MODIFIED APRIORI ALGORITHM USING HASH BASED TECHNIQUE

Kaushal Vyas[1], Shilpa Sherasiya[2]

[1]M.E., Computer Engineering, Kalol Institute of Technology & Research Center, Gujarat, India

[2]Asst. Prof., Computer Engineering, Kalol Institute of Technology & Research Center, Gujarat, India

ABSTRACT

Data mining is a process of extraction of valuable and unknown information from the large databases. The information can be converted into knowledge about historical patterns. Many Algorithms have been proposed to mine association rule that uses support and confidence as constraint. Association rules mining is used to find frequent pattern and correlation existed in item sets through data processing, analysis, synthesis and inference. Association rules mining has achieved a very great application effect in business and other fields and it has become a research hotpot. In the field of association rules mining, Apriori algorithm is most popular. It is based on frequent item sets generation algorithm. Apriori algorithm is based on breadth-first search and its data structure is simple, clear and easy to understand. But, the application of Apriori algorithm needs to scan database many times which leads to a great overhead. so we tries to improve efficiency of this algorithm by removing this limitations and this algorithm is known as modified apriori algorithm which uses hash table approach.

Keyword: *Apriori, Association rules, Frequent itemsets, Support, Confidence*

1.INTRODUCTION

Data mining is used to extract the information from any system by analyzing the present in the form of data. In this paper author focuses on the problem of frequent pattern mining. Frequent patterns are the patterns that that occur in database at least user given number of times. Problem of frequent pattern mining can be defined as: given a large database of transactions, each consists of set of items. Aim of this problem is to find all the frequent itemsets i.e. a set of items Y is frequent if greater than min_supp % of all transaction in database contains Y and finding association rules from these frequent itemsets. Association rules was first introduced by Agarwal. Association rules are helpful for analyzing customer behaviour in retail trade, banking system etc. Association rule can be defined as {X, Y} => {Z}. It means in retail stores if customer buys X, Y he is likely to by Z. this concept of association rule today used in many application areas like intrusion detection, biometrics, production planning etc.

In the field of association rules mining, Apriori algorithm is most popular. It is based on frequent item sets generation algorithm. However, there are also many different aspects, such as searching strategies, scanning databases, data structure and so on. Apriori algorithm is based on breadth-first search and its data structure is simple, clear and easy to understand. But, the application of Apriori algorithm needs to scan database many times which leads to a great overhead.

1.1Association Rules

Association rule are the statements that find the relationship between data in any database. Association rule has two parts „Antecedent" and „Consequent". For example, {egg} => {milk}. Here egg is the antecedent and milk is the consequent. Antecedent is the item that found in database, and consequent is the item that found in combination with the first. Association rules are generated during searching for frequent patterns.

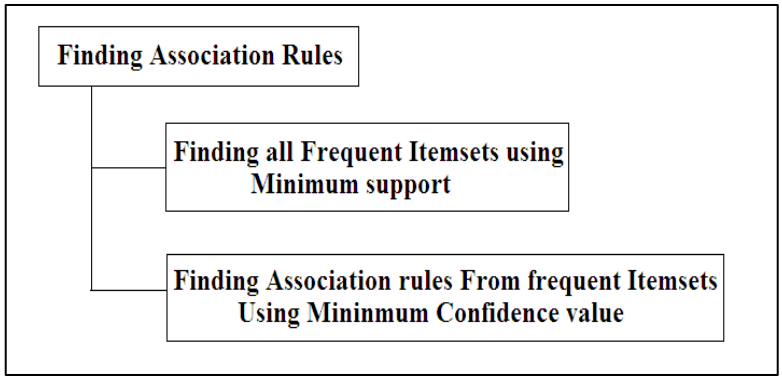

Fig -1 Generating Association rules

The problem of finding association rules is divided into two sub problems: first is to find frequent itemsets and second is to find association rules from these itemsets . For important relationships association rule uses the criteria of "Support" and "Confidence" that are explained below:

- Support (s): it is an indication of item how frequently it occurs in database.
  For a rule A=> B, its support is the percentage of transaction in database that contain AUB (means both A and B)
- Confidence (c): it indicates the no of times the statements found to be true. Confidence of the rule given above is the percentage of transaction in database containing A that also contain B.
- Lift: the lift of rule is defined as: Lift(A=>B)=Supp(AUB)/Supp(B)*Supp(A)

2. APRIORI ALGORITHM

Association rule generation is usually split up into two separate steps:
1. First, minimum support is applied to find all frequent itemsets in a database.
2. Second, these frequent itemsets and the minimum confidence constraint are used to form rules.
While the second step is straight forward, the first step needs more attention.
Finding all frequent itemsets in a database is difficult since it involves searching all possible itemsets (item combinations). The set of possible itemsets is the power set over I and has size $2n − 1$ (excluding the empty set which is not a valid itemset). Although the size of the powerset grows exponentially in the number of items n in I, efficient search is possible using the downward-closure property of support (also called anti-monotonicity) which guarantees that for a frequent itemset, all its subsets are also frequent and thus for an infrequent itemset, all its supersets must also be infrequent. Exploiting this property, efficient algorithms (e.g., Apriori and Eclat) can find all frequent itemsets.
As is common in association rule mining, given a set of itemsets (for instance, sets of retail transactions, each listing individual items purchased), the algorithm attempts to find subsets which are common to at least a minimum number C of the itemsets. Apriori uses a "bottom up" approach, where frequent subsets are extended one item at a time (a step known as candidate generation), and groups of candidates are tested against the data. The algorithm terminates when no further successful extensions are found.

Apriori Algorithm Pseudocode:

```
procedure Apriori (T, minSupport)
{ //T is the database and minSupport is the minimum support
L1= {frequent items};
for (k= 2; Lk-1 !=∅; k++)
        {
        Ck= candidates generated from Lk-1
         //that iscartesian product Lk-1 x Lk-1 and eliminating any k-1 size itemset that is not //frequent
         for each transaction t in database do{
                #increment the count of all candidates in Ck that are contained in t
                Lk = candidates in Ck with minSupport
        } //end for each }
        //end for
return U_k L_k ;
}
```

2.1 Advantages of apriori

- Easy implementation.
- "Apriori", this word is originated from Latin. That means "from what comes before".
- Initial Information- transaction database D and user-defined minimum support threshold Min_supp.
- Algorithm uses information from previous steps to produce the frequent itemsets.

2.2 Limitations of Apriori

- It only explains the presence and absence of an item in transactional databases.
- In case of large dataset, this algorithm is not efficient.
- In Apriori, all items are treated equally by using the presence and absence of items.
- Apriori algorithm requires large no of scans of dataset.
- In this Algorithm, Minimum support threshold used is uniform. Whereas, other methods can address the problem of frequent pattern mining with non-uniform minimum support threshold.
- In case of large dataset, Apriori algorithm produce large number of candidate itemsets. Algorithm scan database repeatedly for searching frequent itemsets, so more time and resource are required in large number of scans so it is inefficient in large datasets.

2.3 Ways to Improve Apriori

- Transaction Reduction: transactions that do not consist of frequent itemsets are of no importance in the next scans for searching frequent itemsets.
- Hash based itemset counting: hashing table is used for counting the occurrences of itemsets.
- Partitioning: for any itemset i.e. frequent in database, then that itemset must be frequent in atleast one of the partition of database.
- By adding attribute Weight and Quantity: means how much quantity of item has been purchased.
- By adding attribute Profit: that can give the valuable information for business and customers.
- By reducing the number of scans.
- By removing the large candidates that cause high Input/output cost.

3. PROPOSED WORK

3.1 Flow of Proposed work

- Step1: Scan a transaction database and find out the 2-itemset combination form of each transaction. For example, *T*= {A, B, C} and the 2-itemset combination form will be {{A, B} {A, C}, {B, C}}.
- Step2: Repeat Step 1 until all the transactions are hashed into hash table. After selecting the items whose count is large than the minimum support, large 2-itemset (*L2*) is obtained.
- Step3: Prune all transactions whose score is less than the minimum support. For example, assume *L2*={{A, C}, {A, D}}, *T* = {{A, B}, {B, C}, {A, C}} and minimum support is 2. Item *A* in set {A, B} gets 1 point, item *C* in set {B, C} gets 1 point, and item A and C in set {A, C} get 1 point respectively. The scores of T will be {A(2), B(0), C(2)}. Then item B will be pruned and item A and C will be preserved. Such operation can reduce the database scanning space significantly.
- Step4: Repeat Step 1 to Step 3 and increase the itemset level (3-item, 4-item and etc.) until no new frequent itemset is found.

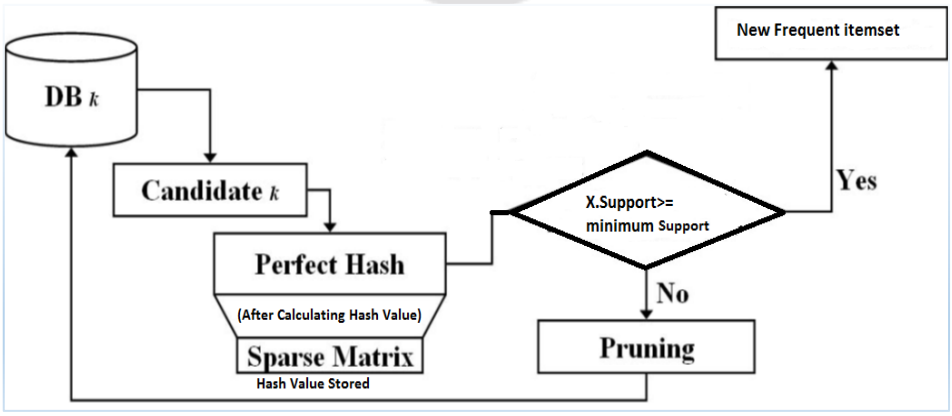

Fig -2 Flow of Proposed work

3.2 Proposed Algorithm

Our hash based Apriori implementation, uses a data structure that directly represents a hash table. This algorithm proposes overcoming some of the weaknesses of the Apriori algorithm by reducing the number of candidate k-itemsets. In particular the 2-itemsets, since that is the key to improving performance. This algorithm uses a hash based technique to reduce the number of candidate itemsets generated in the first pass. It is claimed that the number of itemsets in C2 generated using hashing can be smalled, so that the scan required to determine L2 is more efficient.

When scanning each transaction in the database to generate the frequent 1-itemsets,L1, from the candidate 1-itemsets in C1, we can generate all of the 2- itemsets for each transaction, hash(i.e) map them into the different buckets of a hash table structure, and increase the corresponding bucket counts . A 2-itemset whose corresponding bucket count in the hash table is below the support threshold cannot be frequent and thus should be removed from the candidate set. Such a hash based apriori may substantially reduce the number of the candidate k-itemsets examined.

- 1. Scan all the transaction. Create possible 2-itemsets.
- 2. Let the Hash table of size 8.
- 3. For each bucket assign an candidate pairs using the ASCII values of the itemsets.
- 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.

In this algorithm, each transaction counting all the 1- itemsets. At the same time all the possible 2-itemsets in the current transaction are hashed to a hash table. It uses a hash table to reduce the number if candidate itemsets. When the support count is established the algorithm determines the frequent itemsets. It generates the candidate itemsets as like the Apriori algorithm.

4. EXPERIMENT & RESULTS

We report experimental results on supermarket dataset. We have taken supermarket database as a Text file. In this data set, the average maximal potentially frequent itemset size is set to 16, while the number of transactions in the dataset is set to 25. Apriori and Hash based Apriori were executed for different minimum support level to generate the candidate 2- itemsets. The performance of Apriori and Hash based Apriori algorithms are evaluated for different minimum support levels.

In this section, we present a performance comparison of our development with Apriori. The following table presents the test results of the implementations of Apriori and the Hash based Apriori on the dataset of supermarket for different minimum support level.

Table-1 Memory Usage Comparison

Minimum Support Level	Size of candidate 2-itemsets	
	Apriori Algorithm	Modified Apriori Algorithm
1	136	52
2	78	29
3	48	25
4	45	18
5	28	11
6	20	7
7	15	5

As a result, when comparing with Apriori algorithm the size of candidate 2 Itemsets of Hash based Apriori algorithm is reduced. For minimum support level of 1, the size of candidate 2-Itemsets is 136 while using Apriori. But it is reduced to 52 while using Hash based Apriori. For minimum support level of 7, the size of candidate 2- Itemsets is 15 while using Apriori. While using Apriori with Hashing it is 5. From that the size of c2 is less for Hash based Apriori than Apriori.

Chart 1 illustrate the results of comparing our implementation of Apriori with Hash based Apriori method. Minimum Support is taken as X-axis and the size of C2 is taken as Y-axis.

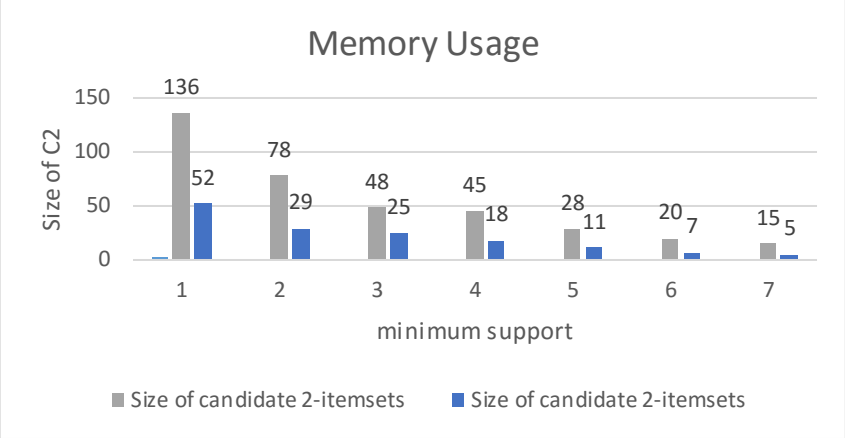

Chart-1 Memory Usage

In these graphs, we see that the memory usage of candidate 2-itemset for both algorithms increases exponentially as the minimum support is reduced. Applying Hashing data structure in apriori reduce the size of candidate 2-itemsets when comparing with apriori. Hence the execution time is reduced .Thus the performance of Apriori with hashing is improved with respect to execution time and memory size.

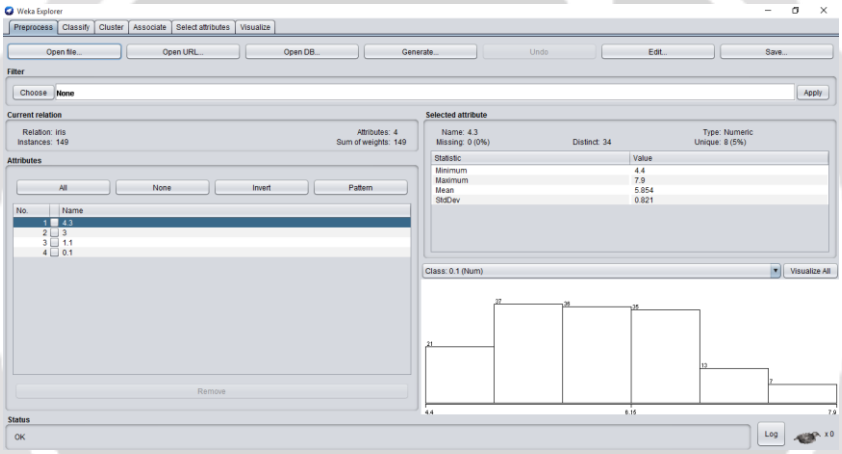

Fig.3 Modified Apriori Result with weka for large database

5. CONCLUSIONS

This modified algorithm can achieve a smaller memory usage than the Apriori algorithm. It is well known that the way candidates are defined has great effect on running time and memory need. Hash based Apriori is most efficient for generating the frequent itemset than Apriori. The proposed algorithm is more efficient than Apriori algorithm for association rules, when the database is dynamically updated.

This algorithm can achieve a smaller memory usage than the Apriori algorithm. It is well known that the way candidates are defined has great effect on running time and memory need. We presented experimental results, showing that the proposed algorithm always outperform Apriori. Hash based Apriori is most efficient for generating the frequent itemset than Apriori.

6. ACKNOWLEDGEMENT

The author would like to thank the reviewers for their precious comments and suggestions that contributed to the expansion of this work.

7. REFERENCES

[1] Ke Zhang, Jianhuan Liu, Yi Chai, Jiayi Zhou, Yi Li , A Method to Optimize Apriori Algorithm for Frequent Items Mining , Seventh International Symposium on Computational Intelligence and Design IEEE,2014 .

[2] Krutikai. K. Jain , A. B. Raut, Finding Association Rule using Apriori Algorithm on Educational Domain , International Journal of Electrical Electronics & Computer Science Engineering Volume 2, Issue 2 (April, 2015).

[3] G. S. Bhamra, A.K.Verma, R. B. Patel, An Encounter with Strong Association Rules, IEEE 2nd International Advance Computing Conference, 2010.
[4] Priyanka, Er. Vinod Kumar Sharma , APRIORI ALGORITHM FOR MINING FREQUENT ITEMSETS –A REVIEW , International Journal of Computer Application and Engineering Technology Volume 3-Issue 3, July 2014.
[5] Ms. Shweta , Dr. Kanwal Garg , Mining Efficient Association Rules Through Apriori Algorithm Using Attributes and Comparative Analysis of Various Association Rule Algorithms, International Journal of Advanced Research in Computer Science and Software Engineering Volume 3, Issue 6 (June, 2013).
[6] Introduction and steps involved in Apriori Algorithm. http://www.slideshare.net/INSOFE/apriori-algorithm-36054672.
[7] Tang, L., Jiang, H., An Improved Incremental Updating Algorithm for Association Rules. Computer Applications and Software, Vol. 29, pp. 246-248, 2012.
[8] Liu, B. Z., Improved Apriori Mining Frequent Items Algorithm. Application Research of Computers, Vol. 29, pp. 475-477, 2012.
[9] Mao, G. J., Data Mining Theory and Algorithm[M]. Beijing: Tsinghua University Press, 2007.
[10] A. Lekha, Dr. C V Srikrishna and Dr. Viji Vinod," Utility of Association Rule Mining: a Case Study using Weka Tool,"2013 IEEE.
[11] Yubo Jia, Guanghu Xia, Hongdan Fan, Qian Zhang and Xu Li, "An Improved Apriori Algorithm Based on Association Analysis," ICNDC 2012, 3rd IEEE International Conference, pp208-211.
[12] Rui Chang and Zhiyi Liu , " An Improved Apriori Algorithm," ICEOE 2011, IEEE International Conference, vol. 1, pp v1- 476 -v1-478.
[13] Sanjeev Rao and Priyanka Gupta," Implementing Improved Algorithm Over APRIORI Data Mining Association Rule Algorithm," IJCST Vol. 3, Issue 1, Jan. - March 2012.
[14] Rachna Somkunwar, "A Study on Various Data Mining Approaches of Association Rules", In: proceeding of International Journal of Advanced Research in Cmputer science and Software Engineering, ISSN 2277-128X, Volume-2, Issue-9, Page-141-144, September-2012.
[15] Weka(2007).http://www.cs.waikato.ac.nz/ml/weka/ dated on May 10, 2013.
[16] Jiawei Han and Micheline Kamber, "Data Mining Concepts and Techniques", Second Edition, Morgan Kaufmann Publishers,2006.