Mining Association Rules using Hash Table

K.Rajeswari
Ph.D Research Scholar
SASTRA University
Tanjore, Tamilnadu
India.

V.Vaithiyanathan,
PhD.
Associate Dean & Professor
School of Computing,
SASTRA university,
Tanjore, Tamilnadu
India.

Swati.Tonge
Assistant Professor
Computer Engineering
Pimpri Chinchwad College of Engg,
Pune University,
Pune,

Rashmi Phalnikar
Assistant Professor
IT Department,
MITCOE,
Pune University,
Pune

ABSTRACT
Data mining is a field which searches for interesting knowledge or information from existing massive collection of data. In particular, algorithms like Apriori help a researcher to understand the potential knowledge, deep inside the data base. But due to the large time consumed by Apriori to find the frequent item sets and generate rules, several applications cannot use this algorithm. In this paper, we describe the modification of Apriori algorism, which will reduce the time taken for execution to a larger extent.

Keywords
Data mining, Knowledge, frequent item sets, rules, Apriori Algorithm.

1. INTRODUCTION
Data mining is an area of research, where useless data are researched to find useful information through many different intelligent algorithms available. Apriori algorithm is an algorithm used to find interesting association between attributes present in a data set. This algorithm was put forward by R.Agarwal and R.Shrikant in 1994, for frequent item set identification from a large collection of data. The simple definition is to generate all subsets of a given item But the problem with Apriori is, it generates all candidates after scanning the complete data set, and prunes the data set if minimum support measure is not met. In this paper we propose a modified Apriori algorithm to solve this problem and a data set collected from University of California, Irvine is used to experiment the same.

2. APRIORI ALGORITHM
Apriori algorithm is an algorithm for finding interesting relationship between attribute with its layer by layer searching technique. Here k- item sets are used to explore (k+1)-item sets. It was developed by IBM to mine correlations between attributes in a very large collection of data base. [1][2][3]. An association rule is an implication expression of the form $A \rightarrow B$, where A and B are disjoint item sets, i.e., $A \cap B = \emptyset$. There are two important basic measures for association rules, minimum support and confidence. Generally minimum support and confidence are predefined by user or analyst so that the rules which are not so interesting or not useful can be deleted. Support is the total count of number of transactions where all items in A and B are together. Confidence determines how frequently items in B appear in transactions that contain A. The formal definitions of these metrics are given below,

Support $(A \rightarrow B) = \sigma (A \text{ and } B)$
Confidence $(A \rightarrow B) = \sigma (A \text{ and } B) / \sigma (A)$

The steps of Apriori algorithm is as follows.
Algorithm: Apriori (minsup, N)
minsup: Minimum support threshold
N: Number of tuples in original data set D
Method:

1) k=1
2) Fk ={i|i∈I∧ Support({i})≥N×minsup}
3) Repeat
4) k = k+1
5) Ck= candidates generated from Fk-1
6) For each instance t ∈ T do
7) C_t=subset(C_k, t)
8) For each candidate itemset c ∈ C_t do
9) Support(c)= Support(c)+1
10) end for
11) end for
12) F_k={c | c ∈ C_k ∧ Support(c) ≥ N × minsup}
13) Until F_k=Null
14) Result=∪F_k

Steps to generate rules [6] are as follows.
Algorithm**:** Rules generation (minconf)
minconf: Minimum confidence threshold
Method:

1. For each frequent k-itemset f_k , k ≥ 2 do
2. H_1={ i |i ∈ f_k}
3. apr-genrules(f_k, H_1)
4. end for

Function apr-genrules(f_k, H_m)

1) kk = |f_k|
2) m = |H_m|
3) If kk > m+1 then
4) H_{m+1} = m+1
5) for each h_{m+1} ∈ H_{m+1} do
6) Conf=Support(f_k)/ Support(f_k-h_{m+1})
7) If Conf ≥ minconf then
8) Return the rule (f_k-h_{m+1})→h_{m+1}
9) Else
10) delete h_{m+1} from H_{m+1}
11) endif
12) endfor
13) apr - genrules(f_k, H_{m+1})

endif

It applies anti-monotonicity property and is used for frequent item set mining for Boolean association rules. Apriori property says "For a frequent item set, all the non empty subsets of the item set are also frequent. Suppose is a itemset, I occurs n times, and minimum support is above n, then this

item set I will be discarded as it does not meet the minimum expectation of support. If an item A is added to I, then definetly, IUA will not have a support greater than minimum support. This property is also is called as AntiMonotone, means if an item set does not pass a test, then its supersets also will fail in the test. As monotonic behaviors are exhibited in failing, it is called as Anti Monotonic property [4][5]. An itemset is a frequent itemset if it occurs atleast satisfying a user expected threshold number of times. A closed itemset is one if none of its immediate supersets has the same support as it. A maximal superset is one if none of its supersets is occurring frequently.

Apriori follows a two step process:

1. Join step
2. Prune step

Join step joins a set of candidate L_k-item sets to generate L_{k+1} candidate. The set obtained is candidate set $C_{k+1.}$ Prune step removes the infrequent item sets from $C_{k+1.}$.

2.1 Problem of Apriori Algorithm

Even though, Apriori finds the interesting correlation between features in a data set, there are two important areas of improvement.

1. To find the frequent item set, it requires repeated scanning of the database
2. Large n umber of candidate sets need to be generated.

The time complexity of the Apriori algorithm is

O ($\sum$ k |Ck |N*v), where| Ck| is the number of times a candidate set occurs in the dataset. K varies from 1 to number of candidate sets found. N is the total number of tuples in the data set and v is the volume of each record. This is very huge and hence affects the efficiency of Apriori algorithm to a larger extent. Hence we propose a modified algorithm which will

- Reduce the number of scans.
- Every transaction is scanned only once.
- Hash table is used to update the count dynamically on encountering the item set.
- Sorting in descending order according to the width of itemset helps to identify maximum number of combinations possible to a larger extent.
- No repeated scans.

2.2 Computational Complexity

The computational complexity of the Apriori algorithm has following parameters [12]:

a. Generation of frequent 1-itemset
b. Generation of candidates
c. Support count calculation

To calculate support count of each generated candidate, the whole transaction database need to be scanned. This increases computational complexity of an apriori algorithm. It can be reduced by decreasing the number of database scans. Our proposed method requires only one scan of database and limited generation of candidates. Hence, this method is far better than traditional apriori algorithm.

3. RELATED WORK

Association rule extraction based on matrix multiplication is proposed in [7]. Boolean matrix of transactions is generated where each column represent one itemset. To get support count of item sets during process of apriori algorithm, these columns are multiplied transaction wise. Improved apriori algorithm suggested in [8] improves apriori algorithm by considering all possible combination of interested itemset only. This method is useful only if all itemsets are not considered. The improved-apriori algorithm suggested in [9] uses hash structure based on their predefined hash function. In this method, hashing is only used for generating 2 itemset. It optimizes 2-items generation which saves the time and space. It directly generates L2 in only one database scan without generation of C1, L1 and C2 by using hash table instead of hash tree the searching cost is reduced. Further k- frequent itemsets are generated similar to traditional apriori algorithm. CHARM [10] is an efficient algorithm for Closed Association Rule Mining, which outperforms Apriori [2], ACLOSE[11] algorithms. It works on the principle of Intersection between transactions and union of itemsets.

In this paper, we propose an improved method for generating association rules by reducing repeated database scans. The key idea is to traverse each transaction, find all possible combination of its item set in lexicographic order, put each combination in hash table if it is not already present and increment its count.

4. IMPROVED ARIORI ALGORITHM USING HASHING

We propose an algorithm that gives all k frequent itemset with one scan of Database D. Since only one scan is needed for this algorithm,it is far better that the traditional Apriori algorithm. During the scan of database the statistical data is collected which is used for mining the important rules. Algorithm GetFrequent() is used to get all k frequent itemsets during scanning the database D. Each transaction T will be scanned and the combination of each itemset in current transaction in lexicographic order will be put into hash table H having following attributes.

a. Itemset String //key for hashing
b. no_of_itemset
c. count

If the combination of itemset in transaction is not present in hash table then this combination is added in hash table and the corresponding attributes are assigned as per algorithm. If it is already present then the count value is simply incremented. Thus after execution of algorithm the hash table will consist of all possible combination of item sets in given database D. Algorithm GetRules is used to generate rules from k frequent item sets generated by GetFrequent algorithm. For this, hash

table must be sorted in descending order of no_of_itemset attribute. One by one frequent itemset should be extracted from sorted hash table and then generate all possible rules for selected combination of frequent item sets , calculate confidence of each such rule r and select the rule in result set R those confidence is greater or equal to minimum Confidence value.

Algorithm GetFrequent (D)

1) For each Transaction Ti repeat
2) Discretize combination C of itemset in transaction in monotonic order.
3) Put each combination C in Hash table H.
    a. If C is not present in H then , set its count =1.
    b. Set no_of_itemset =length(C)
    c. count() function will return number of transaction in combination C.
    d. If C is present in H, then set count=count+1.

Algorithm GetRules(H)

1) Sort H in descending order.
2) Set i=1 ,R= Ø
3) Do
    a. Get the frequent itemset H(i)
    b. If no_of_item(i) > 1 and Count(i) > MinSupport
        i. Generate the rule set RS.
        ii. Calculate Confidence for each rule r in RS.
        iii. If Confidence(r) >= minConfidence then select the rule r in result set R
4) Set i=i+1
5) Repeat while i <= Length(H)
6) Return R

Table 1: Transactional Data for an AllElectronics branch [5]

TID	List of Itemset
T1	I1,I2,I5
T2	I2,I4
T3	I2,I3
T4	I1,I2,I4
T5	I1,I3
T6	I2,I3
T7	I1,I3
T8	I1,I2,I3,I5
T9	I1,I2,I3

For example given in table 1, during scanning given database D, first transaction is T1 having item sets I1, I2 and I5. Possible combinations in lexicographic order are {I1, I2, I5, I1 I2, I1 I5, I2 I5, and I1 I2 I5}. The content of hash table is given below.

Table 2: Hash Table after processing Transaction T1

Itemset String	no_of_itemset	count
I1	1	1
I2	1	1
I5	1	1
I1 I2	2	1
I1 I5	2	1
I2 I5	2	1
I1 I2 I5	3	1

Now scan transaction T2, the content of hash table will be updated as given below.

Table 3: Hash Table after processing Transaction T2

Itemset String	no_of_itemset	count
I1	1	1
I2	1	2
I5	1	1
I1 I2	2	1
I1 I5	2	1
I2 I5	2	1
I1 I2 I5	3	1
I4	1	1
I2 I4	2	1

Similarly process will continue for last transaction T9, the content of hash table will be updated as given below.

Table 4: Hash Table after processing Transaction T9

Itemset String	no_of_itemset	count
I1	1	6
I2	1	7
I5	1	2
I1 I2	2	4
I1 I5	2	2
I2 I5	2	2
I1 I2 I5	3	2
I4	1	2
I2 I4	2	2
I3	1	6
I2 I3	2	4
I1 I4	2	1
I1 I3	2	4
I2 I5	2	2
I3 I5	2	1
I1 I2 I3 I5	4	1
I1 I2 I3	3	2

After sorting the hash tables in descending order of no_of _itemset attribute the table contents will be as shown below.

Table 5: Hash Table after sorting

Itemset String	no_of_itemset	count
I1 I2 I3 I5	4	1
I1 I2 I5	3	2
I1 I2 I3	3	2
I1 I2	2	4
I1 I5	2	2
I2 I5	2	2
I2 I4	2	2
I2 I3	2	4
I1 I4	2	1
I1 I3	2	4
I2 I5	2	2
I3 I5	2	1
I1	1	6
I2	1	7
I5	1	2
I4	1	2
I3	1	6

Let's assume minimum support is 2 and minimum confidence is 100%. Now scan the hash table, first frequent itemset is I1, I2, I3, I5 having support count 1, which is less that the required minimum support (2) hence this frequent itemset is ignored and next frequent item set is taken for analysis.

Frequent item set I1, I2, I5 satisfy the minimum support hence possible rule are generated which are given below and confidence is calculated.

I 1, I2 → I5 Confidence=2/4=50% ----------r1

I 1, I5 → I2 Confidence=2/2=100% ----------r2

I 2, I5 → I1 Confidence=2/2=100% ----------r3

I 1 → I2, I5 Confidence=2/6=33% ----------r4

I2 →I1, I5 Confidence=2/7=29% ----------r5

I5 → I 1, I2 Confidence=2/2=100% ----------r6

From the current frequent item set rule r2, r3 and r6 are selected in result set R. After the execution of GetRules algorithm as given above, the rules in result set R are selected.

R= {"I 1, I5 → I2" ," I 2, I5 → I1" ,"I5 → I 1, I2" }

No rule from frequent item set I1, I2, I3 are selected. Sample analysis is given below.

I 1, I2 → I3 Confidence=2/4=50% ----------r1

I 1, I3 → I2 Confidence=2/4=50% ----------r2

I 2, I3 → I1 Confidence=2/4=50% ----------r3

I 1 → I2, I3 Confidence=2/6=33% ----------r4

I2 →I1, I3 Confidence=2/7=29% ----------r5

I3 → I 1, I2 Confidence=2/6=33% ----------r6

5. CONCLUSION

As more scans are required for traditional Apriori algorithm to generate k-frequent item set, the proposed algorithm is designed in a way that it requires only one scan to find k-frequent item set. Use of hashing technique improves retrieval of item sets, thereby improving overall efficiency of proposed algorithm.

Even though Frequent pattern mining is a area where many publications have come in the recent years, there is still much scope of research. More research can be done in reducing the obtained pattern and improve the quality of retained patterns. Application of frequent pattern mining varies from bioinformatics, web mining, software bug detection and analysis and improves the performance of XML management systems.

6. REFERENCES

[1] Srikant, R and Agarwal, R. 1995. Mining Generalisation Association Rules. In Proceedings of 21st VLDB Conference. pp 407-419.\

[2] Agrawal, Srikant, R. 1994. Fast Algorithms for Mining Association Rules. Proc. of the 20th Int'l Conference on Very Large Databases, Santiago, Chile

[3] Agarwal R, Skikant R. 1996. Mining Quantitative Rules in Large Relational Tables [C] / / Proc of the ACM SIGMOD Conf on Management of Data.1996:1-12.

[4] Han J,Kamber M.Data Mining:Concepts and Techniques.Higher Education Press,200I.

[5] Jiawei Han, Micheline Kamber, Data Mining Concepts and Techniques, 2nd ed. China Machine Press, 2006, pp.155–160.

[6] Xie, Jianhua Wu, Qingquan Qian. 2009. Feature Selection Algorithm Based on Association Rules Mining Method. IEEE.

[7] X. Luo and W. Wang, "Improved Algorithms Research for Association Rule Based on Matrix," 2010 International Conference on Intelligent Computing and Cognitive Informatics, pp. 415–419, Jun. 2010.

[8] Libing Wu , KuiGong , Fuliang Guo "Research on Improving Apriori Algorithm Based on Interested Table",IEEE,2010

[9] R. Chang and Z. Liu, "An Improved Apriori Algorithm," no.Iceoe, pp. 476–478, 2011.

[10] CHARM: An Efficient Algorithm for Closed Itemset Mining by Mohammed J. Zaki and Ching-Jui Hsiao

[11] N. Pasquier, Y. Bastide, R. Taouil, and L. Lakhal. Discovering frequent closed itemsets for association rules. In 7th Intl. Conf. on Database Theory, January 1999.

[12] Pang-Ning Tan, Michael Steinbach, Vipin Kumar "Introduction to Data Mining", Addison Wesley.