PRICES: An Efficient Algorithm for Mining Association Rules

Chuan Wang, Christos Tjortjis

Information Systems Group, Department of Computation,
University of Manchester Institute of Science and Technology,
P.O. Box 88, Manchester, M60 1QD
United Kingdom
E-mail: C.Wang-2@postgrad.umist.ac.uk christos@co.umist.ac.uk

Abstract. In this paper, we present PRICES, an efficient algorithm for mining association rules, which first identifies all large itemsets and then generates association rules. Our approach reduces large itemset generation time, known to be the most time-consuming step, by scanning the database only once and using logical operations in the process. Experimental results and comparisons with the state of the art algorithm *Apriori* shows that PRICES very efficient and in some cases up to ten times as fast as *Apriori*.

1 Introduction

Association rules, is a data mining technique which identifies relationships between items in databases. The process can be decomposed into two steps: large itemsets generation and association rules generation [1]. It is well established that, while association rules generation is rather straightforward, large itemset generation can be a bottleneck in the process. A number of algorithms have been proposed in order to increase the efficiency of the process [1], [2], [3], [4], [5], [6], [7]. We discuss and review the most prominent ones in section 2.

Here we present a new algorithm for mining association rules called PRICES. The algorithm uses the same two steps as in other algorithms; it is however faster as it scans the database only once, to store transactions information in the memory by a succinct form we call *Prices Table*. This table is then pruned by creating a pseudo transaction table called *Pruned Prices Table*, which contains all 1-size large itemsets after eliminating all 1-size small itemsets. Recursion is used to generate k-size (k>1) large itemsets from the Pruned Prices Table and (k-1)-size large itemsets. Finally, association rules are generated using the large itemsets. The innovation of the algorithm is that it uses logical operations, such as AND, OR, XOR and left-shift in the process of generating large itemsets and association rules, thus accelerating the process. Experimental results have shown that PRICES is efficient and outperforms *Apriori* in terms of speed.

A more detailed description of PRICES is given in Section 3. Section 4 presents experimental results and comparisons with *Apriori*. Conclusions and directions for further work are outlined in section 5.

2 Background

Since the introduction of mining association rules in [1], many algorithms that discover large itemsets have been proposed. The number of times an algorithm scans the entire database is a significant factor in terms of speed as it determines the number of time consuming I/O operations involved.

AIS generates all large itemsets by making multiple passes over the database [1]. After reading a transaction, large itemsets are extended by appending lexicographically larger items to generate candidate itemsets. If the *support* for candidate itemsets is above a minimum threshold, they are chosen as large itemsets, and the next pass commences, until there are no more large itemsets. A limitation of AIS is that it only produces one item in the consequent of rules.

Apriori is a well-known improvement over AIS, which utilizes the concept that any subset of a large itemset is a large itemset [2]. It improves candidate generation by joining large itemsets together. DHP is a hash-based Apriori-like algorithm, effective in generating large 2-itemsets [5]. However, both Apriori and DHP scan the database many times producing a substantial I/O overhead.

The Partition algorithm reduces the number of database scans to 2 [6]. It partitions the database into small segments. Local large itemsets of each segment are then united and a further entire database scan is needed to generate the global large itemsets. The Sampling algorithm improves on the Partition algorithm [7]. It reduces the number of database scans to one in the best case and two in the worst. A sample is drawn and large itemsets of it are generated and finally large itemsets are found. The sample is crucial because an unrepresentative one can cause a very big candidate set.

SETM is an algorithm designed for using SQL to generate large itemsets [4]. Large itemsets are in the form of <TID, itemset> where TID is a unique identifier for each transaction. One disadvantage of SETM is that it generates too many candidate itemsets, thus reducing efficiency.

All in all, the number of database scans needed by AIS, Apriori DHP and SETM, depends on the number of items while Partition and Sampling algorithms reduce this number to 2, despite their other limitations.

3 The Algorithm PRICES

PRICES is an algorithm which mines association rules in two steps by use of logical operations. First, large itemsets are identified, and then association rules are generated. Section 3.1 presents basic principles and underlying assumptions used by the algorithm. Large itemsets generation is described in section 3.2, and association rules generation is explained in section 3.3.

3.1 Basic Principles and Assumptions

Prices uses logical operations such as AND, OR, XOR and left-shift to generate large itemsets and association rules. In addition, every item in the transactions is given a

unique *value.* Every transaction can be represented by a *price,* which is the sum of item values it consists of. Items values are assumed be such that that no value can be the sum of other values. Therefore, every price represents a unique itemset pattern.

For example, if there are 5 items, A, B, C, D and E in a database, let the value of item A be 2^4, the value of item B 2^3 and so on. The price of transaction {A, C, D} will be 10110 in binary mode. In this way, an itemset can also be represented as a price.

Under this assumption, we can apply logical operation AND to the price of one transaction and the price of one itemset to determine whether this transaction "contains" this itemset, by comparing the result with the itemset price. For example, transaction {A, C, D} (price $P_T = 10110$) contains itemset {A, C} (price $P_{AC} = 10100$) because the result of P_T AND P_{AC} is equal to P_{AC}. Therefore, our task is to identify all the itemsets prices from {00…01} to {11…11} occurring above a threshold in the prices of transactions.

For a better understanding of the algorithm we shall use the following example for the rest of the discussion: consider a database with transaction information as in Table 1 and assume that minimum support and confidence are both set to 50%.

Table 1. A database example

TID	Items
T_1	ACD
T_2	BCE
T_3	ABCE
T_4	BE

Table 2. The Prices Table (PT)

TID	Items	Prices
T_1	ACD	10110
T_2	BCE	01101
T_3	ABCE	11101
T_4	BE	01001

3.2 Large Itemset Generation

The PRICES algorithm generates large itemsets in three steps. First, the *Pruned Prices Table (PPT)* is created, then all large 2-itemset are created and finally, all large itemsets are generated.

PRICES scans the database, calculates the prices of all transactions and stores these in memory using an array called *Prices Table (PT)*. Table 2 shows the PT for the example given in section 3.1.

It is known that any itemset which contains a small itemset will also be small [2]. Therefore, we can prune the PT by eliminating the column of small items. This is done in two steps: first generate the *Large Bit Mark* (LBM), which is the price of the itemset which contains all large 1-itemsets; then create the *Pruned Prices Table*. To generate the LBM we set the price of the first 1-size candidate to 1 and apply a left-shift operation to generate the second candidate price and so on. We calculate each candidate's support. If a candidate is large, the corresponding position in LBM is set to 1, otherwise to 0. In addition, the large 1-size itemsets, along with the support and size, are stored in L.

Given the LBM and PT, we can generate the Pruned Prices Table by eliminating the columns which have 0 in the corresponding position of LBM. One 0 in the LBM indicates that the corresponding item is small and thus any itemset containing this is also small. Therefore, removing these items shrinks the PT without affecting the generation of large itemsets.

By applying these steps to our example, we obtain the respective LBM as seen in Table 3. Table 4 shows how the pruned price of T_1 is generated from LBM and T_1.

Table 3. Generation of the LBM

	A	B	C	D	E
Support	2	3	3	1	3
comp. with minsup	>=	>=	>=	<	>=
LBM	1	1	1	0	1

Table 4. Price and Pruned Price of T_1

	A	B	C	**D**	E	
P_1	1	0	1	**1**	0	10110
LBM	1	1	1	**0**	1	11101
PP_1	1	0	1	**-**	0	1010

As every single item in the Pruned Prices Table is large, every 2-size itemset composed of different single item is a candidate. We calculate the support of every candidate and if it is large, we record it into L, along with its support and size. We also use the OR operation to compose two different item prices into one price. For example, the price of itemset {A, E} (10001) can be derived by applying OR to itemset {A} (10000) and {E} (00001).

k-size large itemsets can then be generated from (k-1)-size large itemsets and the PPT. We use the XOR operation as a difference indicator from which we can find how many different bits (items) there are between two (k-1)-size large itemsets. To generate a candidate c_k, two (k-1)-size large itemsets must have exactly two different bits. Hence, the fact that 2-size itemset composed by two different bits of two (k-1)-size large itemsets is included in large 2-itemsets (L_2) is a prerequisite of that the itemset composed by these two (k-1)-size large itemsets is a candidate. Furthermore, whether all the other (k-1)-size subsets of this potential c_k are included in L_{k-1} are checked. Finally, the candidate support is calculated and recorded it if large. This is recursively repeated until less than k large (k-1)-size itemsets are found.

Finally, in order to get the large itemsets from L, we restore the prices in L from pruned prices and map those into itemsets. According to the definition of LBM, a 0 is inserted into pruned prices at corresponding positions to restore prices. Once the prices are restored, we can map these into itemsets based on the previous definition of the relationship between price and itemset.

3.3 Association Rules Generation

In this section, we present the way to generate association rules from the final set of large itemsets. We know that an association rule $X \rightarrow Y$ holds if: (1) $X \bigcap Y = \phi$; (2) $X, Y, X \bigcup Y \in L$; (3) $\frac{s(X \bigcup Y)}{s(X)} \geq$ minconf.

Therefore, for every two large itemset l_i and $l_j \in L$, if l_i AND l_j = 0 (1), $l_i \bigcup l_j \in L$ (2) and $\frac{s(l_i \bigcup l_j)}{s(l_i)} \geq$ minconf (3) are all met, then the rule $l_i \rightarrow l_j$ holds.

4. Experimental Results

In order to evaluate the performance of PRICES, we developed a prototype and carried out experiments. We created several synthetic datasets by using Quest dataset generator [9]. The average transaction length is 10 and the average size of potentially maximal large itemsets is 4. Table 5 shows the datasets we generated.

Table 5. Synthetic datasets

#	No. of transactions	No. of items	Name
1	1,000	100	T1K.I100
2	10,000	100	T10K.I100
3	100,000	100	T100K.I100
4	1,000,000	100	T1M.I100

For comparison purposes we used an implementation of the state of the art *Apriori* algorithm obtained from Weka Data Mining System [8]. All the experiments were executed on a Personal Computer at 1800MHz, with 256MB of main memory, running Windows XP Professional. In order to get more accurate results, we executed each experiment three times. Average execution times are shown in Fig. 1.

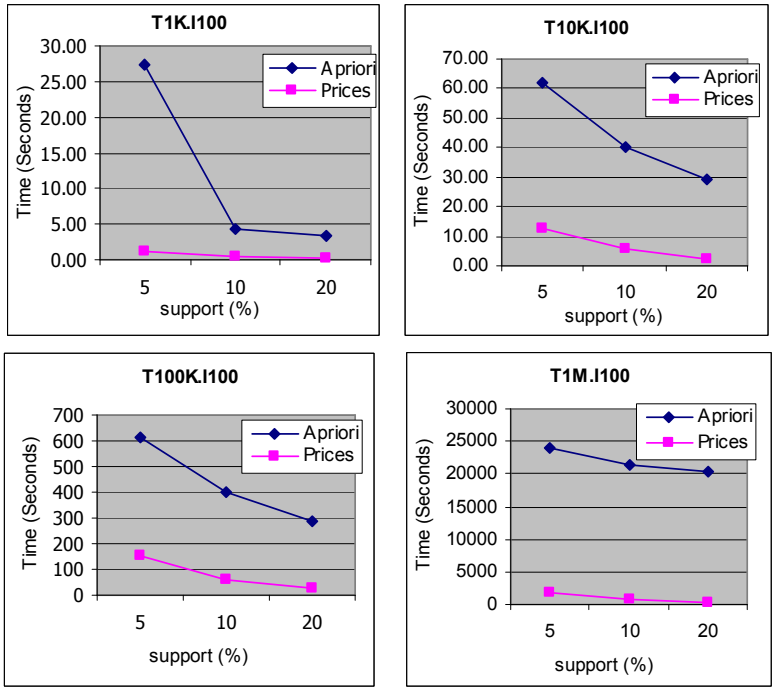

Fig. 1. Experimental results

Results show that PRICES is faster to Apriori as it only scans the database once and uses logical operations performed in the main memory. The pruning technique

used in PRICES also contributes to the high performance. The most important finding of the result analysis is that the larger the database grows in terms of transactions number, the faster PRICES gets compared to Apriori. So for example for a dataset with a million transactions PRICES is more than ten times faster than Apriori.

5. Conclusions and further work

In this paper, we proposed PRICES, a new efficient algorithm for mining association rules. Its major advantage is that it only scans the database once and any consecutive processing takes places in memory using logical operations. Extensive experiments using different synthetic datasets were conducted to assess the performance of the algorithm. Results have been positive and PRICES outperformed *Apriori* in terms of speed.

We are currently experimenting with memory requirements and various techniques to address performance deterioration due to I/O overhead when data do not fit in memory due to the possibly very large size of datasets. Plans for further work include devising an extension of the algorithm to match the needs of different applications, such as document retrieval, information recovery and text mining.

References

[1] R. Agrawal, T. Imielinski, and A.N. Swami. Mining Association Rules between Sets of Items in Large Databases. *Proceedings of the 1993 ACM SIGMOD Int'l Conf. Management of Data,* pp. 207-216, Washington, D.C., May 1993.

[2] R. Agrawal, R. Srikant. Fast Algorithms for Mining Association Rules in Large Databases. *Proc. 20th Int'l Conf. Very Large Data Bases*, September 1994.

[3] L. Dong and C. Tjortjis. Experiences of Using a Quantitative Approach for Mining Association Rules, *Proc.4th Int'l Conf. Intelligent Data Engineering Automated Learning* (IDEAL 03) in Lecture Notes in Computer Science Series Vol. 2690, pp. 693-700

[4] M. Houtsma and A. Swami. Set-Oriented Mining for Association Rules in Relational Databases, *Proc. 11th IEEE Int'l Conf. Data Engineering*, pp. 25-34,Taipei,Taiwan,March, 1995.

[5] J.S. Park, M.S. Chen and P.S. Yu. Using a Hash-Based Method with Transaction Trimming For Mining Association Rules. *IEEE Transactions on Knowledge and Data Engineering*, September/October 1997.

[6] A. Savasere, E. Omiecinski, and S.B. Navathe. An Efficient Algorithm for Mining Association Rules in Large Databases, *Proc. ACM SIGMOD Int'l Conf. Management of Data*, SIGMOD 1998, June 2-4, 1998, Seattle, Washington, USA.

[7] H. Toivonen. Sampling Large Databases for Association Rules, *Proc. 22nd Int'l Conf. Very Large Databases*, pp. 134-145, Mumbai, India, 1996.

[8] www.cs.waikato.ac.nz/~ml/weka. Weka Experiment Environment, Weka Data Mining System. (Last accessed in March, 2004).

[9] http://www.almaden.ibm.com/software/quest/. Intelligent Information Systems, IBM Almaden Research Center. (Last accessed in January, 2004).