
Journal of Intelligent Information Systems 12, 61–73 (1999)

Borders: An Efficient Algorithm for Association Generation in Dynamic Databases

YONATAN AUMANN aumann@cs.biu.ac.il
RONEN FELDMAN feldman@cs.biu.ac.il
ORLY LIPSHTAT okatz@cs.biu.ac.il
Department of Mathematics and Computer Science, Bar-Ilan University, Ramat-Gan 52900, Israel

HEIKKI MANILLA Mannila@cs.helsinki.fi
Department of Computer Science, University of Helsinki, Helsinki, Finland

Abstract. We consider the problem of finding association rules in a database with binary attributes. Most algorithms for finding such rules assume that all the data is available at the start of the data mining session. In practice, the data in the database may change over time, with records being added and deleted. At any given time, the rules for the current set of data are of interest. The naive, and highly inefficient, solution would be to rerun the association generation algorithm from scratch following the arrival of each new batch of data. This paper describes the *Borders* algorithm, which provides an efficient method for generating associations incrementally, from dynamically changing databases. Experimental results show an improved performance of the new algorithm when compared with previous solutions to the problem.

Keywords: association rules, knowledge discovery, data mining

1. Introduction

One of the most commonly used techniques in the new emerging field of Data Mining (Fayyad et al., 1995) is discovery of association rules. Association rules are rules regarding 0-1 data that represent positive correlations between two sets of attributes (Agrawal et al., 1993). Association rules have proven useful in several application areas including: analysis of basket data for customized marketing programs, and telecommunications alarm correlation (Mannila et al., 1994).

Several algorithms have been proposed for discovering association rules (Agrawal et al., 1995; Agrawal et al., 1993; Mannila et al., 1994). These algorithms assume that all transactions are available prier to the first time the algorithm is executed. However, in most cases this assumption does not hold. Rather, information is constantly added (and possibly also deleted) from the database, and we seek the associations relevant to the *current* set of data. A naive solution would be to rerun the algorithm from scratch every time new data arrives. This, however, is highly inefficient, as adding even a very small amount of new data will require running the association generation algorithm on all known transactions. Thus, we seek an *incremental* algorithm, which allows to generate the new associations in an incremental manner. Instead of processing all the records again, such an algorithm would only perform a small fraction of the work on each new set of data and thereby provide the results in a timely manner.

In this paper we describe a new incremental algorithm for association generation, the *Borders* algorithm. The algorithm is based on a new incremental method for generating the *frequent sets,* which are the basis for the association rules. Two main features characterize the Borders algorithm:

1. If the changes do not produce any new frequent sets, then there is no access to the old data. Thus, we guarantee that costly scanning of the entire database will only be performed when *new* frequent sets are obtained. No previous algorithm provides this guarantee.
2. In case the entire database is scanned, the number of passes over the database is generally very small, and there are only relatively few candidates for which support is counted.

We provide three variants of our algorithm: one for the case of additions alone, one for additions and deletions, and one for the case where the analyst wishes to change the support threshold, without having to run the entire algorithm anew.

Incremental algorithms for association generation were previously considered in Cheung et al. (1996), Cheung et al. (1997), Feldman et al. (1997). We compare the performance of the new Borders algorithm to the FUP algorithm (Cheung et al., 1996). In comparison, the Borders algorithm exhibits improved computing times across a wide range of parameters.

The rest of this paper is organized as follows. Section 2 provides the basic definitions. Section 3 describes the Borders algorithm. In Section 4 we provide empirical results, testing the performance Borders for a range parameters, and comparing the performance to that of FUP. We conclude in Section 5 with a short discussion.

2. Definitions

Let $A = \{A_1, \ldots, A_m\}$ be a set of attributes with binary domain ({0,1}). (The attributes are sometime also called "*items*".) A *row*, r, over A is a tuple $r = (r[A_1], \ldots, r[A_m])$, of 0's and 1's. Such a row can also be viewed as a set $\{A_i \mid r[A_i] = 1\}$ of the attributes from A. A *relation*, R, over A is a multiset of rows over A. In the incremental setting, we denote the initial relation by R_O and the new set of added rows by R_N. If records are deleted, we denote the set of deleted records by R_D. We denote by n_O, n_N, and n_D the number of rows in R_O, R_N, and R_D, respectively.

For a given row, r, and set of attributes X, we say that r *includes* X if $X \subseteq r$. The *count* of X in a relation R, denoted by $c(X, R)$, is the number of rows $r \in R$ that include X. The *support* of X in R, denoted by $s(X, R)$ is the fraction of rows which include R, i.e., $s(X, R) = c(X, R)/|R|$ (where $|R|$ is the number of rows in R). An *association rule* or simply an *association* is an expression of the form $X \Rightarrow Y$, where X and Y are attribute sets. The *support* of the association is the support of $X \cup Y$, and the *confidence* of the association, is $s(X \cup Y, R)/s(X, R)$. We search for associations where the support is above some user- defined threshold, denoted by σ, which we call the *minimum support,* and whose confidence is above another user-defined threshold, denoted by γ. An attribute set with at least the minimum support is called a *frequent set*.

Example. Let A = {a,b,c,d}, R = {{a,b,c}, {a,b,d}, {a,c}}, $\sigma = 2/3$, and $\gamma = 0.5$ then the frequent sets are: {a},{b},{c},{a,b},{a,c}, and the association rules are:

$\{a\} \Rightarrow \{b\}$-support 2/3, confidence 2/3

$\{b\} \Rightarrow \{a\}$-support 2/3, confidence 1

$\{a\} \Rightarrow \{c\}$-support 2/3, confidence 2/3

$\{c\} \Rightarrow \{a\}$-support 2/3, confidence 1.

All known algorithms for generating associations operate in two phases. First, all frequent sets are generated. Then the association rules are derived from these sets. The first phase is the most time-consuming (Agrawal et al., 1995). Accordingly, in this paper we present incremental algorithms for generating the frequent sets.

The support of any given attribute set X can be obtained by simply scanning through the relation and counting the number of rows that include X. The problem is that *any* subset of attributes is potentially a frequent set, and the number of these subsets is exponential. Consequently, it is impossible to count the support of all attribute sets simultaneously, in a single scan of the relation. To overcome this problem, algorithms for finding frequent sets operate in *rounds*, scanning the relation many times. Before each scan, a (relatively) small set of *candidates* is generated. During the scan, only the count of the candidates is computed. Scanning the entire relation can be very costly, as a typical relation may be very large. Thus, it is necessary to keep the number of rounds as small as possible. In general, the number of relation scans is the dominating factor in the performance of an algorithm.

3. The borders algorithm

The Borders algorithm is based on the notion of *border sets*, introduced in Mannila and Toivonen (1997). A set X is a *border set* if all its proper subsets are frequent sets (i.e., sets with at least minimum support), but it itself is not a frequent set. Thus, the collection of *border sets* defines the borderline between the frequent sets and non-frequent sets, in the lattice of attribute sets. An example of a relation, it frequent sets and border sets is depicted in figure 1.

The Borders algorithm works by constantly maintaining the count information for all frequent sets and all border sets in the current relation. When an increment, R_N, arrives, the increment alone is scanned to obtain its support for all (previous) frequent and border sets. From this information, we compute (with no extra data access), the support of all frequent and border sets in the combined relation. Additional scans of the entire relation are performed only if the support of some border set has reached the minimum support threshold (thus turning into a frequent set). This policy guarantees that the full relation is never scanned if there is no new frequent set. Furthermore, even when additional scans are required, monitoring the border sets minimizes the amount of counting work performed during these scans. We shall prove that monitoring the border sets alone is sufficient for discovering all frequent sets.

The following is a high-level description of the algorithm. A detailed description is given in figure 2. We describe an incremental stage, where a new set of data R_N is added to the

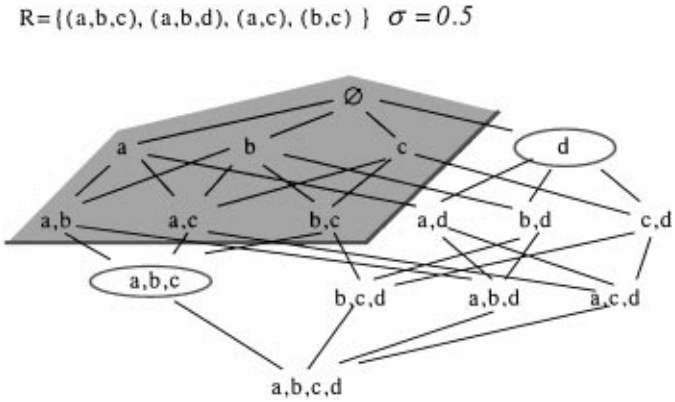

Figure 1. Border sets. The frequent sets are depicted with gray background. Border sets are circled. Lines indicate containment.

Input: R_N , R_O, σ, *Borders* and *FrequentSets* of R_O and their count
Output: *Borders* and *FrequentSets* of $R_O \cup R_N$ and their count

1. Scan new relation R_N and find count $c(X,R_N)$, for all $X \in$ *Borders* $\cup$ *FrequentSets* .
2. **forall** $X \in$ *Borders* $\cup$ *FrequentSets* **do**
3. $c(X,R_O \cup R_N) = c(X,R_O)+c(X,R_N)$
4. $s(X,R_O \cup R_N) = c(X,R_O \cup R_N)/(n_O+n_N)$
5. **end do**
6. *PromotedBorders* = { $X \in$ *Borders* | $s(X,R_O \cup R_N) \geq \sigma$ }
7. *FrequentSets* = { $X \in$ *FrequentSets* | $s(X,R_O \cup R_N) \geq \sigma$ } $\cup$ *PromotedBorders*
8. *Borders* = { X | $\forall x \in X$, $X-\{x\} \in$ *FrequentSets* }
10. m = max { i | *PromotedBorders*$(i) \neq \emptyset$ }
11. $L_0 = \emptyset$, $i = 1$
12. **While** ($L_i \neq \emptyset$ or $i \leq m$) **do**
13. C_{i+1} = {$X = S_1 \cup S_2$ | (i) | X | = $i + 1$,
14. (ii) $\exists x \in X$, $X-\{x\} \in$ *PromotedBorders*$(i) \cup L_i$,
15. (iii) $\forall x \in X$, $X-\{x\} \in$ *FrequentSets*$(i) \cup L_i$ }
16. Scan $R_N \cup R_O$ and obtain $c(X, R_N \cup R_O)$ for all candidates, $X \in C_{i+1}$.
17. L_{i+1}={X | $X \in C_{i+1}$ and $c(X, R_N \cup R_O)/(n_O+n_N) \geq \sigma$}
18. *FrequentSets* = *FrequentSets* $\cup L_{i+1}$
19. *Borders* = *Borders* $\cup (C_{i+1} - L_{i+1})$
20. $i=i+1$
21. **end do**

Figure 2. The Borders algorithm-additions.

old data—R_O. We assume that for each border or frequent set X in R_O, the count $c(X, R_O)$ is already known from the previous stage. (We may assume starting from the empty set, with $\emptyset$ as the only frequent set). The Borders algorithm starts by scanning the new relation R_N and updating the counters of all frequent sets and border sets (lines 1–5). The new support of a set X is its count divided by the new total size. Note that since the size of the relation is now larger, some previous frequent sets may not be frequent any longer. Thus, the new frequent sets and border sets are determined (lines 7–8). Beforehand, the set of *promoted borders* is determined. A border set X (of R_O) is said to be *promoted border* if after the increment R_N its support reaches the minimum support threshold, σ (and hence became frequent sets). Next, the candidates are generated (lines 12–15). Their count is obtained by scanning the entire relation (line 16). Based on the count, the new frequent sets and border sets are determined (lines 17–19).

Candidate generation and counting works in a sequence of rounds, where in round i candidates of size i are generated and checked. The candidates of size $i+1$, denoted by C_{i+1}, are generated based on the new frequent sets of size i (L_i), the promoted borders of size i, and the old frequent sets of size i. The notation *PromotedBorders(i)* denotes the promoted borders of size i. Similarly, *FrequentSets*(i), denotes the frequent sets of size i. The procedure is based on the fact, which we prove next, that a set need be considered as a candidate only if it has a subset that is a promoted border.

Lemma. *Let X be a set of attributes that is a frequent set in $R_N \cup R_O$ but not in R_O. Then there exists a subset $Y \subseteq X$ such that Y is a promoted border.*

Proof: Let Y be a minimal cardinality subset of X that is a frequent set in $R_N \cup R_O$ but not in R_O. Since Y is a frequent set in $R_N \cup R_O$, so are all of its proper subsets. However, by the minimality of Y, none of these subsets is a new frequent set in $R_N \cup R_O$. Thus, Y is a border set in R_O, and $Y \subseteq X$ as claimed. □

Theorem. *The Borders algorithm is correct.*

Proof: We prove that for each update phase, if prior to the beginning of the phase the sets *FrequentSets* and *Borders* consist of all frequent sets and borders of R_O, respectively, and the count for each such set is correct, then the same holds after the completion of the update, with respect to $R_N \cup R_O$.

Let X be a frequent set in $R_N \cup R_O$. There are three possible cases.

1) X was a frequent set in R_O. Then, its count is updated with the extra amount $c(X, R_N)$ (lines 1–5), and it is found to be above threshold (line 7).
2) X was a border in R_O, then its count is updated (line 1–5). It is found to be a promoted border (line 6) and added to the frequent sets (line 7).
3) X was neither a frequent set nor a border in R_O, but is a frequent set in $R_N \cup R_O$. We prove that in this case $X \in L_i$ where i is the size of X. The proof is by induction on i. For $i = 0$ there is nothing to prove since all empty sets are frequent sets (assuming the entire relation has minimum support). Assume for i we prove for $i + 1$. Consider the subsets

of X of size i. Since X is a frequent-set in $R_N \cup R_O$, all its subsets of size i must also be frequent sets in $R_N \cup R_O$. Let $B(X)$ be the subsets of size i of X which were borders in R_O. Similarly, let $F(X)$ be the subsets of size i of X which were frequent sets in R_O. Let $L(X)$ be the remaining subsets of X of size i. Since all elements of $L(X)$ are frequent sets, but not borders or frequent sets in R_O, by the inductive hypothesis $L(X) \subseteq L_i$. Thus, all subsets of X of size i are in $\mathit{FrequentSets}(i) \cup L_i$ (line 15). Finally, if $B(X) \cup L(X) = \emptyset$ then X was a border in R_O, in contradiction to the assumption. Thus, at least one of the subsets of size i of X is in $\mathit{PromotedBorders}(i) \cup L_i$ (line 14). By definition, $|X| = i + 1$ (line 13). Thus, $X \in C_{i+1}$. Thus, its count is determined during the scan of the relation (line 16). It is found to pass the threshold (line 17), and is added to the frequent sets (line 18).

Next, suppose X is a border set in $R_N \cup R_O$. The proof is very similar to that for frequent sets. Again, there are three possible cases.

1) X was a border set in R_O. Then, its count is updated with the extra amount $c(X, R_N)$ (lines 1–5), and it is found to be a border in the new relation (line 8).
2) X was a frequent set in R_O. Then its count is updated (line 1–5). It is found not to be a frequent set in $R_N \cup R_O$ (line 7), but yes a border (line 8).
3) X was neither a frequent set nor a border in R_O, but is border in $R_N \cup R_O$. We prove that in this case $X \in C_{i+1}$ where $i + 1$ is the size of X. Consider the subsets of X of size i. Since X is a border set in $R_N \cup R_O$, all its subsets of size i must be frequent sets in $R_N \cup R_O$. Define $B(X)$, $F(X)$, and $L(X)$ as above. Since all elements of $L(X)$ are frequent sets, but not borders or frequent sets in R_O, we have already shown above that $L(X) \subseteq L_i$. Thus, all subsets of X of size i are in $\mathit{FrequentSets}(i) \subseteq L_i$ (line 15). Similarly, if $B(X) \cup L(X) = \emptyset$ then X was a border in R_O, in contradiction to the assumption. Thus, at least one of the subsets of size i of X is in $B(i) \cup L_i$ (line 14). By definition, $|X| = i + 1$ (line 13). Thus, $X \in C_{i+1}$. Thus, its count is found during the scan of the relation (line 16), it is found not to pass the threshold (line 17), and is added to the border sets (line 19). □

3.1. Deletions

In some cases, rows may also be deleted from the database. This might be due to expiration date, or to errors found in these records. The Borders algorithm can be extended to handle the case of deletions as well. To see this we observe that deletions have two effects on the frequent sets:

1. The count of the attribute sets supported by the deleted rows is decreased. Thus, frequent sets may cease to be frequent after the deletion. These sets can easily be determined by scanning the deleted portion R_D alone.
2. The overall size of the new relation is reduced. Since the support threshold is determined as a *percentage* of the entire relation, the deletion causes a decrease in the *absolute* count necessary for a set to be frequent. Thus, new frequent sets may emerge. In this case, however, it is enough to monitor the border sets to discover all such new frequent sets.

Input: R_O , R_N, and R_D, σ, *Borders* and *FrequentSets* (of R_O) and their count
Output: *Borders* and *FrequentSets* of $R=(R_O - R_D)\cup R_N$ and their count
1. Scan deleted relation R_D and find count, $c(X,R_D)$, for all $X\in$ *Borders*$\cup$*FrequentSets*.
2. Scan added relation R_N and find count, $c(X,R_N)$, for all $X\in$ *Borders*$\cup$*FrequentSets*.
3. **forall** $X \in$ *Borders* $\cup$ *FrequentSets* **do**
4. $c(X,R) = c(X,R_O)-c(X,R_D)+ c(X,R_N)$
5. $s(X,R) = c(X,R)/(n_O-n_D+n_N)$
6. **end do**
7. $PromotedBorders = \{ X \in Borders \mid s(X,R) \geq \sigma \}$
8. $FrequentSets = \{ X \in FrequentSets \mid s(X,R) \geq \sigma \} \cup PromotedBorders$
9. $Borders = \{ X \mid \forall x\in X,\ X-\{x\}\in FrequentSets \}$
10. $m = \max \{ i \mid PromotedBorders(i) \neq \emptyset \}$
11. $L_0 = \emptyset$, $i = 1$
12. **While** ($L_i \neq \emptyset$ or $i \leq m$) **do**
13. $C_{i+1} = \{X = S_1\cup S_2 \mid$ (i) $\mid X \mid = i + 1$,
14. (ii) $\exists\, x \in X$, $X-\{x\}\in PromotedBorders(i) \cup L_i$,
15. (iii) $\forall\, x\in X$, $X-\{x\}\in FrequentSets(i) \cup L_i$ $\}$
16. Scan R and obtain $c(X, R)$ for all candidates, $X \in C_{i+1}$.
17. $L_{i+1}=\{X \mid X\in C_{i+1}$ and $c(X, R)/(n_O+n_N-n_D) \geq \sigma\}$
18. $FrequentSets = FrequentSets \cup L_{i+1}$
19. $Borders = Borders \cup (C_{i+1} - L_{i+1})$
20. $i=i+1$
21. **end do**

Figure 3. The Borders algorithm-addition and deletion.

The full algorithm for deletions and additions combined is provided in figure 3. The correctness proof is essentially identical to that of the addition-alone version and is omitted.

3.2. Changing the threshold

Suppose that frequent sets where generated using a threshold σ, and that we are interested in changing to a different threshold σ'. If $\sigma' > \sigma$ (i.e., the threshold is increased) then the task is easy. All we have to do is to delete the frequent sets that do not pass the new threshold. However, if $\sigma' < \sigma$ (i.e., the threshold is decreased), then new frequent sets may emerge. None of the previously known algorithms provides a solution for this case. Rather, the frequent sets must be regenerated from scratch for the new threshold. *Borders* handles this case with the same ease as deletion. The crucial point, again, is that any new frequent set must have a subset that is a promoted border. In this case, a *promoted border* is a set that was a border using the old threshold σ, but is a frequent set using the new threshold σ'. A full description of the algorithm for handling threshold decrease is provided in figure 4.

4. Experimental evaluation

The performance of the algorithm was evaluated on the Medline database from the years 1989–90. The Medline database consists of a collection of short medical abstracts. Initially

Input: R, old threshold - σ, new threshold $\sigma' < \sigma$, *Borders* and *FrequentSets* (of R) according to the threshold σ, and their count
Output: *Borders* and *FrequentSets* of R according to the threshold σ, and their count
1. $PromotedBorders = \{ X \in Borders \mid s(X,R) \geq \sigma' \}$
2. $FrequentSets = FrequentSets \cup PromotedBorders$
3. $Borders = \{ X \mid \forall x \in X,\ X-\{x\} \in FrequentSets \}$
4. $m = \max \{ i \mid PromotedBorders(i) \neq \emptyset \}$
5. $L_0 = \emptyset$, $i = 1$
6. **While** ($L_i \neq \emptyset$ or $i \leq m$) **do**
7. $C_{i+1} = \{X = S_1 \cup S_2 \mid$ (i) $\mid X \mid = i + 1$,
8. (ii) $\exists\, x \in X$, $X-\{x\} \in PromotedBorders(i) \cup L_i$,
9. (iii) $\forall\, x \in X$, $X-\{x\} \in FrequentSets(i) \cup L_i$ $\}$
10. Scan R and obtain $c(X, R)$ for all candidates, $X \in C_{i+1}$.
11. $L_{i+1} = \{X \mid X \in C_{i+1}$ and $c(X, R)/n \geq \sigma'\}$
12. $FrequentSets = FrequentSets \cup L_{i+1}$
13. $Borders = Borders \cup (C_{i+1} - L_{i+1})$
14. $i = i+1$
15. end do

Figure 4. The Borders algorithm-decreasing the threshold.

term extraction was performed on each of the abstracts. The most representative terms (in *tfidf* metric) were then chosen to represent the abstract (see (Feldman et al., 1998) for details). Thus, each abstract was associated with a set of *terms*. These terms correspond to the attributes in the standard definition.

We performed over 200 experiments, evaluating the performance of the Borders algorithm over a wide range of parameters. We also compared the performance of the Borders to that of the FUP algorithm (Cheung et al., 1996). The results are depicted in figures 5–9, and discussed hereunder.

In each of the experiments we fixed an initial database, consisting of 20,000 rows. The initial set was taken from the 1989 portion of the database. The frequent sets and borders for this base-set were computed separately. We then added an *increment set*, taken from the 1990 portion of the database, and measured the performance of the algorithms when producing the frequent set for the combined data set.

In our first experiment, we measured the performance of the algorithms with varying increment sizes. The support threshold was set to 5%. We varied the increment size from 500 records (2.5% the size of the original set) to 5000 records (25%), in increments of 500. The average times are depicted in figure 5 (each point is averaged over ten experiments). The Borders algorithm exhibits improved performance throughout the entire range. The performance gain ranges from over 50-fold for the small increments (2.5%), to 3–4 fold for very large (10–25%) increments (see figure 6).

The main advantage of Borders algorithm is due to the small number of times the entire database (old and new) must be scanned. Figure 7 depicts the average number of full database passes each of the algorithms required for the different increment sizes. (The

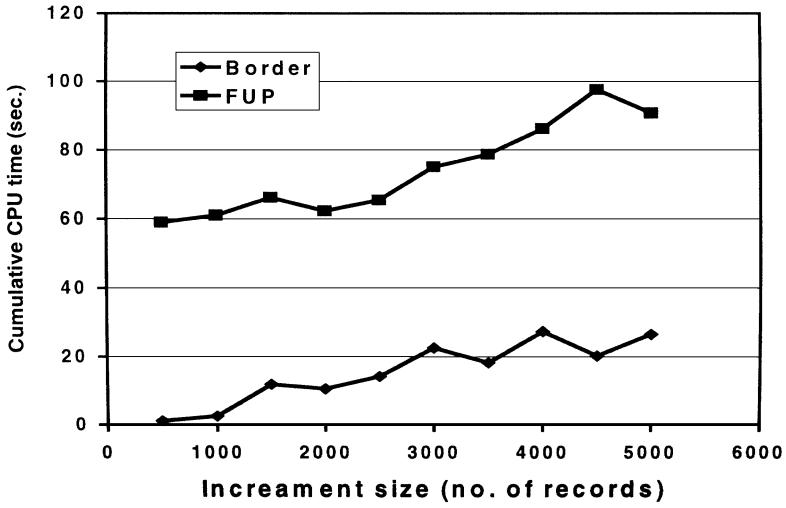

Figure 5. Update time.

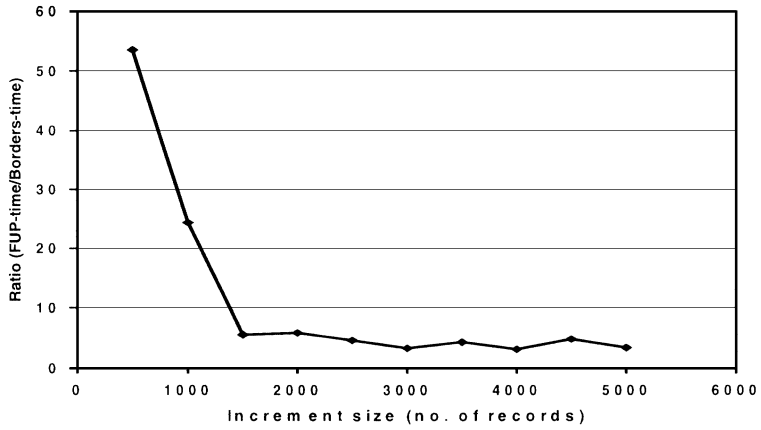

Figure 6. Ratio of FUP time to Borders time.

somewhat rugged shape of the curve results from the statistical variations in the data sets.) Throughout, Borders performs fewer passes than FUP. It is also interesting to note the number of times that it was not necessary to access the old data-set at all. In 79 out of the total of 100 cases considered in the experiment, Borders required *no* access to the old database. In these cases, the update was almost instantaneous. Using the FUP algorithm, in contrast, in only 3 of the 100 cases was it possible not to access the old database.

Next, we examined the affect of varying the support threshold. Using an increment of 1000 (5% of the original data set) we varied the support threshold from 1% to 10%. Average times for ten such trails are depicted in figure 8. As the graph shows, Borders outperforms

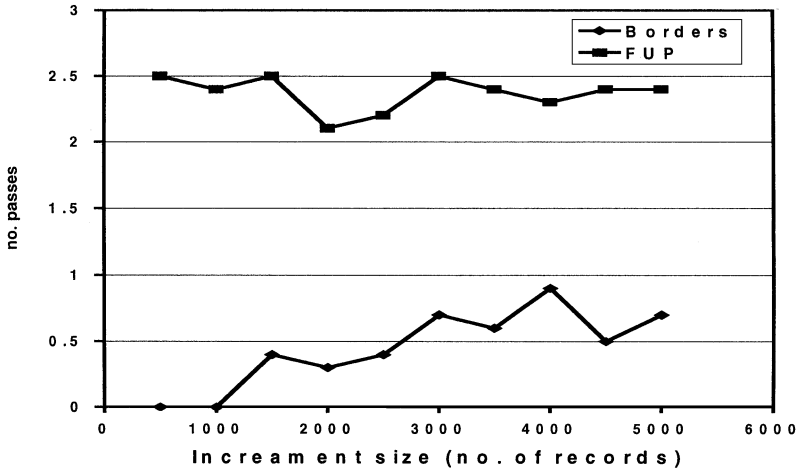

Figure 7. Number of passes; FUP verses Borders.

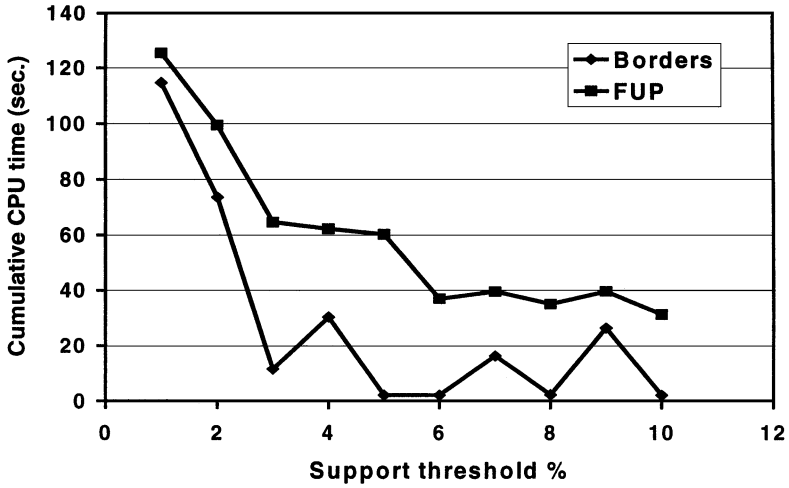

Figure 8. Varying the support threshold.

FUP throughout the entire range. At very small support thresholds (1%) the difference is nominal, but increases with higher thresholds.

Again, the main advantage of Borders stems from the small number of database passes it requires. Figure 9 depicts the average number of passes in the above experiments. The results show that throughout the range the number of passes FUP requires is never less than that of Borders. A detailed analysis proves that this is necessarily so. We can prove that for *any* update session, Borders will never require more database passes than Borders. Furthermore, any candidate in Borders will also be a candidate in FUP (but not necessarily the other way around). The full details of the proof are out of the scope of this paper.

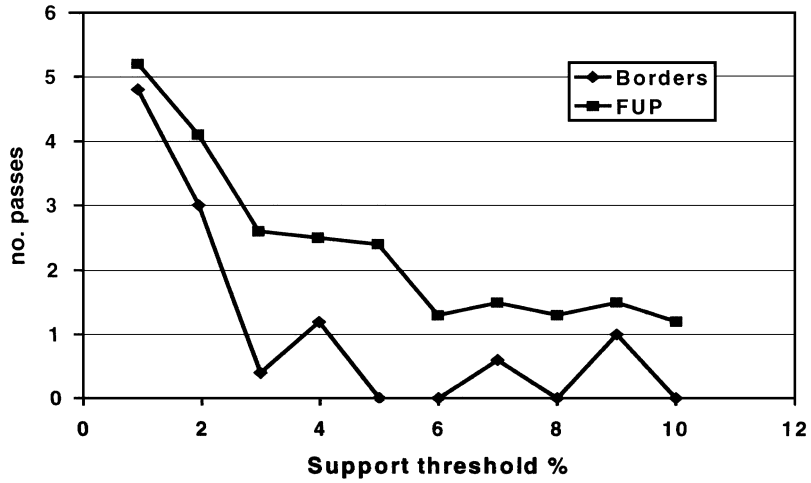

Figure 9. Average number of passes.

5. Discussion

This paper considers the problem of generating association rules in a dynamical setting, where data is constantly being added and/or deleted to and from the database. In this case, at any given time, all associations rules based on the *current* set of data are of interest. We argue that while the incremental arrival of data is common in many applications, most previous algorithms for generating association rules are ill-suited for this setting. In particular, most previous algorithms lack the ability to make use of computations made on the old data to facilitate fast update, and require running the algorithm anew following the arrival of each new set of data. For large data sets, this is prohibitively time-consuming.

In this paper we focus on the task of finding the frequent sets in the data set, which is the main task in generating the association rules. We presented a new algorithm, the Borders algorithm, for fats generation of frequent sets in the dynamical setting. We presented three variants of the algorithm:

1. an algorithm for the case of additions only,
2. an algorithm for additions and deletions, and
3. an algorithm for the case of changing the threshold. (No algorithm for this case was previously known).

All three algorithms are based on the notion of border sets, and on the observation that monitoring the border sets suffices to track any changes in the frequent sets. The three algorithms testify to the usefulness of the border notion in the dynamical setting.

Using the Medline collection as a test bed, we demonstrated the efficiency of the Borders algorithm, and compared its performance to that of the FUP algorithm (Cheung et al., 1996). We show that a large range of parameters (minimum support, update size) the Borders

algorithm outperforms FUP, sometimes by more than an order of magnitude. Furthermore, in most cases (over 75%), Borders requires no access at all to the old data. Thus, in most cases, increments are almost instantaneous.

The algorithms presented in this paper provide a basic tool to allow data mining in the dynamical setting. However, many issues relating this setting call for further research. Among these we mention:

1. Extending the algorithms to relations over a non-binary or continuous domains (see Srikant and Agrawal, 1996). In the case of a continuous domain the problem is that partition of the domain into discrete intervals is in itself dependent on the existing data (Srikant and Agrawal, 1996), and may change over time.
2. Efficient generation of temporal association rules in an incremental setting (see Mannila et al., 1995).

Also, with the introduction of *time* into the system, we may wish to give new data more significance than old data. This introduces the notion of data mining over a *dynamic weighted relation*, were each row has a weight affiliated with it, and the weights vary over time.

Acknowledgments

This research was supported by grant # 8615 from the Israeli Ministry of Sciences.

References

Agrawal, A., Imielinski, T., and Swami, A. (1993). Mining association rules between sets of items in large databases. In *Proc. of the ACM SIGMOD Conference on Management of Data* (pp. 207–216).

Agrawal, R., Mannila, H., Srikant, R., Toivonen, H., and Verkamo, I. (1995). Fast Discovery of Association Rules. In U. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy (Eds.), *Advances in Knowledge Discovery and Data Mining* (pp. 307–328). AAAI Press.

Agrawal, A. and Srikant, R. (1994). Fast algorithms for mining association rules. In *Proceedings of the VLDB Conference*. Santiago, Chile.

Agrawal, R. and Srikant, R. (1995). Mining sequential patterns. *Proc. of the Int'l Conference on Data Engineering (ICDE)*. Taipei, Taiwan.

Cheung, D.W., Han, J., Ng, V., and Wong, C.Y. (1996). Maintenance of discovered association rules in large databases: An incremental updating techniques. *Proc. 12th IEEE International Conference on Data Engineering (ICDE-96)*. New Orleans, Louisiana, U.S.A.

Cheung, D.W., Lee, S.D., and Kao, B. (1997). A general incremental technique for updating discovered association rules. *Proc. International Conference On Database Systems for Advanced Applications (DASFAA-97)*. Melbourne, Australia.

Fayyad, U., Piatetsky-Shapiro, G., Smyth, P., and Uthurusamy, R. (1995). *Advances in Knowledge Discovery and Data Mining*, AAAI Press.

Feldman, R., Amir, A., Aumann, Y., Zilberstein, A., and Hirsh, H. (1997). Incremental algorithms for association generation. In *Proceedings of the 1st Pacific Asia Conference on Knowledge Discovery and Data Mining (PAKDD97)*. Singapore.

Feldman, R., Fresko, M., Kinar, Y., Liphstat, O., Schler, Y., Rajman, M., and Zamir, O. (1998). Text Mining at the Term Level. Department of computer science technical report. Bar Ilan University.

Klemettinen, M., Mannila, H., Ronkainen, P., Toivonen, H., and Verkamo, A. (1994). Finding interesting rules from large sets of discovered association rules. In *Proceedings of the 3rd International Conference on Information and Knowledge Management*.

Mannila, H. and Toivonen, H. (1997). Levelwise Search and Borders of Theories in Knowledge Discovery, *Data Mining and Knowledge Discovery*, 1(3), 241–258.

Mannila, H., Toivonen, H., and Verkamo, A. (1994). Efficient algorithms for discovering association rules. In *Proceedings of KDD94: AAAI Workshop on Knowledge Discovery in Databases* (pp. 181–192).

Savasere, A., Omiecinski, E., Navathe, S. (1995). An efficient algorithm for mining association rules in large databases. In *Proceedings of the 21st VLDB Conference.* Zurich, Switzerland.

Srikant, R. and Agrawal, R. (1996). Mining quantitative association rules in large relational tables. In *Proceedings of the ACM SIGMOD Conference on Management of Data*. Montreal, Canada.