International Journal of Applied Information Systems (IJAIS) – ISSN : 2249-0868
Foundation of Computer Science FCS, New York, USA
International Conference & workshop on Advanced Computing 2013 (ICWAC 2013) – www.ijais.org

A SIMPLE LINEAR COUNTING METHODOLOGY for DETERMINATION of MAXIMUM FREQUENT ITEMSET

Kirti Rajesh kumar Asharani Sharma
Student of B.E (Cmpn)
Thakur College of Engineering and Technology (TCET) Mumbai, India

Prof. Rashmi Thakur
A.P. CMPN, TCET
Mumbai, India

ABSTRACT
In today's fast pacing computer age where everything is digitized, it is imperative to have simple and efficient mechanisms or algorithms that help in analyzing usage patterns of customers. Such an analysis helps in developing profitable marketing strategies to enhance business and to serve customers better. In this paper, we will be presenting results of a simple counting algorithm [1] as compared to the complex Apriori algorithm [3] in order to find the maximum frequent itemsets. Apriori was first proposed by R. Agrawal et al [4, 5]. Many improved algorithms are based on this algorithm [6, 7]. The tedious scans of the Apriori algorithm for candidate generations will be reduced to a single scan in which the entire database will be stored in a bitmap matrix. Thus, the overhead on the system will be greatly reduced. Results will be obtained in a more quicker and efficient manner. Comparisons between the two techniques will be made in the form of a performance graph in an attempt to highlight the most suitable technique. Conclusions will be made following the advantages and future scope of the implementation.

General Terms
Data mining; frequent itemsets.

Keywords
Apriori; frequent itemsets using matrix; association rule mining; maximal frequent patterns.

1. INTRODUCTION
Apriori algorithm is widely used for determining patterns of relations between products. These relations are better known as association rules. An association rule of the form A → R simply means that purchase of product A influences the purchase of product R. The aim of Association Rule Mining (ARM) is to determine all such correlations between different products. The traditional example is that of milk → bread meaning that purchase of milk usually leads to the purchase of bread. Hence, milk vendors are seen with selling bread along with milk.

Thus, the usefulness of this technique is obvious. It can be used to extract patterns of usage of different customers which can be further used to formulate better marketing strategies.

Two important terms related with Apriori or ARM are support and confidence. Based on different values of these terms, a range of analysis can be made for various scenarios. A database on which such an analysis is to be done is usually a transactional database. A transactional database consists of transaction tuples, i.e. records storing the information of various products bought by a customer in a particular bill/transaction. It can be viewed as a two dimensional table with rows representing the transaction's unique identifier and the columns indicating the different items or products being considered.

Determining correlations can also help in making valuable suggestions to customers of the latest products that fall into their category of use/purchase, for example a book by a particular author, a new anti-dandruff hair product etc.

For this it is important to know the frequently occurring patterns. The algorithm usually applied is the Apriori. In this paper we have shown the results of a new algorithm which by simply counting the rows and columns reduces the overhead on the system and determine the maximal frequent itemset accurately. We also propose a way to handle large databases.

2. BASIC TERMINOLOGIES

2.1 Set of items
$I= \{I_1, I_2 \ldots, I_m\}$

2.2 Transactions
$D= \{t_1, t_2 \ldots, t_n\}$, $t_j \subseteq I$, j=1 to n

2.3 Itemset
$\{I_{i1}, I_{i2}, \ldots, I_{ik}\} \subseteq I$

2.4 Support of an itemset
Support of an itemset can be expressed in percentage or as an integer value. It gives the total number of transactions which contain that itemset.

2.5 Minimum Support of an itemset
It is the minimum percentage or number of transactions which are required to contain that itemset.

2.6 Frequent Itemset

Any itemset whose number of occurrences is above the minimum support value can be termed as a frequent itemset.

2.7 Maximal Frequent Itemset

Given a support threshold s, an itemset is maximal frequent if none of its superset is frequent.

2.8 Bitmap Matrix Data Structure

The database is scanned initially to form the associated matrix in bitmap form storing 1's for the presence of an item in a transaction. The absence of an item is represented by a 0. The rows of the matrix represent transactions while the columns represent the items.

2.9 Operations on the Matrix

The count of 1's of a particular column represent the frequency of occurrence of that particular item in the total number of transactions. So if the count is less than the minimum support then we know that the particular item is not a part of the maximal frequent itemset and can be immediately eliminated. This count is represented by the term Pi. Pi for each column i.e. for each item is calculated. Pi gives the one-itemset support [1]. If Pi is equal to the number of rows then it is imperative that the item is frequent since it is present in all the transactions. Hence it can be selected out in advance.

The count of 1's of a particular row represent the relation with the length of maximal frequent itemset. So if the count is zero then we know that the particular transaction is empty and has nothing to contribute to the frequent patterns. Hence it can be immediately eliminated. This count is represented by the term Qj (or Qi [1]). Qj for each row i.e. for each transaction is calculated. Qj gives one-transaction frequent itemset [1]. If Qj is equal to the number of columns then it is imperative that the transaction contains the maximal frequent itemset since it contains all the items. Hence it can be selected out in advance.

3. ALGORITHM [1]

The counting algorithm has the following eight steps:
Step 1: Setting up an associated bitmap matrix. Scanning transaction database D once sets up an associated matrix: setting "0" or "1" of row vector in accordance with a given order for each transaction.

Step 2: Calculating all one-itemset support Pi and all one-transaction frequent itemset Qi. The one-itemset support can be gotten by counting the column value with "1" and the one-transaction frequent itemsets Qi can be gotten by counting the row with "1".

Step 3: Using elimination method to remove the rows and columns of the associated matrix according to following conditions.

There are four conditions.

The first condition: the one-itemset support Pi is less than the minimum support threshold. The "i"th column can be removed because not satisfying the minimum support threshold.

The second condition: the one-transaction frequent itemset Qi equals to 0.

The third condition: the one-itemset support Pi equals to the number of the row.

The fourth condition: one-transaction frequent itemset Qi equals to the number of the column.

The first two conditions can used to eliminate the columns and the rows of the non-frequent itemset. If the rows and columns are met with the latter two conditions, the rows and columns can be selected out in advance in order to improve the efficiency of algorithm.

After the rows are selected out in advance, the standard eliminating columns will be changed. It should be one that minimum support threshold subtracts the number of rows selected out in advance. When the number of rows selected in advance is equal to the minimum support threshold, these rows make up the maximal frequent itemset.

The elimination method is as follows: First of all, eliminating the columns which Pi is less than the minimum support threshold. Then, all Qi of rows are calculated. The rows which Qi is equal to 0 are eliminated. Repeat above process until the rows and columns cannot be further eliminated. In fact, step 3 is one simplifying associated matrix.

Step 4: Qi queue. Qi queue is by descending value of Qi. This step is to find the largest value of Qi.

Step 5: Divided into sub-matrixes. If the value of all Qi equals to the number of columns on the matrix, then stop, otherwise, the matrix is divided into sub-matrixes.
The method is as follows.

At first, the first sub-matrix is divided out according to the largest Qi. This sub-matrix is composed of the corresponding rows with the "1" on the columns of Qi. Then, continuing division from remaining sub-matrix. Similarly, in the left sub-matrix, all Qi is calculated according to the largest Qi, until all the rows and columns are covered.

If there exists across the rows on a division, the columns of these rows with "1" form a new sub-matrix.

If there is more than the largest Qi, the Qi selected is one of Qi from top to bottom. The method is selecting the rows that at least one of their columns includes ones in the row of the largest Qi.

When divided into sub-matrixes, it is important that the union set of rows and columns in all sub-matrixes shall include rows and columns in the original matrix.

Note: the itemset of sub-matrixes after intersection operation must be empty. Otherwise, the intersection of the rows is divided into a new sub-matrix.
Step 6: Using elimination method in step 3 to the submatrixes and Qi queue by decreasing order until no new sub-matrix can be divided out.

Step 7: Selecting the combination of rows in the submatrixes.

According to the minimum support threshold and rows and columns selected in advance in step 3, the number of row which Qi is more than minimum support threshold will be selected out. Selected rows will be combined. The number of possible combinations is C^r_n : n is the number of rows selected; r is the minimum support threshold. Because there are rows or columns selected in advance in step 3, it is necessary to take into account this factor.

There are four cases discussed.

The case 1: there is no selected rows and columns in advance. Because no selected rows and columns in advance, this case can be dealt with according to normal conditions.

The case 2: there are rows selected in advance. Because there are the rows selected in advance, the number of rows to be selected can be reduced. The number of possible combinations is C^{r-t}_{n-t} , t is the number of rows selected in advance. In this way, computation can be reduced to improve the algorithm efficiency.

The case 3: there is columns selected in advance. Because there are the columns selected in advance, these columns become the part of the maximal frequent itemset. These columns do not change the method. In the end, these columns are only added to maximal frequent itemset selected.

The case 4: there are selected rows and columns in advance. The case 4 is the combination of the case 2 and the case 3. According to the case 2, we can use the way to reduce the number of rows. By the case 3, we can use the way to add the columns selected in advance to the maximal frequent itemset, together with the composition of maximal frequent itemset.

Step 8: Searching for maximal frequent itemset. In accordance with the selected combination of rows, all of one-itemset supports Pi are computed out. The maximal frequent itemset is composed of itemset which the one itemset supports Pi equals to minimum support threshold. The number of the items in the itemset is the maximal frequent itemset length

4. EXAMPLE

Consider the transaction database of table 1 with support=2.

Table 1. Transaction Database

TID	ITEMS
T1	I1,I3,I4,I6
T2	I2,I3,I5,I7
T3	I1,I2,I3,I5,I8
T4	I2,I5,I9,I10
T5	I1,I4

Accordingly we set up an associated bitmap matrix by scanning the database once. The matrix is as shown in table 2.

Table 2. Associated Bitmap Matrix

TID	I1	I2	I3	I4	I5	I6	I7	I8	I9I	I10	Qj
T1	1		1	1		1					4
T2		1	1		1		1				4
T3	1	1	1		1			1			5
T4		1			1				1	1	4
T5	1			1							2
Pi	3	3	3	2	3	1	1	1	1	1	

Calculating the one-itemset support Pi and one transaction frequent itemset Qj.

Applying the rules of the algorithm to eliminate rows and columns. The columns I6 TO I10 can be eliminated as their Pi is less than minimum support. Thus, the compressed matrix is in table 3.

Table 3. Compressed Matrix

TID	**I1**	**I2**	**I3**	**I4**	**I5**	**Qj**
T1	1		1	1		3
T2		1	1		1	3
T3	1	1	1		1	4
T4		1			1	2
T5	1			1		2
Pi	3	3	3	2	3	

Dividing into sub-matrices. Arranging Qj in the descending order, two sub-matrices can be formed based on T4 and T1 as in tables 4 and 5.

Table 4. Sub-matrix 1

TID	**I1**	**I2**	**I3**	**I5**	**Qj**
T1	1		1		2
T2		1	1	1	3
T3	1	1	1	1	4
T4		1		1	2
T5	1				1
Pi	3	3	3	3	

Table 5. Sub-matrix 2

TID	sI1	I3	I4	Qj
T1	1	1	1	3
T5	1		1	2
Pi	2	1	2	

In the matrices of table 4 and 5, rows T3 and T1 are selected in advance as their Qj is equal to the number of columns and hence it contributes to maximal frequent itemset.

Matrix of table 5 is now in its simplest form. So leave it there. Whereas matrix of table 4 can be further divided into two new sub-matrices as shown in tables 6 and 7.

Table 6. Sub-matrix 1 of Table 4

TID	I2	I3	I5	Qj
T2	1	1	1	3
T1		1		1
T4	1		1	2
Pi	2	2	2	

Table 7. Sub-matrix 2 of Table 4

TID	I1	I3	Qj
T1	1	1	2
T5	1		1
Pi	2	1	

In table 6, selecting T2 in advance as Qj equals number of columns.

In table 7, eliminating I3 as Pi is less than minimum support.

The matrices cannot be further simplified. Hence, leaving it there and forming combinations of the rows selected in advance i.e. T1, T2 and T3, we have as shown in tables 8, 9 and 10.

Table 8. Combination of T1 and T3

TID	I1	I2	I3	I4	I5	Qj
T1	1		1	1		3
T3	1	1	1		1	4
Pi	**2**	1	**2**	1	1	

Table 9. Combination of T2 and T3

TID	I1	I2	I3	I4	I5	Qj
T2		1	1		1	3
T3	1	1	1		1	4
Pi	1	**2**	**2**	1	**2**	

Table 10. Combination of T1 and T2

TID	I1	I2	I3	I4	I5	Qj
T1	1		1	1		3
T2		1	1		1	3
Pi	1	1	**2**	1	1	

From the three tables it is clearly evident that table 9 forms the maximal frequent itemset with length 3. The items for which the Pi values are greater than or equal to the minimum support will form the maximal frequent itemset.

Thus, the maximal frequent itemset is **I2, I3, and I5** and length of the maximal frequent itemset is 3.

5. PERFORMANCE COMPARISON AND RESULTS

The algorithm is tested to verify the time reduction in the database scans. A transaction database with 10, 20, 30 and 40 transactions is deployed to be tested with the two algorithms. Even if the comparisons to find the support along with candidate generation of the Apriori is assumed to be almost equal to the counting of the rows and columns of the bitmap matrix to find the frequent itemset, the reduction in the database scans is enough to demonstrate the efficiency in terms of the time factor. The below results are true for a database with pattern-information of five items with the length of the frequent itemset being three.

A graph of the Number of transactions **(X-axis)** v/s Database Scan Time **(Y-axis)** in nanoseconds is as shown in Figure 1.

The results are obtained working on JDBC. This is an illustration for just tens of transactions. But it is enough to demonstrate how adverse the overhead will be imposed on the system for thousands of transactions which is the case in practical scenarios.

We will further be testing this with a database containing lakhs of records wherein the advantage of this algorithm will be much more clearly evident.

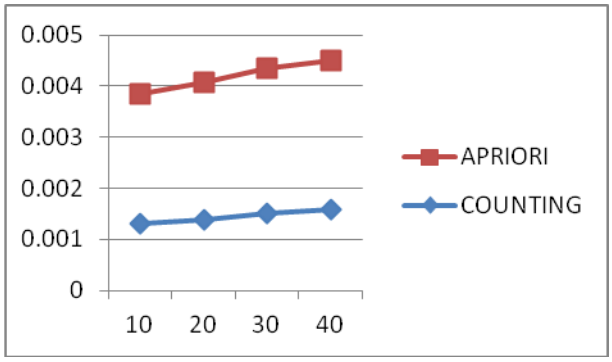

Fig 1: Relationship of No. of Transactions with Time Consumption

6. FUTURE SCOPE

6.1 For Business Analysis

Data mining will widen the horizons in the business world. Analysts will be able to explore the business at hand at a greater depth. This will help in knowing the impact over the masses. For example, when a particular product is launched, business analysis helps in knowing how effective it is among the people, whether the sales are as expected, the measures that can be taken to increase the popularity among the masses etc.

6.2 For Developing Marketing Strategies

Profits are the main goal of any business organization. Data mining algorithms like Apriori or the counting algorithm as discussed in this paper provide a means to know the frequent data items being purchased. This helps in the traditional "Market-Basket" [8] analysis. Thus, better marketing strategies can be developed to influence customers. Loss-leader techniques can also be implemented.

6.3 For Devising Usage Patterns

Data mining can prove very useful for devising usage patterns of customers using the World Wide Web. Information regarding the websites they visit, the products they purchase online etc can be extracted. This can further help in providing better offers/customized services to the customers.

6.4 For Customized Services

Better services can be provided to customers as per their requirement. Thus, customers can be made to feel special by providing them with customized services. They will not receive any spam since only potential customers will be targeted. As a result, if a customer browses mostly for jewellery, then she/he will be targeted with various discounts and new arrivals of the same.

7. PROPOSED IMPROVEMENT

In practical situations, databases encountered contain thousands of transactions. It might be a little difficult to maintain the matrix of such a large database as it will consume too much memory. Processing speed might also be affected. In order to efficiently handle large transactional databases with the counting algorithm, we propose to deploy the following steps:

7.1 Partition the database

7.2 Apply the algorithm on each partition

7.3 Combine the results of each partition to form a new transactional database

7.4 Apply the algorithm on the new database

7.5 Obtain the final result

8. CONCLUSION

The basis of association mining is mining frequent itemset. This helps in knowing various patterns of usage which eventually help in developing better marketing strategies. The Apriori algorithm poses a high overhead on the system with greater time consumption. Our implemented algorithm involves simple counting of the rows and columns of the transaction matrix thus consuming linear time. This counting algorithm is less complex and gives the same results with greater efficiency. Implementation and analysis demonstrate the advantages of this algorithm. The graph of the results supports our point clearly.

9. ACKNOWLEDGMENTS

We would like to extend our warm gratitude to our family for their patience and motivational words throughout the development of the paper. Last but not the least; we thank God for his spiritual bliss on us.

10. REFERENCES

[1] Haiwei Jin. A Counting Mining Algorithm of MaximumFrequent Itemset Based on Matrix; 2010 Seventh International Conference on Fuzzy Systems and Knowledge Discovery (FSKD 2010).

[2] Apriori algorithm by Agrawal et al at IBM Almaden Research Centre.

[3] Margaret H. Dunham, Data Mining Introductory and Advanced Topics 166-170.

[4] R Agrawal. Mining Association Rules Between Sets ofItems in Large Databases[C] .Washington :Proceedings ofthe ACM SIGMOD International Conference Managementof Data,1993 :207- 216.

[5] Agrawal R, Srikant R. Fast algorithms for miningassociation rules in large databases [A].Proc. of the 20thInt'l Conf on Very Large Data Bases [C]. Santiago: MorganKaufmann, 1994:478~49.

[6] Z.Xu,S. Zhang. Mining Association Rules in an optimized Apriori algorithm [J] Computer Engineering.2003, 29(19):83-85.

[7] G. Grahne, J.Zhu, Efficiently using prefix-trees in mining frequent itemsets. In Proc. ICDM'03 Int. Workshop on Frequent Itemsets Mining Implementations (FIMI'03), Melbourne, FL, Nov. 2003.

[8] Yen-Liang Chen, Kwei Tang, Ren-Jie Shen, Ya-Han Hu, Market basket analysis in a multiple store environment, Decision Support Systems, Volume 40, Issue 2, August 2005, Pages 339-354, ISSN 0167-9236, http://dx.doi.org/10.1016/j.dss.2004.04.009