International Journal of Computer Applications (0975 – 8887)
Volume 159 – No 6, February 2017

A Matrix based Maximal Frequent Itemset Mining Algorithm without Subset Creation

Balwant Kumar
Dept. of Computer Science and Engineering
GJUS&T, Hisar,
Haryana, India

Dharmender Kumar, PhD
Dept. of Computer Science and Engineering
GJUS&T, Hisar,
Haryana, India

ABSTRACT

Frequent pattern mining is main step in association rule mining. Several algorithms have been proposed for this, but the majority of these algorithms have two main problems that is large number of database scan and generating large candidate itemsets. This process is time intense because these algorithms first mine the minimal frequent itemsets and then generate maximal frequent itemsets from minimal frequent itemsets. Present paper proposes a new top down approach based on compressed matrix for mining maximal frequent itemsets directly without the help of subset. The proposed algorithm performs better than Maximal Frequent Itemset First (MFIF) algorithms with datasets of different size and on different threshold.

Keywords

Association Rules, Frequent Itemset, Matrix based Maximal Frequent Itemset Mining (MB-MFIM), Maximal Frequent Itemset (MFI), Maximal Frequent Itemset First (MFIF).

1. INTRODUCTION

Discovery of association rule from huge datasets is a vital data mining task. It consists of two steps, first mine frequent itemsets and then generate association rule from these itemsets. An itemset is said to be frequent if its support count (total number transactions that contain the itemset in database) is greater than predetermined minimum support threshold specified by the user. Association rule is like (X→Y) where X and Y are two frequent itemsets. Support (X→Y) = X(X∪Y), and Confidence is the conditional probability Confidence (X→Y) =X(X/Y) ={X(X∪Y)/support count(X)}. Association rule (X→Y) is said to be interesting if (X→Y) satisfy user defined thresholds (min support, min confidence). Frequent itemsets play a very central role in association rule mining. In literature various methods to find frequent patterns has been proposed. In 1994 R. Agrawal and R. Srikant proposed Apriori algorithm for finding frequent itemsets [1]. After that many variations have been proposed to enhance the performance of Apriori algorithm. Kumar D. and Singh S. analyze various algorithms to generate frequent itemsets [2]. These algorithms are not time intensive because they first mine minimal frequent itemsets and then generate maximal frequent itemsets from these frequent minimal itemsets. The major shortcoming of Apriori and its variations are large number of database scan and generating large candidate itemsets. In most of the cases users are interested only in Maximal frequent itemsets due to their wide applicability. An itemset is said to be Maximal Frequent Itemset if none of its superset is frequent mean this is largest frequent itemset. To generate Maximal Frequent Itemset Apriori and many other frequent itemset mining algorithms required more database scan and time intense candidate generation. As the size of the datasets growing, so, fast and more efficient algorithms are required to generate Maximal frequent itemsets.

The present paper proposes an efficient algorithm named as Matrix Based Maximal Frequent Itemset Mining Algorithm without Subset Creation. This algorithm is based on top down approach and it uses compressed matrix for finding maximal frequent itemsets directly without the help of subset. A Boolean matrix is generated from the transaction database; column sum and weight concept are used to compress the Matrix so that search space for mining Maximal frequent itemsets can be reduced. The paper is divided in different sections, section II describes the related work and section III outlines the proposed algorithm. Experimental results and comparisons with proposed algorithm and MFIF are discussed in section IV and the last section provides the conclusions.

2. RELATED WORK

Several methods have been proposed to discover the maximal frequent itemsets. R.J. Bayardo proposed a MaxMiner algorithm which extends the Apriori algorithm [3]. MaxMiner reduces the search space by using two pruning steps (subset infrequency pruning and look-aheadpruning). MaxMiner first time use dynamic reordering approach. After that lots of enhancements have been suggested for mining the MFI for dataset of erratic type and size. Pincer-Search algorithm combined the top-down and bottom-up [4] technique to discover the maximal frequent itemset. R.C. Agarwal et al. implements a depth first search technique with bitmap representation (DepthProject)[5]. In which, column denotes the items and rows denotes the transactions. Like MaxMiner algorithm, they used dynamic reordering and look-ahead pruning. A projection mechanism used to reduces the size of database. They efficiently find the support counts and give a superset of the MFI. D. Burdick et al. further extended DepthProject and named it as Mafia [6]. They used vertical bit-vector data format. Compression and projection on bitmaps are applied to increase the performance. Unlike DepthProject and MaxMiner pruning technique, Mafia used Parent Equivalence Pruning. Algorithm GenMax [7] presented by Gouda and Zaki which combine pruning with mining and give the exact MFI. They used vertical data format representation and backtracking to specify all maximal patterns; a progressive focusing technique to remove non-maximal itemsets and diffset propagation for fast support counting. To search MFI, SmartMiner [8] uses tail information. It also uses an augmented heuristic that generates a smaller search tree which reduces the cost of support counting. Don-Lin Yang et al. [9] suggested a hash-based way to mine maximal frequent itemset called as HMFS. HMFS integrates DHP and Pincer-Search technique. HMFS shrinks the database scan and also filtered out the uncommon itemsets which lessens the whole computing time. Zubair Rahman et al. proposed HBMFI-LP [10] approach which stores the database in vertical data format with the help of hashing and hash collisions are avoided by suing linear probing. Therefore compute MFI directly. Similarly M. Krishnamurthy et al. proposed HBMFI-QP [11] which use quadratic probing to avoid hash collision, that occur in linear probing. HK Jnanamurthy et al. presented a top down approach to find maximal frequent itemset by using subset named as MFIF [12].

3. PROPOSED WORK: MB-MFIM WITH-OUT SUBSET ALGORITHM

The proposed technique, MB-MFIM without subsetis centered on top down methodology and directly mine maximal frequent itemset by using compressed matrix. From the transaction database, A Boolean matrix BM[] is generated and weight concept is used to compress the BM[]. The proposed method is discussed below.

3.1 Procedure MB-MFIM with-out Subset

1) Columnsum []:=sum of all column of matrix BM[].

2) Delete Columnsum[i] where Columnsum[i] <minsupport.

3) Z:=unique rows of matrix BM[] and weight []:=total no of occurrences for all rows of Z in BM[].

4) Rowsum []:=sum of all rows of Z; Max: =Maximum value in Rowsum [] and j []: = index of Max in Rowsum [].

5) Support []:= weight[j]

6) If minsupport<support [k] for any value of k then Maximal frequent itemset: =Z (kth row) and exit.

7) S := Z (Kth row) for all K := j[]; Max := Max-1 and j[] index of Max in Rowsum[]; R := Z (Kth row) for all

   K := j[]; Support1 := Sum (Support []) + Sum (weight [i] for all i := j[]),

8) If minsupport ≤ Support1, then take one by one row of R and find that each of its element are present in any row of S. If yes then add their corresponding weight values. If this value is greater than or equal to minsupprot, then it is frequent and Exit.

9) Goto 7;

3.2 Example

Let a matrix BM[] of database:-

Item / Transaction	I1	I2	I3	I4	I5	I6	I7	I8
1	1	0	1	1	1	0	1	0
2	0	1	1	1	1	0	0	1
3	1	1	1	1	1	0	1	0
4	0	0	0	0	1	0	1	1
5	1	0	1	1	1	0	1	0
6	1	0	0	1	1	0	1	0
7	0	0	1	1	1	1	0	0
8	1	0	1	1	1	0	0	1
9	1	1	0	1	1	1	0	0
10	0	0	1	1	1	0	0	1

Step1: Columnsum of Matrix BM [] is given in below table:-

Columnsum							
6	3	7	9	10	2	5	4

Step2: Let minsupport = 4 (40 %) and Delete BM's column where Columnsum < minsupport so 2nd, 6th and 8th rows are deleted. Now BM [] is following.

Item / Transaction	I1	I3	I4	I5	I7
1	1	1	1	1	1
2	0	1	1	1	0
3	1	1	1	1	1
4	0	0	0	1	1
5	1	1	1	1	1
6	1	0	1	1	1
7	0	1	1	1	0
8	1	1	1	1	0
9	1	0	1	1	0
10	0	1	1	1	0

Step3: BM [] with unique rows assign to Z and total weight of each rows assign to Weight are following:-

Weight=	3	3	1	1	1	1

	I1	I3	I4	I5	I7
Z=	1	1	1	1	1
	0	1	1	1	0
	0	0	0	1	1
	1	0	1	1	0
	1	1	1	1	0
	1	0	1	1	1

Step4: Rowsum of Matrix Z is given in below table:-

Rowsum=	5	3	2	3	4	4

Step5: Max in Rowsum [] is 5 and weight of 5 is 3.

Step6: Support is 3 < minsupport. So assign row having Rowsum 5 to S:-

S=	I1	I3	I4	I5	I7
	1	1	1	1	1

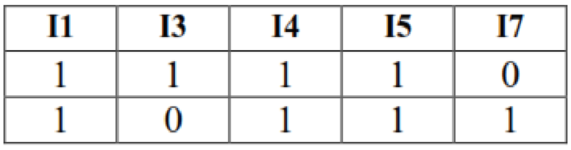
Step7: find Max-1; which is 4 and which have weight [1,1] total support=3+1+1.

R	I1	I3	I4	I5	I7
	1	1	1	1	0
	1	0	1	1	1

Step8: As total support ≥ minsupport, take first row of R and finds that each of its element are present S. Therefore add up the R's row weight value and S's row weight value. Like, 1+3

= 4, which is equal to minsupport, so it is frequent. Similarly check for all rows of R.

Both rows of R have support ≥ minsupport so Maximal frequent itemset are these following rows:-



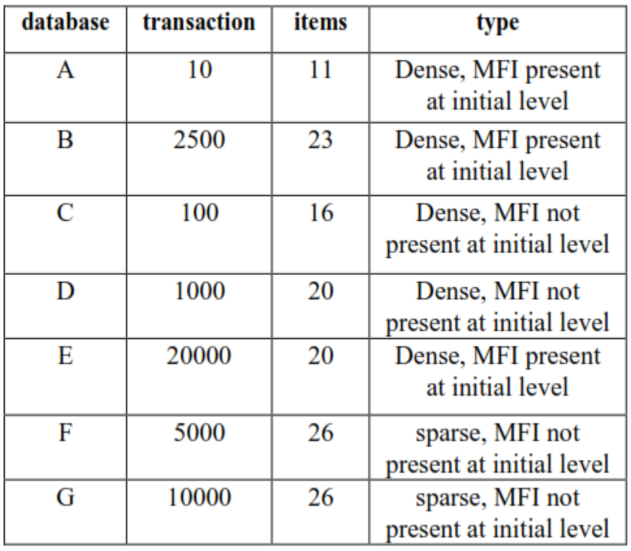
I1	I3	I4	I5	I7
1	1	1	1	0
1	0	1	1	1

So MFI are {I1, I3, I4, I5} and {I1, I4, I5, I7}.

4. COMPARISON AND EXPERIMENT RESULTS

By using the unique rows concept and weight concept, the proposed method perform better than simple MFIF algorithm. MB-MFIM algorithm without subset is compared with simple MFIF. Both algorithms are implemented in MATLAB version 7.12.0.384over window vista ultimate 32 bit operating system and Intel dual core (1.86 GHz) processor with 2 GB RAM.For fair evaluation; both procedures use the identical database with transaction Matrix. To take a detail evaluation, both procedures are run on several types of database with different user thresholds. The following tables show the type of database i.e. total number of transaction, number of item, database type dense or sparse and level of MFI present.

Table 1. Type of database



database	transaction	items	type
A	10	11	Dense, MFI present at initial level
B	2500	23	Dense, MFI present at initial level
C	100	16	Dense, MFI not present at initial level
D	1000	20	Dense, MFI not present at initial level
E	20000	20	Dense, MFI present at initial level
F	5000	26	sparse, MFI not present at initial level
G	10000	26	sparse, MFI not present at initial level

Table 2. Comparison Result on Execution Time

Type of database	Min support (percent)	MB-MFIF without subset Time (second)	MFIF Time (second)
A	30	0.0036	0.0067
	40	0.0042	0.0094
	50	0.0171	0.0252
	60	0.0450	0.0795
B	30	0.0220	0.0567
	40	0.03660	0.1814
	50	0.0470	0.1664
	60	0.1190	0.7434
C	30	0.0211	0.1947
	40	0.0590	0.738
	50	0.3298	3.8488
	60	0.7810	3.8632
D	30	0.0053	0.178
	40	0.0197	0.606
	50	0.2010	2.391
	60	0.3487	13.942
E	30	0.2412	0.375
	40	0.3211	0.522
	50	0.2744	6.971
	60	2.780	51.647
F	30	0.2384	27.575
	40	0.3082	266.337
	50	2.3407	1443.03
	60	9.2911	5936.13
G	30	1.244	43.280
	40	2.0623	523.4790
	50	3.3200	3513.723
	60	11.624	10587.262

MFIF is better where MFI present at preliminary level in dense database but proposed technique works well in both type of database and in both case i.e. MFI present at preliminary level or not. MB-MFIM without subset perform best when number of duplicate rows is high in matrix than its running time does not depend on database size and minsupport threshold. MB-MFIM without subset is also memory efficient due to compressed Matrix and weight concept.

The following factors make proposed algorithm MB-MFIM efficient and faster than MFIF:-

1. Proposed algorithm use one level of Apriori property to eliminate the one infrequent item from the transaction Matrix which reduce the search space but MFIF take original Matrix to find MFI.
2. It eliminates the duplicate rows of transaction Matrix by assign weight to each row, it again compress the Matrix but MFIF not remove redundancy from database.
3. The proposed algorithm is much efficient due to no sub-setting. Time and memory both saved.

5. DISCUSSION AND CONCLUSION

In this paper, a new algorithm MB-MFIM without subset is proposed which use a transaction Boolean MatrixBM [] and first apply one step Apriori property for one itemset to remove non-frequent one item. Then all the matching rows are store as single row and assign weight according to their incident in Matrix; these steps compact the size of Matrix very much and maximal frequent itemsets are mined from this compressed Matrix. Comparison Results show that this is fast as well as memory efficient than MFIF, so it can be used very efficiently, in all type of problem where we needed MFI. Although it is very simple, fast and efficient for all type of database but its best case is where redundancy of itemset in transaction is high. Its performance not depended on size or type of database or user threshold but depends on redundancy of transaction. More the redundancy, more efficient it will be.

6. REFERENCES

[1] R. Agrawal and R. Srikant, "Fast algorithms for mining association rules," *in Proc. of Int. Conf. Very Large Data Bases (VLDB'94)*, Santiago, Chile, pp: 487–499, Sept. 1994.

[2] Sukhbir Singh, Dharmender Kumar "Frequent Pattern Mining Algorithms: A Review", *International Journal of Advanced Engineering and Nano Technology*, Vol -1, No -10, 2014, pp 1-7.

[3] Bayardo RJ. "Efficiently Mining Long Patterns from Databases," in Proceedings *ACM SIGMOD International Conference on Management of Data*, pp.85–93, 1998.

[4] Lin DI, Kedem ZM. "Pincer-Search: A New Algorithm for Discovering the Maximum Frequent Set," in *Proceedings of 6th International Conference on Extending Database Technology*, pp.105–119, 1998.

[5] Agarwal RC, Aggarwal CC, Prasad VVV. "Depth First Generation of Long Patterns," in *Proceedings of the 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp.108–118,2000.

[6] Burdick, D., Calimlim,M., and Gehrke, J. "MAFIA: A maximal frequent itemset algorithm for transactional databases," in *Proceedings of IEEE International Conference on Data Engineering*, pp. 443–452, 2001.

[7] K. Gouda and M. J. Zaki. Efficiently Mining Maximal Frequent Itemsets. *Proc. of the IEEE Int. Conference on Data Mining*, San Jose, 2001.

[8] QinghuaZou, Wesley W. Chu and Baojing Lu, "SmartMiner: A depth first algorithm guided by tail information for mining maximal frequent itemsets," in *Proceedings of the IEEE International Conference on Data Mining.* pp.570 – 577, 2002.

[9] Don-Lin Yang, Ching-Ting Pan and Yeh-Ching Chung , "An efficient hash-based method for discovering the maximal frequent set," *25th Annual International Conference on Computer Software and Applications*, pp.511-516, 2001.

[10] A.M.J. Md. ZubairRahman, P. Balasubramanieand P. VenkataKrihsna, "A Hash based MiningAlgorithm for Maximal Frequent Itemsets using Linear Probing,"*Infocomp Journal of Computer Science*, Vol.8, No.1,pp.14-19, 2009.

[11] M. Krishnamurthy, A. Kannan , R. Baskaran, R. Deepalakshmi "Frequent Item set Generation Using Hashing-Quadratic Probing Technique," *European Journal of Scientific Research* Vol.50, No.4, pp. 523-532, 2011.

[12] Jnanamurthy HK, Vishesh HV, Vishruth Jain, Preetham Kumar, Radhika M. Pai, "Discovery of Maximal Frequent Item Sets using Subset Creation,"*International Journal of Data Mining & Knowledge Management Process (IJDKP)* Vol.3, No.1, Jan. 2013.