Progressive Sampling for Association Rules *
(Paper ID:247)

Srinivasan Parthasarathy
Computer and Information Science
Ohio State University,
Columbus, OH 43235
srini@cis.ohio-state.edu
Paper ID: 247

In data mining, sampling has often been suggested as an effective tool to reduce the size of the dataset operated at some cost to accuracy. However, this loss to accuracy is often difficult to measure and characterize since the exact nature of the learning curve (accuracy vs. sample size) is data dependent, i.e., we do not know apriori what sample size is needed to achieve a desired accuracy on a particular dataset. In this article we propose the use of progressive sampling to determine the required sample size for association rule mining. We first show that a naive application of progressive sampling is not very efficient for association rule mining. We then present a refinement based on equivalence classes, that seems to work extremely well in practice and is able to converge to the desired sample size very quickly and very accurately. An additional novelty of our approach is the definition of a support-sensitive, interactive measure of accuracy across progressive samples.

1. INTRODUCTION

As our ability to collect, store, and distribute huge amounts of data increases with advancing technology, discovering the knowledge hidden in these ever-growing databases has become a pressing problem. This problem referred to as data-mining, an effort to derive interesting conclusions from large bodies of data, is an interactive process. In fact, interactivity is often the key to facilitating effective data understanding and knowledge discovery. In such an environment response time is crucial because lengthy time delay between responses of two consecutive user requests can disturb the flow of human perception and formation of insight. However, extracting knowledge from these massive databases is a compute and I/O intensive process which makes the task of guaranteeing quick response times difficult.

In order to to minimize the I/O traffic involved in such data-intensive applications researchers have evaluated the viability of using sampling[9] to reduce the dataset size. While such methods have shown quite a lot of promise it has been observed by several researchers[19, 20, 27] that it is often very difficult to quantify, apriori, the quality of the results obtained for a given sample size. Recently, to address this problem some researchers have proposed and evaluated progressive sampling[19] for select data mining tasks. Progressive sampling starts with a small sample and uses progressively larger ones until model accuracy no longer improves beyond a user specified threshold. In this paper we study progressive sampling methods as they apply to association rule mining, a key data mining task. Realizing an efficient method to progressively sample a dataset for association rule mining poses several interesting challenges.

First, and foremost, one needs to define a notion of model accuracy. In other words, for a particular dataset how does one define how good the sample is? This goodness criterion should be sensitive to relevant interaction parameters (e.g. support, confidence, important items[1] to the user) as well as the inherent properties of the dataset in question. A naive approach could be to compare the set of associations generated by the sample with the set of associations generated on the entire dataset. Obviously, this is self-defeating and does not take into account all aspects of user interaction.

Second, while defining a notion of model accuracy is important, one must also be able to compute it efficiently. Zaki *et al*[27], have observed, that at low sample sizes, there is a tendency to detect a large number of false positives. This property can increase the computational time to such an extent that in some datasets mining for associations on 5% of a dataset could take longer than mining for associations on 10% of the dataset. This above property of association rules adds overhead and can significantly limit the effectiveness of progressive sampling.

Third, as noted by Provost and Kolluri [20] "most discussions on sampling assume that producing random samples efficiently from large datasets is not difficult. This is

*This work was partially supported by an Ameritech Faculty Fellowship.

[1]For example a user may be interested in associations pertaining to a specific item (say diapers).

simply not true.". In fact most implementations require O(N) time where N represents the size of dataset and not the sample size (S). Naive implementations often may be much worse. This problem is critical especially for any kind of progressive sampling approach. One simply cannot afford to spend O(N) time to generate each progressive sample.

We address these three problems in the context of progressively sampling for association rules. Specifically our contributions are:

- A novel measure of model accuracy for progressively sampling association rules. The measure is designed in such a way to be sensitive to user parameters and interactions while not requiring execution on the entire dataset.
- An efficient technique for identifying the optimal sample size. This key result is based on the identification and tracking of a representative set of frequent itemsets (a small subset of the entire set of frequent itemsets). Essentially the computational element (computing the associations for a given sample size) is reduced significantly enabling faster convergence to the optimal sample size. This technique also addresses the high false-positive problem for low support values.
- An efficient technique based on asynchronous I/O operations and a novel application of a well known sampling methodology to improve the efficiency of generating a random sample, as perceived by the processor.

The rest of this article is organized as follows. In Section 2 we provide some background on progressive sampling and association rules. In Section 3 we present our progressive sampling approach. We empirically evaluate the proposed approach on synthetic and real datasets in Section 4. Finally we conclude with directions for future work in section 5.

2. BACKGROUND

2.1 Association Rules

Discovery of association rules is an important problem in database mining. The prototypical application is the analysis of sales or *basket* data [4]. Basket data consists of items bought by a customer along with the transaction identifier. Application domains for association rules range from decision support to telecommunications alarm diagnosis, and prediction.

The problem of mining association rules over *basket* data was introduced in [3, 4]. It can be formally stated as: Let $\mathcal{I} = \{i_1, i_2, \cdots, i_m\}$ be a set of m distinct attributes, also called *items*. Each transaction T in the database $\mathcal{D}$ of transactions, has a unique identifier, and *contains* a set of items, such that $T \subseteq \mathcal{I}$. An *association rule* is an expression $A \Rightarrow B$, where $A, B \subset \mathcal{I}$, are sets of items called *itemsets*, 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., $C = S(A \cup B)/S(A)$, i.e., the conditional probability that transactions contain the itemset B, given that they contain itemset A.

Data mining of association rules from such databases consists of finding the set of all rules which meet the user-specified minimum confidence and support values.

2.2 Database Layout

There are two possible layouts of the database for association mining. The *horizontal* layout consists of a list of transactions, where each transaction has an identifier followed by a list of items. The *vertical* layout consists of a list of items where each item contains a list of transactions that transacted on that particular item. Approaches based on the horizontal format include the popular Apriori algorithm[5] and its variants[6, 16, 12, 4, 22]. The Apriori algorithm uses the downward closure property of itemset support to prune the itemset lattice - the property that all subsets of a frequent itemset must themselves be frequent. Thus only the frequent k-itemsets are used to construct candidate (k+1)-itemsets. A pass over the dataset is made to identify which of the candidate (k+1)-itemsets are actually frequent. This process is repeated until there are no more frequent itemsets.

The vertical format has the advantage that the support for candidate k-itemset can be computed by simple tid-list intersections. The tid-lists cluster relevant transactions, and avoid scanning the whole databases to compute support, and the larger the itemset, the shorter the tid-lists, resulting in faster intersections[28]. An additional optimization of compressing the vertical lists results in additional gains due to lower memory and I/O traffic[7, 23].

From the perspective of sampling the horizontal format makes more sense as sampling in the vertical format will be inefficient. Essentially sampling in the vertical context will necessarily have to keep track of which transactions are in the sample and which are out and one would need to to scan through each tid-list and mark the transactions in the sample and those out of it. One can of course sample in the horizontal format and then convert to the vertical format on the fly.

2.3 Equivalence Class Partitioning

One way to improve the vertical approach is to use it in conjunction with equivalence class partitioning. Let the set of large two-itemsets, L_2, be {AB, AC, AD, AE, BC, BD, BE, DE}. Equivalence class partitioning partitions these itemsets by their (k-1 length where k= 2 in the above example) prefixes resulting in the following three partitions:S_A = [**A**] = {AB, AC, AD, AE}, S_B = [**B**] = {BC, BD, BE}, and S_D = [**D**] = {DE}. The same partitioning scheme can be recursively repeated to find all the associations. For instance, in the above example, partition [**B**] yields candidates {BCD,BCE,BDE}. Assuming that all of the candidates are deemed frequent then the level 3 partitions for [**B**] are S_{BC} = [**BC**] = {BCD,BCE}, and S_{BD} = [**BD**] = {BDE}. The advantage of this approach is that the algorithm can process an entire equivalence class partition before proceeding to the next partition. This improves memory locality and minimizes I/O traffic in the ECLAT algorithm[28], an approach based on the above equivalence class partitioning. When compared with the baseline apriori approach this also reduces the number of passes that need to be made over the entire dataset.

Note that for the present work it is important to define a notion of an equivalence superclass. Informally, an equivalence superclass is defined as the set of all frequent itemsets that can recursively be enumerated from a given partition. From the above example, the equivalence superclass from partition S_B, denoted as $ES_B = \{$BC,BD,BE,BCD,BCE,BDE$\}$.

2.4 Sampling in the Association Mining Context

While several authors have proposed various strategies on the use of sampling for KDD[11, 19, 14] and database tasks[17, 18], we limit the discussion in this section to those relevant to association mining. Toivonen [24] presents an association rule mining algorithm using sampling. The approach can be divided into two phases. During phase 1 a sample of the database is obtained and all associations in the sample are found. These results are then validated against the entire database. Since the approach is probabilistic (i.e. dependent on the sample containing all the relevant associations) not all the rules may be found in this first pass. Those associations that were deemed not frequent in the sample but were actually frequent in the entire dataset are used to construct the complete set of associations in phase 2. To maximize the effectiveness of the overall approach, the author makes use of lowered minimum support on the sample. The sample size is determined based on error tolerances from Chernoff's bounds. A detailed theoretical analysis of sampling (using Chernoff bounds) for association rules was presented by Zaki *et al* [27]. 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. Based on itemset frequencies, using Chernoff bounds one can obtain a sample size[27]. However, the sample size is independent of the original dataset size and can be quite large, sometimes larger than the original dataset! Empirical evidence in the same paper also showed that Chernoff bounds may be too pessimistic for association mining. It was also shown that sampling can be effective for association mining if the sample size were known apriori for the corresponding dataset and input parameters. However, determining the optimal sample size was left as an open problem. Note, that determining the optimal sample size efficiently can significantly improve on the overall performance of Toivenen's approach[24] since with a good estimate one could minimize the computational and I/O aspects of the second pass.

2.5 Progressive Sampling

In order to quickly estimate the optimal sample size, researchers have recently turned to progressive sampling. Before we detail this procedure we first define the notion of a learning curve. A learning curve is a mapping between sample size and model accuracy. Typically a learning curve is depicted with the vertical axis representing the accuracy of the model and the horizontal axis representing the sample size. Most learning curves typically have steeply sloping portion early in the curve, and a plateau late in the curve[19, 8]. The cost-performance tradeoff is best at the knee of the curve. Such curves exhibit the property that the slope of the curve is monotonically non-increasing with n (excepting for small local variance). While most learning curves exhibit the above behavior some curves can misbehave especially at small sample sizes[13].

The goal of progressive sampling is to start with small samples and progressively increase them as long as model accuracy improves sufficiently. Using such a technique one can identify the knee of the learning curve using basic slope characterization across recently evaluated samples. One problem in the association rule mining context is how does one quantify model accuracy? A simple metric would be to compare the set of associations found for a given sample size with the set of associations found for the entire dataset. However, to obtain the latter we would have to run the algorithm on the entire dataset!

The efficiency of progressive sampling is governed by the average case execution time performance of the algorithm, and by the sampling schedule. In the case of association rule mining algorithms while the worst case complexity is exponential the average case behavior typically tends to be linear, or in some cases quadratic. The issue of determining an optimal sampling schedule was addressed by Provost et al[19], where the authors show that a simple geometric sampling schedule is efficient in an asymptotic sense for most induction algorithms with a run time complexity of O(n) or worse as long as the maximum sample size is $n/2$.

However, the above result is not useful if the desired accuracy is met only at a a sample size of 70% (for a linear time algorithm). This reduction (from 100%) may still result in significant performance benefits. Another problem with the above theoretical model is that sampling overheads are ignored. Efficiently obtaining a sample from a large dataset, is often ignored by most researchers as pointed out by Provost and Kolluri[20]. If one were to account for this the benefits of progressive sampling would decrease. Another overhead, induced within the context of association rules, is the avalanche effect of detecting false positives at very small sample sizes (see Zaki et al for details[27]). We address these issues in the next section.

3. METHODOLOGY

In this section we present our approach for enabling efficient progressive sampling of association rules. We first describe our measure of model accuracy which is based on the notion of self-similarity of associations across progressive samples. A novelty of the proposed measure is that it is functionally dependent on user input parameters (support, constrained online associations etc.). This is important since optimal sample size (the knee of the learning curve) can vary for the same dataset with changes in user parameters.

We then address the efficiency issues raised in the previous section. We first identify a representative subset of frequent itemsets such that the behavior of our measure of model accuracy on this representative subset mimics the behavior of our measure on the entire set of associations. The primary advantage of this method is that it is more than an order of magnitude faster than tracking the learning curve of the entire set of associations.

The final problem we address relates to that of sampling overhead. To address this problem we rely on a combination of ideas from sampling methods[26], active disks[21], asynchronous I/O and file operations to efficiently sample a large dataset and minimizing the overheads that affect the performance of progressive sampling-based approaches.

3.1 Model Accuracy

Absolute model accuracy, for a given sample is difficult to measure without running the algorithm on the entire dataset. Since this is not possible to do due to efficiency constraints we define a measure of accuracy that is based on the following key intuitions:

For progressive samples d_1 and d_2, where size$(d_2) >$ size(d_1), the self-similarity between the set of associations generated under the two samples is likely to be low during the growth phase and is likely to be much higher during the "plateau" phase. Therefore identification of the knee of the curve can be done by measuring the self-similarity between progressive samples.

We now define the our notion of interactive self-similarity: Let A and B respectively be a the set of frequent itemsets for a database sample d_1 and that for a database sample d_2. For an element $x \in A$ (respectively in B), let $\sup_{d_1}(x)$ (respectively $\sup_{d_2}(x)$) be the frequency of x in d_1 (respectively in d_2). Our metric is:

$$Sim(d_1, d_2) = \frac{\sum_{x \in A \cap B} \max\{0, 1 - \alpha |\sup_{d_1}(x) - \sup_{d_2}(x)|\}}{\|A \cup B\|}$$

where α is a scaling parameter. The parameter α has a default value of 1 and can be modified to reflect the significance the user attaches to variations in supports. For $\alpha = 0$ the similarity measure is identical to $\frac{\|A \cap B\|}{\|A \cup B\|}$, i.e., support variance carries no significance. Sim values are bounded and lie in [0,1]. Sim also has the property of *relative ordinality*, wherein if $Sim(X, Y) > Sim(X, Z)$, then X is more similar to Y than it is to Z.

An important point raised by Das, Manilla and Ronkainen[10] while evaluating the similarity between two attributes was that using a different set of external probes to measure similarity could potentially yield a different similarity measure. In our case the action of modifying the external probe set corresponds to modifying the association sets evaluated over different samples of the input database. This is achieved either by modifying the minimum support or by restricting that associations should satisfy certain conditions (Boolean properties over attributes). Supporting such constraints and and interactions in our tool is accomplished by leveraging past work on online and interactive association mining [2, 15]. Again this aspect is quite relevant in our case since the optimal sample size (the knee of the curve) is different for a different set of parameter values.

3.2 Picking a Representative Set

The above metric suggests using the entire association set from consecutive samples to measure the self-similarity between progressive samples. However, as observed earlier, computing complete association sets for each sample may eventually defeat the purpose of sampling since evaluating each sample has various overheads associated with it. To overcome this problem what we need is a representative class of itemsets which has a behavior similar to the self-similarity curve of the original set of associations. Moreover, this representative class of itemsets should additionally satisfy the criteria that it should be efficient to compute.

Our proposed approach is to use equivalence superclasses generated from the most frequent item(s)[2] The key intuition here is that such equivalence classes in all likelihood most completely captures the entire set of frequent itemsets and the distribution of associated supports across all levels (based on itemset lengths) of frequent itemsets. As we shall see from the empirical results in the next section this representative class based approach yields a very good predictor of the self similarity measure. The other nice feature of this is that it can easily be computed using a trivial modification to the basic ECLAT algorithm and has all the locality-enhancing properties of the base ECLAT algorithm.

The simple modification to ECLAT involved re-labeling the k most frequent items according to their base frequencies[3] (i.e., the item with the most frequency was labeled itemset 1, the next most frequent is labeled itemset 2). This re-labeling is necessary to ensure that all frequent itemsets involving these k items will be part of the first k equivalence superclasses. See [28] for more details on why this is necessary. Then on using the first k equivalence superclasses as the representative set we found that it yielded very good predictions of the self-similarity measure on the entire set of associations. Empirically we found k=1 or k=2 produced excellent results.

We also considered an approach based on the apriori algorithm. Basically we tried to use k-itemsets where k was a parameter we varied as the representative class. We found that for most of the datasets we evaluated k=2,3,4, were not very useful predictors. While certain larger values k=5,6 and beyond for some datasets were decent predictors, there are two problems with using such a representative set. First, computing k=5 frequent itemsets using either the ECLAT[28] algorithm or the Apriori[5] algorithm is almost tantamount, to executing the entire association mining algorithm, in terms of computational expense. Second, these representative sets were not uniformly good predictors across all datasets, how then is one to decide which representative set to pick?

Note, no changes are required to the self-similarity measure, we are just replacing the base association set with the representative set as determined by the first few equivalence class partitions.

3.3 Efficient Sampling Methodology

Provost and Kolluri[20] point out that most sampling algorithms require O(N) or greater time to execute where N is the size of the database and that researchers rarely account for this overhead. For generating samples of the database we use the Method A algorithm presented by Vit-

[2]We considered using randomly selected equivalence superclasses as the representative class of itemsets. However, we found that while some equivalence superclasses were good predictors others were not. The ones that were good were generated by the frequent items.

[3]Note that computing the most frequent items can be accomplished on the fly (based on the first sample one can identify one of the more frequent items, if not the most frequent) or it can be pre-computed.

ter[26]. 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 probability (n-m)/N-t. This algorithm generates N random variates. Method A significantly speeds up the sampling process by efficiently determining the number of records to be skipped before the next one is chosen. It generates exactly n random variates. In conjunction with appropriate support for database indexing the method A scheme allows the sampling procedure to take O(n) time[4].

Note that even with direct indexing creating a sample still creates some level of overhead. This overhead can be split into two components computational overhead (determining which transactions are in the sample) and I/O overhead (reading in said transactions). The I/O component can be overcome with suitable systems support in the form of asynchronous I/O, a technique which allows I/O operations to overlap with useful computation. Essentially one can overlap the I/O required for the next progressive sample with the computation required for processing the current sample. In other words while we compute the association set for the current sample one can do the I/O for the next sample size (in the sampling schedule).

With additional systems support, such as those promised by active disk-like approaches[1, 21], one can move the computational overhead associated with sampling off the critical path as well. Essentially this would allow for computing the association set for the current sample to happen simultaneously with producing the next sample. This approach lends itself to accessing data over the network as well, since even that can be effectively overlapped with useful computation.

3.4 Overall Approach

The basic steps to the progressive sampling approach are highlighted in Figure 1. We present two approaches, one based on estimating self-similarity between the current sample and the subsequent sample in the schedule (RC-SS), and the other based on estimating self-similarity between the current sample and the entire dataset (but only computing the first k equivalence super-classes) (RC-S).

Step 0 for the RC-S algorithm computes the representative set on the entire dataset. Step 1 for the RC-SS algorithm computes the representative set on the lowest sample size in the schedule. Step 2 is an iterative for loop. Each iteration of this for loop first computes the next sample (Step3) in the schedule. Then the representative set for this sample is constructed (Step4). After this the self-similarity measure is computed. For the RC-SS algorithm we compute the self similarity between this representative set and the previous one. For the RC-S algorithm we compute the self similarity between this representative set and the one pre-computed in Step 0. If the similarity metric is above a user-specified threshold, then convergence is achieved (Step 5). For RC-SS we modified the convergence criteria slightly after viewing empirical results. We found that the self-similarity curve for RC-SS is not always well behaved (is non-monotonic). For this algorithm we declared convergence only if the similarity metric is above the user-specified threshold for two consecutive iterations. This avoids premature convergence due to local variances and also protects against the possible misbehavior of self-similarity curves (found mainly at small sample sizes). If the convergence criteria is met we break out of the for loop (Step 6) else we continue (Step 7). Once convergence is determined we compute the overall association set for the sample size we converged at (this is the first of the two sample sizes for which the user specified threshold was met) in Step 8.

Step 0: Compute representative set
on entire dataset (RC-S only)
Step 1: Compute initial sample and
representative set on sample. (RC-SS only)
Step 2: For each sample size in the schedule:
Step 3: Compute sample.
Step 4: Compute representative set.
Step 5: Is convergence criteria met?
Step 6: If yes set effective sample
size (ESS) and break;
Step 7: If no continue;
Step 8: Compute the rest of the entire set
of associations for the ESS.

Figure 1: Proposed Approach

We implemented the progressive sampling approach on a dual processor linux box. Linux does not support asynchronous I/O efficiently (it actually forks of a new process to handle asynchronous I/O requests), and there is currently no vendor support for active disks. In order to evaluate the effectiveness of our sampling methodology we used the second processor on our system to compute the samples from a remote dataset while the first processor was working on the current sample. One limitation of this simulated environment is that the fact that the active disk processor is as fast as the compute processor. In the next section we evaluate our results keeping this limitation in mind.

4. EXPERIMENTAL METHODOLOGY

In this section we empirically evaluate the proposed methodology on several synthetic and real datasets. We first describe the experimental setup (machine configuration, dataset properties etc.). We then evaluate the proposed measure of model accuracy: self similarity between consecutive samples; and the impact on this measure when using the first equivalence superclass as a representative set. We then quantify the performance gains from using this representative class (as opposed to the entire set of associations). At the end of this section we quantify the gains from our sampling methodology.

4.1 Experimental Setup

We used three synthetic datasets generated from the IBM dataset generator program[5]. These datasets mimic the transactions in a retailing environment. The properties of these three datasets are described in Table 2. The prop-

[4]Note that if each transaction in the database is directly indexable proving this bound for the *worst-case* is trivial.

Database	$numT$	Size
T20I6D1M	1000000	100MB
T10I4D5M	5000000	260MB
T8I5D25M	25000000	1.25GB
Gazelle	59602	1.35MB
Vbook	136809	4.3MB

Figure 2: Database properties

erties of the synthetic datasets are inherent in the names. The number following the **T** refers to the average transaction length, the number following the **I** refers to the average maximal potentially frequent itemset size, the alphanumeric-code following the **D** refers to the total number of transactions (1M means 1 Million). The dataset sizes of these synthetic datasets ranged from 100MB to 1.25GB. We also used two real datasets for our experiments. The first real dataset is the Gazelle dataset which formed a part of the KDD Cup 1999 competition. This dataset contains 59602 transactions and is a little over 1.35 MB in size. The second real dataset is the VBook dataset which is a dataset from a prominent online bookstore retailer in South America. It contains a little over 136,809 transactions and is a little over 4.3MB in size. For a detailed description of this dataset see[25].

All experiments unless otherwise noted, were performed on a dual pentium node machine, 1GHz Pentium III, having 512 MB RAM and running Linux 2.4.

4.1.1 Impact of Representative Set on Self Similarity Measure

For all the experiments in this section we assumed a roughly geometric sampling schedule up to 40% of the original dataset and an arithmetic progressive schedule from that point on, if required. Specifically we used the following sampling schedule for all the experiments on the synthetic datasets unless otherwise noted: 0.5%, 1%, 2%, 5%, 10%, 20%, 40%, 50%, 60%, 70%, 80%. For the two real datasets the schedule we adopted started from 2% (since they have fewer transactions, going to very small sample sizes, resulted in absolute support frequencies of 1 or 2 transactions which are not very useful).

We first consider the two real datasets. Both of these datasets are not very large, so the relative benefits from sampling is not likely to be great. The overall quality of results from small samples is likely to be terrible. The self similarity plots for all the datasets are described in Figure 3. In each graph the Y-axis corresponds to the self similarity and the X-axis corresponds to the sample size. in each graph there are four plots, the RC-S and RC-SS plots have already been explained in the previous section. The A-S plots (in bold) is the learning curve that we are trying to estimate. They represent the naive approach[5] of comparing the set of associations generated by the sample with the set of associations generated by the entire dataset. The A-SS plots mirror the procedure described for R-SS, the only difference being that the self-similarity measure is computed on the entire set of associations.

On viewing the plots for the Gazelle dataset (Figure3a and Figure3b) one can observe that the plots for the representative set (prefix RC) closely follow the plots for the entire set of associations (prefix A) for both support values considered. Furthermore, note that if we required a self similarity cut off of 0.9, for 0.1% support we would never obtain this value (even if we went the distance), whereas at 0.25% support this would be achieved at a 50% sample size. This shows that the user-specified parameters has an important role to play in determining whether sampling is useful or not and if so at what level. We observed similar trends if instead of modifying the support value, we chose to identify templates of interest (for example all associations including item 5 and excluding item 7). Overall the plots also follow the expected pattern of low self similarity at smaller samples, and higher self similarity at larger samples. The phase transition from lower self-similarities to higher self-similarities coincide with the knee of the learning curve.

The results for the VBook (Figure 3c and Figure 3d) dataset are similar for 0.1% support values. At 0.25% support we observe that there is a sharp decline in the self-similarity curve (for RC-S,not for RC-SS) followed by a rise then it stabilizes. While this is a local variation (and one can see this in a minor way in other graphs as well) we should point out that if the user had specified a self-similarity parameter of 0.9 and we did not require as part of our convergence algorithm two or more consecutive samples to have this requirement, our algorithm would have converged (incorrectly) at a sample size of 5% for this dataset. This convergence criteria is also reinforced by the fact that some learning curves are not well-behaved at lower sample sizes[13].

The results for the synthetic datasets are better (see Figures 3e-j). The shape of the self similarity plots for the representative class almost exactly mirror the self similarity plots for the entire set of associations. Unlike the plots for the real datasets, the self similarities are quite high even at low sample sizes. This is mainly due to the fact that the synthetic datasets are much larger than the real datasets therefore the absolute number of transactions for a given sample size (expressed as percentage of the original) is much larger. Another reason for this difference could be traced to the fact that the real datasets have different properties from the synthetic datasets[29].

Overall the RC-S plots most closely mirrors A-S plots. It also seems to have desirable monotonic properties. However, due to the requirement of precomputing the representative set on the entire dataset (Step 0 of the algorithm) the RC-S method is more expensive (execution time wise). For many of the datasets (at the support values considered) the RC-SS and A-SS plots yield very good approximations.

Another interesting trend that is observed across both real and synthetic datasets is that there seems to be a relatively consistent overestimation or underestimation of the self-similarity across RC and A graphs. For example in Figure 3c until a sample size of 40%, the RC curves consistently overestimate their A-curve counterparts. After sample size 40% they underestimate their A-curve counterparts. This would seem to indicate that the curve we want to estimate (A-S) can be best estimated by using the RC-S curve and a translation factor that can be derived from the translations computed from the RC-SS and A-SS curves. This aspect will be investigated as part of future work.

[5]Note, that actually computing the A-S plot you will need to execute the algorithm on the entire dataset.

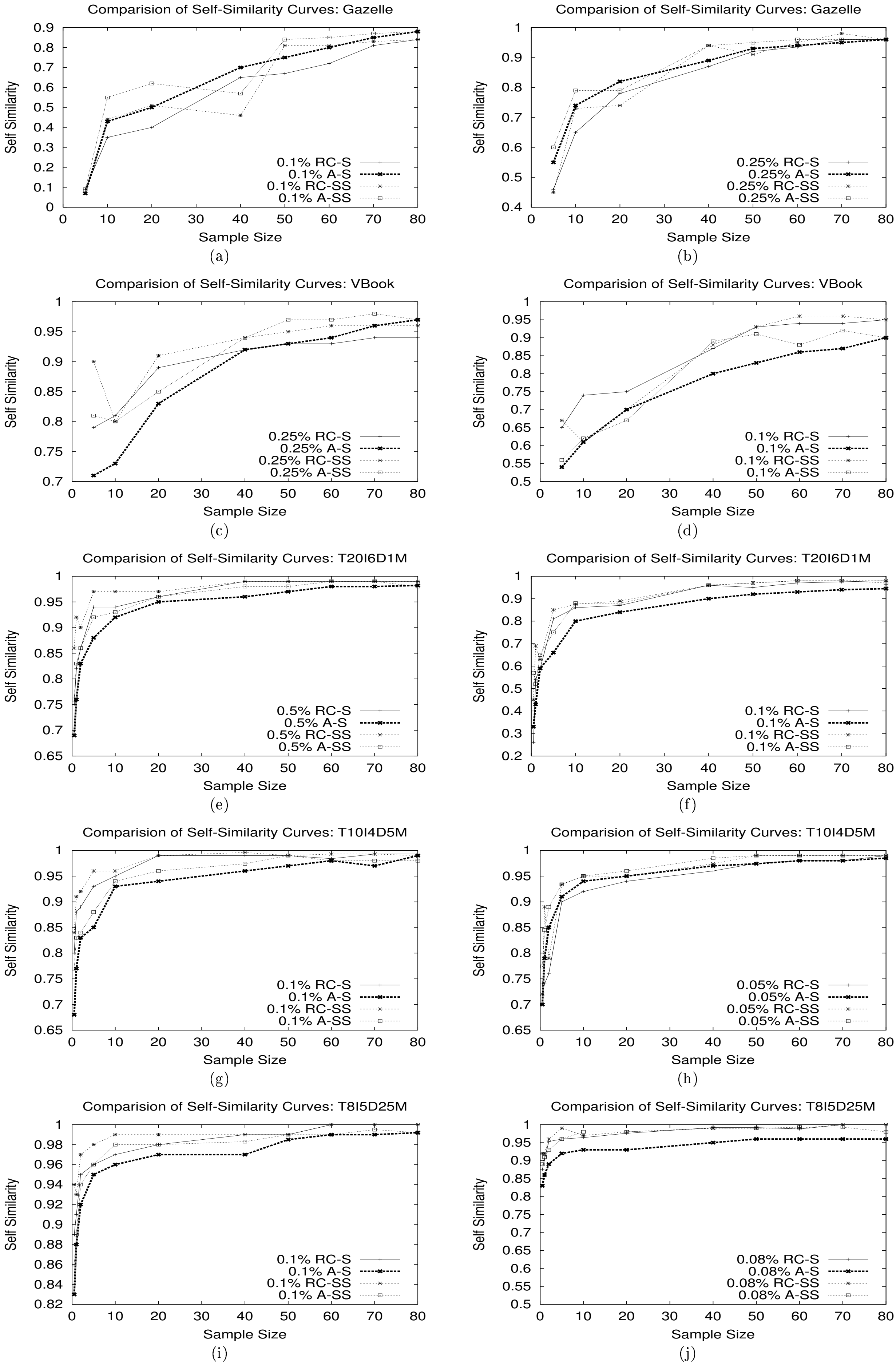


Figure 3: Evaluation of Proposed Method: Similarity plots

4.2 Impact of Representative Set on Performance

In this section we highlight the performance gains from using the representative class of itemsets to compute self similarities over using the entire set of associations. We first consider the performance benefits on the larger of the two real datasets: Vbook. Figure 4a plots the cumulative execution time for different sample sizes. The cumulative execution time is the execution time of the algorithm (RC-S or RC-SS) when the convergence criteria is satisfied at the corresponding sample size. The *Baseline Execution Time* corresponds to the execution time of the association mining algorithm on the entire dataset. Essentially the break even point for progressive sampling, when computing on the entire set of associations (represented by the prefix A) at 0.5% support, is when the cumulative execution time intersects with this baseline. As we can see from the figure the break even point when evaluating self similarities on the entire set of associations is reached at a sample size of around 10% for this dataset at this value of support. The poor performance here can be traced to the fact that for this support value at low sample sizes the basic association mining algorithm detects a large number of false positives. However, for the representative class approach, after determining the effective sample size, the algorithm still has to be completely executed on that sample size (sans the representative class). For this dataset if the effective sample size determined was 50% (corresponding to a value of 0.9 in the RC-SS (RC-S) plot from Figure 3d) then the total execution time is equal to 1.1s (1.15s for RC-S) which is still below the baseline of 1.5s. This result is especially encouraging and shows that even for a small dataset the approach presented can be quite effective.

As can be observed from Figures 4b-d, the results on the synthetic datasets are even better. For T10I4D5M for a self similarity cut off of 0.95 or higher the optimal sample size is 50% (this can be seen from Figure 3h). The RC-SS approach required a total of 448 (496 for RC-S) seconds to compute the results (see Figure 4b). The approach using the entire association set requires 1037 seconds (which is below the baseline of 950s) to compute the same result. This is a factor of 2.1 improvement over the baseline execution and a little over a factor of 2.4 improvement over the approach using the entire set of associations. For a self similarity cut off of 0.9 the effective sample size is 20%. The RC-SS approach requires a total of 166 seconds (214s for RC-S) to compute the result, whereas the approach using the entire set of associations requires 300s (an improvement factor of 1.8).

Similar results were observed for the other synthetic datasets (see Figure 4c-d). For T20I6D1M (Figure4c) for a self similarity cut off of 0.9 or higher the optimal sample size is 50% (from Figure 3f). The representative class approach requires a total of 686 (RC-SS requires 736) seconds to compute the results. The approach using the entire association set requires 1669 seconds (well above the baseline of 1316 seconds) to compute the same result.

For T8I5D25M (Figure4d) for a self similarity cut off of 0.95 or higher the optimal sample size is 10% (from Figure 3j). The representative class approach requires a total of 73.17 (125 for RC-SS) seconds to compute the results. The approach using the entire association set requires 140 seconds to compute the same result. When compared with the baseline execution time (1730s) the representative class approach's improvement factor is around 25.

4.3 Impact of Sampling Methodology on Performance

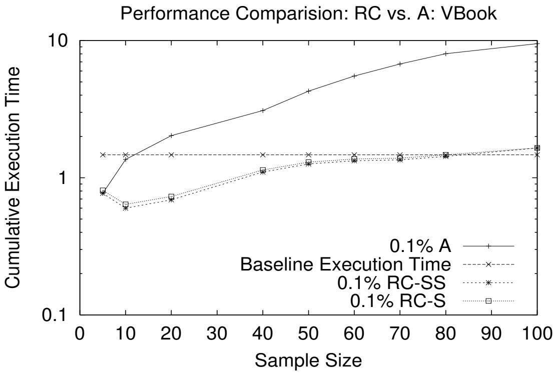
In Figure 5 we consider the performance of the sampling methodology presented in Section 3.3. Figure 5a compares the cumulative execution time performance of our the approach using the representative class to identify the ideal sample size while overlapping the sampling and I/O operations of the next stage with the computation of the current stage. In Figure 5a we compare the performance of ideal situation (where there is no sampling overhead: TC-ideal) with the actual performance with overlapping (TC-overlap) and without overlapping (TC-nooverlap). For the overlapping computation we used two scenarios, one where the processor handling the I/O and sampling was a 300Mhz Pentium II[6] (i.e. a much slower processor than the compute processor as discussed in Section 3.4) and one where the processor handling the I/O and sampling was the same configuration as the compute processor (1GHz).

On viewing the graphs it is clear that the TC-ideal graph and the TC-overlap (1GHz) are almost identical, reflecting the fact that almost all of the sampling overhead is overlapped with useful computation (computing the association set for the previous sample size). On relaxing the assumption and allowing for the fact that the processor doing the sampling and I/O is much slower (300 Mhz) we observe a marginal drop in performance (slightly under 4%), so still most of the sampling overhead is overlapped with useful computation. Note that if we did not overlap the sampling overhead with computation then we perform 10-12% worse than the ideal (comparing the TC-Ideal and TC-noverlap graphs).

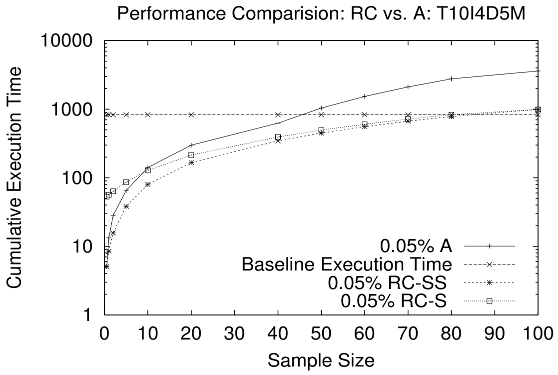
The above experiment assumed the best possible sampling algorithm (the Sample A algorithm). We further broke down the performance of the sampling overheads in Figure 5b for the two configurations (1GHz and 300Mhz). Here again the Y-axis represents cumulative sampling overheads and the X axis represents sample size. The performance of the naive algorithm (discussed in section 3.3) is much worse than the Sample A algorithm. This highlights the importance of the computational overhead of the sampling. Just invoking N random variates vs. invoking n random variates makes a huge difference in performance. We have shown this overhead (computational and I/O) can be effectively overlapped with computation to efficiently sample in a progressive manner for association rules.

5. CONCLUSIONS AND FUTURE WORK

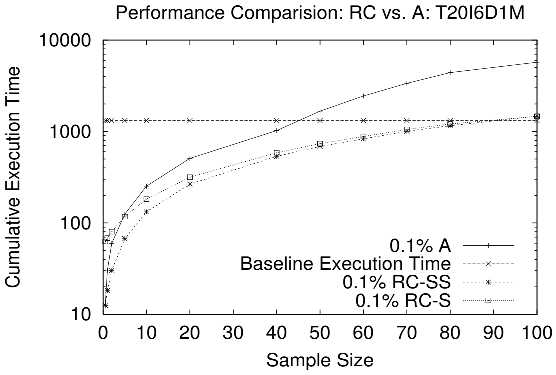
[6]We did not have a dual processor system where one processor was fast and the other was slow. We timed the sampling overheads for each of the different sample sizes for this dataset on the slower machine, and we used these times (by precomputing the samples and busy waiting for the necessary amount of time) while evaluating the performance on this configuration. This this approach at least gives us an idea of the performance under an active disk scenario where the processor attached to disk is not state-of-the-art!



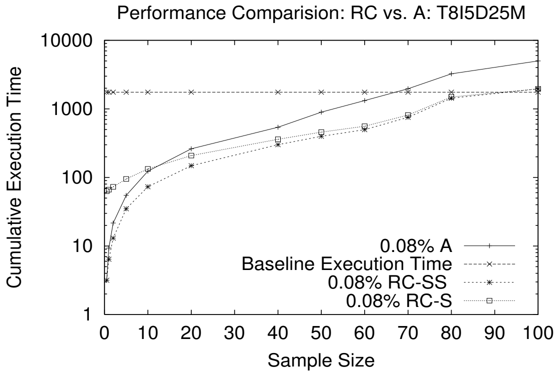
(a) VBook



(b) T10I4D5M

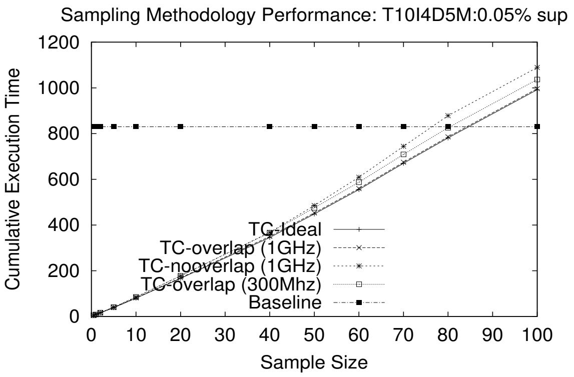


(c) T2OI6D1M

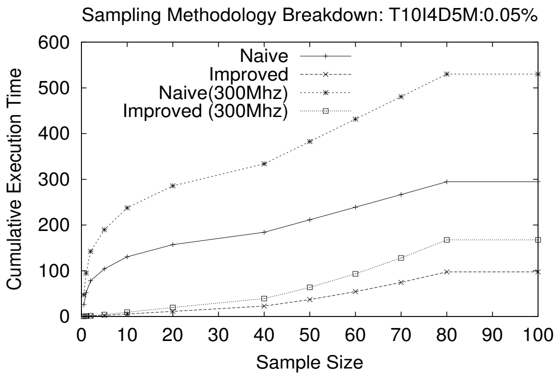


(d) T8I5D25M

Figure 4: Impact of Representative Set on Performance



(a) Overall Performance



(b) Breakdown

Figure 5: Sampling Methodology on Performance Breakdown

We have presented an efficient method to progressively sample for association rules. Our approach relies on a novel measure of model accuracy (self-similarity of associations across progressive samples), the identification of a representative class of frequent itemsets that mimic (extremely accurately) the self-similarity values across the entire set of associations, and an efficient sampling methodology that hides the overhead of obtaining progressive samples by overlapping it with useful computation.

We evaluated the results on a set of real and synthetic datasets. We extensively benchmarked each aspect of our algorithm and obtained uniformly good performance (several factor-fold execution time improvements) across both real and synthetic datasets.

As part of ongoing work we are currently bootstrapping the proposed approach with the work proposed by Toivonen[24] as suggested in Section 2.4. In fact for this bootstrapping exact estimates of the learning curve are not needed, but detecting the knee of the curve is very important. Our work shows that this can be done quite effectively using the methods proposed in this paper.

In the current work we have considered each sample to be independent of the other. We would like to see if the proposed method can be improved by using adaptive sampling techniques[11]. While there are likely to be obvious performance gains from this approach, it remains to be seen if the self-similarity measure quality drops as a result. In Section *4.1.1* we observed that while the self similarity plots of the representative class closely mirror the self similarity plots of the actual association sets, there is often a consistent translation factor separating the two plots. An interesting line of future work would be to estimate the translation factor to improve the accuracy of the proposed approach.

6. REFERENCES

[1] Anurag Acharya, Mustafa Uysal, and Joel H. Saltz. Active disks: Programming model, algorithms and evaluation. In *Architectural Support for Programming Languages and Operating Systems*, pages 81–91, 1998.

[2] C. Aggarwal and P. Yu. Online generation of association rules. In *IEEE International Conference on Data Engineering*, February 1998.

[3] 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.

[4] R. Agrawal, H. Mannila, R. Srikant, H. Toivonen, and A. Inkeri Verkamo. Fast discovery of association rules. In U. Fayyad and et al, editors, *Advances in Knowledge Discovery and Data Mining*. MIT Press, 1996.

[5] R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In *20th VLDB Conf.*, September 1994.

[6] Roberto J. Bayardo. Efficiently mining long patterns from databases. In *SIGMOD 1998*, pages 85–93, 1998.

[7] Doug Burdick, Manuel Calimlim, and J. E. Gehrke. Mafia: A maximal frequent itemset algorithm for transactional databases. In *ICDE*, 2001.

[8] Jason Catlett. Megainduction: A test flight. In *Machine Learning*, pages 596–599, 1991.

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

[10] G. Das, H. Mannila, and P. Ronkainen. Similarity of attributes by external probes. In *Proceedings of the 4th Symposium on Knowledge Discovery and Data-Mining*, 1998.

[11] Carlos Domingo and Osamu Watanabe. Scaling up a boosting-based learner via adaptive sampling. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 317–328, 2000.

[12] J. Han and J. Pei. Yiwen yin: Mining frequent patterns without candidate generation. In *SIGMOD Conference 2000: 1-12 24*, 2000.

[13] Haussler, Kearns, Seung, and Tishby. Rigorous learning curve bounds from statistical mechanics. In *COLT: Proceedings of the Workshop on Computational Learning Theory, Morgan Kaufmann Publishers*, 1994.

[14] George H. John and Pat Langley. Static versus dynamic sampling for data mining. In Evangelos Simoudis, Jiawei Han, and Usama M. Fayyad, editors, *Proc. 2nd Int. Conf. Knowledge Discovery and Data Mining, KDD*, pages 367–370. AAAI Press, 2–4 1996.

[15] M. Klemettinen, H. Mannila, P. Ronkainen, H. Toivonen, and A. I. Verkamo. Finding interesting rules from large sets of discovered association rules. In *3rd Intl. Conf. Information and Knowledge Management*, pages 401–407, November 1994.

[16] Dao-I Lin and Zvi M. Kedem. Pincer search: A new algorithm for discovering the maximum frequent set. In *Extending Database Technology*, pages 105–119, 1998.

[17] Y. Ling and W. Sun. An evaluation of sampling-based size estimation methods for selections in database systems. In *11th Intl. Conf. on Data Engg.*, pages 532–539, 1995.

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

[19] Foster J. Provost, David Jensen, and Tim Oates. Efficient progressive sampling. In *Knowledge Discovery and Data Mining*, pages 23–32, 1999.

[20] Foster J. Provost and Venkateswarlu Kolluri. A survey of methods for scaling up inductive algorithms. *Data Mining and Knowledge Discovery*, 3(2):131–169, 1999.

[21] E. Riedel, G. Gibson, and C. Faloutsos. Active storage for large-scale data mining and multimedia. In *Proceedings of the 24th International Conference on Very large Databases*, 1998.

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

[23] Pradeep Shenoy, Jayant R. Haritsa, S. Sundarshan, Gaurav Bhalotia, Mayank Bawa, and Devavrat Shah. Turbo-charging vertical mining of large databases. In *SIGMOD*, pages 22–33, 2000.

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

[25] A. Veloso, W. Meira, M. Carvalho, B. Possas, S. Parthasarathy and M.Zaki. Mining Frequent Itemsets in Evolving Databases To appear in the SIAM International Conference on Data Mining 2002.

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

[27] M. J. Zaki, S. Parthasarathy, W. Li, and M. Ogihara. Evaluation of sampling for data mining of association rules. In *7th Intl. Wkshp. Research Issues in Data Engg*, April 1997.

[28] M. J. Zaki, S. Parthasarathy, M. Ogihara, and W. Li. New algorithms for fast discovery of association rules. In *3rd Intl. Conf. on Knowledge Discovery and Data Mining*, August 1997.

[29] Zijian Zheng, Ron Kohavi, and Llew Mason Real World Performance of Association Rule Algorithms. KDD 2001.