Further Pruning for Efficient Association Rule Discovery

Songmao Zhang and Geoffrey I. Webb

School of Computing and Mathematics, Deakin University
Geelong, Victoria 3217, Australia

Abstract. The Apriori algorithm's frequent itemset approach has become the standard approach to discovering association rules. However, the computation requirements of the frequent itemset approach are infeasible for dense data and the approach is unable to discover infrequent associations. OPUS_AR is an efficient algorithm for association rule discovery that does not utilize frequent itemsets and hence avoids these problems. It can reduce search time by using additional constraints on the search space as well as constraints on itemset frequency. However, the effectiveness of the pruning rules used during search will determine the efficiency of its search. This paper presents and analyses pruning rules for use with OPUS_AR. We demonstrate that application of OPUS_AR is feasible for a number of datasets for which application of the frequent itemset approach is infeasible and that the new pruning rules can reduce compute time by more than 40%.

Keywords: machine learning, search.

1 Introduction

Association rule discovery has been dominated by the frequent itemset strategy as exemplified by the Apriori algorithm [2]. OPUS_AR utilizes an alternative association rule discovery strategy to find associations without first finding frequent itemsets [16]. This avoids the need to retain the set of frequent itemsets in memory, a requirement that makes the frequent itemset strategy infeasible for dense data [4]. This paper presents and evaluates pruning rules and other strategies that improve the computational efficiency of OPUS_AR.

We characterize the association rule discovery task as follows.

- A *dataset* is a finite set of *records* where each record is an element to which we apply Boolean predicates called *conditions*.
- An *itemset* is a set of conditions. The name *itemset* derives from association rule discovery's origins in market basket analysis where each condition denotes the presence of an item in a market basket.
- $coverset(I)$ denotes the set of records from a dataset that satisfy itemset I.
- An *association rule* consists of two conjunctions of conditions called the *antecedent* and *consequent* and associated statistics describing the frequency with which the two co-occur within the dataset. An association rule with antecedent A, consequent C, and statistics S is denoted as $A \rightarrow C[S]$.

The task involves finding all association rules that satisfy a set of user defined constraints with respect to a given dataset.

The frequent itemset strategy has become the standard approach to association rule discovery. This strategy first discovers all *frequent itemsets*. A frequent itemset is an itemset whose support exceeds a user defined threshold. The association rules are then generated from the frequent itemsets. If there are relatively few frequent itemsets this approach can be very efficient. However, it is subject to a number of limitations.

1. The user is required to nominate a minimum frequency. Associations with support lower than this frequency will not be discovered. For some applications there may not be any natural lower bound on support and hence pruning the search space on minimum frequency in this manner may not be appropriate. Also, for some applications infrequent itemsets may actually be especially interesting. For example, especially high value transactions are likely to be both relatively infrequent and of high interest. This is known as the *vodka and caviar problem*.
2. Even when a minimum frequency is applicable, there may be too many frequent itemsets for computation to be feasible. The frequent itemset approach requires that all frequent itemsets be maintained in memory. This imposes unrealistic memory requirements for many applications [4].
3. It is difficult to utilize search constraints other than minimum frequency to improve the efficiency of the frequent itemset approach. Where other constraints can be specified, potential efficiencies are lost.

Most research in association rule discovery has sought to improve the efficiency of the frequent itemset discovery process [1,9, for example]. This has not addressed any of the above problems, except the closed itemset approaches [11,17], which reduce the number of itemsets required, addressing point 2, but not 1 or 3.

OPUS_AR provides an alternative approach to association rule discovery based on the efficient OPUS search algorithm [15]. This extends previous work in rule discovery search [5,8,10,12,13,14,15] by searching for rules that optimize an objective function over a space of rules that allows alternative variables in the consequent. Previous algorithms have all been restricted to a single target consequent variable per search.

OPUS_AR does not have significant memory requirements other than the requirement that all data be retained in memory. While it does not achieve the same degree of pruning as Apriori from a constraint on minimum frequency, it can utilize other constraints more effectively than Apriori. In particular, it can utilize constraints on the number of associations to be discovered, returning the n associations that optimize some criterion of interestingness. This provides a desirable contrast to the frequent itemset approach that is prone to generate extraordinarily large numbers of associations. In practice, only a small number associations are likely to be utilized by a user. A large number of associations is more likely to be a hindrance than an asset.

Search space pruning rules are critical to the efficiency of OPUS_AR. Webb [16] utilized four such pruning rules. This paper presents two new pruning rules and additional mechanisms for reducing the computational requirements of OPUS_AR.

This paper is organised as follows. Section 2 introduces the Apriori algorithm and analyzes its advantages and disadvantages. Section 3 introduces the OPUS search algorithm on which OPUS_AR is based. Section 4 presents the OPUS_AR algorithm for discovering association rules. Section 5 describes the new pruning rules and other efficiency measures and presents experiments that demonstrate the effectiveness of these measures when discovering association rules on several large datasets. Section 6 presents conclusions.

2 The Apriori Algorithm

The Apriori algorithm discovers associations in a two-step process. First, it finds the frequent itemsets $\{I \subseteq \mathcal{C} : \frac{|coverset(I)|}{|\mathcal{D}|} \geq min\_support\}$, where $\mathcal{C}$ is the set of all available conditions, $\mathcal{D}$ is the dataset, and *min_support* is a user defined minimum support constraint. In the second stage the frequent itemsets are used to generate

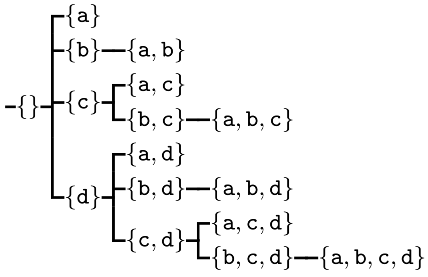

Fig. 1. A fixed-structure search space

the association rules. The minimum support constraint on the frequent itemsets guarantees that all associations generated will satisfy the minimum support constraint. Other constraints, such as minimum confidence are enforced during the second stage.

The frequent itemset strategy can limit the number of rules that are explored, and cache the support values of the frequent items so that there is no need to access the dataset in the second step. It is very successful at reducing the number of passes through the data. The frequent itemset approach has become the predominant approach to association rule discovery.

However, the frequent itemset approach is only feasible for sparse data. For dense datasets where there are numerous frequent itemsets, the overheads for maintaining and manipulating the itemsets are too large to make the system efficient and feasible [4]. This is also apparent in the experiments presented below. Dense datasets are common in applications other than basket data analysis or when basket data is augmented by other customer information. Another problem of Apriori is that it lists numerous association rules to the user and it may be very difficult for the user to identify the interesting rules manually. Take the covtype dataset for example. Covtype has 581,012 records containing 125 items. The number of the association rules generated by Apriori with the minimum support set to 0.01, minimum confidence 0.8, and maximum itemset size 5 is 88,327,710. Since the Apriori algorithm generates itemsets by considering features of itemsets in isolation, the inter-relationships between the itemsets are not taken into account. In consequence, many association rules generated may not be of interest to the user.

3 The OPUS Search Algorithm

OPUS [15] provides efficient search for subset selection, such as selecting a subset of available conditions that optimizes a specified measure. It was developed for classification rule discovery. Previous algorithms ordered the available conditions and then conducted a systematic search over the ordering in such a manner as to guarantee that each subset was investigated once only, as illustrated in Fig. 1.

Critical to the efficiency of such search is the ability to identify and prune sections of the search space that cannot contain solutions. This is usually achieved by identifying subsets that cannot appear in a solution. For example, it might be determined that $\{b\}$ cannot appear in a solution in the search space illustrated in Fig. 1. Under previous search algorithms [8,10,12,13,14], subsets that appear below such a subset were pruned, as illustrated in Fig. 2. In this example, pruning removes one subset from the search space.

This contrasts with the pruning that would occur if all subsets containing the pruned subset were removed from the search space, as illustrated in Fig. 3. This optimal pruning almost halves the search space below the parent node.

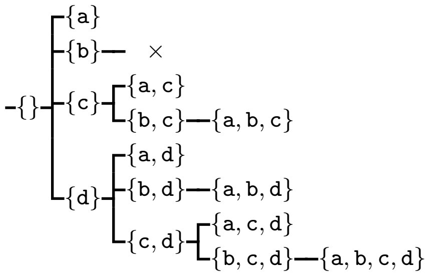

Fig. 2. Pruning a branch from a fixed-structure search space

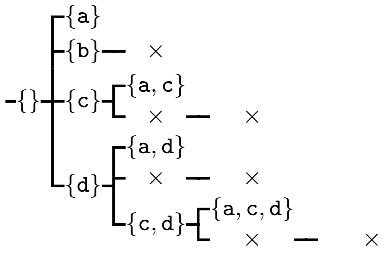

Fig. 3. Pruning all nodes containing a single condition from a fixed-structure search space

OPUS achieves the pruning illustrated in Fig. 3 by maintaining a set of available items at each node in the search space. When adding an item i to the current subset s results in a subset $s \cup \{i\}$ that can be pruned from the search space, i is simply removed from the set of available items at s which is propagated below s. As supersets of $s \cup \{i\}$ below s can only be explored after $s \cup \{i\}$, this simple mechanism with negligible computational overheads guarantees that no superset of a pruned subset will be generated in the search space below the parent of the pruned node. This greatly expands the scope of a pruning operation from that achieved by previous algorithms which only extended to the space below the pruned node. Further pruning can be achieved by reordering the search space, but this proves to be infeasible in search for association rule discovery, as explained by Webb [16].

4 The OPUS_AR Algorithm

OPUS_AR extends the OPUS search algorithm to association rule discovery [16]. To simplify the search problem, the consequent of an association rule is restricted to a single condition. Association rules of this restricted form are of interest for many data mining applications.

Whereas OPUS supports search through spaces of subsets, the association rule search task requires search through the space of pairs $\langle I \subseteq \mathit{conditions}, c \in \mathit{conditions} \rangle$, where I is the antecedent and c the consequent of an association. OPUS_AR achieves this by performing OPUS search through the space of antecedents, maintaining at each node a set of potential consequents, each of which is explored at each node.

The algorithm relies upon there being a set of user defined constraints on the acceptable associations. These are used to prune the search space. Such constraints can take many forms, ranging from the traditional association rule discovery constraints on support and confidence to a constraint that only the n associations that maximize some statistic be returned. To provide a general mechanism for handling a wide variety of constraints, we denote associations that satisfy all constraints *target associations*. Note that it may not be apparent when an association is encountered whether or not it is a target. For example, if we are seeking the 100 associations with the highest lift, we may not know the cutoff value for lift until the search has been completed. Hence, while we may be able to determine in some circumstances that an association is not a target, we may not be able to determine that an association is a target until the search is completed. To accommodate this, pruning is only invoked when it is determined that areas of the search space cannot contain a target. All associations encountered are recorded unless the system can determine that they are not targets. However, these associations may be subsequently discarded as progress through the search space reveals that they cannot be targets. When seeking the n best associations with respect to some statistic, we can determine that a new association is not a target if its value on that statistic is lower than the value of the n^{th} best recorded so far, as the value of the n^{th} best for the search space cannot be lower than the value of the n^{th} best for the subset of the search space examined so far.

Table 1 displays the algorithm that results from applying the OPUS search algorithm [15] to obtain efficient search for this search task. The algorithm is presented as a recursive procedure with three arguments:

CurrentLHS: the set of conditions in the antecedent of the rule currently being considered.

AvailableLHS: the set of conditions that may be added to the antecedent of rules to be explored below this point.

AvailableRHS: the set of conditions that may appear on the consequent of a rule in the search space at this point and below.

Table 1. The OPUS search algorithm adjusted for search for association rules

```
Algorithm: OPUS AR (CurrentLHS,AvailableLHS,AvailableRHS)

1. SoFar := {}
2. FOR EACH P in AvailableLHS
2. 1 IF pruning rules cannot determine that $\forall x \subseteq$ AvailableLHS: $\forall y \in$
   AvaiableRHS: $\neg$target (x $\cup$ CurrentLHS $\cup$ {P} $\rightarrow$ y) THEN
2. 1.1 NewLHS := CurrentLHS $\cup$ {P}
2. 1.2 NewAvailableLHS := SoFar - P
2. 1.3 IF pruning rules cannot determine that $\forall x \subseteq$ NewAvailableLHS: $\forall y \in$
   AvailableRHS: $\neg$target (x $\cup$ NewLHS $\rightarrow$ y) THEN
  (a) NewAvailableRHS := AvailableRHS - P
  (b) IF pruning rules cannot determine $\forall y \in$ NewAvailableRHS: $\neg$target
      (NewLHS $\rightarrow$ y) THEN
  (b. 1) FOR EACH Q in NewAvailableRHS
        i. IF pruning rules determine that $\forall x \subseteq$ NewAvailableLHS: $\neg$target (x
           $\cup$ NewLHS $\rightarrow$ Q) THEN
            A. NewAvailableRHS := NewAvailableRHS - Q
       ii. ELSE IF pruning rules cannot determine that $\neg$target (NewLHS $\rightarrow$ Q)
           THEN
            A. IF target (NewLHS $\rightarrow$ Q) THEN
               A.1 record NewLHS $\rightarrow$ Q
               A.2 tune the settings of the statistics
            B. IF pruning rules determine that $\forall x \subseteq$ NewAvailableLHS: $\neg$target
               (x $\cup$ NewLHS $\rightarrow$ Q) THEN
               NewAvailableRHS := NewAvailableRHS - Q
  (c) IF NewAvailableLHS $\neq$ {} and NewAvailableRHS $\neq$ {} THEN
          OPUS AR (NewLHS,NewAvailableLHS,NewAvailableRHS)
  (d) SoFar := SoFar $\cup$ {P}
```

The initial call to the procedure sets CurrentLHS to {}, and AvailableLHS and AvailableRHS to the set of conditions that are to be considered on the antecedent and consequent of association rules, respectively.

The algorithm OPUS_AR is a search procedure that starts with the associations with one condition in the antecedent and searches through successive associations formed by adding conditions to the antecedent. It loops through each condition in AvailableLHS, adds it to CurrentLHS to form the NewLHS. For the NewLHS, it loops through each condition in AvailableRHS to check if it could be the consequent for NewLHS. After the AvailableRHS loop, the procedure is recursively called with the arguments NewLHS, NewAvailableLHS and NewAvailableRHS. The two latter arguments are formed by removing the pruned conditions from AvailabeLHS and AvailableRHS, respectively. Step 2.1.3(b.1)ii.A.1 records the potential target associations.

5 Pruning in Search for Association Rules

Webb [16] utilized four pruning rules to prune the search space explored by OPUS_AR. We present two new pruning rules and two data access saving rules for improving the efficiency of OPUS_AR. In order to evaluate their impact, experiments are performed on five large datasets from the UCI ML and KDD repositories [6,3]. These datasets are listed in Table 2.

The four pruning rules presented in Webb [16] are taken as the basic pruning rules in our experiments. Column "basic pruning" of Table 3 lists the times of running OPUS_AR with these basic pruning rules on the five datasets. We test on the same datasets the running times of OPUS_AR with the basic pruning plus each of the pruning mechanisms introduced below. We also compare the performance with the publicly available apriori system developed by Borgelt [7]. In all the experiments OPUS_AR seeks the top 1000 associations on lift within the constraints of minimum confidence set to 0.8, minimum support set to 0.01, and the maximum number of conditions in antecedent of an association set to 4. The same minimum support, minimum confidence, and maximum antecedent size are used for Apriori, thus the maximum itemset size is 5 for Apriori because itemsets are required that contain up to 4 antecedent conditions as well as the single consequent condition. The experiments were performed on a Linux server with 2 CPUs each 933MHz in speed, 1.5G RAM, and 4G virtual memory.

5.1 Formal Description of Association Rule Discovery Based on OPUS_AR

A formal description of association rule discovery based on OPUS_AR is given in the following.

Definition 1. *An association rule discovery task based on OPUS_AR (abbreviated as AR_by_OPUS) is a 4-tuple* $(\mathcal{C}, \mathcal{D}, \mathcal{A}, \mathcal{M})$, *where*
$\mathcal{C}$: *nonempty set of conditions;*
$\mathcal{D}$:*nonempty set of records, called the dataset, where for each record* $d \in \mathcal{D}$, $d \subseteq \mathcal{C}$. *For any* $S \subseteq \mathcal{C}$, *let* $coverset(S) = \{d | d \in \mathcal{D} \wedge S \subseteq d\}$, *and let* $cover(S) = \frac{|coverset(S)|}{|\mathcal{D}|}$;
$\mathcal{A}$: *set of association rules, where each association rule takes the form*

$$X \rightarrow Y[coverage, support, confidence, lift]$$

where $X \subset \mathcal{C}$, $X \neq \emptyset$, $Y \subset \mathcal{C}$, $|Y| = 1$, $X \cap Y = \emptyset$, *and* $coverage, support, confidence,$ *and* $lift$ *are statistics for the association rule,*

Table 2. Datasets for experiments

name	**records**	**attributes**	**values**
covtype	581012	55	125
ipums.la.99	88443	61	1883
ticdata2000	5822	86	709
connect-4	67557	43	129
letter-recognition	20000	17	74

satisfying $coverage(X \rightarrow Y) = cover(X)$, $support(X \rightarrow Y) = cover(X \cup Y)$, $confidence(X \rightarrow Y) = \frac{support(X \rightarrow Y)}{coverage(X \rightarrow Y)}$, *and* $lift(X \rightarrow Y) = \frac{confidence(X \rightarrow Y)}{cover(Y)}$;
$\mathcal{M}$: *constraints, composed of* $maxAssocs$ *denoting the maximum number of target association rules (which will consist of the association rules with the high-*

est values for lift of those that satisfy all other constraints), $maxLHSsize$ denoting maximum number of conditions allowed in the antecedent of association rule, $minCoverage$ denoting the minimum coverage, $minSupport$ denoting the minimum support, $minConfidence$ denoting the minimum confidence, and $minLift = \max(1.0, \beta(RS, maxAssocs))$, where RS is the set of associations $\{R : coverage(R) \geqslant minCoverage \wedge support(R) \geqslant minSupport \wedge confidence(R) \geqslant minConfidence\}$, and $\beta(Z, n)$ is the lift of the n^{th} association in Z sorted from highest to lowest by lift. An association rule $X \rightarrow Y[coverage, support, confidence, lift]$ is a target iff it satisfies $|X| \leqslant maxLHSsize$,$coverage(X \rightarrow Y) \geqslant minCoverage$, $support(X \rightarrow Y) \geqslant minSuport$, $confidence(X \rightarrow Y) \geqslant minConfidence$, and $lift(X \rightarrow Y) \geqslant minLift$.

Theorem 1. *Suppose $AR\_by\_OPUS = (\mathcal{C}, \mathcal{D}, \mathcal{A}, \mathcal{M})$. For any $S_1 \subseteq \mathcal{C}$, $S_2 \subseteq \mathcal{C}$, and $S_1 \subseteq S_2$, $coverset(S_2) \subseteq coverset(S_1)$ holds. This is to say, $cover(S_2) \leqslant cover(S_1)$ holds.*

Proof. For any $d \in coverset(S_2)$, according to Definition 1, $S_2 \subseteq d$ holds. Since $S_1 \subseteq S_2$, $S_1 \subseteq d$ holds. Hence $d \in coverset(S_1)$. So $coverset(S_2) \subseteq coverset(S_1)$ holds. □

Theorem 2. *Suppose $AR\_by\_OPUS = (\mathcal{C}, \mathcal{D}, \mathcal{A}, \mathcal{M})$. For any nonempty $S_1, S_2, S_3 \subseteq \mathcal{C}$ satisfying $S_1 \cap S_2 = \emptyset$, $S_2 \cap S_3 = \emptyset$, and $S_1 \cap S_3 = \emptyset$, if*

$$cover(S_1) = cover(S_1 \cup S_2) \tag{1}$$

the following holds.

$$cover(S_1 \cup S_3) = cover(S_1 \cup S_2 \cup S_3) \tag{2}$$

Proof. From (1) and Definition 1, we have

$$|coverset(S_1)| = |coverset(S_1 \cup S_2)| \tag{3}$$

From Theorem 1,

$$coverset(S_1) \supseteq coverset(S_1 \cup S_2) \tag{4}$$

From (3) and (4), we get

$$coverset(S_1) = coverset(S_1 \cup S_2) \tag{5}$$

For any $d \in \mathcal{D} \wedge S_1 \cup S_3 \subseteq d$, $S_1 \subseteq d$ and $S_3 \subseteq d$ hold. From $S_1 \subseteq d$ and (5), we get $S_1 \cup S_2 \subseteq d$. From $S_3 \subseteq d$, $S_1 \cup S_2 \cup S_3 \subseteq d$ holds. Hence

$$coverset(S_1 \cup S_3) \subseteq coverset(S_1 \cup S_2 \cup S_3) \tag{6}$$

From Theorem 1, we have

$$coverset(S_1 \cup S_2 \cup S_3) \subseteq coverset(S_1 \cup S_3) \tag{7}$$

From (6) and (7), $coverset(S_1 \cup S_3) = coverset(S_1 \cup S_2 \cup S_3)$ holds. Hence (2) is proved. □

5.2 Pruning the Consequent Condition Before the Evaluation of Association Rule

One of the pruning rules at Step 2.1.3(b.1)i is used to prune the consequent condition according to the current lower bound on *minSupport* before the evaluation of the association rule. This pruning rule is based on the following theorem.

Theorem 3. *Suppose* $AR\_by\_OPUS = (\mathcal{C}, \mathcal{D}, \mathcal{A}, \mathcal{M})$. *For any association rule* $X \to Y$, *if* $cover(Y) < minSupport$, $X \to Y$ *is not a target.*

Proof. According to Definition 1 and Theorem 1, we get

$$support(X \to Y) = cover(X \cup Y) \leqslant cover(Y) < minSupport$$

Hence $X \to Y$ is not a target. □

From this theorem, we get the following pruning rule.

Pruning 1 *In OPUS_AR for* $AR\_by\_OPUS = (\mathcal{C}, \mathcal{D}, \mathcal{A}, \mathcal{M})$, *for any condition* $Q \in AvailableRHS$, *if* $cover(Q) < minSupport$, *then* Q *can be pruned from* $NewAvailableRHS$.

According to Theorem 3, any association rule with such Q as the consequent can not be a target, therefore Q can be pruned. The "pruning 1 added" column of Table 3 lists the times for OPUS_AR on the five datasets with the basic pruning and pruning 1.

5.3 Pruning the Consequent Condition after the Evaluation of Association Rule

This pruning rule at Step 2.1.3(b.1)ii.B is used to prune the consequent condition after the evaluation of the current association rule. It is based on the following theorem.

Theorem 4. *Suppose* $AR\_by\_OPUS = (\mathcal{C}, \mathcal{D}, \mathcal{A}, \mathcal{M})$. *For any association rule* $X \to Y$, *if* $confidence(X \to Y) = 1$, *for any* $X_1 \subset \mathcal{C}$ *satisfying* $X_1 \cap X = \emptyset \wedge X_1 \cap Y = \emptyset \wedge cover(X \cup X_1) \neq 0$, *the following holds.*

$$lift(X \cup X_1 \to Y) = lift(X \to Y)$$

Proof. From $confidence(X \to Y) = 1$, we get

$$support(X \to Y) = coverage(X \to Y)$$

that is to say,

$$cover(X) = cover(X \cup Y) \tag{8}$$

From (8) and Theorem 2, $cover(X \cup X_1) = cover(X \cup X_1 \cup Y)$ holds. Since $cover(X \cup X_1) \neq 0$, hence

$$support(X \cup X_1 \to Y) = coverage(X \cup X_1 \to Y) \neq 0$$

Therefore $confidence(X \cup X_1 \to Y) = 1$. From Definition 1, the following two equations hold.

$$lift(X \cup X_1 \to Y) = \frac{confidence(X \cup X_1 \to Y)}{cover(Y)} = \frac{1}{cover(Y)}$$

$$lift(X \to Y) = \frac{confidence(X \to Y)}{cover(Y)} = \frac{1}{cover(Y)}$$

Hence $lift(X \cup X_1 \to Y) = lift(X \to Y)$ holds. □

From this theorem, we get the following pruning rule.

Pruning 2 *In OPUS_AR for AR_by_OPUS* $= (\mathcal{C}, \mathcal{D}, \mathcal{A}, \mathcal{M})$*, after the evaluation of the current association rule* $NewLHS \to Q$*, if* $confidence(NewLHS \to Q) = 1$ *and* $lift(NewLHS \to Q) < minLift$*,* Q *can be pruned from* $NewAvailableRHS$*.*

According to the above theorem, all of the association rules with Q as the consequent in the search space below the current node take the same lift value as $NewLHS \to Q$. Therefore if $lift(NewLHS \to Q) < minLift$, none of these rules can be target association, Q can be pruned from $NewAvailableRHS$. The "pruning 2 added" column of Table 3 lists the times for OPUS_AR on the five datasets with the basic pruning and pruning 2. For "covtype," the compute time is reduced by this pruning to less than 55% of that supported by the basic pruning rules, for "ipums.la.99," the compute time is reduced to less than 66% of the basic pruning.

5.4 Saving Data Access for the Current Association Rule by $minConfidence$

In order to evaluate the number of records covered by set of conditions, the dataset is normally accessed by OPUS_AR at least once for each association rule antecedent and once for the union of the antecedent and consequent. Techniques for saving such data access can improve the efficiency of the algorithm. Whereas the pruning rules save data access by discarding the region of the search space below a node, the saving rules save data access for a node without removing its branch.

Step 2.1.3(b.1)ii is for saving data access for the current association rule $NewLHS \to Q$. We are going to introduce two of the saving rules adopted at this step, one is by $minConfidence$, based on the following theorem, and the other is by the antecedent of the current association rule, described in the next section.

Theorem 5. *Suppose AR_by_OPUS* $= (\mathcal{C}, \mathcal{D}, \mathcal{A}, \mathcal{M})$*. For any association rule* $X \to Y$*, if* $\frac{cover(Y)}{cover(X)} < minConfidence$*,* $X \to Y$ *is not a target.*

Proof. According to Definition 1, we have

$$confidence(X \to Y) = \frac{support(X \to Y)}{coverage(X \to Y)} = \frac{cover(X \cup Y)}{cover(X)}$$

According to Theorem 1, $cover(X \cup Y) \leqslant cover(Y)$ holds. Since $\frac{cover(Y)}{cover(X)} < minConfidence$,

$$confidence(X \to Y) \leqslant \frac{cover(Y)}{cover(X)} < minConfidence$$

Therefore $X \to Y$ is not a target. □

From this theorem, we get the following data access saving rule.

Saving 1 *In OPUS_AR for AR_by_OPUS* $= (\mathcal{C}, \mathcal{D}, \mathcal{A}, \mathcal{M})$*, for the current association* $NewLHS \to Q$*, if* $|NewLHS| = maxLHSsize$ *and* $\frac{cover(Q)}{cover(NewLHS)} < minConfidence$*, there is no need to access data to evaluate* $NewLHS \to Q$*, as it is not a target.*

The reason that the saving is adopted instead of pruning under this situation is in the branch below the current $NewLHS \to Q$, some of the supersets of $NewLHS$ with lower values of coverage might make the association have confidence larger than $minConfidence$. While saving data access, the pruning based on the results of the data access is not available anymore, thus the overall efficiency might be slowed down accordingly. Due to this, $|NewLHS| = maxLHSsize$ is added to the above saving rule to ensure that it is applied only at the maximum search depth where no pruning is necessary.

The "saving 1 added" column of Table 3 lists the times for OPUS_AR on the five datasets with the basic pruning and this saving rule.

5.5 Saving Data Access for the Current Association Rule by the Antecedent

Another saving rule at Step 2.1.3(b.1)ii for the current associations rule $NewLHS \to Q$, where $NewLHS = CurrentLHS \cup \{P\}$, $P \in AvailableLHS$, functions according to the relation between $CurrentLHS$ and P. It is based on the following theorem.

Theorem 6. *Suppose AR_by_OPUS* $= (\mathcal{C}, \mathcal{D}, \mathcal{A}, \mathcal{M})$*. For any association rule* $X \to Y$ *and* $X \cup \{P\} \to Y$ *where* $P \in \mathcal{C}$*,* $P \notin X$ *and* $P \notin Y$*, if* $cover(X) = cover(X \cup \{P\})$*, the following hold.*

$$coverage(X \to Y) = coverage(X \cup \{P\} \to Y) \tag{9}$$

$$support(X \to Y) = support(X \cup \{P\} \to Y) \tag{10}$$

$$confidence(X \to Y) = confidence(X \cup \{P\} \to Y) \tag{11}$$

$$lift(X \to Y) = lift(X \cup \{P\} \to Y) \tag{12}$$

Proof. According to $cover(X) = cover(X \cup \{P\})$, (9) holds. From $cover(X) = cover(X \cup \{P\})$ and Theorem 2, the following holds.

$$cover(X \cup Y) = cover(X \cup \{P\} \cup Y) \tag{13}$$

From (13), (10) holds. Hence (11) and (12) are proved. □

From this theorem, we get the following data access saving rule.

Saving 2 *In OPUS_AR for AR_by_OPUS* $= (\mathcal{C}, \mathcal{D}, \mathcal{A}, \mathcal{M})$, *for the current association* $NewLHS \rightarrow Q$ *where* $NewLHS = CurrentLHS \cup \{P\}$, $P \in AvailableLHS$, *if* $|NewLHS| = maxLHSsize$, *the number of current target associations is less than* $|coverset(NewLHS)|$, *and* $cover(CurrentLHS) = cover(NewLHS)$, *instead of accessing data to evaluate* $NewLHS \rightarrow Q$, *check if* $CurrentLHS \rightarrow Q$ *exists in the current target associations, and if yes, copy all the statistic values of* $CurrentLHS \rightarrow Q$ *to* $NewLHS \rightarrow Q$, *otherwise,* $NewLHS \rightarrow Q$ *is not a target.*

Since $CurrentLHS \rightarrow Q$ is investigated before $NewLHS \rightarrow Q$ in OPUS_AR, and they share the same statistic values, $NewLHS \rightarrow Q$ will be a target if and only if $CurrentLHS \rightarrow Q$ is a target. Due to the same reasons as in the above section, we add $|NewLHS| = maxLHSsize$ in the saving rule to make sure that application of the saving rule can not slow down the overall efficiency. If the number of current target associations is larger than $|coverset(NewLHS)|$, searching current target associations might become less efficient than accessing data of the amount of $|coverset(NewLHS)|$ for computing $cover(NewLHS \cup Q)$.

The "saving 2 added" column of Table 3 lists the times for OPUS_AR on the five datasets with the basic pruning and this saving rule. For both "covtype" and "connect-4," the compute times are reduced by this saving to less than 66% of that supported by the basic pruning rules.

Table 3. Efficiency improvements by pruning in OPUS_AR and efficiency of Apriori

datasets	OPUS_AR						Apriori
	basic pruning	**pruning 1 added**	**pruning 2 added**	**saving 1 added**	**saving 2 added**	**all added**	
covtype	7:33:50	5:28:21	4:6:58	6:25:16	4:59:19	3:4:19	77:56:3
ipums.la.99	11:38:31	9:2:37	7:40:12	11:27:9	9:25:16	6:28:38	19:45:5
ticdata2000	25:28:43	24:34:12	23:41:12	24:56:10	22:34:7	23:18:29	—
connect-4	1:48:51	1:24:59	1:10:9	1:30:35	1:11:37	0:48:33	3:15:26
letter-recognition	0:0:23	0:0:20	0:0:20	0:0:23	0:0:22	0:0:20	0:0:35

5.6 Efficiency Comparison between OPUS_AR and Apriori

The "all added" column of Table 3 lists the times on the datasets for OPUS_AR with the new pruning mechanisms composed of pruning 1 and 2 and saving rule 1 and 2 all added to the four original pruning rules. For all datasets other than "ticdata2000," combining all rules results in more efficient search than utilizing any of the rule alone. The interaction between rules than increases compute times for "ticdata2000" merits further investigation. For "covtype," "connect-4," and "ipums.la.99," the compute times are reduced to less than 41%, 45% and 56% of that supported by the original pruning rules, respectively.

The CPU times of running Borgelt's Apriori system on the five datasets are listed in the "Apriori" column of Table 3. The inefficiency of Apriori for dense datasets is demonstrated by the fact that on every dataset OPUS_AR is more efficient than Apriori, and that for "ticdata2000," Apriori runs out of memory when processing itemsets of size 4.

6 Conclusions

OPUS_AR provides an alternative to the frequent itemset approach to association rule discovery. Our experiments have demonstrated that OPUS_AR can provide more efficient association rule discovery than apriori for dense datasets, and can make association rule discovery feasible where the memory requirements of the frequent itemset approach can make its application infeasible. OPUS_AR has the further advantage that it can utilize constraints other than minimum frequency to prune the search space. This makes feasible association rule discovery where there is no natural lower limit on the support for an association.

This paper has presented new pruning rules and data access saving rules for OPUS_AR, which result in the reduction of compute times by as much as 41% compared with those resulting from the original mechanisms only. These results again demonstrate that OPUS_AR can support fast association rule discovery from large dense datasets.

References

1. R. Agarwal, C. C. Aggarwal, and V. V. V. Prasad. Depth first generation of long patterns. In *Proc. Sixth ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining (KDD2000)*, pages 108–118, Boston, MA, August 2000. ACM.
2. R. Agrawal, T. Imielinski, and A. Swami. Mining associations between sets of items in massive databases. In *Proc. 1993 ACM-SIGMOD Int. Conf. Management of Data*, pages 207–216, 1993.
3. S. D. Bay. The UCI KDD archive. [http://kdd.ics.uci.edu] Irvine, CA: University of California, Department of Information and Computer Science., 2001.
4. R. J. Bayardo. Efficiently mining long patterns from databases. In *Proc. 1998 ACM-SIGMOD Int. Conf. Management of Data*, pages 85–93, 1998.
5. R. J. Bayardo, R. Agrawal, and D. Gunopulos. Constraint-based rule mining in large, dense databases. *Data Mining and Knowledge Discovery*, 4(2/3):217–240, 2000.
6. C. Blake and C. J. Merz. UCI repository of machine learning databases. [Machine-readable data repository]. University of California, Department of Information and Computer Science, Irvine, CA., 2001.
7. C. Borgelt. apriori. (Computer Software) http://fuzzy.cs.Uni-Magdeburg.de/ borgelt/, February 2000.
8. S. H. Clearwater and F. J. Provost. RL4: A tool for knowledge-based induction. In *Proc. Second Intl. IEEE Conf. on Tools for AI*, pages 24–30, Los Alamitos, CA, 1990. IEEE Computer Society Press.
9. J. Han, J. Pei, and Y. Yin. Mining frequent patterns without candidate generation. In *Proc. 2000 ACM-SIGMOD Int. Conf. on Management of Data (SIGMOD'00)*, Dallas, TX, May 2000.
10. S. Morishita and A. Nakaya. Parallel branch-and-bound graph search for correlated association rules. In *Proc. ACM SIGKDD Workshop on Large-Scale Parallel KDD Systems*, volume LNAI 1759, pages 127–144. Springer, Berlin, 2000.
11. J. Pei, J. Han, and R. Mao. CLOSET: An efficient algorithm for mining frequent closed itemsets. In *Proc. 2000 ACM-SIGMOD Int. Workshop on Data Mining and Knowledge Discovery (DMKD'00)*, Dallas, TX, May 2000.
12. F. Provost, J. Aronis, and B. Buchanan. Rule-space search for knowledge-based discovery. CIIO Working Paper IS 99-012, Stern School of Business, New York University, New York, NY 10012, 1999.

13. R. Rymon. Search through systematic set enumeration. In *Proc. KR-92*, pages 268–275, Cambridge, MA, 1992.
14. R. Segal and O. Etzioni. Learning decision lists using homogeneous rules. In *AAAI-94*, Seattle, WA, 1994. AAAI press.
15. G. I. Webb. OPUS: An efficient admissible algorithm for unordered search. *Journal of Artificial Intelligence Research*, 3:431–465, 1995.
16. G. I. Webb. Efficient search for association rules. In *The Sixth ACM SIGKDD Int. Conf .Knowledge Discovery and Data Mining*, pages 99–107, Boston, MA, 2000. The Association for Computing Machinery.
17. M. J. Zaki. Generating non-redundant association rules. In *Proceedingsof the Sixth ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining (KDD2000)*, pages 34–43, Boston, MA, August 2000. ACM.