theoretical analysis. We consider that the algorithm misclassifies an itemset of a certain frequency if at least one of the 30 rounds misclassifies the itemset. For the other case shown in Figure 5 we have found that algorithms BSAR and



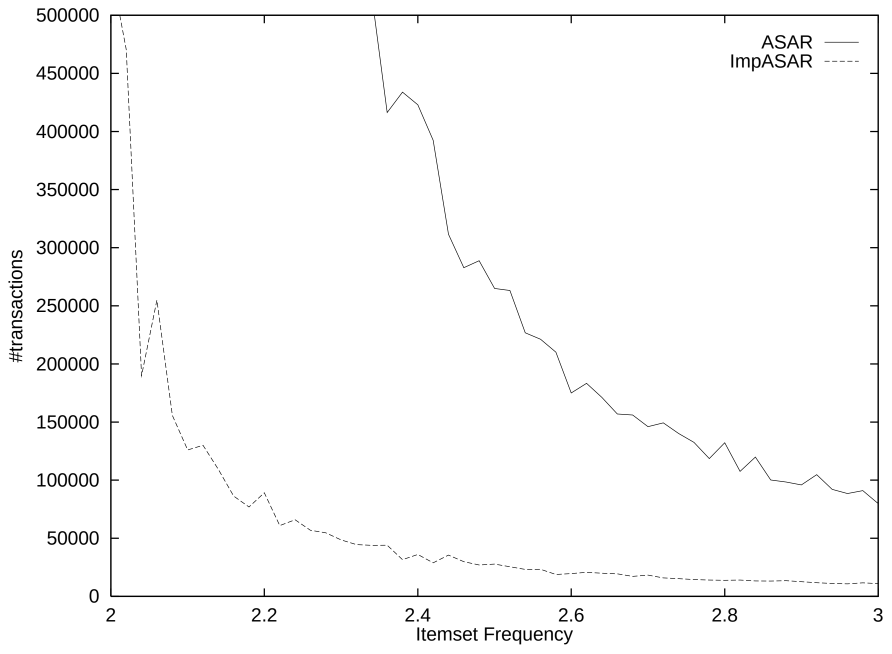
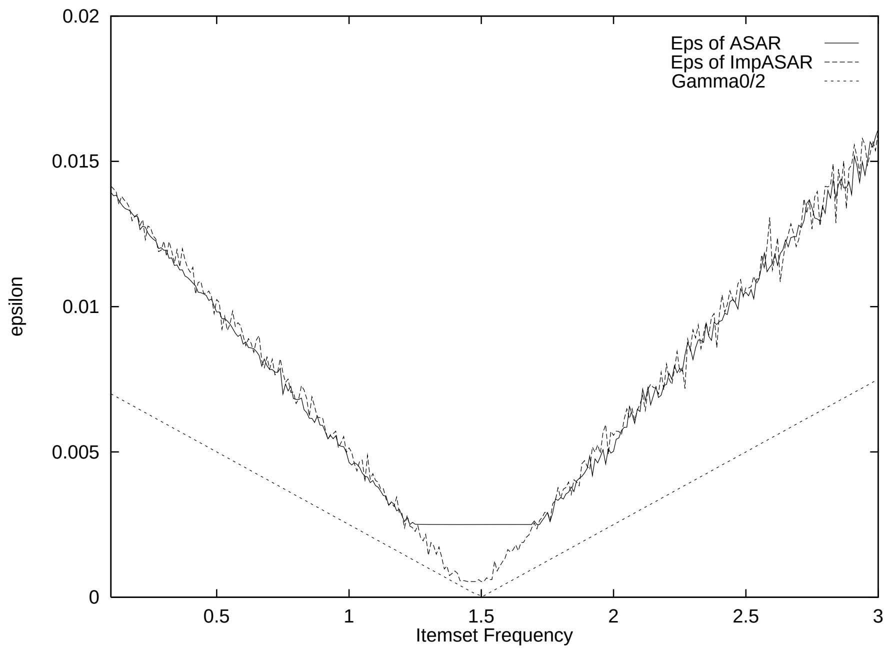


Figure 5: Number of transactions used by BSAR, ASAR, and ImpASAR for $\sigma = 2\%$ and $\sigma' = 1.9\%$.



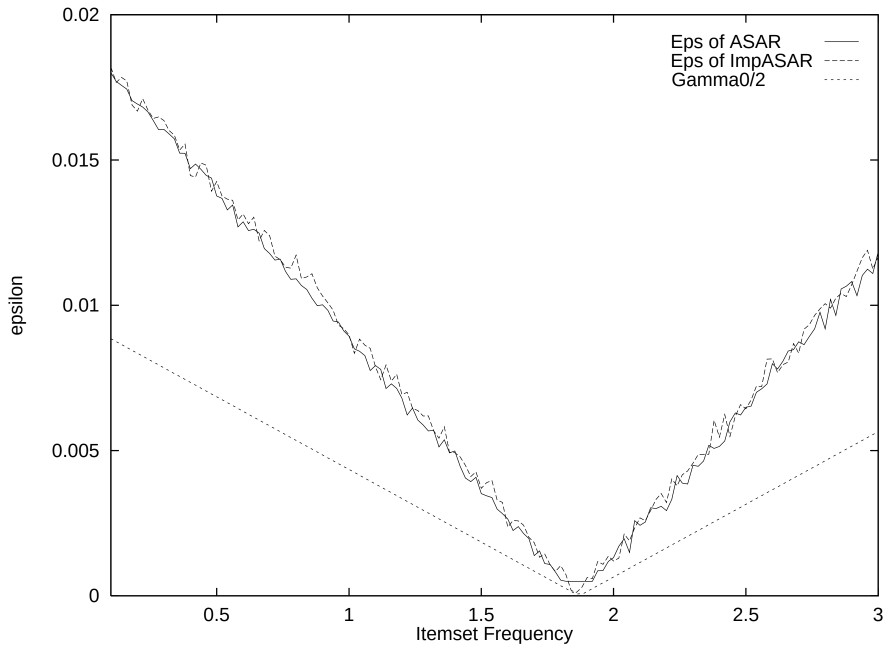


Figure 6: Value of ϵ_t when the algorithms stop for $\sigma' = 1.5\%$ and $\sigma' = 1.9\%$.

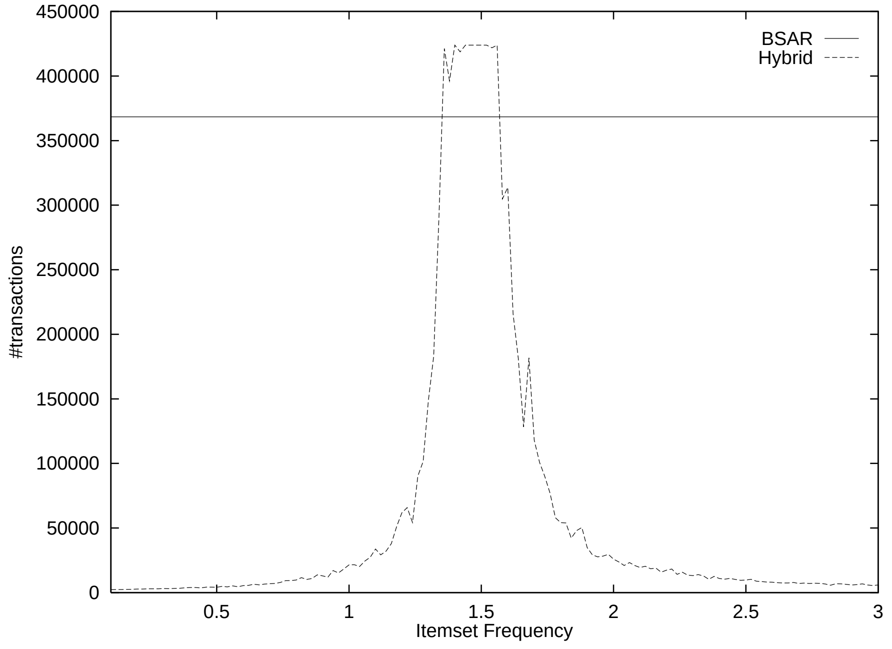
ASAR only misclassify itemsets with frequencies between 1.94% and 2% while algorithm ImpASAR misclassifies itemsets with frequencies between 1.91% and 2%.

These results might be indicating that the analysis of the confidence of algorithms BSAR and ASAR is not very tight while for algorithm ImpASAR is much tighter. So we do not expect to find a much better analysis for that algorithm since the theoretical worst case bound is pretty much the same as the one found experimentally. Thus, these experiments also indicate that algorithm ImpASAR is not only better but it also reflects better the theory. In other words, if we allow a sampling algorithm to make mistakes in a certain range of frequencies, algorithm ImpASAR will make a more economic use of the transactions in order to guarantee that it only makes mistakes on that range and not anywhere else. On the other hand, algorithms BSAR and ASAR are overestimating the number of transactions needed to achieve the same goal.

6.4 A hybrid algorithm

We have just seen that our algorithms, in particular algorithm ImpASAR, greatly outperform algorithm BSAR for most of the frequencies of the input itemsets. However, when the frequency of the itemset is around σ', algorithm ImpASAR becomes the worst requiring much more transactions than algorithm BSAR. An obvious improvement is the following [4]. For a given X, σ, σ' and δ, let B the number of examples required by algorithm BSAR with confidence $\delta/2$. We run algorithm ImpASAR with inputs X, σ, σ', and $\delta/2$ until either it finishes and classifies itemset X or it uses B transactions. In the later case we use those B transactions to feed algorithm BSAR and classify X. Clearly, the probability that this algorithm fails is at most δ. We call this algorithm Hybrid. An example of an execution of Hybrid for $\sigma = 2\%$, $\delta = 0.01$ and $\sigma' = 1.5\%$ and 1.9% for a range of frequencies is shown in Figure 7. Algorithm Hybrid is only marginally worse than BARS in values very close to σ' and immensely better everywhere else. It is therefore the most appropriate algorithm to be implemented as part of an automatic tool.

[4] For simplicity, we describe it for the case where we have as input a single itemset. To extend the argument to a set of itemsets is routine.



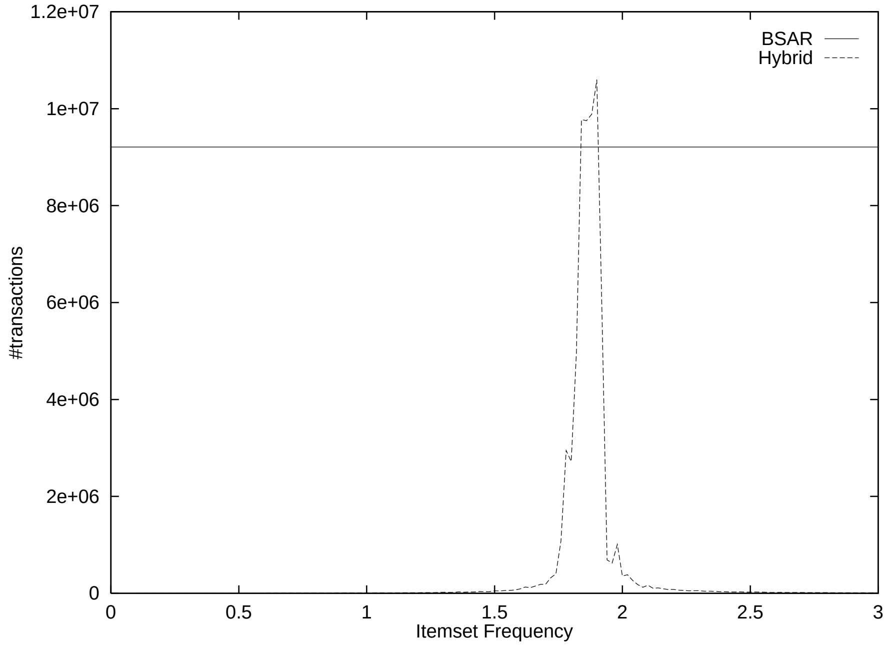


Figure 7: Number of transactions used by algorithm Hybrid for $\sigma' = 1.5\%$ and $\sigma' = 1.9\%$.

7 Conclusions

In this paper we have presented some algorithms for the problem of efficiently sampling for association rules. Our algorithms are designed for deciding whether an itemset is frequent or not and have the advantage that, for frequent itemsets, the sample size is proportional to the inverse of the real frequency not to the minimum support threshold of the frequency. The main feature is that our algorithms do not need any previous knowledge on the real frequency of the itemset. We have argued that an immense improvement in the number of necessary transactions can be achieved by using our algorithms instead of the usual batch approach described in the literature and we have provided experimental evidence to support this claim. The algorithms are not only efficient, they are also very simple and thus it is very easy to incorporate them into existent software just by substituting the frequency calculation step by them.

It remains to perform some more realistic experiments to see whether our algorithms have a real advantage in practical situations. This involves two different problems. One is studying how to best combine sampling algorithms with existing algorithms such as Apriori; Toivonen [15] reports some work in this direction using the batch sampling approach. The other one, particular to our algorithms, is the technological problem of picking random transactions on-line with an acceptable cost. In view of our experimental results, we believe that, even if the additional cost for on-line sampling is non-negligible, we can still obtain a significant reduction on the overall running time by using our algorithms.

References

[1] R. Agrawal, H. Mannila, R. Srikantand H. Toivonen, and I. Verkamo. *Fast discovery of association rules.* AAAI Press, 1996.

[2] R. Agrawal and R. Srikant. Fast algorithms for mining association rules in large databases. In *Proc. of the 20th International Conference on Very Large Data Bases, VLDB'94*, pages 487–499, 1994.

[3] Rakesh Agrawal, Tomasz Imielinski, and Arun N. Swami. Mining association rules between sets of items in large databases. In *Proc. of the 1993 ACM SIGMOD International Conference on Management of Data*, pages 207–216, 1993.

[4] Carlos Domingo, Ricard Gavaldà, and Osamu Watanabe. Practical algorithms for on-line sampling. In *Proc. of the First International Conference on Discovery Science*, Lecture Notes in Computer Science, December 1998. To appear.

[5] D. Gunopulos, Mannila, and S. Saluja. Discovering all most specific sentences by randomized algorithms. In *Proc. of the International Conference on Database Theory*, Lecture Notes in Computer Science 1186, pages 215–229, 1997.

[6] Dimitrios Gunopulos, Roni Khardon, Heikki Mannila, and Hannu Toivonen. Data mining, hypergraph transversals, and machine learning. In *Proceedings of the Sixteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, pages 209–216, 1997.

[7] Michael J. Kearns and Umesh V. Vazirani: *An Introduction to Computational Learning Theory.* The MIT Press, 1994.

[8] H. Mannila and H. 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, 1996.

[9] Heikki Mannila and Hannu Toivonen. Levelwise search and borders of theories in knowledge discovery. *Data Mining and Knowledge Discovery*, 1(3):241–258, 1997.

[10] Heikki Mannila, Hannu Toivonen, and A. Inkeri Verkamo. Efficient algorithms for discovering association rules. In *Proc. of the AAAI Workshop on Knowledge Discovery in Databases (KDD-94)*, pages 181–192, Seattle, Washington, July 1994. AAAI Press.

[11] Rajeev Motwani and Prabhakar Raghavan. *Randomized Algorithms.* Cambridge University Press, 1995.

[12] Jong Soo Park, Ming-Syan Chen, and P. S. Yu. An effective hash-based algorithm for mining association rules. *SIGMOD Record (ACM Special Interest Group on Management of Data)*, 24(2):175–186, June 1995.

[13] Ashok Savasere, Edward Omiecinski, and Shamkant Navathe. An efficient algorithm for mining association rules in large databases. In *Proc. of the 21st International Conference on Very Large Databases*, pages 432–444, 1995.

[14] R. Srikant and R. Agrawal. Mining generalized association rules. In *Proc. of the 21st International Conference on Very Large Databases*, 1995.

[15] Hannu Toivonen. Sampling large databases for association rules. In *Proceedings of the 22nd VLDB Conference*, pages 134–145, 1996.

[16] Mohammed Javeed Zaki, Srinivasan Parthasarathy, Wei Li, and Mitsunori Ogihara. Evaluation of sampling for data mining of association rules. In *7th International Workshop on Research Issues in Data Engineering*, 1997.