International Journal of Computer Applications Technology and Research
Volume 2– Issue 2, 166 - 169, 2013, ISSN: 2319–8656

An improvised tree algorithm for association rule mining using transaction reduction

Krishna Balan
Pondicherry Engineering College
Pondicherry, India

Karthiga
Pondicherry Engineering College
Pondicherry, India

Sakthi Priya
Pondicherry Engineering College
Pondicherry, India

Abstract: Association rule mining technique plays an important role in data mining research where the aim is to find interesting correlations between sets of items in databases. The apriori algorithm has been the most popular techniques in finding frequent patterns. However, when applying this method a database has to be scanned many times to calculate the counts of the huge umber of candidate items sets. A new algorithm has been proposed as a solution to this problem. The proposed algorithm is mainly concentrated to reduce the candidate sets generation and also aimed to increase the time of execution of the process.

Keywords: Apriori; association; candidate sets; data mining; itemsets;

1. INTRODUCTION

As the rapid growth of the information technology the data's has been stored in the form of digital systems. As tremendous amounts of data are thus generated due to the full digitization. Data mining plays an important role to extract meaningful information from the scattered data. Association rule is a popular technique which is used for finding interesting relationship between variables in large databases. R. Agrawal and R. Srikant in 1994 presented the apriori algorithm for mining frequent itemsets which is based on the generation of candidate itemset. Several different algorithms have been proposed for association rule. In this paper a new algorithm has been proposed for association rule. The proposed algorithm has been implemented by comparing all the demerits of the existing systems. The main goal of the proposed system is to speed up the computation process.

2. RELATED WORKS

There are various algorithms were proposed for finding the frequent itemsets . The best well known for the rule mining is the apriori algorithm which was proposed by the Agrawal and Srikant (1994) [2]. This uses the concept of candidate generation. Although the apriori algorithm is efficient in finding the item sets the execution time gets longer when the database size increases since it has to generate the candidate item-sets. Many algorithms have been proposed to overcome the drawbacks of the a priori such as the FP-Growth algorithm [4] were proposing a new idea for the candidate set generation problem where it introduces a Tree structure concept where it distributes the workload as it relies on the depth first search. It implies that it is faster than the apriori where there is no candidate generation as it uses the divide and conquer approach such that the database is scanned only twice. The matrix Apriori algorithm[6] is proposed in order to improve the efficiency time of the apriori algorithm where this uses the combined approach of the apriori and the fp-growth algorithm. The description and implementation of the above algorithms are briefly explained below.

2.1 Apriori Algorithm [2]

One of the first algorithms to evolve for frequent itemset and Association rule mining was Apriori. Two major steps of the Apriori algorithm are the join and prune steps. The join step is used to construct new candidate sets. A candidate itemset is basically an item set that could be either Frequent or infrequent with respect to the support threshold. Higher level candidate itemsets (Ci) are generated by joining previous level frequent itemsets are Li-1 with it. The prune step helps in filtering out candidate item-sets whose subsets (prior level) are not frequent. This is based on the anti-monotonic property as a result of which every subset of a frequent item set is also frequent.Thus a candidate item set which is composed of one or more infrequent item sets of a prior level is filtered(pruned) from the process of frequent itemset and association mining.[4]

2.2 FP-Growth Algorithm [6]

The FP-Growth methods adopts a divide and conquer strategy as follows: compress the database representing frequent items into a frequent-pattern tree, but retain the itemset association information, and then divide such a compressed database into a set of condition databases, each associated with one frequent item, and mine each such database [8].

First, a scan of database derives a list of frequent items in descending order. Then the FP - tree is constructed as follows. Create the root of the tree and scan the database second time. The items in each transaction are processed in the order of frequent items list and a branch is created for each transaction. When considering the branch to be added

to a transaction, the count of each node along a common prefix is incremented by 1. After constructing the tree the mining proceeds as follows. Start from each frequent length-1 pattern, construct its conditional pattern base, then construct its conditional FP-tree and perform mining recursively on such a tree. The support of a candidate (conditional) itemset is counted traversing the tree. The sum of count values at least frequent item's nodes gives the support value.

2.3 DH Algorithm [1]

In order to improve the execution time of the apriori algorithm there are many algorithms have been implemented. The DH (Direct Hashing) algorithm has been proposed for reducing the database rescanning. The DHCP algorithm is an effective hash based algorithm for the candidate set generation. It reduces the size of the candidate set of filtering any k item set out of the hash table if the hash entry does not have minimum support. The hash table structure contains the information regarding the support of each item set. The DHP algorithm consists of three steps. The first step is to get a set of large itemsets and constructs a hash table for 2 itemset. The second step generates the set of candidate itemsets Ck. The third step is the same as the second step except it does not use the hash table in determining whether to include a particular itemset into the candidate itemsets.

2.4 Transaction Reduction Algorithm:

The classical Apriori algorithm generates a large number of candidate sets if the database is large. And due to large number of records in database results in much more I/O cost. In this project, we proposed an optimized method for Apriori algorithm which reduces the size of the database. In our proposed method, we introduced an attribute Size_Of_Transaction (SOT) , containing a number of items in individual transaction in the database. The deletion process of transaction in database will made according to the value of K. Whatever the value of K, the algorithm searches the same value for SOT of the database. If the value of K matches with a value of SOT then delete only those transactions from the database.

3. PROPOSED SYSTEM

3.1 Improved Apriori Algorithm

Our Improved Apriori algorithm which uses the data structure which represents the hash table. This algorithm proposes to overcome the weakness of Apriori by reducing the candidate sets. The proposed algorithm does a three stage process where the first process is a hash based step is used to reduce the candidate itemsets generated in the first phase. We assure that the number of itemset generated using hashing can be reduced. In this algorithm each transaction counts all the itemset at the same time possible 2-itemsets in the current transactions are hashed to a hash map. After the 2-itemset the individual items which has less frequent are deleted from the transaction database and the final step is the construction of a tree where we apply a divide and conquer strategy for mining the frequent itemsets from the transaction database. And in this process the frequent itemsets are listed in descending order. A root of the tree is constructed first and then the branches are added according to the count of the itemset. Once the tree is constructed the frequent itemsets are minined by traversing through the tree. Since the construction of the tree is made simple as by reducing the items from the transaction database.

The algorithm is as follows

Input: The Transaction database and the minimum support.
Output: All the frequent itemsets in the transaction database.
The following is the description of the algorithm

1. The transaction database is scanned and create a possible 2-itemsets.
2. Let the Hash table of size 8.
3. For each bucket assign a candidate pair using the ASCII values of the item sets.
4. Each bucket in the hash table has a count, which is increased by 1 each item an item set is hashed to that bucket.
5. If the bucket count is equal or above the minimum support count, the bit vector is set to 1. Otherwise it is set to 0.
6. The candidate pairs that hash to locations where the bit vector bit is not set are removed.
7. Modify the transaction database to include only these candidate pairs
8. Then the candidate itemsets which has less frequent are then removed from the transaction database.
9. And the database is scanned for minimum support threshold, frequent items are selected and sorted.
10. Initialization of the FP-tree is done. From the frequent items a node list is created which will be connected to nodes of the tree. After initialization the database is read again. This time, if an item in a transaction is selected as frequent then it is added to the tree structure.
11. Beginning of the least frequent item, a frequent pattern finder procedure is called recursively. The support count of the patterns is found and displayed if they are frequent.
12. End.

Table 1. Transaction Database

TID	ITEMS
T1	I1, I3, I7
T2	I2, I3, I7
T3	I1, I2, I3
T4	I2, I3
T5	I2, I3, I4, I5
T6	I2, I3
T7	I1, I2, I3, I4, I6
T8	I2, I3, I4, I6
T9	I1
T10	I1, I3

Table 2.Hash Table Data Format

TID	ITEM SET
T1	I1I3, I1I7, I3I7
T2	I2I3, I2I7, I3I7
T3	I1I2, I1I3, I2I3
T4	I2I3
T5	I2I3,I2I4,I2I5,I3I4,I3I5,I4I5
T6	I2I3
T7	I1I2,I1I3,I1I4,I1I6,I2I3,I2I4,I2I6,I3I4,I3I6,I4I6
T8	I2I3,I2I4,I2I6,I3I4,I3I6,I4I6
T9	I1
T10	I1I3

HASH COUNT:

{I1I3}=4,{I1I7}=1,{I3I7}=2,{I2I3}=7,
{I2I7}=1,{I3I7}=2,{I1I2}=2,{I1I3}=3,
{I2I4}=3,{I2I5}=1,{I3I4}=3{I3I5}=1,
{I4I5}=1,{I1I4}=1,{I1I6}=1,
{I2I6}=2,{I3I6}=2,{I4I6}=2.

MINIMUM SUPPORT=3 ,

FREQUENT ITEM SET {I1I3, I2I3, I3I4},
FREQUENT ITEM SET= {I1, I2, I3, I4}

Table 3. Transaction Reduction Table

TID	ITEMS
T1	I1, I3
T2	I2, I3
T3	I1,I2,I3
T4	I2,I3
T5	I2,I3
T6	I2,I3
T7	I1,I2,I3,I4
T8	I2,I3,I4
T9	I1
T10	I1,I3

Table 4. Maximum Item Set Count

ITEMS	COUNT
I3	8
I2	7
I1	5
I4	3

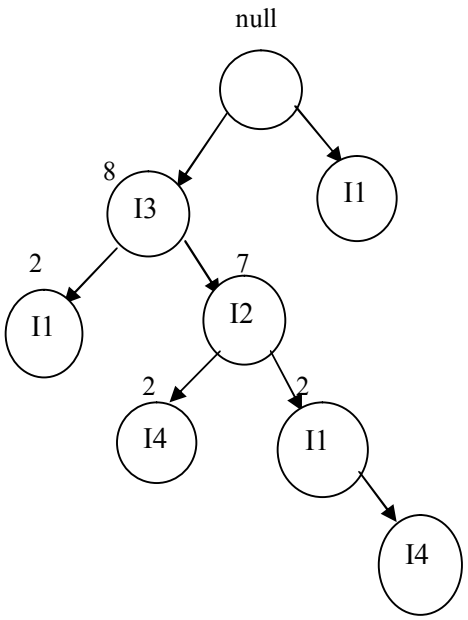

Figure 1. Tree Structure

3. EXPERIMENTAL RESULTS

In this section we have taken the market basket analysis and compare the efficiency of the proposed method to the existing algorithms which is mentioned above. All these algorithms are coded using the eclipse IDE which uses JAVA programming language. The data sets have been generated for testing these algorithms.

Two case studies have been done in analyzing the algorithm

i) the execution time of the algorithm is tested to the number of transactions,

ii) The execution time is executed to the number of the support.

Case i:

In this case where we are comparing the execution time of the transaction where any transaction may a contain more than one frequent itemsets. Here the minimum support is made constant. Here we assume the minimum support is being 40% and the comparison table is shown below.

Table:6 Execution Time based on Transactions

Transactions	Exec Hash Apriori	Transaction Reduction	Exec Apriori	Improved Apriori
1000	0.326	0.986	1.247	0.238
750	0.275	0.731	1.136	0.19
500	0.186	0.051	1.041	0.13
300	0.165	0.192	0.961	0.05

Case ii:

Now the execution time of different algorithms is compared by varying the minimum support. The comparison table is shown below.

Table:7 Execution Time Based on Support

Support %	Exec Hash Apriori	Exec Apriori	Transaction reduction	Improved Apriori
70	0.065	0.146	0.056	0.04
60	0.066	0.151	0.06	0.045
50	0.061	0.301	0.096	0.055
40	0.165	0.406	0.205	0.135

4. CONCLUSION

In this paper a new algorithm has been proposed for association rule mining for finding the frequent itemsets.

The present apriori algorithm has some bottlenecks we need to optimize and the proposed algorithm will give a new way for association rule where it reduces the candidate item sets. And we have also done some case studies about the existing algorithm above and we also listed the demerits of the existing systems and our proposed work is assured to overcome these bottlenecks we mainly concentrated to reduce the candidate itemset generation and also to increase the execution time of the process.

5. REFERENCES

[1] J. S. Park, M.S. Chen, and P.S. Yu. “An Effective Hash Based Algorithm for Mining Association Rules”. Proceedings of the 1995 ACM SIGMOD International Conference on Management of Data, San Jose, CA, USA, 1995, 175186.

[2] R. Agrawal and R. Srikant, “Fast Algorithms for Mining Association Rules in Large Databases,” Prof. 20th Int’l Conf. Very Large Data Bases, pp. 478499, 1994.

[3] A. Savasere, E. Omiecinski, and S. Navathe. “An Efficient Algorithm for Mining Association Rules in Large Databases”. Proceedings of 21th International Conference on Very Large Data Bases (VLDB’95), September 1115, 1995, Zurich, Switzerland, Morgan Kaufmann, 1995, 432444.

[4] J. Han and J Pei, “Mining frequent patterns by pattern growth: methodology and implications”. ACM SIGKDD Explorations Newsletter 2, 2, 1420. 2000.

[5] G. PiatetskyShapiro. “Discovery, analysis, and presentation of strong rules. Knowledge Discovery in Databases”, 1991: p. 229248.

[6] Barış Yıldız, Belgin Ergenç. “Comparison Of Two Association Rule Mining Algorithms Without Candidate Generation”, In IASTED International Conference on Artifical Intelligence and Applications (AIA 2010), Austria, Feb 15-17, 2010.

[7] Jiawei Han, Hong Cheng, Dong Xin, Xifeng Yan, “Frequent pattern mining: current status and future directions”, In the Journal of Data Min Knowl Disc (2007) 15:55–86,Springer Science+ Business Media, LLC 2007.