ISSN:0975-9646

A Parallel Incremental Mining Algorithm using Modified Inverted Matrix

Siddharth Shah* Aniruddha K* Prerak Thakkar* Chinmay Joshi* Gopi Bhatt*

**Department of Computer Engineering,*
A.D Patel Institute of Technology

Abstract- The association rule mining has been very useful in many business applications, such as market analysis, web data analysis, decision making, knowing customer purchase behavior etc. In almost all transactional databases, new transactions are added with time. So an efficient algorithm is required to be developed in order to avoid re-scanning of old datasets. Incremental mining [18] deals with generating association rules based on available knowledge (obtained from mining of previously stored databases) and incremented databases, without scanning the previously mined databases again. In this paper a novel approach of using horizontal and vertical database layout, a new representation of transactional database, has been proposed (modified version of Inverted Matrix [17]) to mine large database incrementally. One of the major advantage of this approach is that it generates the data structure in a single scan of the database and whenever new database is added incrementally, the generated structure is updated to consider the effect of incremented database. In this paper the Modified Inverted Matrix is distributed amongst parallel nodes. Frequent item from the modified inverted matrix is assigned to parallel nodes in alternate splitting fashion. In parallel implementation, a Co-Occurrence Frequent Item (COFI) tree [17] for assigning frequent item is generated by the parallel nodes. Mining process is accomplished by all nodes which generate all frequent items in which the assigned items are participated. Here, less communication is required amongst the master node and all parallel node to generate all frequent item sets. Moreover, one of the additional advantage is that the algorithm still responds correctly without the need for changing the data structure, whenever desired support value changes. In this paper we provide the theoretical proof of concept for our proposed approach.

Keywords-Parallel mining, Incremental mining, Inverted matrix, COFI tree.

1. INTRODUCTION

Due to the increasing use of large amount of data in various applications, the importance of data mining has grown rapidly. With respect to business applications, analysis of previous transactional data can provide valuable information on purchase behavior of customer, and thus help in making business decisions. Thus it is necessary to collect and analyze a sufficient data properly before making any decisions. Since the amount of data being processed is large, it is important for the mining algorithms to be very computationally efficient. Various data mining algorithms have been explored in the literature [1–6]. Recently many important applications have created the need of incremental mining. This is due to the increasing use of the such databases where data is being continuously added e.g., super market data, stock market data, sales data, and weather/traffic records, etc. In the incremental mining, data are continuously being added with time. The aim of incremental mining techniques is to re-run the mining algorithm on the only updated database. However, it is obviously less efficient since previous mining rules are not utilized for discovering new rules while the updated portion is usually small compared to the whole dataset. Consequently, the efficiency and the effectiveness of algorithms for incremental mining are both crucial issues. Algorithms should be such that only updated transactions and previous mined rules to be taken into account for generating new rules. The process of incremental mining is described in Fig. 1. The next few sections discuss the related work and the proposed approach for Incremental Mining.

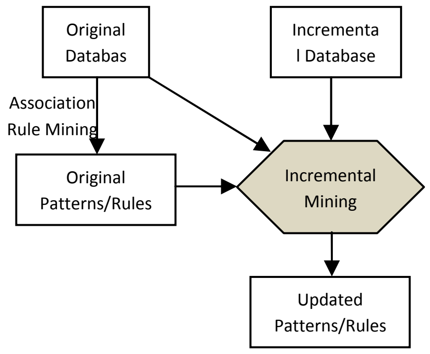

Fig 1: Process of incremental mining [4]

2. RELATED WORK

2.1 Inverted Matrix

The concept of Inverted Matrix and COFI Tree [17] was used for Interactive mining, where if the support change, re-scan of complete dataset is not required. In Inverted Matrix [17] the transactional data is converted into a new database layout called Inverted Matrix that prevents multiple scanning of the database during the mining phase, in which finding frequent patterns could be achieved in less than a full scan of Inverted Matrix. The Inverted Matrix is then mined using different support levels to generate association rules using the Inverted Matrix algorithm. Inverted Matrix layout is a combination of both horizontal and vertical layouts. Each item associates with all transactions in which it occurs, and each transaction with all its items using pointers. The item is the key of each record in this layout. Each attribute on the Inverted Matrix is a pointer that points to the location of the next item on the same transaction. The pointer is a pair where the first element indicates the address of a line in the matrix and the second element indicates the address of a column. Each line in the matrix has an address and is prefixed by the item it represents with its frequency in the database. The lines are ordered in ascending order of the frequency of the item. Building the Inverted Matrix is done in two phases, in which phase one scans the database once to find the frequency of each item and orders them into ascending order. The second phase scans the database again once to sort each transaction into ascending order according to the frequency of each item.

2.2 Co-occurrence Frequent-Item (COFI) Trees

Computing the frequencies is done by reading sub-transactions for frequent items directly from the Inverted Matrix, then building independent trees called COFI Trees [17] for each frequent item in the transactional database. Each one of the trees are mined separately as soon as they are built, without building conditional sub-trees recursively and are then discarded. The COFI-tree for a given frequent item *x* contains only items that are more frequent than *x* or as frequent as *x*. For each given frequent 1-itemset all frequent k-item sets are found. For this, a COFI-tree is built for each frequent item except the most frequent one, starting from the least frequent.

The mining process [17] is done for each tree independently. From each branch of the tree, using the support count and the participation count, candidate frequent patterns are found and non-frequent patterns are discarded at the end when all branches are processed.

Tid	Items				
T1	A	G	D	C	B
T2	B	C	H	E	D
T3	B	D	E	A	M
T4	C	E	F	A	N
T5	A	B	N	O	P
T6	A	C	Q	R	G
T7	A	C	H	I	G
T8	L	E	F	K	B
T9	A	F	M	N	O
T10	C	F	P	J	R
T11	A	D	B	H	I
T12	D	E	B	K	L
T13	M	D	C	G	O
T14	C	F	P	Q	J
T15	B	D	E	F	I
T16	J	E	B	A	D
T17	A	K	E	F	C
T18	C	D	L	B	A

Figure 2: Transaction Database [17]

3. Parallel Implementation Of Frequent Pattern Mining With Proposed Modified Inverted Matrix

Inverted matrix generation requires two full I/O scans of the dataset and generates a special disk based data structure. One scan is required to find frequencies of all items and then other scan for storing the items in transactional array. In our work, it was found that instead of two scan, a modified inverted matrix in which entries are not sorted according to support count can be build with single scan. This approach reduces one I/O read of database. Below Figure 3(a) and 3(b) are an image of Modified Inverted Matrix with single scan of transactional database given in Figure 2. Then using the information of inverted matrix, COFI Tree can be build which can then be mined to generate frequent patterns. Since Modified Inverted Matrix (MIM) is not sorted, whenever any new transactions are added, the information of the same can be easily incorporated into Modified Inverted Matrix without scanning the original database again which makes mining incrementally. Then COFI Tree can be generated from MIM and then mined where the procedure remains same as in [17]. In Modified Inverted Matrix with single scan, each item as read from database makes an entry into Modified Inverted Matrix with its support counter equal to 1. If an item is already present, its support count is incremented. Along with this the transactional array consists of two entries, one (backward pointer) for item which is before current item and another (forward pointer) for item which is after current item in transactional database. The entries made are same as in Inverted matrix [17]. Once an item is read its entry is made into Modified Inverted Matrix and consequently its backward pointer is set in its corresponding transactional array and previous item's forward pointer is set at the same time. Same process is repeated until all transactions are read. The item location field only is sorted in order to reduce searching complexity while building COFI Tree. The index column contains 5 entries namely sorted item, its support and its original location in transactional array and actual item with its support count. These five entries are kept to make searching of item easier. The Modified Inverted Matrix for first 13 transactions of Figure 2 is shown in the Figure 3(a) given below. The Modified Inverted Matrix built as shown in figure 3(b) is same as that of one build after reading complete set of transactions at once. Thus it makes incremental mining process possible. The algorithm for creating MIM is as given below

Modified Inverted Matrix Algorithm
Input : Transactional Database (*D*)
Output : Modified Inverted Matrix
Method :
Pass I

```
1. While there is a transaction T in the database (D)
do
1.1  while there are items si in the transaction do
 1.1.1Create the index part of the MIM
        1.1.1.1 Add an entry in transactional array row
        with 4- parameters
        (A) Location in index part of the IM of the next
        item si+1 in T null if si+1 does not exist.
         (B) Location of the next empty slot in the
        transactional array row of si+1, null if si+1 does
        not exist.
        (C) Location in index part of the IM of the
        previous item si-1 in T null if si-1 does not exist.
        (D) Location of the next empty slot in the
        transactional array row of si-1, null if si-1 does not
        exist.
1.2  Goto 1.1
2. Goto 1
3. Sort index part of IM and keep original entry of item and
its frequency with sorted one.
```

Creating and Mining COFI-Trees from MIM
Input: Modified Inverted Matrix (IM) and a minimum support threshold
Output: Full set of frequent patterns
Method:

```
1. Frequency Location = Apply binary search on the
index part of the IM to find the Location of the
first frequent item based on min_sup.
2. While (Frequency Location < IM Size) do
2.1 A = Frequent item at
```

```
location (Frequency Location)
2.2 A Transactional = The Transactional array of item A
2.3 Create a root node for the (A)-COFI-Tree with frequency-count and participation-count = 0
2.4 Index Of TransactionalArray = 0
2.5 While (Index Of TransactionalArray < Frequency of item A)
2.5.1 B = item from Transactional array at location (Index Of TransactionalArray)
2.5.2 Follow the chain of item B to produce sub-transaction C (forward and backward)
2.5.3 Items on C form a pre_x of the (A)-COFI-Tree.
2.5.4 If the pre_x is new then
2.5.4.1 Set frequency-count= 1 and participation-count= 0 for all nodes in the path
Else
2.5.4.2 Adjust the frequency-count of the already exist part of the path.
2.5.5 Adjust the pointers of the Header list if needed
2.5.6 Increment Index Of TransactionalArray
2.5.7 Goto 2.5
2.6 MineCOFI-Tree (A)
2.7 Release (A) COFI-Tree
2.8 Increment Frequency Location //to build the next COFI-Tree
3. Goto 2
```

Function: MineCOFI-Tree (A)

```
1. nodeA = select next node //Selection of nodes will start with the node of most frequent item and following its chain, then the next less frequent item with its chain, until we reach the least frequent item in the Header list of the (A)-COFI-Tree
2. while there are still nodes do
2.1 D = set of nodes from nodeA to the root
2.2 F= frequency-count-participation-count of nodeA
2.3 Generate all Candidate patterns X from items in D. Patterns that do not have A will be discarded
2.4 Patterns in X that do not exist in the A-Candidate List will be added to it with frequency = F otherwise just increment their frequency with F
2.5 Increment the value of participation-count by F for all items in D
2.6 nodeA = select next node
2.7 Goto 2
3. Based on support threshold _ remove non-frequent patterns from A Candidate List.
```

Loc	**Index**	**Transactional Array**							
		1	**2**	**3**	**4**	**5**	**6**	**7**	**8**
1	J,1,15 A, 8	(@,@) (2,1)	(7,2) (@,@)	(13,1) (9,1)	(@,@) (5,4)	(@,@) (4,4)	(@,@) (4,5)	(@,@) (13,3)	(@,@) (3,4)
2	Q,1,18 G, 4	(1,1) (3,1)	(16,1) (@,@)	(17,1) (@,@)	(4,7) (10,3)				
3	M,2,8 D, 6	(2,1) (4,1)	(7,1) (@,@)	(5,3) (7,2)	(1,8) (5,6)	(@,@) (7,5)	(8,2) (4,7)		
4	P,2,11 C, 7	(3,1) (5,1)	(6,1) (5,2)	(@,@) (7,3)	(1,5) (18,1)	(1,6) (6,2)	(@,@) (13,4)	(3,6) (2,4)	
5	L,2,12 B, 7	(4,1) (@,@)	(@,@) (4,2)	(@,@) (3,3)	(1,4) (9,2)	(14,1) (@,@)	(3,4) (6,3)	(7,5) (14,2)	
6	K,2,14 H, 3	(4,2) (7,1)	(4,5) (17,1)	(5,6) (17,2)					
7	R,2,16 E, 5	(6,1) (3,2)	(3,3) (1,2)	(4,3) (13,1)	(12,1) (13,2)	(3,5) (5,7)			
8	I,2,17 M, 2	(13,3) (9,3)	(@,@) (3,6)						
9	H,3,6 N, 3	(1,3) (@,@)	(5,4) (10,1)	(8,1) (10,2)					
10	N,3,9 O, 3	(9,2) (11,1)	(9,3) (@,@)	(2,4) (@,@)					
11	O,3,10 P, 2	(10,1) (@,@)	(13,4) (15,1)						
12	G,4,2 L, 2	(@,@) (7,4)	(14,2) (@,@)						
13	F,4,13 F, 4	(7,3) (1,3)	(7,4) (14,1)	(1,7) (8,1)	(4,6) (11,2)				
14	E,5,7 K, 2	(13,2) (5,5)	(5,7) (12,2)						
15	D,6,3 J, 1	(11,2) (16,2)							
16	C,7,4 R, 2	(18,1) (2,2)	(15,1) (@,@)						
17	B,7,5 I, 2	(6,2) (2,3)	(6,3) (@,@)						
18	A,8,1 Q, 1	(4,4) (16,1)							

Figure 3 (a): Proposed Modified Inverted Matrix generated with single scan of database of first 13 transactions

Loc	Index	Transactional Array										
		1	2	3	4	5	6	7	8	9	10	11
1	R,2,16A, 11	(@,@) (2,1)	(7,2) (@,@)	(13,1) (9,1)	(@,@) (5,4)	(@,@) (4,4)	(@,@) (4,5)	(@,@) (13,3)	(@,@) (3,4)	(5,9) (3,8)	(@,@) (14,3)	(5,10) (@,@)
2	Q,2,18G, 4	(1,1) (3,1)	(16,1) (@,@)	(17,1) (@,@)	(4,7) (10,3)							
3	H,3,6 D, 9	(2,1) (4,1)	(7,1) (@,@)	(5,3) (7,2)	(1,8) (5,6)	(@,@) (7,5)	(8,2) (4,7)	(5,8) (7,6)	(1,9) (@,@)	(4,10) (12,3)		
4	M,3,8 C, 10	(3,1) (5,1)	(6,1) (5,2)	(@,@) (7,3)	(1,5) (18,1)	(1,6) (6,2)	(@,@) (13,4)	(3,6) (2,4)	(@,@) (13,5)	(13,7) (@,@)	(@,@) (3,9)	
5	N,3,9 B, 10	(4,1) (@,@)	(@,@) (4,2)	(@,@) (3,3)	(1,4) (9,2)	(14,1) (@,@)	(3,4) (6,3)	(7,5) (14,2)	(@,@) (3,7)	(7,7) (1,9)	(12,3) (1,11)	
6	O,3,10 H, 3	(4,2) (7,1)	(4,5) (17,1)	(5,6) (17,2)								
7	P,3,11E, 8	(6,1) (3,2)	(3,3) (1,2)	(4,3) (13,1)	(12,1) (13,2)	(3,5) (5,7)	(3,8) (13,6)	(15,3) (5,9)	(14,3) (13,7)			
8	L,3,12M, 3	(13,3) (9,3)	(@,@) (3,6)									
9	K,3,14N, 3	(1,3) (@,@)	(5,4) (10,1)	(8,1) (10,2)								
10	J,3,15 O, 3	(9,2) (11,1)	(9,3) (@,@)	(2,4) (@,@)								
11	I,3,17 P, 3	(10,1) (@,@)	(13,4) (15,1)	(13,5) (18,2)								
12	G,4,2 L, 3	(@,@) (7,4)	(14,2) (@,@)	(3,9) (5,10)								
13	F,7,13 F, 7	(7,3) (1,3)	(7,4) (14,1)	(1,7) (8,1)	(4,6) (11,2)	(4,8) (11,3)	(7,6) (17,3)	(7,8) (4,8)				
14	E,8,7 K, 3	(13,2) (5,5)	(5,7) (12,2)	(1,10) (7,8)								
15	D,9,3 J, 3	(11,2) (16,2)	(18,2) (@,@)	(@,@) (7,7)								
16	C,10,4 R, 2	(18,1) (2,2)	(15,1) (@,@)									
17	B,10,5 I, 3	(6,2) (2,3)	(6,3) (@,@)	(13,6) (@,@)								
18	A,11,1Q, 2	(4,4) (16,1)	(11,3) (15,2)									

Figure 3 (b): Proposed Modified Inverted Matrix generated with single scan of database after reading next 5 transactions

In COFI Tree building and mining, the same process as explained in [17] is carried out with a modification of considering backward pointers also. Since new transactions can be easily added in Inverted Matrix without re-scanning the original database, Incremental Mining can be achieved. Now for building the COFI Tree for E considering Modified Inverted Matrix, for the first column reading forward pointers item D is discovered and from backward pointers items H, C and B are discovered. Discarding item H being not frequent and sorting the rest creates a branch of E which is EDCB. This branch is same as generated from Inverted Matrix. The similar process is carried out for building COFI Tree and the final tree for 'E' is shown in Figure 4 which is same as that of generated from Inverted matrix in [17]. So the mining process remains same. Moreover since for each item an independent COFI Tree is built, the mining process can run in parallel. To mine frequent patterns alternate splitting technique is used.

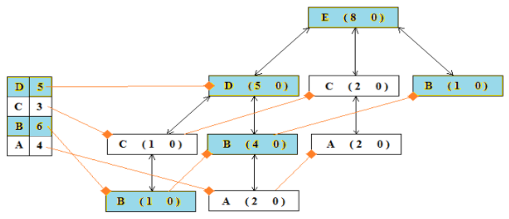

Figure 4: COFI Tree for 'E' using Modified Inverted Matrix [17]

3.1 Alternate Splitting Technique

In this technique, all frequent items from the modified inverted matrix are evenly assigned to m nodes, one by one, i.e. least frequent item is assigned to node 1, and next least frequent item is assigned to node 2 and so on up to m nodes. After that, (m+1)th item is assigned to node 1 and so on until all items are distributed amongst all m nodes. The implementation architecture of this technique is shown in Figure 5. Since the mining process is distributed among parallel nodes the running time of mining process can be significantly reduced. By this we state that this is the best algorithm of its kind which is both incremental and interactive in nature and can be implemented parallel.

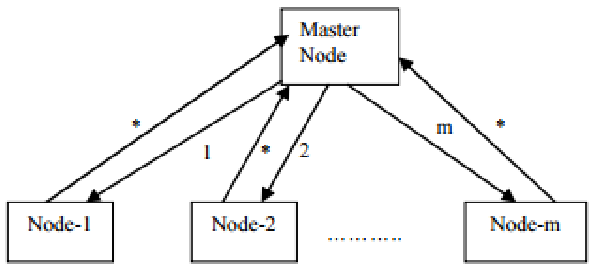

Figure 5: Implementation Architecture of Alternative Splitting

4. CONCLUSION

Association rule mining has been an important part in many of the business, commercial and non-commercial applications. But as the database updates, the maintenance of association rules is an important and critical problem. The overall objective of this work was to achieve the task of incremental mining whenever database changes, without referring to the previous database again and reduce the time to mine frequent items. The proposed Modified Inverted Matrix has been generated with single scan of database. In the paper, through theoretical justifications, it was demonstrated that using Modified Inverted Matrix, same COFI Tree can be developed which we can generate using original Inverted matrix. Also since independent trees are built, the same MIM can be mined in parallel nodes. The proposed modification to the inverted matrix serves two important purposes. First, it can be updated easily by reading the incremented database without referring to the original database. Second, the COFI tree can be generated from the modified inverted matrix. The generated COFI tree can be used for extracting the frequent item-sets. Hence, for generating updated association rules, re-scanning of the original database is not required, which achieves the task of incremental mining. Third, the proposed MIM also supports interactive mining, i.e. whenever desired support changes, it is not required to rebuild the MIM.

REFERENCES

[1] R. Agrawal, T. Imielinski, and A. Swami. Mining Association Rules between Sets of Items in Large Databases. Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data, pages 207—216, May 1993.

[2] J. Han, J. Pei, and Y. Yin. Mining Frequent Patterns without Candidate Generation. Proceedings of the 2000 ACM-SIGMOD International Conference on Management of Data, May 2000.

[3] W.-G. Teng and M.-S. Chen. Incrememtal Mining on Association Rules, National Taiwan University Taipei, 2005

[4] V. Pudi. Data Mining: Concepts and Techniques, Oxford University Press, Jan-2009

[5] C.-C. Chang et al., An Efficient Algorithm for Incremental Mining of Association Rules, Proceedings of the 15th International Workshop on Research Issues in Data Engineering, 2005

[6] D. W. Cheung, J. Han, V. T. Ng, and C. Y. Wong, "Maintenance of discovered association rules in large databases: an incremental updating technique," In *Proc. 12th Intl. Conf. on Data Engineering*, New Orleans, LA, pp. 106-114, Feb. 1996.

[7] C. K.-S. Leung, Q. I. Khan and T. Hoque. CanTree: A Tree Structure for Efficient Incremental Mining of Frequent Patterns, Proceedings of the Fifth IEEE International Conference on Data Mining (ICDM'05), 2005.

[8] N. L. Sarda, N. V. Srinivas. An Adaptive Algorithm for Incremental Mining of Association Rules,

[9] J. Qian, X.-P. Meng. An Adaptive Algorithm for Incremental Mining Association Rules, Proceedings of the Second International Conference on Machine Learning and Cybernetics, Xi'an, November 2003

[10] M.A.B. Tobji, A. Abrougui, B.B. Yaghlane, GUFI: a new algorithm for General Updating of Frequent Itemsets, The 11th IEEE International Conference on Computational Science and Engineering – Workshops, 2008

[11] D. Cheung, J. Han, V.T. Ng, and C. Y. Wong. Maintenance of Discovered Association Rules in Large Databases: An Incremental Updating Technique. Proceedings of the 12th International Conference on Data Engineering, pages 106—114, February 1996.

[12] W. Cheung and O. R. Zaiane. Incremental Mining of Frequent Patterns without Candidate Generation or Support Constraint. Proceedings of the 7th International Database Engineering and Application Symposium, July 2003.

[13] H. Lu, J. Han, and L. Feng. Stock Movement Prediction and N-Dimensional Inter-Transaction Association Rules. Proceedings of the 1998 ACM SIGMOD Workshop on Research Issues on Data Mining and Knowledge Discovery, pages 12:1—12:7, June 1998.

[14] N. F. Ayan, A. U. Tansel, and M. E. Arkun. An Efficient Algorithm to Update Large Itemsets with Early Pruning. Proceedings of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pages 287- 291, August 1999.

[15] H. Toivonen. Sampling Large Databases for Association Rules. Proceedings of the 22th International Conference on Very Large Data Bases, pages 134—145, September 1996.

[16] S. D. Lee, D. W. Cheung, and B. Kao. Is Sampling Useful in Data Mining? A Case Study in the Maintenance of Discovered Association Rules. Data Mining and Knowledge Discovery, 2(3):233—262, 1998.

[17] M. El-Hajj and O. R. Zaıane. Parallel Association Rule Mining with Minimum Inter-Processor Communication. Proceedings of the 14th International Workshop on Database and Expert Systems Applications, 2003

[18] Siddharth Shah, N. C Chauhan and S.D Bhanderi. "Incremental Mining of Association Rules: A Survey",) International Journal of Computer Science and Information Technologies, Vol. 3 (3) , 2012,4071-4074