Journal of Theoretical and Applied Information Technology

www.jatit.org

EFFECTIVE MINER OF TIME-VARIANT DATA USING WEIGHT FACTOR

[1] RICHA SHARMA [2] PURUSHOTTAM SHARMA [3] DEEPAK NAGARIA

[1]Research Scholar (M.Tech) B.I.E.T Jhansi (U.P)
[2]Lecturer Amity University, Noida (U.P)
[3]Lecturer B.I.E.T, Jhansi (U.P)

Emails: richa_sharma06@ rediffmail.com , psharma5@amity.edu, deepaknagaria@gmail.com

ABSTRACT

There are so many important results toward finding the association rules. But, when we consider the sale of seasonal items then no algorithm existed till now that can able to mine the interesting pattern on time–variant seasonal database. In view of this, we propose an Effective Miner (abbreviatedly as EM) algorithm to perform the mining for this problem as well as I conduct the corresponding performance studies.
Furthermore, without fully considering the time-changing characteristics of items and transactions, it is noted that some discovered rules may be expired from user's interest i.e rules generated in one season can not give the useful information in other season. Under EM we first partition the database on the bases of season i.e Yearly, Half Yearly or Quarterly etc according to user's requirements and then we apply weighted mining on each partition. In EM (Effective Miner) the cumulative information of mining previous partitions is selectively carried over toward the generation of candidate itemsets for the subsequent partitions I have also applied scan reduction technique in EM due to that only two scan of database are required, means saving lots of time.

Keywords: *Association Rules, Effective Miner, Database, Partition, Saving Time*

1. INTRODUCTION

The discovery of association relationship among the data in a huge database has been known to be useful in selective marketing, decision analysis, and business management. A popular area of applications is the market basket analysis, which studies the buying behaviors of customers by searching for sets of items that are frequently purchased either together or in sequence. For a given pair of confidence and support thresholds, the problem of mining association rules is to identify all association rules that have confidence and support greater than the corresponding minimum support threshold (denoted as min_supp) and minimum confidence threshold (denoted as min_conf). Association rule mining algorithms [5] work in two steps:
(1) Generate all frequent itemsets that satisfy min_supp;
(2) Generate all association rules that satisfy min_conf using the frequent itemsets

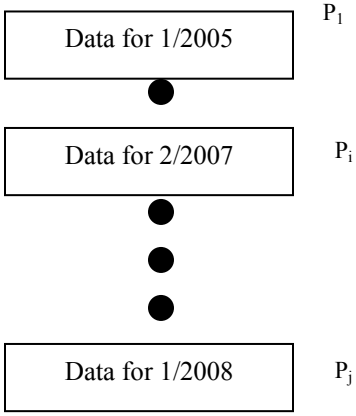


Fig 1.1 A time-variant transaction database

On the other hand, a time-variant database, as shown in Figure 1.1, consists of values or events varying with time. Time-variant databases are popular in many applications, such as daily fluctuations of a stock market, traces of

a dynamic production process, scientific experiments, medical treatments, weather records, to name a few. In our opinion, the existing model of the constraint-based association rule mining is not able to efficiently handle the time-variant database due to two fundamental problems, i.e.,
(1) Lack of consideration of the exhibition period of each individual transaction;
(2) Lack of an intelligent support counting basis for each item.

However, some phenomena are observed when we take the "item information" in Figure 1.2 into consideration.

1.1 An early product intrinsically possesses a higher likelihood to be determined as a frequent itemset.

As a result, the association rules we usually get will be those with long-term products such as "milk and bread are frequently purchased together", which, while being correct by the definition, is of less interest to us in the association rule mining. In contrast, some more recent products, such as new books, which are really "frequent" and interesting in their exhibition periods, are less likely to be identified as frequent ones if a traditional mining process is employed.

1.2 Some discovered rules may be expired from users' interest.

Some discovered knowledge may be obsolete and of little use. This is especially true when we perform the mining schemes on a transaction database of short life cycle products such as CPU, RAM, etc.

Such mining results could be of less interest to our on-going mining works. For example, most researchers tend to pay more attention to the recently published papers.

1.3 Different transactions are usually of different importance to the user.

From the above discussions, it is noted that mining long period transaction data could have less contribution to making future business decisions because, for example, the selling items could be out of date. Since a new coming data is usually viewed more important than an old one, without fully considering this aspect, the knowledge discovered from the traditional mining framework may lead to wrong decisions.

Item	Starting Date
A	Jan-07
B	Apr-07
C	Jul-07
D	Aug-07
E	Feb-07
F	Mar-07

			Transaction Database	
		Date	TID	Itemset
D	P_1	Jan-08	T_1	B D
			T_2	B C D
			T_3	B C
			T_4	A D
	P_2	Feb-08	T_5	B C E
			T_6	D E
			T_7	A B C
			T_8	C D E
	P_3	Mar-08	T_9	B C E F
			T_{10}	B F
			T_{11}	A D
			T_{12}	B D F

Figure 1.2: An illustrative transaction database and the corresponding item information

Specifically, we propose an Effective Miner (abbreviatedly as EM) algorithm to perform the mining for this problem as well as conduct the corresponding performance studies. In algorithm EM, the importance of each transaction period is first reflected by a proper weight assigned by the user. Then, EM partitions the time-variant database in light of weighted periods of transactions and performs weighted mining.

2. PROBLEM DESCRIPTIONS

Let n be the number of partitions with a time granularity, e.g., business-week, month, quarter, year, etc., in database D. In the model considered, Pi denotes the part of the transaction database where Pi is a subset of D. Explicitly, we explore in this paper the mining of transaction-weighted association rules (abbreviatedly as weighted association rules), i.e., $(X \Rightarrow Y)^W$, where $X \Rightarrow Y$ is produced by the concepts of weighted − support and weighted confidence.

Further, instead of using the traditional support threshold $min\_S^T = \lceil |D| \times min\_supp \rceil$ as a minimum support threshold for each item , a weighted minimum support for mining an association rules is determined by $min\_S^W = \{\Sigma |Pi| \times W(Pi)\} \times min\_supp$ where |Pi| and W (Pi) represent the amount of partial transactions and their corresponding weight values by a weighting function W (·) in the weighted period Pi of the database D. Formally, we have the following definitions.

Definition 2.1:

Let NPi(X) be the number of transactions in partition Pi that contain itemset X. Consequently, the weighted support value of an itemset X can be formulated as $S^W(X) = \Sigma NPi(X) \times W(Pi)$. As a result, the weighted support ratio of an itemset X is $supp^W(X) = S^W(X) / \Sigma |Pi| \times W(Pi)$.

In accordance with Definition 2.1, an itemset X is termed to be frequent when the weighted occurrence frequency of X is larger than the value of min_supp required, i.e., $supp^W(X) > min\_supp$, in transaction set D. The weighted confidence of a weighted association rule $(X \Rightarrow Y)^W$ is then defined below.

Definition 2.2:

$conf^W(X \Rightarrow Y) = supp^W (X \cup Y) / suppW (X)$.

Definition 2.3:

An association rule $X \Rightarrow Y$ is termed a frequent weighted association rule $(X \Rightarrow Y)^W$ if and only if its weighted support is larger than minimum support required, i.e., $supp^W(X \cup Y) > min\_supp$, and the weighted confidence $conf^W(X \Rightarrow Y)$ is larger than minimum confidence needed, i.e., $conf^W(X \Rightarrow Y) > min\_conf$.

3. EFFECTIVE MINING

It is noted that most of the previous studies, including those in [5], belong to Apriori-like approaches. Basically, an Apriori-like approach is based on an anti-monotone Apriori heuristic [5], i.e., if any itemset of length k is not frequent in the database, its length (k + 1) superitemset will never be frequent. The essential idea is to iteratively generate the set of candidate itemsets of length (k+1) from the set of frequent itemsets of length k (for $k \geq 1$), and to check their corresponding occurrence frequencies in the database.

As a result, if the largest frequent itemset is a j-itemset, then an Apriori-like algorithm may need to scan the database up to (j +1) times. This is the basic concept of an extended version of Apriori-based algorithm, referred to as $Apriori^W$, whose performance will be comparatively evaluated with algorithm EM in our experimental studies later. In fact, as will be validated by experimental results later, the increase of candidates often causes a drastic increase of execution time and a severe performance degradation, meaning that without utilizing the partitioning and Time support counting techniques proposed, a direct extension to priori work is not able to handle the weighted association rule mining efficiently.

In [6], the technique of scan-reduction was proposed and shown to result in prominent performance improvement. By scan reduction, C_k is generated from $C_{k-1} * C_{k-1}$ instead of from $L_{k-1} * L_{k-1}$. Clearly, a C'_3 generated from $C_2 * C_2$, instead of from $L_2 * L_2$, will have a size greater than $|C_3|$ where C_3 is generated from $L_2 * L_2$. It can be seen that using this concept, one can determine all L_ks by as few as two scans of the database (i.e., one initial scan to determine L_1 and a final scan to determine all other frequent itemsets), assuming that C'_k for $k \geq 3$ is generated from C'_{k-1} and all C'_k for $k > 2$ can be kept in the memory. It will be seen that the Progressive mining technique used in algorithm EM will enable EM to obtain candidate set C_k with the size very close to that of L_k. This feature of EM allows itself of fully utilizing the technique of scan reduction and leads to prominent performance improvement over $Apriori^W$.

3.1 Algorithm of EM

In general, databases are too large to be held in main memory. Thus, the data mining techniques applied to very large databases have to be highly scalable for efficient execution. As mentioned above, by partitioning a transaction database into several partitions, algorithm EM is devised to employ a Time filtering scheme in each partition to deal with the candidate itemset generation and process one partition at a time. For ease of exposition, the processing of a partition is termed a phase of processing. Explicitly, a Time candidate set of itemsets is composed of the following two types of candidate itemsets, i.e.,

(1) The candidate itemsets that were carried over from the previous progressive candidate set in the previous phase and remain as candidate itemsets after the current partition is included

into consideration (Such candidate itemsets are called type **α** candidate itemsets); and
(2) The candidate itemsets that were not in the Progressive candidate set in the previous phase but are newly identified after only taking the current data partition into account (Such candidate itemsets are called type **β** candidate itemsets).

Under EM, the cumulative information in the prior phases is selectively carried over toward the generation of candidate itemsets in the subsequent phases. After the processing of a phase, algorithm EM outputs a Progressive candidate set of itemsets, their occurrence counts and the corresponding partial supports required.

Initially, a time-variant database D is partitioned into n partitions based on the weighted periods of transactions. The procedure of algorithm EM is outlined below, where algorithm EM is decomposed into four sub-procedures for ease of description. C_2 is the set of candidate 2-itemsets generated by database D. Recall that NPi(X) is the number of transactions in partition Pi that contain itemset X and W (Pi) is the corresponding weight of partition Pi.

Algorithm EM (n, min_supp): Effective Miner

Procedure I: Initial Partition, based on time i.e. yearly, half yearly, Quarterly etc.
1. $|D| = P_{i=1, n} |Pi|$;
2. $C_2 = \varnothing$;

Procedure II: Candidate 2-Itemset Generation
1. begin for i = 1 to n // 1st scan of D
2. begin for each 2-itemset $X_2 \in Pi$
3. if ($X_2 \notin C_2$)
4. X_2.count=NPi(X_2)×W(Pi);
5. X_2.start = i;
6. if (X_2.count ≥ min_supp × |Pi|×W(Pi))
7. $C_2 = C_2 \cup X_2$;
8. if ($X_2 \in C_2$)
9. X_2.count = X_2.count + NPi(X_2) ×W (Pi);
10. if (X_2.count < min_supp ×$\sum$m=X_2.start,i(|Pm|×W(Pm)))
11. $C_2 = C_2 - X_2$;
12. end
13. end

Procedure III: Candidate k-Itemset Generation
1. begin while (Ck ≠∅ & k ≥ 2)
2. $C_{k+1} = C_k * C_k$;
3. k = k +1;
4. end

Procedure IV: Frequent Itemset Generation
1. begin for i = 1 to n
2. begin for each itemset $X_k \in C_k$
3. X_k.count = X_k.count + NPi (X_k)×W(Pi);
4 end
5. end
6. begin for each itemset $X_k \in C_k$
7. if (X_k.count ≥ min_supp ×$\sum$m=1,n(|Pm|×W(Pm)))
8. $L_k = L_k \cup X_k$;
9. end
10. return Lk;

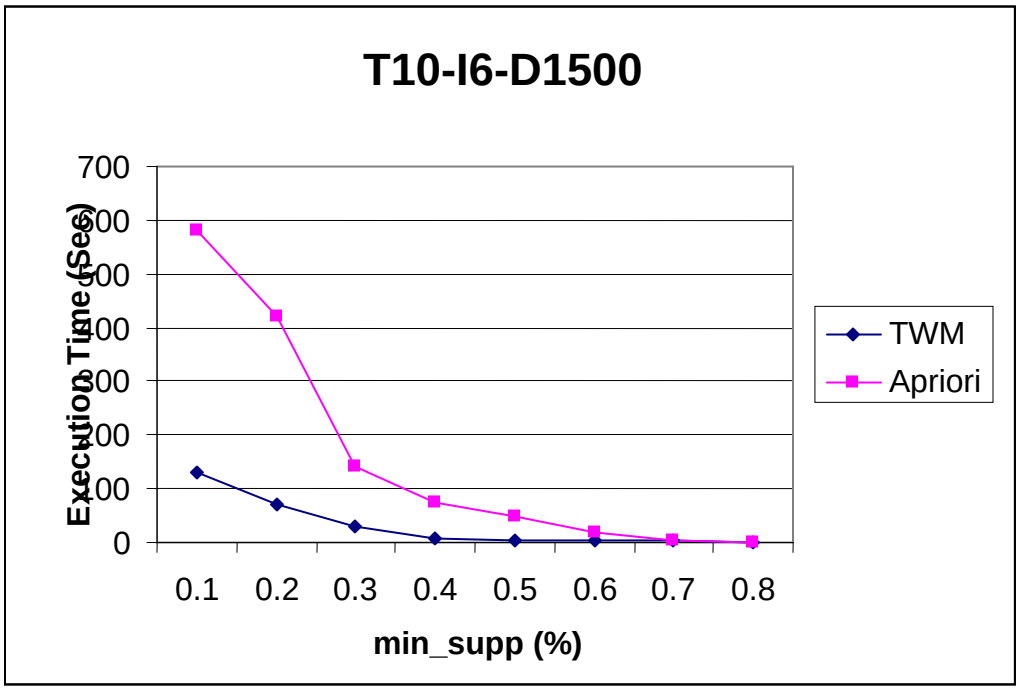
4. PERFORMANCE STUDIES

To assess the performance of algorithm EM, we performed several experiments on a computer. I have implemented EM in Visual Basic and MS Access as a backend. The performance comparison of EM and Apriori^W is presented in Section 4.1

4.1 Relative performances

Note that as pointed out earlier, there is essentially no restriction on the form of weighting functions. In all the experiments shown we take transaction length 10, average length of frequent itemset is 5 and 1500 transactions in database D.

We use the notation Tx−Iy −Dm to represent a database in which D = m transaction, |T| = x, and |I| = y ie T10-I5-D1500 for x=10, y=5 and m=1500.
We take a synthetic database of 4 years of transaction for our experimental results.



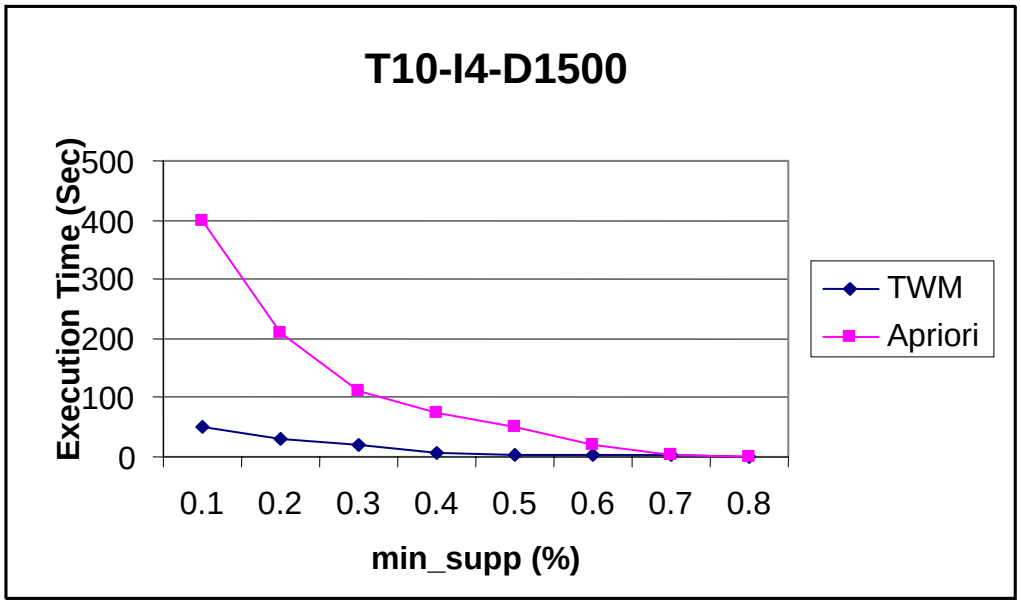


Figure 4.1: Relative performance studies between EM and Apriori[W]

5. CONCLUSIONS

In this paper we design and evolutes the performance of EM (Time Weight Miner) with a synthetic data set in which time constraint play a major role. In experimental result it is shown that the three problems can be effectively solve with the help of EM.

Problem 1. An early product intrinsically possesses a higher likelihood to be determined as a frequent itemset Problem 2. Some discovered rules may be expired from users' interest
Problem 3. Different transactions are usually of different importance to the user

In addition to above problem we have seen that EM generate candidate itemsets as minimum as possible when support value decrease gradually to a lower value. Due to this in candidate generation step time taken by EM is very less in comparison to older algorithm in which we don't consider time factor.

We also had shown the execution time comparison by graphs with Apriori Algorithm, that shows EM takes less time to Apriori.

REFERENCES:

[1] Arun K Pujari. Data Mining: Techniques. University Press (India) Private Ltd, 2005

[2] C.-H. Lee, C.-R. Lin, and M.-S. Chen. Sliding-Window Filtering: An Efficient Algorithm for Incremental Mining. Proc. of the ACM 10th Intern'l Conf. on Information and Knowledge Management, November 2001

[3] J. Ale and G. Rossi. An Approach to Discovering Temporal Association Rules. ACM symposium on Applied Computing, 2000

[4] J. Han and M. Kamber. Data Mining: Concepts and Techniques. Morgan Kaufmann Publishers, 2000

[5] R. Agrawal, T. Imielinski, and A. Swami. Mining Association Rules between Sets of Items in Large Databases. Proc. of ACM SIGMOD, pages 207—216, May 1993

[6] W. Wang, J. Yang, and P. Yu. Efficient mining of weighted association rules (WAR). Proc. of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2000