Temporal Weighted Association Rule Mining for Classification

Purushottam Sharma and Kanak Saxena

***Abstract*—There are so many important techniques towards finding the association rules. But, when we consider the sale of seasonal items (Winter, Summer, Spring etc.) or weighted items then no effective algorithm or model existed till now that can able to mine the interesting pattern on time–variant seasonal database. In view of this, we propose an optimized Temporal Weighted Association Rule miner (abbreviatedly as TWARM) algorithm to perform the mining for these problems and we conduct the corresponding performance studies by implementing this algorithm and then we used these weighted association rules for designing a good classifier to classify the items.**

Furthermore, without fully considering the time-changing characteristics or behavior 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 and required information in other season. Under TWARM we first partition the database on the bases of seasons (winter, summer, spring) or Yearly, Half Yearly or Quarterly etc. according to user's requirements and then we apply temporal weighted mining on each partition. In TWARM (Temporal Weighted Association Rule Miner) the cumulative occurrence count of mining previous partitions is selectively carried over toward the generation of candidate itemsets for the subsequent partitions. We have also applied scan reduction technique in TWARM due to that only two scan of database are required, means saving lots of time.

Now we have all Temporal Weighted Association Rules for Classification (TWARC) with the help of TWARM. By using these Temporal Weighted Association Rules we design a classifier to classify the items towards the appropriate class symbol.

***Index Terms*—Weighted transaction, time weighted mining, seasonal mining, temporal weighted association rule miner, Classification based on weighted association.**

I. Introduction

Association rule mining (ARM) is an important mining technique in the history of data mining. We try to find out the hidden relationship among the different attributes of a dataset in association rule mining for decision analysis, marketing analysis and business management purpose.

One popular application of ARM is the market basket analysis, in which we try to find out the customer's buying pattern for better customer management. In association rule mining, basically two term confidence threshold and support thresholds are used.

Manuscript received June 16, 2012; revised August 2, 2012.

P. Sharma is with the S. A. T. I. Vidisha (M. P), India (e-mail: puru_mit2002@rediff.com)

K. Saxena is with the Department of Computer Application, S. A. T. I. Vidisha (M. P), India (e-mail: kanak.saxena@gmail.com)

According to association rule mining algorithm, For a given pair of confidence and support thresholds, the problem of finding 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 [1] 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.

On the other hand, a seasonal database consists of values or events varying with time. Seasonal databases are popular in many applications, such as medical treatments, sales for seasonal items, 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 seasonal occurrences of individual transactions;
2) Lack of weighted support calculation for each item.

Some of the effects of this phenomenon may be like this.

1) Some old age product would be more frequent in comparison to new ones.

If we apply the traditional association rule mining approach then the early product comes out to be more frequent in comparison to the later one.

2) Some association rules are not so interested for the users.

This happens due to the time span factor of short life products.

3) Some transaction can be more important in comparison with other.
4) Effective classification after effective association rule mining can produce more accurate and faster useful results.

In this regard, we propose a Temporal Weighted Association Rule miner (abbreviatedly as TWARM) algorithm to perform mining for this problem. We also conducted the corresponding performance studies of TWARM.

II. Problem Descriptions

Initially divide the database D based on seasonal time granularities into n partition. In the model TWARM, Px denotes the part of the transaction database where Px is a subset of D. We explore in this research work, the mining of weighted association rules, i.e., $(A \Rightarrow B)^W$, where $A \Rightarrow B$ is a weighted association rule produced by the concepts of weighted support and weighted confidence. In TWARM we are not using traditional support threshold as in Apriori, i.e support=Probability (*AUB*), instead of that we are using a weighted minimum support for finding an association rules, and that is determined by $\text{min\_S}^W = \{\Sigma\ |Px|\ \times W\ (Px)\} \times$ min_supp where $|Px|$ and $W\ (Px)$ represent the number of partial transactions and their corresponding weight values by a weighting function $W\ (Px)$ in the corresponding weighted period of the database *D*.

III. Temporal Weighted Association Rule Mining

It is noted that most of the previous studies, including those in [1], [2] belong to Apriori-like approaches. Basically, an Apriori-like approach uses an anti-monotone Apriori heuristic [3], 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, performance of which will be comparatively evaluated with algorithm TWARM 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 [4], 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. Here * symbol is used to show the set theoretic join operation with a condition, and the condition is that candidate itemset C_k is joinable with C_k if there first k-1 items are in common. In TWARM candidate itemsets are sorted in alphabetical order first so that joining process becomes faster.

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 has been seen that the incremental mining technique used in algorithm TWARM will enable TWARM to obtain candidate set C_k with the size very close to that of L_k. This feature of TWARM allows itself of fully utilizing the technique of scan reduction and leads to prominent performance improvement over Apriori^W.

A. Algorithm of TWARM

Initially, a time-variant database D is partitioned into n partitions based on the weighted periods of transactions. The procedure of algorithm TWARM is outlined below, where algorithm TWARM 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 NPx(X) is the number of transactions in partition Px that contain itemset X and $W\ (Px)$ is the corresponding weight of partition *Px*.

Range of *W (Px):* In this experiment we have taken the range of $W\ (Px)$ from 1 to 5 but values of $W\ (Px)$ can be any values in the specified range and should be decided by expert of that dataset because if we have two events in a temporal database then weights can be different for both the events even the events are same, it may depends on the occurrence order of events also. Like in a medical database if two events fever and surgical operation occurs on a patient, then the weight of these two events may be different based on the occurrence instance. i.e if fever come before the surgical operation then it can be due to fear of surgical operation to the patient but if fever comes after surgical operation then it can be the result of infection. So the weights will be different for both the situation even though events are same.

Algorithm TWARM (n, min_supp): Temporal Weighted Association Rule miner

Procedure I: Partition the database according to season:

Initial Partition, based on time i.e. yearly, half yearly, Quarterly etc based on seasons.

Procedure II: Generation of Candidate 2-Itemset:

Starting from the first partition, generate all candidate-2 itemset with frequency in the first scan of database.

Procedure III: Generation of Candidate k-Itemset:

Generate candidate-k itemset by using scan reduction technique

Procedure IV: Frequent Itemset Generation:

Starting with first partition

Find out all frequent itemset if minimum weighted support count is satisfied.

IV. Performance Studies

To assess the performance of algorithm TWARM, we performed several experiments on a computer with Pentium-4 3.2GHz processor in Microsoft Windows environment. We have implemented TWARM in Visual Basic as front end and MS Access as a backend. We have also performed some experiments with real dataset with categorical attributes. For continuous valued attributes we have to first apply the Discretization rules to convert into categorical format. After discretization we use direct coding techniques to replace the actual attribute name with a single alphabet (i.e *A*, *B*, *C*…) so that in candidate generation step sorting would be easy and candidate generation would be faster. But for accurate time measurement we have taken here a synthetic data set for around 15000 transactions with different attributes. The performance comparison of TWARM and Apriori^W is presented in Section IV *A*.

A. 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 15000 transactions in database *D*.

We use the notation *Tx*−I*y* −Dm to represent a database in which *D* = m transaction, |*T*| = *x*, and |*I*| = *y* ie *T*10-*I*5-*D*15000 for *x*=10, *y*=5 and *m*=15000.

Here we take a synthetic database of 4 years of transaction for our experimental results.

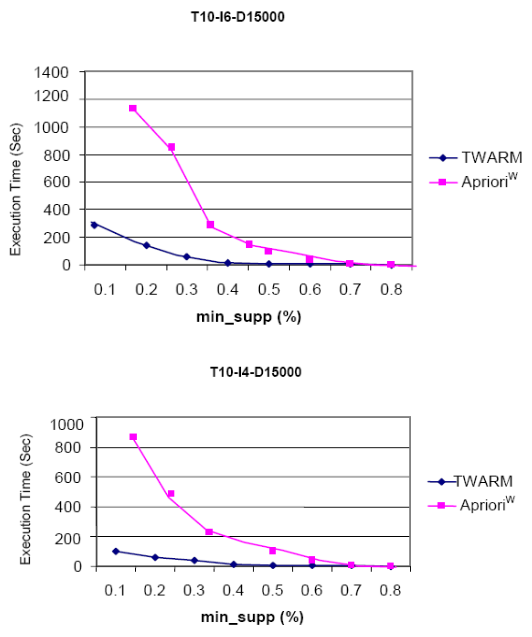

Fig. 1. Relative performance studies between TWARM and Apriori^{W}

V. Comparative Study of Twarm Between Different Time Granularities

A. *Comparative Study of TWARM between Time Granularity Half Yearly and Quarterly for Average Weights and with Different Minimum Support Values.*

TABLE I: Half Yearly Results of Average Weights with different minimum Support Values Half Yearly。

Weights		MinSupp	$\|C_2\|$	$\|C_k\|$	$\|L_k\|$	Length (C_k)	Length (L_k)
Jan-01	1.5	0.4	1869	1070	102	5	3
Jul-01	1.5						
Jan-02	2						
Jul-02	2						
Jan-03	2.5	0.3	1982	1095	112	6	4
Jul-03	2.5						
Jan-04	3						
Jul-04	3	0.2	2188	1236	132	9	5

TABLE II: Quarterly Results of Average Weights with Different Minimum Support Values Quarterly.

Weights		Min Supp	$\|C_2\|$	$\|C_k\|$	$\|L_k\|$	Length (Ck)	Length (L_k)
Jan-01	1	0.4	1755	1030	101	5	3
May-01	1						
Sep-01	1						
Jan-02	1.3						
May-02	1.3						
Sep-02	1.3	0.3	1913	1072	107	6	4
Jan-03	1.7						
May-03	1.7						
Sep-03	1.7						
Jan-04	2						
May-04	2						
Sep-04	2	0.2	2085	1133	114	9	5

B. *Comparative Study of TWARM between Time Granularities Half Yearly and Quarterly for Varying Weights and with Different Minimum Support Values.*

TABLE III: Half Yearly Results of varying Weights with Different Minimum Support Values Half Yearly.

Weights		MinSupp	$\|C_2\|$	$\|C_k\|$	$\|L_k\|$	Length (C_k)	Length (L_k)
Jan-01	1	0.4	2064	1188	101	5	3
Jul-01	2						
Jan-02	1						
Jul-02	3						
Jan-03	2	0.3	2079	1193	112	6	4
Jul-03	3						
Jan-04	2						
Jul-04	4	0.2	2088	1237	132	9	5

TABLE IV: Quarterly Results of Varying Weights with Different Minimum Support values Quarterly.

Weights		Min _Supp	$\|C_2\|$	$\|C_k\|$	$\|L_k\|$	Length (C_k)	Length (L_k)
Jan-01	1	0.4	1969	1098	100	5	3
May-01	1						
Sep-01	1						
Jan-02	1						
May-02	1						
Sep-02	2	0.3	1994	1103	103	6	4
Jan-03	1						
May-03	2						
Sep-03	2						
Jan-04	1						
May-04	2						
Sep-04	3	0.2	2026	1127	111	8	5

C. Analysis of Tables

We can see by the values of Table I (Half Yearly) and Table II (Quarterly) that when we lower down the values of minimum support from 0.4 to 0.2, number of candidate-k itemsets is reduced from 1236 (min_sup=0.2, half early) to 1133 (min_sup=0.2, Quarterly) even though weights equally scattered in database and count and length of frequent itemsets are same. So mining time will be smaller. This shows that Time factor play a major role if considered appropriately.

In the same manner we can also see by the values of Table III (Half Yearly) and Table IV (Quarterly) that when we decrease the values of minimum support from 0.4 to 0.2, number of candidate-k itemsets is reduced from 1237(min_sup=0.2, half early) to 1127(min_sup=0.2, Quarterly) when weights scattered integrally in database and count and length of frequent itemsets are same. So mining time will be smaller.

These results imply that if we appropriately consider and use the time information on the dataset then the mining result we get will be more appropriate and mining time will be lesser.

VI. Classification.

After getting the entire weighted association rule set we have a complete rule space. Now we have to do classification with this rule space. Initially we select those rules that have class symbol in their consequent.

Let A be the set of all attributes i.e. fields in data set. Let *C* be the set of all possible classes which are used to categorize transactions in *D*, where $C \subset A$. Then the rule $(X => Y)^W$ is said to be Temporal weighted classification association rule if $Y \subset C$. According to the classification process Y needs to consist of only one $A_i \in$ A since any transaction can not belongs to multiple classes.

To find out the Temporal Weighted Association Rules, we have two techniques, first one is to find out all the weighted association rules first and then perform classification and second one is to perform association and classification simultaneously in a controlled way. In this work we used first technique.

A. Ordering TWARCs and Building a Classifier

There are several methods existed in data mining literature to order the rule set such as CSA, WRA, and Laplace Accuracy [5], [6]. In this work, CSA (confidence, support, and size of antecedent) approach is used. The first factor that determines rules order is their confidence value: the higher rule's confidence is, the lower its order number is. If we have two rules having exactly same confidence then we look for their supports. Again higher support is preferred. Finally, for the rules with same confidence and support, the smaller rule is placed before the longer one. We have also used a default class in TWARC for most seen class in data set [7]-[17].

The classifier is presented here

B. Temporal Weighted Association Rule Classifier

1) Select only those weighted association rules generating by TWARM that have a class attributes in that rule. Let's call this rule set Temporal Weighted Association Rules for Classification (TWARC).
2) Order temporal weighted association rules according to CSA (confidence, support, and size of antecedent)
3) Test the given transaction starting from higher priority rules to lower priority rules to find out the corresponding class.
4) If no rule is capable to decide the class then default class is assigned to that transaction.

VII. Conclusions

In this paper we design and evolutes the performance of TWARM (Temporal Weighted Association Rule miner) with a synthetic data set in which time constraint play a major role. In experimental result it is shown that the some problems can be effectively solved with the help of TWARM and TWARC.

1) Some old age product would be more frequent in comparison to new ones.
2) Some association rules are not so interested for the users
3) Some transaction can be more important in comparison with other.
4) Effective classification after optimized association rule mining can produce more accurate and faster useful results

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

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

References

[1] J. Ale and G. Rossi, "An approach to discovering temporal association rules," *ACM symposium on Applied Computing*, 2000

[2] J. Han and M. Kamber, "Data mining: Concepts and techniques," *2 nd ed, Morgan Kaufmann Publishers*, 2000, ch. 5 and 6 pp. 227-377.

[3] 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

[4] X. Yin and J. Han, "CPAR: Classification based on predictive association rule," *In SDM 2003*, San Francisco, CA

[5] F. Thabtah, P. Cowling, and Y. Peng, "Comparison of classification techniques for a personnel scheduling problem," In *Proceeding of the 2004 International Business Information Management Conference*, Amman, July 2004.

[6] B. Tunc and H. DAG, "Generating classification association rules with modified Apriori," In *International Conference on Artificial Intellegence, Knowledge Engineering and Data Bases*, Madrid, Spain, February 15-17, 2006

[7] R. Agrawal, T. Imielinski, and A. Swami, "Mining association rules between sets of items in large databases," *Proc. of ACM SIGMOD*, pp. 207-216, May 1993

[8] C.-H. Lee, J. C. Ou, and M.-S. Chen, "Progressive weighted miner: An efficient method for time-constraint mining."

[9] M. J. Zaki, S. Parthasarathy, M. Ogihara, and W. Li, "New algorithms for fast discovery of association rules," *3rd KDD Conference*, pp. 283-286, August 1997.

[10] M.-S. Chen, J.-S. Park, and P. S. Yu, "Efficient data mining for path traversal patterns," *IEEE Transactions on Knowledge and Data Engineering*, vol. 10, no. 2, pp. 209-221, April 1998.

[11] I. Witten and E. Frank, "Data mining: practical machine learning tools and techniques with Java implementations," *Morgan Kaufmann, San Francisco*, 2000.

[12] 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.

[13] C. Leng, "An evaluation of approaches to classification rule selection," In *Proc. of the IEEE ICDM*, 2004, pp. 359-362.

[14] R. J. B. Jr., R. Agrawal, and D. Gunopulos, "Constraint-based rule mining in large, dense databases," *Data Mining and Knowledge Discovery*, vol. 4, no. 2/3, pp. 217-240, 2007.

[15] Ayad, N. El-Makky, and Y. Taha, "Incremental mining of constrained association rules," *Proc. of the First SIAM Conference on Data Mining*, 2008.

[16] CBA. Data mining tool. Downloading page. [Online]. Available: http://www.comp.nus.edu.sg/~dm2/p_download.html. Viewed on February 2010

[17] Weka. Data Mining Software in Java. [Online]. Available: http://www.cs.waikato.ac.nz/ml/weka. Viewed on February 2010.