IJCSE International Journal of Computer Sciences and Engineering Open Access
Research Paper Volume-5, Issue-2 E-ISSN: 2347-2693

An Efficient Algorithm for Mining Frequent Itemsets from Compressed Transactions using Matrix Approach

Ameta G.[1*] and Bhatnagar D.[2]

[1*]Department of Computer Science & Engineering, Sir Padampat Singhania University, Udaipur, Rajasthan, India
[2] Department of Computer Science & Engineering, Sir Padampat Singhania University, Udaipur, Rajasthan, India

[*]*Corresponding Author: gaurav.ameta@spsu.ac.in, Tel.: +91-94136-64420*

Available online at: www.ijcseonline.org

Received: 28/Jan/2017 Revised: /04/Feb/2017 Accepted: 22/Feb/2017 Published: Feb/28/2017

***Abstract*— Mining of frequent itemsets from large databases has been an interesting area for data miners from the beginning of data mining research. Knowing frequent patterns, data miners can determine interesting relationships among the items. In the proposed work, the original database is scanned once and the encoded database transactions are stored as a matrix. All frequent patterns are then determined from this matrix of coded transactions. An efficient algorithm has been developed to mine all frequent itemsets directly from this encoded matrix with the help of a reference matrix. The proposed approach reduces the memory size required for the database and the number of database scans to one. The algorithm finds its application in distributed data mining and secure data publishing.**

***Keywords*- Mining Frequent Pattern; Matrix Approach; Reference Matrix; Compressed Database; Market Basket Analysis; Apriori Algorithm.**

I. INTRODUCTION

An organization collects and analyzes transactional databases received from multiple locations. This analysis is useful for an organization in deciding new strategies or to modify the existing business policies. After mining frequent itemsets association rules are applied which are used to know how frequently one item is purchased with another? Association rules are mainly used to disclose interesting relationships in unrelated data of a database. Interestingness of an association rule depends on the support and confidence that guides to decide the usefulness and the certainty of the discovered rules. Association rule mining is an important task in data mining. Several algorithms exist to determine frequent itemsets in data mining. Apriori Algorithm is used for determining frequent itemsets and association rules from large databases [1]. Apriori Algorithm has an advantage of being clear and simple. The main shortcoming of Apriori is the time it consumes to hold a large number of candidate sets with much frequent itemsets. Several improved optimized methods were discovered on the foundation of Apriori Algorithm [2]. An approach by using matrix based technique to improve Apriori Algorithm is also discussed. In this approach transactions are considered in Boolean matrix form [3]. Liu et. al., proposed a method based on sampling to determine association rules. by improving the use of hash table technology and sampling algorithms [4]. Another Algorithm was proposed for mining association rules from large databases by compact grouping of database transactions (CGDBT). In this technique original database is transformed to compact groups of database transactions and these groups are efficient from mining point of view because it stops the recurring scanning of database and saves time [5]. Different ways were discussed for increasing the effectiveness of existing Association Rule Mining Algorithms. It also talks about various categories of databases on which association rules can be applied [6]. Implementation of Apriori Algorithm using WEKA tool is elaborated [7]. Problem of finding association rules in large databases is discussed. An algorithm is also proposed as a solution [8]. FP-Tree construction approach is used for saving summarized information about frequent patterns. Advantages of FP-Tree method and factors responsible for increasing efficiency of FP Growth Algorithm also discussed [9]. Cost vector matrix based mining algorithm is developed to generate frequent patterns from database. After that comparison with existing techniques has been done [10]. Graph Structure is used to compress the large database. Adjacency Matrix is used at various levels of mining procedure [11]. Package arules is provided in R which provides infrastructure which helps miner to find associations and analyze and evaluate results [12]. A technique is developed to discover association rules using Boolean Algorithm in large databases. In this technique candidate itemsets are not produced, it uses different technique for completion of its process [13]. Using MapReduce model hadoop cluster is implemented. Apriori algorithm is implemented on a singly node Hadoop cluster. This method has an advantage that it can be parallelized and also be

extensible and applicable to large databases [14]. Various algorithms used for discovering association rules. In this paper comparative study has been done based on efficiency of these algorithms [15]. Studies have emphasized on generating candidate itemsets with a focus on reducing processing time, memory requirement and the number of iterations required to scan the database [16]. To overcome such disadvantages various other approaches and techniques are discovered. Efforts have been made to minimize the time and space requirements for frequent item set mining algorithms. Apriori algorithm has some limitations. One limitation is that it scans the original database several times. Second issue is with database size, storing the original database in memory is more space consuming. Both issues are also correlated if database size is large, then the time needed to scan complete database will also be large. Apriori Algorithm traverses complete database again and again so if the size of database is large, then it takes very long time. The proposed work attempts to minimize the number of database scans to find frequent itemsets from transactional database. Second the original database is transformed to a matrix which will be the compressed form of the database. Reduced size of database will be helpful to reduce memory necessities during the scanning process. Section II presents the concepy of frequent pattern mining, section III presents tahe approach developed to do so in an efficeint manner, section IV presents a running example, section V presents the results obtained and section VI discusses the advantages and applications of the proposed work.

II. RELATED TERMS

A. Frequent Patterns

Frequent patterns are the itemsets that appear in a data set with not less than a user-specified minimum support.

B. Association Rules

Association rules are used to discover frequent patterns, uncover interesting relationships and correlations to analyze interesting patterns in the data bases. Support and Confidence are two pattern interestingness measures. Support S for an Association rule A➔B is the percentage of transactions in database that contain both A and B. Confidence is ratio of number of transactions that hold both A and B to the number of transactions that hold A.
Confidence (A➔B) = Support (AUB)/Support (A).

C. Apriori Algorithm

Apriori algorithm is used for mining frequent itemsets and association rules from large databases. The algorithm works on some prior knowledge. Algorithm starts its functioning by generating candidate itemsets of a particular size and after that scanning the database to count these to see if they are large. Algorithm uses the iterative approach named as a level-wise search, where k-itemsets are required to discover (k+1) itemsets.

To increase the efficiency of level-wise generation of frequent itemsets, an essential property is used known as Apriori Property which is used to shrink the search space. Apriori property uses two step processes, consisting of join and prune actions. Consider a Transactional Database for Market Basket Analysis of a Shopping Market as shown in Table I.

Let the minimum support be 2. The itemsets having support count equal or more than minimum support are the frequent itemsets.

TABLE I TRANSACTIONAL DATABASE

TID	List of Items
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

1. Find support of all items. Hence from Table III sup{I1}=6, sup{I2}=7, sup{I3}=7, sup{I4}=2 and sup{I5}=2.All items are frequent.
2. Generate candidates two Itemsets from the frequent item and find the support sup{I1,I2} =4, sup{I1,I3}=4, sup{I1,I4}=1, sup{I1,I5}=2, sup{I2,I3}=4, sup{I2,I4}=2, sup{I2,I5}=2, sup {I3,I4}=0, sup{I3,I5}=1, sup{I4,I5}=0.
3. Support count when compared with minimum support, it gives frequent Itemsets {I1,I2}, {I1,I3},{I1,I5},{I2,I3},{I2,I4},{I2,I5}.
4. Candidates for three itemsets are {I1, I2, I3} and {I1, I2, I5}. Here comparison of candidate support count with minimum support count is done. Both are frequent.
5. Since the 4-itemset {I1, I2, I3, I5} is not frequent, the algorithm will stop working here.

III. METHODOLOGY

A technique has been developed for mining frequent item sets from transactional databases. In this, the database is transformed into codes. Based on the number of items, a Reference Matrix is created. The database in Boolean Matrix form is further converted into relevant hex codes. Database in Hex Code format is known as Hex Matrix. A Reference Matrix is also generated to help mine frequent items.

A. Creation of Boolean Matrix Form from Transactional Database

In this proposed method, a transactional database having Ti (i=1, 2, 3…, n) transactions and Ij (j=1, 2, 3,…, m) items are represented as Table II.

TABLE II TRANSACTIONAL DATABASE (D)

TID	Items
T1	I1,I4,I6, I7, I9,I 10,I 11,I 14, I15, I16
T2	I1, I2, I3, I5, I8, I10, I12, I13, I15
T3	I1, I5, I6, I7, I9, I11, I12, I15, I16
T4	I2, I3,I4, I7, I 11, I12, I13, I14

This Transactional Database is then transformed into Boolean matrix as shown in Table III.

TABLE III BOOLEAN MATRIX FORM (M)

TID	I1	I2	I3	I4	I5	I6	I7	I8	I9	I10	I11	I12	I13	I14	I15	I16
T1	1	0	0	1	0	1	1	0	1	1	1	0	0	1	1	1
T2	1	1	1	0	1	0	0	1	0	1	0	1	1	0	1	0
T3	1	0	0	0	1	1	1	0	1	0	1	1	0	0	1	1
T4	0	1	1	1	0	0	1	0	0	0	1	1	1	1	0	0

The transactional database is first converted into a Boolean matrix of size T by I. In Boolean Matrix form T is the number of transactions and I is the maximum index of item sequence in Transaction Matrix. The existence and absence of an item in a transaction is represented by 1 and 0 respectively. Here it is also to be checked that maximum item index is the exact multiple of 4. If it is not exact multiple of 4, then we have to add minimum number of dummy items to make it exact multiple of 4. It is necessary to insert 0 (absence) bits corresponding to each dummy item added to it. Making exact multiple of 4 will be helpful in flawless conversion of boolean matrix to hex matrix. It is required because boolean values are converted into hex numbers and for hence it must be completely divisible by 4. So, Hexadecimal values are generated by dividing boolean transactions in a group of four as shown in Table IV.

B. Conversion of Boolean Matrix to Hex Matrix

For efficient conversion of Boolean Transactions into hex codes, we do vertical partitioning of transactions where each partition contains four items. After partitioning, Boolean values are transformed into a single matrix, called Hex Matrix (HM) as shon in Table V. Let D be the transactional database having T= 4 and N = 16 means that number of transactions is 4 and number of items are 16. So there will be 16/4= 4 blocks in a single row of hex matrix for that. First column of hex matrix will contain details of Items (I1,..., I4) second column of hex matrix will contain details of items (I5,...., I8) and so on.

Items I1-I4, I5-I8, I9-I12, I13-I16 are coded with single Hex Column respectively is Hex Matrix. Advantage of this matrix is that it saves the memory by ¼ factor. This form is useful in frequent data mining along with reference matrix we discussed above.

TABLE IV BLOCK CONTENTS

Items	Block
I1, I2,I3,I4	1
I5,I6,I7,I8	2
I9,I10,I11,I12	3
I13, I14,I15,I16	4

TABLE V HEX MATRIX

	Col-1	Col-2	Col-3	Col-4
Row-1	9	6	E	7
Row-2	E	9	5	A
Row-3	8	E	B	3
Row-4	7	2	3	C

C. Reference Matrix Creation

To construct Reference Matrix as shown in Table VI, binary equivalent of Hexadecimal number is used. Reference Matrix will be refered to compute support counts. In Reference Matrix, item indeces are placed in rows and Hexadecimal numbers are placed in columns.

TABLE VI REFERENCE MATRIX

Item Index	Hexadecimal Code															
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
I1, I5, I9, I13	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
I2, I6. I10, I14	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
I3, I7, I11, I15	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
I4, I8, I12, I16	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

Here, the Reference Matrix is matrix of size 4 by 16 which remains constant throughout the process. A unique number is provided to the items referred as Item Index. In this matrix rows are representing item index and columns are representing hexadecimal codes from 0 to F. This matrix has a unique property that items are placed in a sequential and row wise in iterative manner. From Table VI it can be observed that I1 has been placed in first row, I2 in second row, I3 in third row and I4 in fourth row. At this stage starting from row 1 to row 4 each containing one Item Index from I1 to I4 respectively, after one round. In the next round, Item Index I5 is placed in first row, I6 in second row, I7 in third and so on. In the same way all sixteen Item Indexes are placed in row wise manner. If more number of Items like I17, I18 ... required to be placed in reference matrix, then I

17 will be placed in first row and I 18 in second row and so on in sequential manner. It means that Item index {I1, I5, I9, I13…..} will be represented by the first row in the reference matrix. In the same way Item Index {I2, I6, I10, I14…, {I3, I7, I11, I15} and {I4, I8, I12, I16….} are represented by second, third and fourth rows.

D. Proposed Algorithm for Mining Frequent Itemsets

The algorithm is based on Apriori Property. In each step, support is counted for each k-itemset, and only frequent itemsets will be considered for the k+1-itemset mining.

Algorithm:

Step 1: Prepare Reference Matrix
Assign Rows→Items
Assign Columns→Hex Codes (0 to F)

Step 2: Prepare Boolean Transaction Matrix and Hex Matrix. Divide the items into blocks of four, when Hex Matrix is prepared.

Step 3: For 2-itemsets one of the two cases occurs:

Case 1: If both the items (say I1 and I2) representing to same hex code columns/ blocks / Intra block itemsets in Hex Matrix. We begin searching from reference matrix to hex matrix, for which do the following:

(i) Intersect the rows of Reference Matrix.
(ii) Fetch the data from the relevant block in Hex Matrix

Case 2: If both item (ex Item1 and Item 2) representing to different hex code columns/ blocks/ Inter block itemsets in Hex Matrix.

(i) Select the Hex Matrix columns for Item 1 and 2.
(ii) Now check the reference matrix for Item 1 and Item 2. Firstly, the first column element for both Item 1 and Item 2 are checked. If it has 1 in both the places then support count is increased for the itemset by 1, otherwise not.
(iii) Step (ii) is repeated for second, third and fourth columns of Itemset 1 and Itemset 2.

Step 4: For 3 Item Sets (If Item1 and Item 2 refer to same hex code column and Item 3 refer to other)

(i) For Item 1 and Item 2 same hex column is selected from hex matrix.
(ii) Item 3 refers to different hex column.
(iii) For support count we take first hex column element for Item1 and Item2 and first hex column element of Item 3. We check with reference matrix for three positions of hex code, if it's 1 at all the places. Then we increase support count, otherwise not. Same process is repeated for second column and so on.

Step 5: For 4 Item Sets (Items are Item1, Item 2, Item 3 and Item 4 having same hex column)

(i) Select the respective hex columns for Item 1, Item 2, Item 3 and Item 4.
(ii) Select the first hex column for all the Items and check with reference matrix. If we get 1 at all four places, the support count will increase by 1 for four itemset, otherwise not.

IV. RUNNING EXAMPLE

A. Items within same block : Find support count of itemsets where the items belong to the same block

i) In case of finding the support count of {I1, I3}, both the items belong to the same block.
ii) Intersecting the rows of I1 and I3 in Reference Matrix, we get the result R as A, B, E, F.
iii) Check the hex matrix's first column,i.e., 9, 8, E, 7.
We match 9, 8, E, 7 and R=A, B, E and F. Only one match for E is found. Therefore, the support count of both the items will be increased by 1. We can also verify it from Table VI that there is only one place where both I1 and I3 are 1.

B. Two Itemsets (Items with two different blocks)

i) Item I 6 and I 11 are Inter block items.
ii) For I6, Hex Matrix Column elements are 6, 9, E, 2 and for I 11 Hex Matrix column elements are E, 5, B, 3.
iii) Check in the reference matrix according to above codes first we check for I 6 in the second row at 6, its 1. Now we check I 11 at the place of E it is also 1 in its place so we will increase support count by 1.
iv) Check for I6 element 9 and for I11 element 5 there is a mismatch so we will not increase support count here.
v) Check for I6 element E and for I11 element B there is an exact match (both are 1 and 1) so we will increase support count here by 1.
vi) Check for I6 element 2 and I 11. Element 3 there is a mismatch so we will not increase support count here.
Hence, the overall support count will be increased by 2.

C. Three Itemsets (2Items in same block and1 in the other)

i) Items are I2, I3 and I6
ii) For I2 and I3 within the same block so the column we select from hex matrix is 9, E, 8 and 7.
iii) For I6 there will be a different column 6, 9, E, 2.
iv) So for 3 Item set we check in Reference matrix 9 for I2 and I3 and for 6 in I6. If all three are one then support count will be increased by 1 for three item sets. Here all are 0 so no increments here.
v) Check E in reference matrix for I 2 and I 3 and 9 for I6. Mismatch here so no increment in support count here.
vi) Check for 8 for I2 and I3 and E for I 6, mismatch here so no increment. Finally we check I2 and I3 for 7 and 2 for I 6, mismatch no increment.

D. Four Itemsets

i) Items are I1, I 10, I 14 and I 16
ii) The hex matrix block for I1 is 9, E, 8, 7, for I10 is E, 5, B ,3 and for I14 and I16 is 7, A, 3,C.

iii) Check the first element, for each item I1 for 9, I 10 for E, and 7 for I14 and I16. Since all are 1 in the reference matrix, support count is increased by 1.

In the next iteration comparison will be done for items I1 for E, I 10 for 5 and A for I14 and I16. For support count increment, all 1's are needed. If mismatch occurs, there will be no increment in support count for four item-set. Same procedure is followed for remaining two comparisons.

V. RESULT

The proposed algorithm requires single database scan, after which, all computations are done from Hex Matrix. It is better as compared to apriori algorithm because the later requires multiple scans. Also the database is reduced in size by one fourth and the details of transactions are confined to a single digit hex code matrix. Thereafter, there is no need to scan the database. It saves memory required at run time. Reference Matrix used in matching procedure is static in nature and prepared one time before starting the algorithm. Its size does not vary even if number of items is very large.

VI. CONCLUSION

The proposed method to mine frequent itemsets proves to be an efficient approach as the number of database scan is reduced to one. The space required to store matrices reduces to one fourth due to hex code conversion. Hex coded form can be used for secure data publishing by using various encryption techniques. This algorithm can be applied to distributed data mining in which miner gets data for analysis from multiple locations. It also has its application in secure data publishing. Research domain can be expanded to the area of Privacy Preserving Data Mining (PPDM).

ACKNOWLEDGMENT

We thank all authors of technical papers for sharing their work that helped us carry out the research successful and Sir Padampat Singhania University, Udaipur for providing with all laboratory facilities required.

REFERENCES

[1]. Al-Maolegi M., Arkok B., "*An improved Apriori Algorithm for Association Rules*", International Journal on Natural Language Computing, Vol. 3(1), pp. 21-29, 2014.

[2]. Yabing J., "*Research of an Improved Apriori Algorithm in Data Mining Association Rules*", International Journal of Computation and Communication Engineering, Vol. 2(1), pp. 25-27, 2013.

[3]. Rehab A., Alva H., Anasuya B., Patil V., "*New Matrix based approach to improve Apriori Algorithm*", International Journal of Computer Science & Network Solutions, Vol. 1(4), pp. 102-109, 2013.

[4]. Liu Z., Sun T., Sang G., "*An Algorithm of Association Rules Mining in Large Databases Based on Sampling*", International Journal of Database Theory and Application, Vol. 6(6), pp. 95-104, 2013.

[5]. Al-Shoreman H., Jbara Y., "*An Efficient Algorithm for Mining Association Rules for Large Itemsets in Large Databases*", International Journal of Engineering and Innovative Technology, Vol. 3(10), pp. 237-240, 2014.

[6]. Kotsiantis S., Kanellopoulos D., "*Association Rule Mining: A Recent Overview, GESTS*", International Transactions on Computer Science and Engineering, Vol. 32(1) , pp. 71-82, 2006.

[7]. Tanna P., Ghodasara Y., "*Using Apriori with WEKA for frequent Pattern Mining*", International Journal of Engineering Trends and Technology, Vol. 12(3), pp. 127-134, 2014.

[8]. Mishra S., Pattanaik S., Patnaik D., "*Application of association rules to determine item sets from large databases*", International Journal of Computer Science Engineering, Vol. 2(6), pp. 276-278, 2013.

[9]. Han J., Pei J., Yin Y., "*Mining Frequent Patterns without Candidate Generation ACM SIGMOD*", International Conference on Management of Data, pp. 1-12, 2000.

[10]. Jain S., Dave M., Agrawal A., "*Cost Vector Matrix – A new approach to Association Rule Mining*", International Journal of Recent research and Review, Vol. 7(2), pp. 68-73, 2014.

[11]. Choubey A., Patel R., Rana J., "*Graph based new approach for frequent pattern mining*", International Journal of Computer Science and Information Technology, Vol. 4(1), pp. 221-235, 2012.

[12]. Grun B., Hahsler M.., "*arules- A Computational Environment for Mining Association Rules and Frequent Item Sets*", Journal of Statistical Software, Vol. 14(15), pp. 1-23, 2005.

[13]. Rao C., Babu D., Shankar R., Kumar V., Rajanikanth J., Shekhar C., "*Mining Association Rules Based on Boolean Algorithm – a Study in Large Databases*", International Journal of Machine Learning and Computing Vol. 3(4), pp. 347-351, 2013.

[14]. Ezhilvathani A., Raja K., "*Implementation of Parallel Apriori Algorithm on Hadoop Cluster*", International Journal of Computer Science and Mobile Computing, Vol. 2(4), pp. 513-516, 2013.

[15]. Kumar D., Jayaveeran N., " *A Survey on Association Rule Mining Algorithms for Frequent Itemsets*", International Journal of Computer Sciences and Engineering, Vol. 4(10), pp. 120-125, 2016.

[16]. Mani K., Akila R., "*Enhancing the Performance in Generating Association Rules using Singleton Apriori*", International Journal of Information Technology and Computer Science, Vol. 9(1), pp. 58-64, 2017.

Authors Profile

Mr. Gaurav Kumar Ameta is a Research Scholar, Department of Computer Science & Engineering at Sir Padampat Singhania University, Udaipur, India. His specialization areas are Data Mining, Computer Graphics, Cryptography and Privacy Preservation.

Dr. Divya Bhatnagar is working as Professor in the department of Computer Science and Engineering at Sir Padampat Singhania University, Udaipur, India. She holds 18 years of teaching and research experience. Her areas of interest include data mining, Big Data and Neural Networks.

