International Journal of Electronic and Electrical Engineering.
ISSN 0974-2174 Volume 6, Number 2 (2013), pp. 77-85

http://www.irphouse.com

Efficient Association Rule Mining using Hybrid Techniques

Venu Mishra[1] and Dr. Meenu Dave[2]

[1]M.Tech. Scholar, [2]Assistant Professor,
Department of CSE, Jagan Nath University, Jaipur, India
Venu0110mishra@gmail.com, meenu.dave@jagannathuniversity.org

Abstract

As the information is increasing day by day, there is abundance of data but it is not very easy to draw the accurate or required knowledge. Improved and efficient mining techniques are needed to acquire the useful knowledge. In this paper, an algorithm is proposed that is based on count and record filtering techniques. Count based method is used for pruning of candidate itemset and record filtering technique is used to reduce data scan. Experiments show that this algorithm reduces the overall time of mining.

Keywords: ARM, AIS, SETM, Apriori, DIC, support, confidence

INTRODUCTION

With the betterment of technology and increase in the data generated, new data storage techniques like cloud computing and service oriented architecture are needed for integrating scattered data and finding interesting information out of that, and this is a new growing challenge. This information cannot be extracted by the traditional methods such as firing of queries or statistical analysis. It can be used in classification of the data and in prediction of future events etc. So far many techniques have been implemented that are used in data mining. Association rule mining is one of the techniques.

Classical algorithms for mining have two major problems first is the redundant candidate generation and the other is the number of database scans. In this paper some classical and improved mining algorithms are discussed but still there is a need to improve the quality of the mining algorithms by reducing the candidate generation and database scans. In section II, frequent pattern mining is discussed. Section III covers the proposed algorithm concepts. Section IV shows the experimental results of the proposed algorithm and the Apriori algorithm.

FREQUENT PATTERN MINING

Frequent pattern mining is to find the frequent and interesting patterns from the large databases. Frequent patterns are then used to create the association rules for further predictions, that is called association rule mining. We can define association rule mining problem as: Let DB_T be a database of transactions; each transaction consists of I, where I is $\{i_1, i_2, i_3 \ldots i_n\}$ items. Association rules are in the form of

$A \rightarrow B$, where $A \subset I, B \subset I$ *and* $A \cap B = \phi$.

Each association rule has support and confidence that specifies the significance of the association rule. Support denotes the occurrence of item in the database DB_T. Confidence is the proportion of the data items containing B in all the items containing A also, in DB_T.

$$Sup(A) = Count(A) / Count(DB_T)$$

$$Sup(A \rightarrow B) = Sup(A \cup B)$$

$$Conf(A \rightarrow B) = Sup(A \cup B) / Sup(A)$$

If the support and confidence of the rule is greater or equal to the threshold value minimum support and minimum confidence, then the rule is considered as a valid rule, otherwise it is discarded. The objective of the ARM is to find the set of the valid association rules [1, 2].

Many algorithms were proposed to perform ARM on transactional databases. Comparative analysis of some of them is described below:

Apriori Algorithm

Apriori algorithm states that itemset x containing subset itemset y is never frequent if y is not frequent. Based on this principle, Apriori generates new itemsets of length K+1 using k frequent itemsets. And eliminates rest of the elements, which have infrequent itemsets. So Apriori generates new itemsets by using frequent itemsets of previous itemset without considering transaction. But it considers transaction for counting of support of the new candidates. Problem with this algorithm is that, it is more time consuming because it generates redundant candidates itemsets, and it requires database scan at each pass for counting support of the itemset[3, 4].

AprioriTid & AprioriHybrid

This algorithm uses the concept of the Apriori, and is better than Apriori in terms of database scan. It makes TIDs at each pass. It just scans database in the first pass. After that, it uses Tid's for counting, but the major downside of this algorithm is the generation of the Tid table at each pass.

Another algorithm, called Apriori Hybrid, introduced in [6], uses the combination of Apriori and aprioriTid. Idea behind this algorithm is to run the Apriori algorithm initially, when transactions are large and then switch to the AprioriTid algorithm when the generated database, i.e. large k itemset in the transaction with identifier TID, would fit in the memory[10].

Partition Algorithm

Partitioning algorithm[8], uses the concept of partitioning the database in parts. Algorithm consists of two phases. In the first phase, we partition the database in n partitions that are not overlapping and can fit into the memory; and start mining on the partitions. With every iteration, one partition is considered which finds frequent itemsets that are local to that partition. In second phase, these local frequent itemsets are combined to find global frequent itemsets[4, 10].

Dynamic Itemset Counting(DIC)

In DIC, candidate (k+1) itemsets are counted as soon as the algorithm discovers that all its subsets of size k have exceeded the support threshold and will be frequent. This is done by stopping at various points in the database to examine the possibility of including other itemsets in the counting procedure. It has been found that such techniques, with reasonable setting of the number of transactions passed before stopping for recalculation, can reduce the number of database passes dramatically while maintaining the number of candidate sets that need to be counted relatively less compared to other proposed techniques [5, 10].

Algorithms discussed so far are all mining algorithms but still all have some deficiencies and none is perfect on all type of datasets. Some of the shortcomings of the ARM algorithms are given in Table 1:

TABLE 1: Shortcomings of the Existing Algorithms [8, 9, 10]

S.No.	Algorithm	Drawbacks
1	AIS	AIS algorithm unnecessarily generates and counts too many candidate item sets. This algorithm requires too many passes over the whole database.
2	SETM	The problem with the SETM algorithm is that candidates are replicated for every transaction in which they occur, which results in huge sizes of intermediate results.
3	Apriori	Problem with this algorithm is that it is more time consuming because it generates redundant candidate itemsets and it requires database scan at each pass for counting support of the itemset.
4	AprioriTid & AprioriHybrid	The size of database is limited to the main memory size. Second problem is the pruning of the database in the later stages of the algorithm. i.e., removing the part that will not be used further for mining process.
5	Dynamic itemset counting	Used only to find the frequent items; can't used to generate association rules.
6	Partitioning algorithm	Sometimes, all frequent patterns are not found.

7	FP-growth	It requires several scans over the database for the construction of the FP-Tree. Whole mining process should be repeated whenever the support value is changed, as well when a new dataset is inserted into the database.
8	Sampling	All frequent patterns are not found.

PROPOSED ALGORITHM (ICA)

Process of this algorithm can be divided in two steps after the generation of 1st candidate set. One step is the prune step and the other is count step. So we will use count based technique to improve the pruning process and e record filter technique improves the database scan to count occurrence of candidates. Details of these two methods are described below:

Count based candidate pruning method

Let L_k denote the set of k-dimensional frequent itemsets and C_k denote the set of frequent itemsets. $\overline{L_k}$ denotes the complement of L_k set.

Theorem1: If any itemset X is frequent then every subset of X will also be frequent [1].

Deduction 1: If itemset X is infrequent then every superset of X is also infrequent.

To check if c is a frequent itemset or not, the apriori algorithm uses L_{k-1}, if $\exists s \notin L_{k-1}$, where s is subset of c, then c is not frequent itemset. In contrast to Apriori, HDO Apriori algorithm [9] uses $\overline{L_{k-1}}$ to remove infrequent itemsets. The pruning time of Apriori and HDO Apriori algorithm is based on the reference collection of L_{k-1} *and* $\overline{L_{k-1}}$. Proposed algorithm uses a count based candidate prune operation.

Deduction 2: All k-dimensional frequent candidates are generated from (k-1)-dimensional frequent itemsets that have only one different itemset.

Proof: Let {L, a, b} and {L, d, c} be (k-1)-dimensional frequent itemsets with two different itemsets, so if k-dimensional itemsets {L, a, b, c} is frequent, using Theorem 1, {L, a, b} must also be frequent.

But {L, a, b, c} can be generated from {L, a, b} and {L, b, c}, so it concludes that generated set of {L, a, b} and {L, d, c} is redundant.

If k-dimensional itemset c is frequent, all its (k-1) dimensional itemsets are also frequent and two of these will generate *c* once. So if the total count is less than two, then it is sure that the item generated is not frequent itemset. Thus, it could be removed from further processing.

Count-based pruning function:

Input: (k-1)-dimensional frequent itemsets L_{k1}
Output: k-dimensional frequent itemsets C_k
Begin

$C_k = \phi$

For each $l_i \in L_{k-1}, l_j \in L_{k-1}, l_i \neq l_j$ {

If($l_i[1]=l_j[1] \wedge \ldots \wedge l_i[k-2]=l_j[k-2] \wedge l_i[k-1] \neq l_j[k-1]$)

$c = l1 \cup l2$

If $(c \in C_k)$

Increment count of c by 1.

Else

count=1, insert c into C_k

Endif

Endif

For each $c \in C_k$ {

If c.count<2

Remove c from C_k

Endif

}}

End

Figure 1: counting based method for pruning

Record Filter Approach

Record Filter approach improves the efficiency of Apriori by memory management and removes the complexity of process. This uses a different approach in Apriori algorithm to count the support of candidate item set. In the classical Apriori algorithm, counting process covers all the transactions in each pass. But, record filter approach counts the support of candidate set only in the transaction record whose length is greater than or equal to the length of candidate set. As candidate set of length k cannot exist in the transaction record of length k-1, it may exist only in the transactions of length greater than or equal to k.

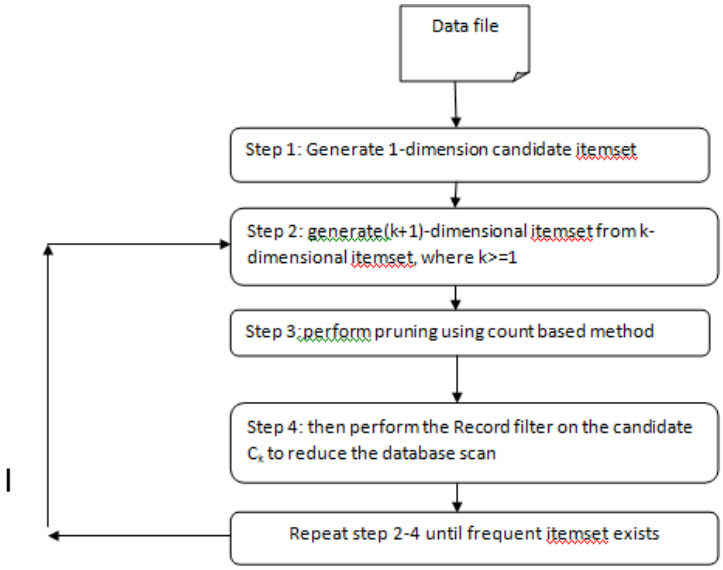

Figure 2: Flow chart to show steps of the algorithm

Improved counting based algorithm (ICA)

Input: DB_T, Sup_{min} *and* $Conf_{min}$
Output: association rules.
Begin
scan database get $L1$ / * L1 list of 1-itemset
k=2;

While ($L_{k-1} \neq \phi$){

//to perform pruning use count based method

C_k=count Based Pruning (L_{k-1})

For each $c \in C_k$ {

for all $t \in DB_T$ where length of $t.length >= k$ {
Increment the count of all candidates in C_K that are contained in t;}
}
Store L_k:=All candidates in $c \in C_k \mid Sup_c \geq \sup_{min}$;
k := k + 1;
}
Use L_k to generate association rules in $C \Rightarrow R$

Figure 3: Complete Proposed Algorithm

EXPERIMENTAL RESULTS

Analysis of this algorithm is done on 5.3GHz Intel® on window 7. Programs are coded on the platform of MATLAB 7.10.0(R2010a).

The experiment compares the runtime of the two algorithms Apriori and the proposed algorithm ICA.

First analysis is with the increasing transaction size and next one is with increasing the support count. Through given result we can see that there is Approximate 50% mining times is reduced by the ICA algorithm.

Next comparison is using the sampling technique that also shows the good results.

Measuring Performance With Increasing Database size

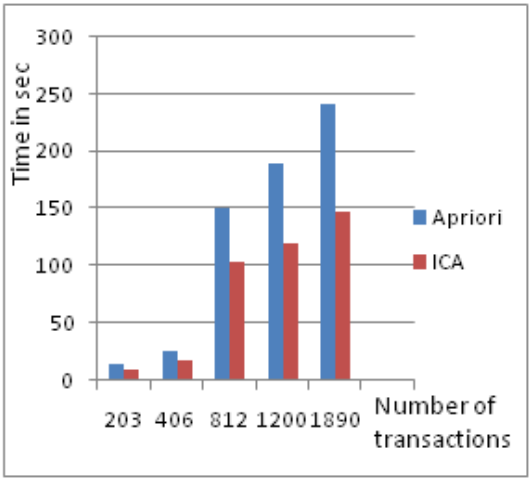

Figure 4: Comparative Analysis between Apriori and ICA with increasing number of transactions

Table 2: Comparative time for ICA and Apriori with increasing transactions

Number of transactions	Apriori	ICA
203	13.76	8.99
406	25.37	17.66
812	149.67	103.317
1200	188.55	118.691
1890	240.3	146.71

Measuring Performance With Increasing Support Count

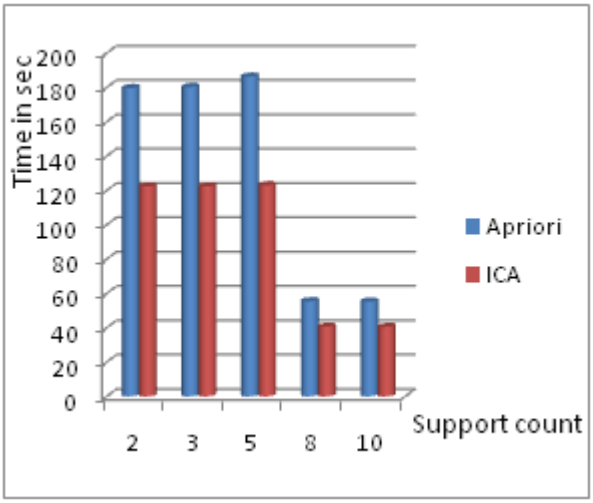

Figure 5: Comparative analysis of Apriori and ICA with variant support count

Table 3: Comparative time for ICA and Apriori with variant support count

Support count	Apriori (time in sec)	ICA (time in sec)
2	179.644	122.359
3	180.319	122.376
5	186.35	123.049
8	55.4627	40.62
10	55.2617	40.53

Measuring Performance With Sampling

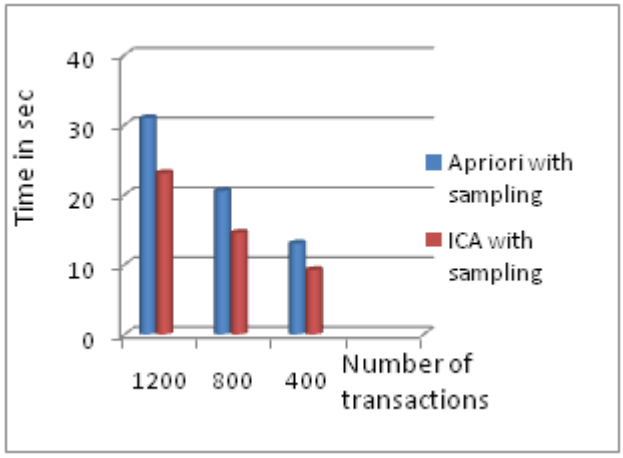

Figure 6: Comparative analysis of Apriori and ICA with sampling

Table 4: Comparative time for ICA and Apriori with sampling

Number of transactions	Apriori with sampling (time in sec)	ICA with sampling
1200	30.94	23.05
800	20.52	14.52
400	13.01	9.2

CONCLUSION

In this paper, an improved apriori algorithm is proposed that is based on counting based and record filter techniques. Pruning process of ARM is improved through count based technique and reduces the database scans through record filtering technique. This is notable using the experimental results and comparisons, where the proposed algorithm has improved the overall performance by reducing the time. This algorithm needs to be tested for incremental mining and constraint based mining and further research is required to implement this in the distributed environment.

ACKNOWLEDGEMENT

The authors thank Mr. Manish Gupta, Chancellor, Jagan Nath University, Mr. Deepak Gupta, Vice-Chairman, Jagan Nath Gupta Memorial Education Society, Prof. V.K. Agarwal, Vice Chancellor, Jagan Nath University, and Prof. Y.S. Shishodia, Pro-Vice Chancellor, Jagan Nath University, Jaipur, for providing encouragement and support for this research work.

REFERENCES

[1] Farah Hanna AL-Zawaidah and Yosef Hasan Jbara, “An Improved Algorithm for Mining Association Rules in Large Databases” in World of Computer Science and Information Technology Journal (WCSIT) ISSN: 2221-0741 Vol. 1, No. 7, 311-316, 2011

[2] P.Velvadivu1and Dr.K.Duraisamy, “An Optimized Weighted Association Rule Mining On Dynamic Content” in IJCSI International Journal of Computer Science Issues, Vol. 7, Issue 2, No 5, March 2010.

[3] Qian Wan, Student Member, IEEE, and Aijun An, Member, IEEE “Discovering Transitional Patterns and Their Significant Milestones in Transaction Databases” in IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, VOL. 21, NO. 12, DECEMBER 2009

[4] Qin Ding, Qiang Ding, and William Perrizo “PARM—An Efficient Algorithm to Mine Association Rules From Spatial Data” in IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS—PART B: CYBERNETICS, VOL. 38, NO. 6, DECEMBER 2008

[5] R. Agrawal, and R. Srikant, "Fast Algorithms for Mining Association Rules", In Proc. VLDB 1994, pp.487-499.

[6] Kun-Ming Yu, Jia-Ling Zhou, “A Weighted Load-Balancing Parallel Apriori Algorithm for Association Rule Mining”, Granular Computing, 2008. GrC 2008. IEEE International Conference on 26-28 Aug. 2008, pp.756-761

[7] Dongme Sun, Shaohua Teng, Wei Zhang, and Haibin Zhu, “An Algorithm to Improve the Effectiveness of Apriori”, Cognitive Informatics, 6th IEEE International Conference on 6-8 Aug. 2007, pp.385-390

[8] Colin Cooper, and Michele Zito, “Realistic Synthetic Data for TestingAssociation Rule Mining Algorithms for Market Basket Databases”, Knowledge Discovery in Databases: PKDD 2007, Volume4702/2007, pp.398-405

[9] Lei Ji, Baowen Zhang, and Jianhua Li, “A New Improvement on Apriori Algorithm”, Computational Intelligence and Security, International Conference on Volume 1, Nov. 2006, pp.840-844

[10] Venu Mishra, Tarun Mishra, Arun Mishra, ” Algorithms for Association rule mining: A General Survey on Benefits and Drawbacks of algorithms”, International Journal of Advanced Rearch in Computer Science, Volume 4, No. 8, May-June 2013 pp. 155-159