Evaluation of Sampling for Data Mining of Association Rules *

Mohammed Javeed Zaki, Srinivasan Parthasarathy, Wei Li, Mitsunori Ogihara
Computer Science Department, University of Rochester, Rochester NY 14627
{zaki,srini,wei,ogihara}@cs.rochester.edu

Abstract

Discovery of association rules is a prototypical problem in data mining. The current algorithms proposed for data mining of association rules make repeated passes over the database to determine the commonly occurring itemsets *(or set of items). For large databases, the I/O overhead in scanning the database can be extremely high. In this paper we show that random sampling of transactions in the database is an effective method for finding association rules. Sampling can speed up the mining process by more than an order of magnitude by reducing I/O costs and drastically shrinking the number of transactions to be considered. We may also be able to make the sampled database resident in main-memory. Furthermore, we show that sampling can accurately represent the data patterns in the database with high confidence. We experimentally evaluate the effectiveness of sampling on different databases, and study the relationship between the performance, and the accuracy and confidence of the chosen sample.*

1. Introduction

With large volumes of routine business data having been collected, business organizations are increasingly turning to the extraction of useful information from such databases. Such high-level inference process may provide information on customer buying patterns, shelving criterion in supermarkets, stock trends, etc. Data mining is an emerging research area, whose goal is to extract significant patterns or interesting rules from such large databases. It combines research in machine learning, statistics and databases. In this paper we will concentrate on the discovery of association rules.

The problem of mining association rules over *basket* data was introduced in [1]. Basket data usually consists of a record per customer with a transaction date, along with items bought by the customer. The main computation step consists of finding the frequently occurring item sets via an iterative process. In the k-th scan of the database all frequent items sets of length k are obtained. For disk resident databases, the I/O overhead in scanning the database during each iteration can be extremely high for large databases.

Random sampling from databases has been successfully used in query size estimation. Such information can be used for statistical analyses of databases, where approximate answers would suffice. It may also be used to estimate selectivities or intermediate result sizes for query optimization [11]. In the context of association rules, sampling can be utilized to gather quick preliminary rules. This may help the user to direct the data mining process by refining the criterion for "interesting" rules.

In this paper we show that random sampling of transactions in the database is an effective way for finding association rules. We empirically compare theory and experimentation, present results on the percentage of errors and correct rules derived at different sampling values, the performance gains, and also the relationship between performance, accuracy and confidence of the sample size. More specifically, we make the following contributions:

- Sampling can reduce I/O costs by drastically shrinking the number of transaction to be considered. We show that sampling can speed up the mining process by more than an order of magnitude.

- Sampling can provide great accuracy with respect to the association rules. We show that the theoretical results (using Chernoff bounds) are extremely conservative, and that experimentally we can obtain much better *accuracy* for a given *confidence*, or we can do with a smaller sample size for a given *accuracy*.

We begin by formally presenting the problem of finding association rules in section 2. Section 3 presents an analysis of random sampling from databases. The effectiveness of sampling is experimentally analyzed in section 4, and section 6 presents our conclusions.

*This work was supported in part by an NSF Research Initiation Award (CCR-9409120) and ARPA contract F19628-94-C-0057.

2. Data mining for association rules

We now present the formal statement of the problem of mining association rules over basket data. The discussion below closely follows that in [1, 3].

Let $\mathcal{I} = \{i_1, i_2, \cdots, i_m\}$ be a set of m distinct attributes, also called *items*. A set of items is called an *itemset*, and an itemset with k items is called a k-$itemset$. Each transaction T in the database $\mathcal{D}$ of transactions, has a unique identifier TID, and *contains* a set of items, such that $T \subseteq \mathcal{I}$. An *association rule* is an expression $A \Rightarrow B$, where itemsets $A, B \subset \mathcal{I}$, and $A \cap B = \emptyset$. Each itemset is said to have a *support* s if $s\%$ of the transactions in $\mathcal{D}$ contain the itemset. The association rule is said to have *confidence* c if $c\%$ of the transactions that contain A also contain B, i.e., $c = support(A \cup B)/support(A)$, i.e., the conditional probability that transactions contain the itemset B, given that they contain itemset A.

The data mining task for association rules can be broken into two steps. The first step consists of finding all *large* itemsets, i.e., itemsets that occur in the database with a certain user-specified frequency, called *minimum support*. The second step consists of forming implication rules among the large itemsets [3]. In this paper we only deal with the computationally intensive first step.

Many algorithms for finding large itemsets have been proposed in the literature [1, 7, 3, 10, 12, 6, 13, 2]. In this paper we will use the *Apriori* algorithm [2] to evaluate the effectiveness of sampling for data mining. We chose *Apriori* since it fast and has excellent scale-up properties. We would like to observe that our results are about sampling, and as such independent of the mining algorithm used.

2.1. The *Apriori* algorithm

The naive method of finding large itemsets would be to generate all the 2^m subsets of the universe of m items, count their support by scanning the database, and output those meeting minimum support criterion. It is not hard to see that the naive method exhibits complexity exponential in m, and is quite impractical. *Apriori* follows the basic iterative structure discussed earlier. However the key observation used is that any subset of a large itemset must also be large. In the initial pass over the database the support for all single items (1-itemsets) is counted. During each iteration of the algorithm only candidates found to be large in the previous iteration are used to generate a new candidate set to be counted during the current iteration. A pruning step eliminates any candidate which has a small subset. *Apriori* also uses specialized data structures to speed up the counting and pruning (hash trees and hash tables, respectively.) The algorithm terminates at step t, if there are no large t-itemsets. Let L_k denote the set of Large k-itemsets and $C_k = L_{k-1}\text{x}L_{k-1}$, the set of candidate k-itemsets. The general structure of the algorithm is given in figure 1. We refer the reader to [2] for more detail on *Apriori*, and its performance characteristics.

$L_1 = \{\text{large 1-itemsets }\}$;
for $(k = 2; L_{k-1} \neq \emptyset; k++)$
  C_k = Set of New Candidates;
  for all transactions $t \in \mathcal{D}$
    for all k-subsets s of t
      if $(s \in C_k)$ $s.count++$;
  $L_k = \{c \in C_k | c.count \geq \text{minimum support}\}$;
Set of all large itemsets = $\bigcup_k L_k$;

Figure 1. The *Apriori* algorithm

We now present a simple example of how *Apriori* works. Let the database, $\mathcal{D} = \{T_1 = (1,4,5), T_2 = (1,2), T_3 = (3,4,5), T_4 = (1,2,4,5)\}$. Let the minimum support value $MS = 2$. Running through the iterations, we get

$$\begin{aligned} L_1 &= \{\{1\},\{2\},\{4\},\{5\}\} \\ C_2 &= \{\{1,2\},\{1,4\},\{1,5\},\{2,4\},\{2,5\},\{4,5\}\} \\ L_2 &= \{\{1,2\},\{1,4\},\{1,5\},\{4,5\}\} \\ C_3 &= \{\{1,4,5\}\} \\ L_3 &= \{\{1,4,5\}\} \end{aligned}$$

Note that while forming C_3 by joining L_2 with itself, we get three potential candidates, {1,2,4},{1,2,5}, and {1,4,5}. However only {1,4,5} is a true candidate, and the first two are eliminated in the pruning step, since they have a 2-subset which is not large (the 2-subset {2,4}, and {2,5} respectively).

3. Random sampling for data mining

Random sampling is a method of selecting n units out of a total N, such that every one of the $\mathcal{C}_n^N$ distinct samples has an equal chance of being selected. In this paper we consider *sequential* random sampling *without replacement*, i.e., the records are selected in the same order as they appear in the database, and a drawn record is removed from further consideration.

3.1. Sampling algorithm

For generating samples of the database, we use the **Method A** algorithm presented in [15], which is simple and very efficient for large sample size, n. A simple algorithm for sampling generates an independent uniform random variate for each record to determine whether that record should be chosen for the sample. If m records have

been chosen from the first t records, then the next record will be chosen with the probability $(n-m)/(N-t)$. This algorithm, called **Method S** [9], generates $\mathcal{O}(N)$ random variates, and also runs in $\mathcal{O}(N)$ time. **Method A** significantly speeds up the sampling process by efficiently determining the number of records to be skipped over before the next one is chosen for the sample. While the running time is still $\mathcal{O}(N)$, only n random variates are generated (see [15] for more details).

3.2. Chernoff bounds

Let τ denote the support of an itemset I. We want to select n transactions out of the total N in the Database $\mathcal{D}$. Let the random variable $X_i = 1$ if the i-th transaction contains the itemset I ($X_i = 0$, otherwise). Clearly, $P(X_i = 1) = \tau$ for $i = 1, 2, \cdots n$. We further assume that all $X_1, X_2, \cdots, X_n$ are independent 0-1 random variables. The random variable $\mathbf{X}$ giving the number of transactions in the sample containing the itemset I, has a binomial distribution of n trials, with the probability of success τ (note: the correct distribution for finite populations is the *Hypergeometric distribution*, although the Binomial distribution is a satisfactory approximation [4]). Moreover, $\mathbf{X} = \sum_i^n X_i$, and the expected value of $\mathbf{X}$ is given as $\mu = E[\mathbf{X}] = E[\sum_{i=1}^n X_i] = \sum_{i=1}^n E[X_i] = n\tau$, since $E[X_i] = 0 \cdot P(X = 0) + 1 \cdot P(X = 1) = \tau$.

For any positive constant, $0 \leq \epsilon \leq 1$, the Chernoff bounds [5] state that

$$P(\mathbf{X} \leq (1-\epsilon)n\tau) \leq e^{-\epsilon^2 n\tau/2} \quad (1)$$

$$P(\mathbf{X} \geq (1+\epsilon)n\tau) \leq e^{-\epsilon^2 n\tau/3} \quad (2)$$

Chernoff bounds provide information on how close is the actual occurrence of an itemset in the sample, as compared to the expected count in the sample. This aspect, which we call as the *accuracy* of a sample, is given by $1-\epsilon$. The bounds also tell us the probability that a sample of size n will have a given accuracy. We call this aspect the *confidence* of the sample (defined as 1 minus the expression on the right hand size of the equations). Chernoff bounds give us two set of confidence values. Equation 1 gives us the lower bound – the probability that the itemset occurs less often than expected (by the amount $n\tau\epsilon$), while equation 2 gives us the upper bound – the probability that the itemset occurs more often than expected, for a desired accuracy. A low probability corresponds to high confidence, and a low ϵ corresponds to high accuracy. It is not hard to see that there is a trade-off between accuracy and confidence for a given sample size. This can been seen immediately, since $\epsilon = 0$ maximizes the right hand side of equations 1,2, while $\epsilon = 1$ minimizes it.

3.3. Sample size selection

Given that we are willing to accommodate a certain accuracy, $\mathcal{A} = 1-\epsilon$, and confidence $\mathcal{C} = 1-c$ of the sample, the Chernoff bounds can be used to obtain a sample size. We'll show this for equation 1, by plugging in $c = e^{-\epsilon^2 n\tau/2}$, to obtain

$$n = -2\ln(c)/(\tau\epsilon^2) \quad (3)$$

If we know the support for each itemset we could come up with a sample size n_I for each itemset I. We would still have the problem of selecting a single sample size from among the n_I. One simple heuristic is to use the user specified minimum support threshold for τ. The rationale is that by using this we guarantee that the sample size contains all the large itemsets contained in the original database. For example, let the total transactions in the original database $N = 3,000,000$. Let's say we desire a confidence $\mathcal{C} = 0.9 (c = 0.1)$, and an accuracy $\mathcal{A} = 0.99 (\epsilon = 0.01)$. Let the user specified support threshold be 1%. Using these values in equation 3, we obtain a sample size of $n = 4,605,170$. This is even greater than the original database! The problem is that the sample size expression is independent of the original database size. Moreover the user specified threshold is also independent of the actual itemset support in the original database. Hence, using this value may be too conservative, as shown above. In the next section we will compare experimental results obtained versus the theoretical predictions using Chernoff bounds.

4. Experimental evaluation

In this section we describe the experiments conducted in order to determine the effectiveness of sampling. We demonstrate that it is a reasonably accurate technique in terms of the associations generated by the sample, as compared to the associations generated by the original database. At the same time sampling can help reduce the execution time by more than an order of magnitude.

4.1. Experimental framework

All experiments were conducted on a 233MHz DEC Alphaserver 2100 processor, with 256MB of main memory. The databases are stored on an attached 2GB disk, and data is obtained from the disk via an NFS file server. We used four different databases to evaluate the effectiveness of sampling. These are:

• **SYNTH800, SYNTH250:** These are synthetic databases which mimic the transactions in a retailing environment. They have been used as benchmark databases for many association rules algorithms [3, 6, 12, 13, 2]. Each transaction has a unique ID followed by a list of items bought in

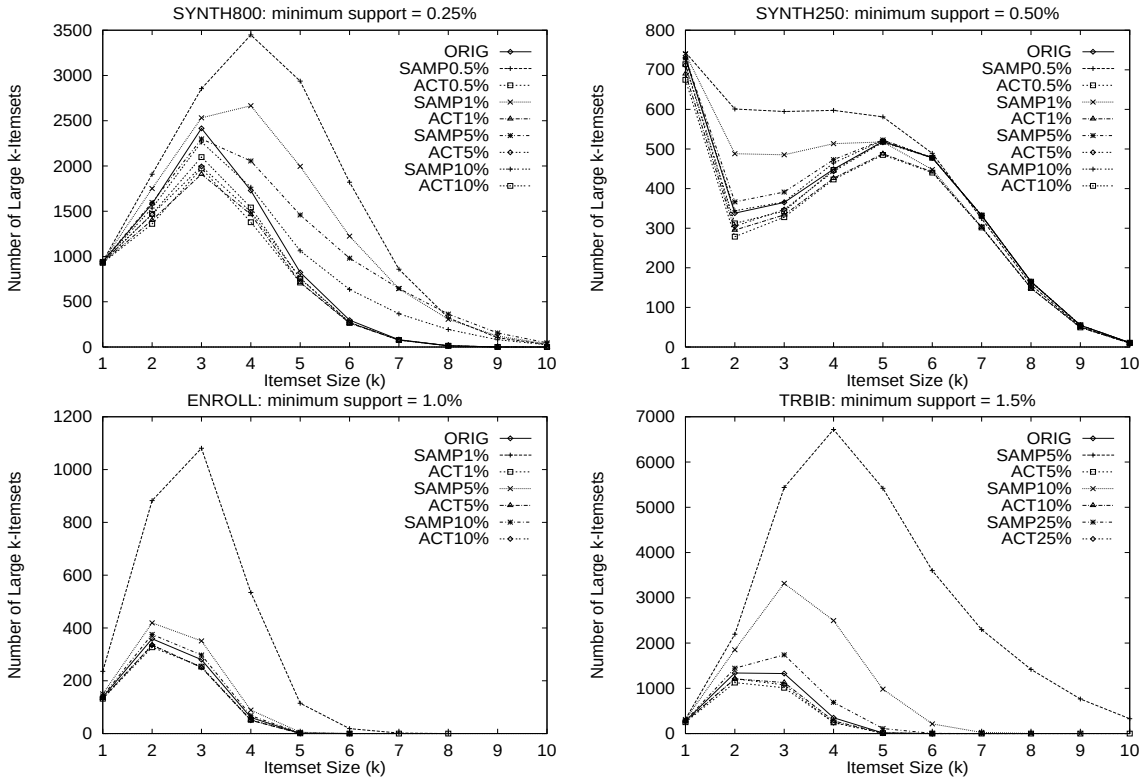

Figure 2. Itemset size vs. number of large itemsets

that transaction. We obtained the database $T10.I6.D800K$, by setting the number of transactions $|\mathcal{D}| = 800,000$, average transaction size $|T| = 10$, average maximal potentially large itemset size $|I| = 6$. For $T10.I4.D250K$, $|\mathcal{D}| = 250000$, $|T| = 10$, $|I| = 4$. For both databases the number of maximal potentially large itemsets $|L| = 2000$, and the number of items $N = 1000$. We refer the reader to [3] for more detail on the database generation.

• **ENROLL:** This is a database of student enrollments for a particular graduating class. Each transaction consists of a student ID followed by information on the college, major, department, semester, and a list of courses taken during that semester. There are 39624 transactions, 3581 items and the average transaction size is 9.

• **TRBIB:** This is a database of the locally available technical report bibliographies in computer science. Each item is a key-word which appears in a paper title, and each transaction has a unique author ID followed by a set of such keywords (items). There are 13793 transactions, 10363 items, and the average transaction size is 22.

4.2. Accuracy measurements

We report experimental results for the databases described above. Figure 2 shows the number of large itemsets found during the different iterations of the *Apriori* algorithm, for the different databases, and sample size. In the graphs, ORIG indicates the actual number of large itemsets generated when the algorithm operates on the entire database. SAMPx refers to the large itemsets generated when using a sample of size x% of the entire database. ACTx refers to the number of itemsets generated by SAMPx that are *true* large itemsets in the original database. The number of *false* large itemsets is given as (SAMPx − ACTx). From figure 2 we can observe that the general trends of sampled databases resemble actual results. Smaller sample sizes tend to over-estimate the number of large itemsets, i.e., they find more false large itemsets. On the other hand, larger sample sizes tend to give better results in terms of fidelity or the number of true large itemsets. This is indicated by the way ACT.x comes closer to ORIG as x (the sample percentage) is increased.

More detailed results are shown in figure 3, which shows the percentage of true and false itemsets generated for different values of sampling and minimum support. The values of minimum support were chosen so that there were enough large k-itemsets, for $k >= 2$. For example, for SYNTH800 and SYNTH250, only large 1-itemsets were found at support more than 1%. Therefore, only support values less than those were considered. Furthermore, support values were chosen so that we don't generate too many large itemsets.

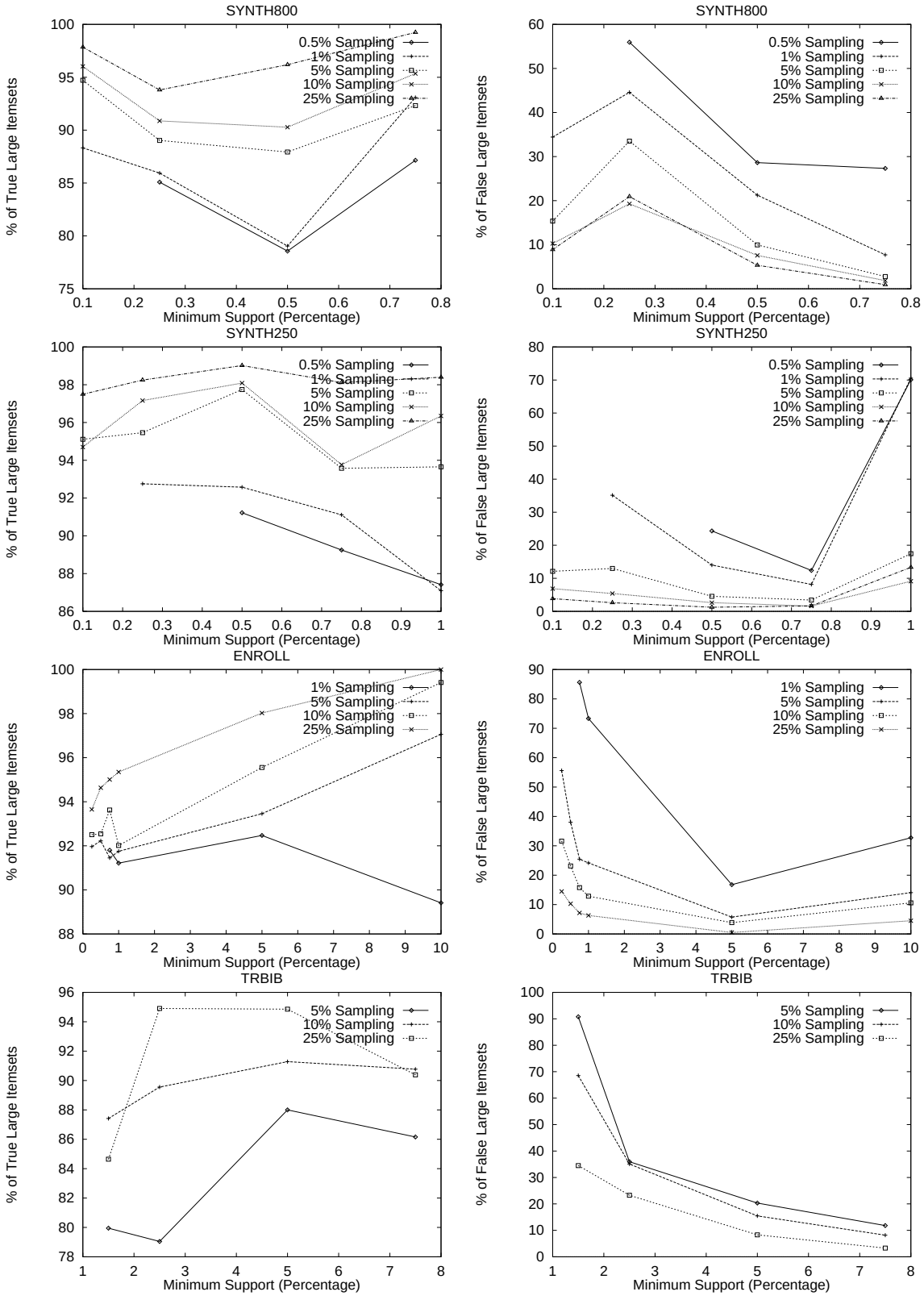

Figure 3. % of True and false large itemsets vs. % threshold for various sample sizes

For example, for ENROLL at 1% sampling size, we get a sample of 396 transactions. For support of 0.5%, we must find all itemsets which occur at least 2 times, in effect finding all possible large itemsets. Thus only support values greater than 0.5% were used.

The figure shows that at higher sampling size we generate a higher percentage of true large itemsets, and a smaller number of false large itemsets. It is interesting to note that in all cases we found more than 80% of all the large itemsets. We further observe that for other than very small sampling size, we can keep the false large itemsets under 20%.

4.3. Performance

Figure 4 shows the speedup obtained for the databases on different minimum support and different sampling size values. The speedup is relative to the algorithm execution time on the entire database. For SYNTH800 we obtain a speedup of more than 20 at small sample size and high support. For SYNTH250 we get more than 10 speedup in the same range. The performance at lower support is poor due to the large number of false large itemsets found. At higher sampling we get lower performance, since the reduction in database I/O is not that significant, and due to the introduction of more inaccuracies. For the smaller databases (ENROLL and TRBIB), at small sample size, we get no speedup, due to the large number of false large itemsets generated. We can observe that there is a trade-off between sampling size, minimum support and the performance. The performance gains are negated due to either a large number of false large itemsets at very low support or due to decreased gains in I/O vs. computation. We can conclude that in general sampling is a very effective technique in terms of performance, and we can expect it to work very well with large databases, as they have higher computation and I/O overhead.

4.4. Confidence: comparison with chernoff bounds

In this section we compare the Chernoff bound with experimentally observed results. We show that for the databases we have considered the Chernoff bound is very conservative.

Consider equations 1 and 2. For different values of accuracy, and for a given sampling size, for each itemset I, we can obtain the theoretical confidence value by simply evaluating the right hand side of the equations. For example, for the upper bound the confidence $\mathcal{C} = 1 - e^{-\epsilon^2 n\tau/3}$. Recall that confidence provides information about an item's actual support in the sample being away from the expected support by a certain amount ($n\tau\epsilon$). We can also obtain experimental confidence values as follows. We take s samples of size n, and for each item we compute the confidence by evaluating the left hand side of the two equations, as follows. Let $\imath$ denote the sample number, $1 \leq \imath \leq s$. Let

$$l_I(\imath) = \begin{cases} 1 & \text{if}(n\tau - \mathbf{X}) \geq n\tau\epsilon \ \text{ in sample } \imath \\ 0 & \text{otherwise} \end{cases}$$

$$h_I(\imath) = \begin{cases} 1 & \text{if}(\mathbf{X} - n\tau) \geq n\tau\epsilon \ \text{ in sample } \imath \\ 0 & \text{otherwise} \end{cases}$$

The confidence can then be calculated as $1 - \sum_{\imath=1}^{m} h_I(\imath)/s$, for the upper bound, and $1 - \sum_{\imath=1}^{m} l_I(\imath)/s$, for the lower bound.

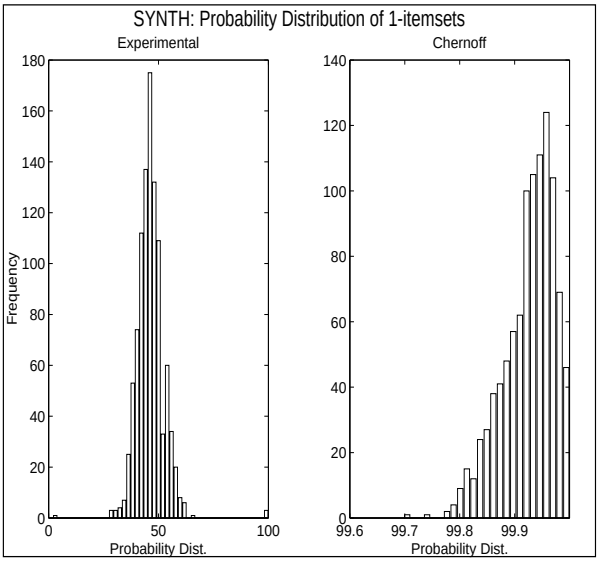

Figure 5. Probability distribution: experiment vs. chernoff

Figure 5 compares the distribution of experimental confidence to the one obtained by Chernoff upper bounds, for all m 1-itemsets or single items. It is possible (though impractical) to do this analysis for all the 2^m itemsets, however we present results for only single items. This should give us an indication whether the sample faithfully represents the original database. The results shown are for the SYNTH250 database with $\epsilon = 0.01$, $n = 2500$ (1% of total database size), and the number of samples taken, $s = 100$. We can see that the probability distribution across all items varies from 0.30 to 0.60 for the experimental case, with a mean probability close to 0.43. The Chernoff bounds produce a distribution clustered between 0.998 and 1.0, with an average probability of 0.9992. Chernoff bounds indicate that it is very likely that the sample doesn't have the given accuracy, i.e., with high probability, the items will be overestimated by a factor of 1.01. However, in reality, the probability of being over-estimated is only 0.43. The obvious difference in confidence depicts the limitation of Chernoff bounds in this setting. This was observed in all of the databases we looked at.

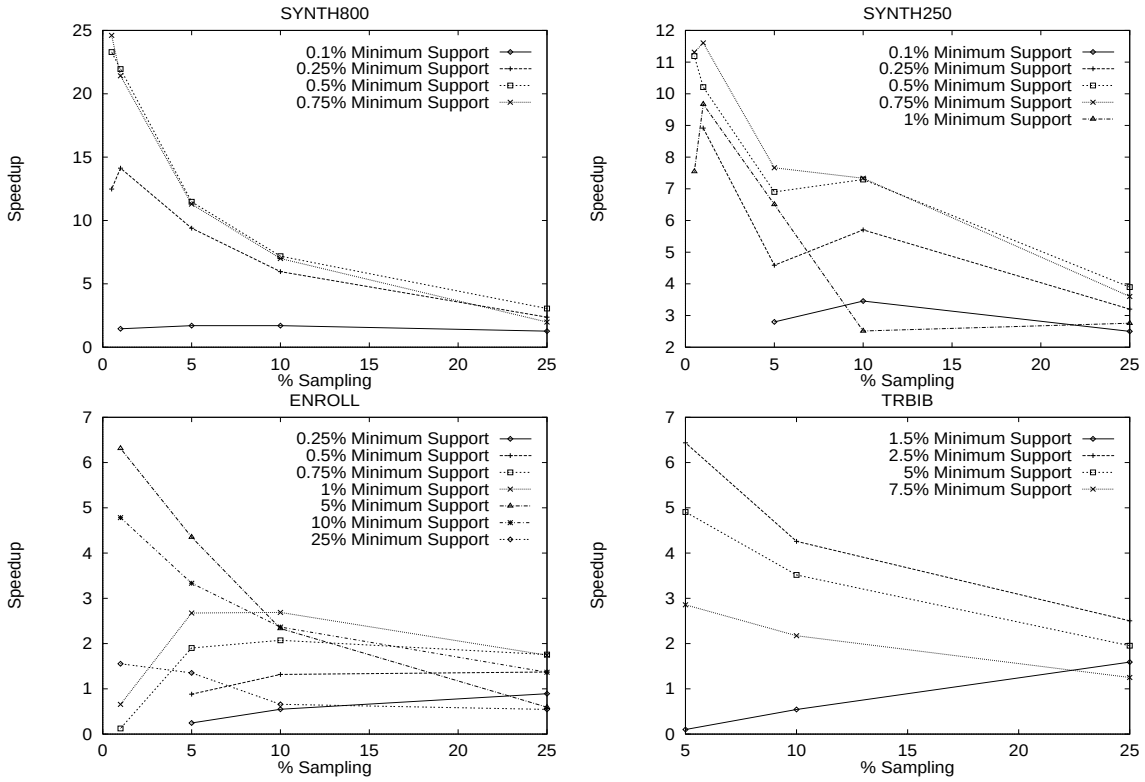

Figure 4. Sampling performance

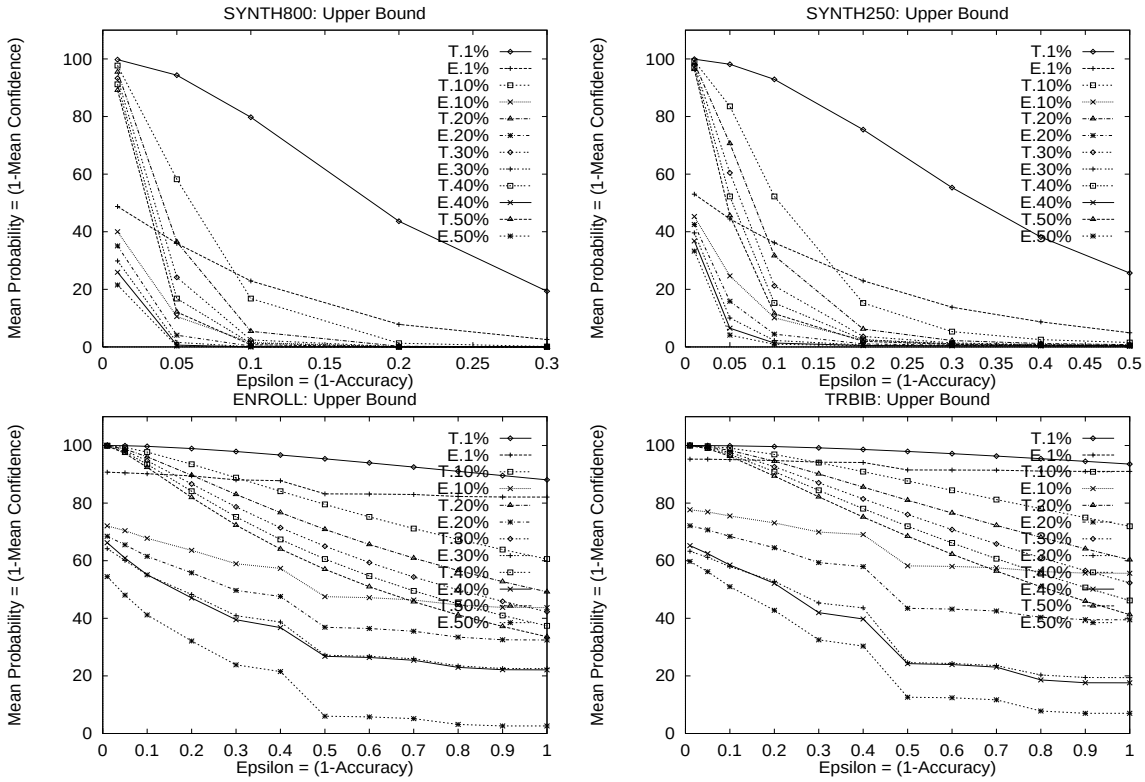

Figure 6. Accuracy vs. mean confidence for single items

Figure 6 gives a broader picture of the large gap between Chernoff bounds and experimentally obtained effectiveness of sampling. For all four databases we plot the mean of the confidence or the probability distribution for different accuracies ($1-\epsilon$). The mean confidence obtained from Chernoff bounds is marked as T.x, and that obtained experimentally is marked as E.x. Different values of the sample size x are plotted (from 1% to 50%), and results for only the upper bound are shown. For all the databases the upper and lower bounds give similar results. There is a small difference in the Chernoff bound values due to the asymmetry in equations 1 and 2. This is also true for the experimental results. For both cases the lower bounds give a slightly higher confidence for the same value of accuracy, as expected from the Chernoff bounds.

For SYNTH800 and SYNTH250 we observe that as the accuracy is compromised (as ϵ increases) the mean confidence across all items increases exponentially (therefore, only ϵ values upto 0.5 are shown). Furthermore, as the sample size increases, the curve falls more rapidly, so that we have higher confidence even at relatively higher accuracies. For SYNTH800 we get higher confidence for higher accuracy, when compared to SYNTH250. For both ENROLL and TRBIB we get the same general trends, however the increase in confidence for lower accuracies is not as rapid. This is precisely what we expect. For example, consider the right hand side of Chernoff upper bounds (equation 2), $e^{-\epsilon^2 n\tau/3} = \mathcal{C}$. For a given ϵ and τ (the support for an item), a higher value of n gives us high confidence, as it results in a lower value for $\mathcal{C}$. For a given sampling percentage, since SYNTH800 and SYNTH250 are large, we expect a higher confidence than that for ENROLL or TRBIB (for example, with sampling = 10%, $\epsilon = 0.1$, and $\tau = 0.01$, we get $n = 80000$, $\mathcal{C} = 0.07$ for SYNTH800; $n = 25000$, $\mathcal{C} = 0.43$ for SYNTH250; $n = 3962$, $\mathcal{C} = 0.88$ for ENROLL; and $n = 1379$, $\mathcal{C} = 0.96$ for TRBIB). We get the same effect for the experimental results.

We can observe that for all the databases, the experimental results predict a much higher confidence, than that using Chernoff bounds. Furthermore, from the above analysis we would expect sampling to work well for larger databases. The distribution of the support of the itemsets in the original database also influences the sampling quality.

5. Related Work

Many algorithms for finding large itemsets have been proposed in the literature since the introduction of this problem in [1] (AIS algorithm). The *Apriori* algorithm [2] reduces the search space effectively, by using the property that any subset of a large itemset must itself be large. The DHP algorithm [12] uses a hash table in pass k to do efficient pruning of $(k+1)$-itemsets to further reduce the candidate set. The *Partition* algorithm [13] minimizes I/O by scanning the database only twice. In the first pass it generates the set of all potentially large itemsets, and in the second pass their support is obtained. Algorithms using only general-purpose DBMS systems and relational algebra operations have also been proposed [6, 7].

A theoretical analysis of sampling (using Chernoff bounds) for association rules was presented in [2, 10]. We look at this problem in more detail empirically, and compare theory and experimentation. In [8] the authors compare sample selection schemes for data mining. They make a claim for collecting the sample dynamically in the context of the subsequent mining algorithm to be applied. A recent paper [14] presents an association rule mining algorithm using sampling. A sample of the database is obtained and all association rules in the sample are found. These results are then verified against the entire database. The results are thus exact and not approximations based on the sample. They also use Chernoff bounds to get sample sizes, and lowered minimum support values for minimizing errors. Our work is complementary to their approach, and can help in determining a better support value or sample size. We also show results on the percentage of errors and correct rules derived at different sampling values, the performance gains, and also the relationship between performance, accuracy and confidence of the sample size.

6. Conclusions

We have presented experimental evaluation of sampling for four separate databases to show that it can be an effective tool for data mining. The experimental results indicate that sampling can result in not only performance savings (such as reduced I/O cost and total computation), but also good accuracy (with high confidence) in practice, in contrast to the confidence obtained by applying Chernoff bounds. However, we note that there is a trade-off between the performance of the algorithm and the desired accuracy or confidence of the sample. A very small sample size may generate many false rules, and thus degrade the performance. With that caveat, we claim that for practical purposes we can use sampling with confidence for data mining.

References

[1] R. Agrawal, T. Imielinski, and A. Swami. Mining association rules between sets of items in large databases. In *ACM SIGMOD Intl. Conf. Management of Data*, May 1993.

[2] R. Agrawal, H. Mannila, R. Srikant, H. Toivonen, and A. I. Verkamo. Fast discovery of association rules. In *Advances in Knowledge Discovery and Data Mining, U. Fayyad, G. Piatetsky-Shapiro, P. Smyth, R. Uthurusamy (Eds.)*. AAAI Press, Melo Park, CA, 1996.

[3] R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In *20th VLDB Conference*, Sept. 1994.

[4] W. G. Cochran. *Sampling Techniques*. John Wiley & Sons, 1977.

[5] T. Hagerup and C. Rüb. A guided tour of chernoff bounds. In *Information Processing Letters*, pages 305–308. North-Holland, 1989/90.

[6] M. Holsheimer, M. Kersten, H. Mannila, and H. Toivonen. A perspective on databases and data mining. In *1st Intl. Conf. Knowledge Discovery and Data Mining*, Aug. 1995.

[7] M. Houtsma and A. Swami. Set-oriented mining of association rules. In *RJ 9567*. IBM Almaden, Oct. 1993.

[8] G. John and P. Langley. Static versus dynamic sampling for data mining. In *2nd Intl. Conf. Knowledge Discovery and Data Mining*, Aug. 1996.

[9] D. E. Knuth. *The Art of Computer Programming. Volume 2. Seminumerical Algorithms*. Addison-Wesley, 1981.

[10] H. Mannila, H. Toivonen, and I. Verkamo. Efficient algorithms for discovering association rules. In *AAAI Wkshp. Knowledge Discovery in Databases*, July 1994.

[11] F. Olken and D. Rotem. Random sampling from database files - a survey. In *5th Intl. Conf. Statistical and Scientific Database Management*, Apr. 1990.

[12] J. S. Park, M. Chen, and P. S. Yu. An effective hash based algorithm for mining association rules. In *ACM SIGMOD Intl. Conf. Management of Data*, May 1995.

[13] A. Savasere, E. Omiecinski, and S. Navathe. An efficient algorithm for mining association rules in large databases. In *21st VLDB Conference*, 1995.

[14] H. Toivonen. Sampling large databases for association rules. In *22nd VLDB Conference*, 1996.

[15] J. S. Vitter. An efficient algorithm for sequential random sampling. In *ACM Trans. Mathematical Software*, volume 13(1), pages 58–67, Mar. 87.