IACSIT International Journal of Engineering and Technology, Vol.2, No.6, December 2010
ISSN: 1793-8236

Modified Approach for Online Data Mining

Pramod S., *Member, IACSIT*

***Abstract*—The association rule mining and its usage put forwarded lots of hopes in the field of data mining. The researchers in the field are going after the association rule mining techniques to find fastest as well as more precise association rules. Now a day's, online association rule mining is getting its importance due to the popularity of internet as well as the changing behavior of the customer to depend internet for almost everything. The time required for generating frequent itemsets plays an important role in the online mining. Some algorithms are designed as considering only the time factor. Here in this paper our effort is to improve the performance of the online data mining algorithm as by making some important changes in its approach. The results are proved that the new approach improved the performance in the association rule mining in its online environment. The implementation of this algorithm has been tested as by used the dataset from Freequent Itemset Mining(FIM) dataset repository.**

***Index Terms*—Frequent Itemset, Freequent Itemset Mining , Online Data Mining.**

I. Introduction

In recent years the size of the database has increased rapidly. This has led to a growing interest in the development of tools capable in the automatic extraction of knowledge from data. The term data mining or knowledge discovery in database has been adopted for a field of research dealing with the automatic discovery of implicit information or knowledge within the databases. The implicit information within databases, mainly the interesting association relationships among sets of objects that lead to association rules may disclose useful patterns for decision support, financial forecast, marketing policies, even medical diagnosis and many other applications.

The problem of mining frequent itemsets came out first as a sub-problem of mining association rules[1]. Frequent itemsets play an essential role in many data mining tasks that try to find interesting patterns from databases such as association rules, correlations, sequences, classifiers, clusters and many more of which the mining of association rules is one of the most popular problems. The original motivation for searching association rules came from the need to analyze so called supermarket transaction data, that is, to examine customer behavior in terms of the purchased products. Association rules describe how often items are purchased together. For example, an association rule states that four out of five customers that bought beer also bought chips. Such rules can be useful for decisions concerning product pricing, promotions, store layout and many others.

Manuscript received October 28, 2010.
Author is with the Computer science Department, Pt.Ravishankar Shukla University, Raipur,C.G., INDIA. (e-mail: pramodsnair@yahoo.com).

II. Preliminaries

A. Need of Freequent Itemset Mining

Studies of Frequent Itemset (or pattern) Mining is acknowledged in the data mining field because of its broad applications in mining association rules, correlations, and graph pattern constraint based on frequent patterns, sequential patterns, and many other data mining tasks. Efficient algorithms for mining frequent itemsets are crucial for mining association rules as well as for many other data mining tasks. The major challenge found in frequent pattern mining is a large number of result patterns. As the minimum threshold becomes lower, an exponentially large number of itemsets are generated. Therefore, pruning unimportant patterns can be done effectively in mining process and that becomes one of the main topics in frequent pattern mining. Consequently, the main aim is to optimize the process of finding patterns which should be efficient, scalable and can detect the important patterns which can be used in various ways.

B. Preliminaries of new Approach

In the estdec[2] method it examines the transactions one by one without the candidate generation and keeps track of the occurrence count of each item in a monitoring lattice called prefixtree[3][,4]. The estdec method is having two major operations called delayed-insertion and pruning operations. The new itemset will be counted in two different cases:

1) When a new 1-Itemset appears in a newly generated transaction T_k, the monitoring start by incrementing its count and immediately adds that in to the monitoring lattice P_k without any other prior estimation.
2) The second case is that when an insignificant itemset become the significant itemset due to its appearance in the newly generated transactions. So this has to be inserted into the lattice. As by calculating the support using the n-1 subsets stored in the lattice and compare it with the estimated support and the predefined support $S_{ins.}$ The estimated current count $EC_k(e)$ of the itemset e is obtained by the largest among the all subset count. The upper bound of the estimation error for e can be calculated as by subtracting the $C_k(e)$ form $EC_k(e)$ and it is called as delayed insertion operation in estdec method.

The pruning operation will take place while the support of the existing itemset in the P_k becomes less than the predefined pruning support S_{prn}. The itemset regarded as the insignificant itemset that could not be frequent in the near future. So the identified insignificant itemset and its nodes

are pruned from P_k based on the anti-monotonicity of a frequent itemsets[5]. In this paper we have used the sorting of transaction before it considering for any operation to enter in the lattice. The second change we made in the lattice as by keeping all the information including all the items in the itemset at the node. This change has made a significant change in the frequent itemset generation. The third modification we have made by separating the rule generation from the frequent itemset generation. This has improved the performance of both the rule generation as well as the monitoring lattice updation.

III. Itemset-Tree

The Itemset-Tree is the tree used as the data structure. The temporary storage of itemset can be reduced by storing of the node item along with the predecessors. This can be achieved by adding an item in the tree immediately after the root node if it is a single item in the transaction and not in the lattice as the first node after the null node. Let there is a single item transaction as {a} and a is already existing in the lattice then no need to add the same item otherwise add that immediately after the root node by creating a new node. If there is another transaction as {b,a,c} then in our new data structure the transaction has to be ordered and after that it can be entered in the Itemset-Tree in the new order, it will be {a},{ab},{abc} while traverse in depth.

A. Definition 1. Itemset-Tree

Let T_k be an Itemset-Tree and let e denote the itemset and its elements are e= {e_1, e_2, e_3….}. Set of its subsets are added in to the trees after reordering the elements in e.

1) The new node created if $e_1 \subseteq$ e and $e_1 \notin T_k$. From the newly created node e_1 create the child node and the items e_1e_2.
2) Any of the node available then its count incremented and create the non existing element node.
3) Add a transaction under one node along with all the items of its parents. The same transaction with n-1 itemset is in the tree then the n^{th} item to be added and increment the counter for all other.
4) The concatenation of the parent node items to the new node to reduce the temporary storage.

B. Definition 2 Itemset-tree node structure

Given an Itemset-Tree T_k. The node of T_k is node n. For finding frequent itemsets, a data stream can be defined as follows:

Let I_I={i_1,i_2, … ,i_n} be a set of items in the transaction of any application domain.

1) An itemset *e* is a set of items such that e∈(2^I- {Ø}) where 2^I is the power set of *I*. The length |*e*| of an itemset *e* is the number of items that form the itemset and it is denoted by |*e*|-itemset. An itemset {*a,b,c*} is denoted by *abc* in the node while store the last item c in the node.
2) A transaction is a subset of *I* and each transaction has a unique transaction identifier *TID*. A transaction generated at the k^{th} turn is denoted by I_k.
3) The stream D_k is composed of all transactions that have ever been generated so far i.e. $D_k = <T_1, T_2, \ldots, T_k>$ and the total number of transactions in D_k is denoted by $|D|_k$.
4) When a transactions I_k is generated, the current count $C_k(e)$ of an itemset *e* is the number of transactions that contain the itemset among the *k* transactions.

The below given fig:1 is the example of a Itemset-Tree.

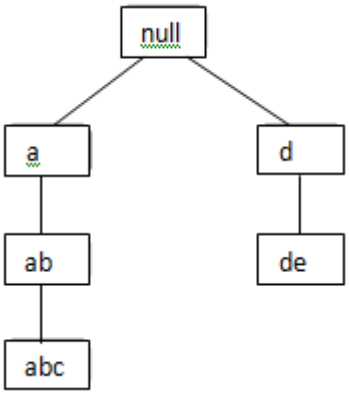


Figure 1: Itemset-Tree T_k

IV. MOnline Method

The decay mechanism mentioned in the estdec method can be used in this paper for MOnline method with little modification. In this paper the decay has been restricted as d= b^{-1} (b>1, b^{-1}<=d<2).

A. Definition 3

Given a decay rate d=b^{-1} (b>1, b^{-1}<=d<2) the total number of transaction $|D|_k$ in the current stream D_k is found as follows.

$$|D|_k = \begin{cases} 1 & \text{if } k=1 \\ |D|_k * d+1 & \text{if } k>=2 \end{cases}$$

The value of $|D|_k$ converges to 1/(1-d) as the value of k increases infinitely.

The delayed insertion and pruning operations are used to trace the current supports of the significant itemsets. The two thresholds used by the estdec S_{ins} and S_{prn} for pruning is also used in the MOnline method for the same purpose. The MOnline method consists of 4 phases.

Sort operation phase: The items in the itemsets are sorted and the sorted stream D_{sort} will be available as output. The items are saved with integer number representation to save the memory space as well as to do the sorting easily.

Add itemset with merge operation: The itemset insertion in the MOnline method is as by finding the first item in the itemset. If the first item in the itemset is in the node it is not added in the lattice as a new node but the count value will be incremented. For the second item in the itemset the new node is created as by adding the first item with second in the newly created node.

Prune Operation: In this step if the cnt/$|D|_{sort}$ goes below the support S_{prn} then the prune operation has to apply on the lattice to prune that itemset along with its entire child.

Delayed insertion with merge operation: In this operation as first step one window was made for finding at what point the write operation has to be made. This step is to improve the performance of the lattice updation.

B. *Algorithm MOnline*

Sort operation

Streamsort(stream D)

For each new transaction in D

sort the read items and store to D_{sort}

Add/Update with merge operation to the lattice

Itemset $e_1 \in e$ and $\notin T_k$ then add at first

Add item e_2 as the sibling of e_1 as $\{e_1e_2\}$ along with parameters

If item $e_1 \in e$ and $\in T_k$ then traverse for e1's child node to search for e_2, if it is found, increment all the parameters otherwise add the new node as $\{e_1e_2\}$

Prune Operation

prune operation(stream D_{sort})

$|D|_{sort}=|D|_{sort}*d+1$

For each itemset update $cnt=cnt*d^{(k-MRtid)}+1$

$err=err* d^{(k-MRtid)}$

$M_{Rtid}=k$

If$(cnt/|D|_{sort})< S_{prn}$ and $|e|>1$ prune e and its child nodes from lattice

Delayed insertion with merge operational

Set a window to monitor the total transactions w

For all itemset e in $\in D_{sort}$ and $e \notin T_k$

Insert e to the T_k if $|e|==1$ then

$C_{nt}=1$; $err=0$; $M_{Rtid}=k$;

If $|e|>1$ then estimate the maximum and the minimum count of eFind the maxcount as by find the node that contain all items in the itemset e

If $(maxcount(e)/|D|_{sort})>=S_{ins}$ then insert e into T_k.

C_{nt}= maxcount(e); $M_{Rtid}=k$;

Frequent itemset selection with write operation

For all itemset $e \in Tk$

$cnt=cnt*d^{(k-MRtid)}+1$; $M_{Rtid}=k$;

if$(cnt/|D|_{sort})>= S_{min}$

$L_k=L_k \cup \{e\}$

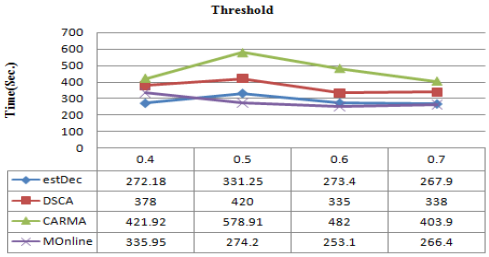
If window size exceeds the predefined window size then write

L_k to another file.

Reset the window.

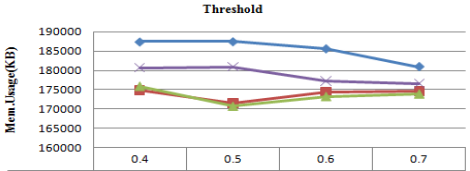
V. Implementation and Comparisons

In this section, the performance of the MOnline method is analyzed by the data set T10I4D100K. In the experiment, the transactions of a data set are looked up one by one in sequence to simulate the environment of an online data stream and the newly generated frequent itemsets are transferred to another location after every 500,1000,5000 and 10000 transactions in different experiments. This is made to find out the changes that influence the lattice structure. In the experiment, compared in all for scenarios the performance with the different thresholds as .4,.5,.6 and .7. Similarly we have compared the memory usage with different thresholds as .4,.5,.6,.7. In all the experiment it is proved that for finding the association rule the process can be divided in two steps as frequent item generation and association rule mining. This two have to be considered as two different process doing all most all independently. The association rule mining will depend on the lattice made for finding frequent itemsets. This approach will help to do more accurate online mining.



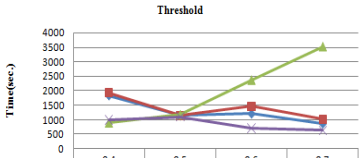
	0.4	0.5	0.6	0.7
estDec	272.18	331.25	273.4	267.9
DSCA	378	420	335	338
CARMA	421.92	578.91	482	403.9
MOnline	335.95	274.2	253.1	266.4

Figure 2 : Threshold Vs. Time (Write, after every 500 Transactions)



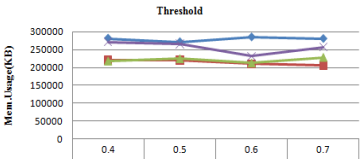
	0.4	0.5	0.6	0.7
estDec	187400	187460	185600	180900
DSCA	174950	171564	174550	174654
CARMA	175950	170750	173300	173950
MOnline	180750	180950	177250	176650

Figure 3 : Threshold Vs. Time/ Mem.Usage (Write, after every 500 Transactions)



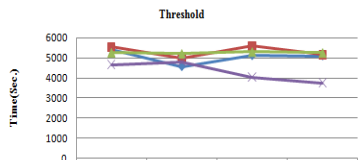
	0.4	0.5	0.6	0.7
estDec	1821.33	1142.7	1208.33	873.43
DSCA	1931	1154	1463	1023
CARMA	905	1188.76	2373.12	3517.82
MOnline	1009.3	1105.12	718.73	649.46

Figure 4: Threshold Vs. Time (Write, after every 1000 Transactions)



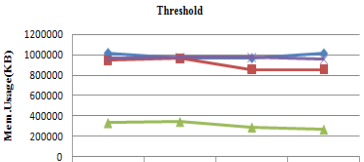
	0.4	0.5	0.6	0.7
estDec	281333.33	271566.66	284600	280300
DSCA	221133.33	220366.66	210500	205900
CARMA	217900	224300	212160	228040
MOnline	271365	265456	231564	256782

Figure 5: Threshold Vs. Time/Mem.Usage (Write, after every 1000 Transactions)



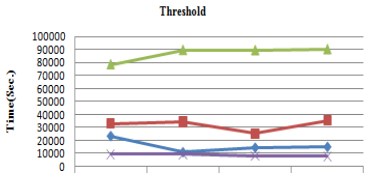
	0.4	0.5	0.6	0.7
estDec	5432	4563	5123	5112.56
DSCA	5548	4956.5	5591.92	5162.07
CARMA	5268.5	5210	5305	5256.87
MOnline	4661.78	4810.28	4055.78	3757.78

Figure 6: Threshold Vs. Time(Write, after every 5000 Transactions)



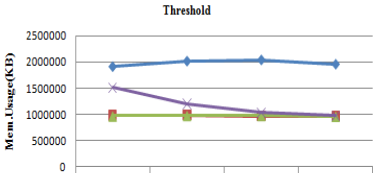
	0.4	0.5	0.6	0.7
estDec	1015928.57	964071.42	966357.14	1012142.8
DSCA	945634	965454	856424	854624
CARMA	332984.64	343591.78	289698	268412
MOnline	966375	974875	973375	956500

Figure 7: Threshold Vs. Time/Mem.Usage (Write, after every 5000 Transactions)



	0.4	0.5	0.6	0.7
estDec	23435.62	11250	14707	15253.87
DSCA	33456	34652	25564	35654
CARMA	78485.9	89576.5	89439.1	90189
MOnline	9335.87	9648.37	8279.25	7845.75

Figure 8: Threshold Vs. Time (Write, after every 10000 Transactions)



	0.4	0.5	0.6	0.7
estDec	1907375	2010125	2032625	1952625
DSCA	975641	985641	965456	965425
CARMA	966375	974875	973375	956500
MOnline	1514125	1206047	1034906	980718.8

Figure 9: Threshold Vs.Mem.Usage (Write, after every 10000 Transactions)

VI. Conclusion

Considering the continuity of a data stream, the general definition of finding frequent itemsets used in conventional data mining methodology may not be valid in a data stream. This is because the old information of a data stream may be no longer useful or possibly incorrect at present. In this paper we propose a modified approach for online data mining, the algorithm named as MOnline. In this algorithm we have proved the change in the item storage can improve the performance. The window used in the algorithm help to reduce the frequent intervenes of the rule generation process to lattice. This allows the two different process to work almost independently.

References

[1] R. Agrawal, T. Imielienski, and A. Swami. Mining Association Rules between Sets of Items in Large Databases. Proc. Conf. on Management of Data, 207–216. ACM Press, New York, NY, USA 1993.

[2] H. Chang and W.S. Lee. Finding recent frequent itemsets adaptively over online data streams. In *Proc.of the 9th ACM SIGKDD*, pp. 487 -492, 2003.

[3] S. Brin, R. Motwani, J. D. Ullman, and S. Tsur. Dynamic itemset counting and implication rules for market basket data. In *Proc. of the ACM SIGMOD Int'l Conference on Management of Data*, pages 255 -264, Tucson, AZ, May1997.

[4] R. C. Agarwal, C. C. Aggarwal and V.V.V. Prasad. Depth first generation of long patterns. In *Proc. of the 6th ACM SIGKDD Int'l Conference on Knowledge Discovery and Data Mining*, pages 108-118, Boston, MA, Sept. 2000.

[5] R. Agrawal, T. Imielinski, and A. N. Swami. Mining Association Rules between Sets of Items in Large Databases. In Proc. of SIGMOD, 1993.

[6] M. Garofalakis, J. Gehrke and R. Rastogi. Querying and mining data streams: you only get one look. In *the tutorial notes of the 28th Int'l Conference on Very Large Databases*.

[7] G. S. Manku and R. Motwani. Approximate frequency counts over data streams. In *Proc. of the 28th Int'l Conference on Very Large Databases*, Hong Kong, China, Aug. 2002.

[8] M. Charikar, K. Chen and M. Farach-Colton. Finding frequent items in data streams. In *Proc. of the 29th Int'l Colloq. on Automata, Language and Programming*, 2002.

[9] C.-H. Lee, C.-R. Lin and M.-S. Chen. Sliding-window filtering: An efficient algorithm for incremental mining. In *Proc. of the 10th Int'l Conference on Information and Knowledge Management*, pages 263-270, Atlanta, GE, Nov. 2001.

[10] B.-K. Yi, N. D. Sidiropoulos, T. Johnson, H. V. Jagadish, C. Faloutsos, and A. Biliris Online data mining for co-evolving time sequences. In *Proc. of the 16th Int'l Conference on Data Engineering*, pages 13-22, San Diego, CA, Feb. 2000.

[11] H. S. Javitz and A. Valdes. The NIDES statistical component description and justification. Annual report, March 1994.

[12] C. Hidber. Online association rule mining. In *Proc. of the ACM SIGMOD Int'l Conference on Management of Data*, pages 145-156, Philadelphia, PA, May 1999.

[13] R. Agrawal, and R. Srikant. Fast algorithms for mining association rules. In *Proc. of the 20th Int'l Conference on Very Large Databases*, Santiago, Chile, Sept. 1994.

Pramod S. is a Ph.D. student in Computer Science Department, Pt. Ravishankar Shukla University, Raipur, C.G., INDIA. His main area of research is Data mining. For Data mining, he is interested in Association Rule Mining and Online Data Mining.