IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 5, Issue 6, June 2016

Distributed Association Rule Mining on Batchwise Data

Nisha Kapoor[1], Nidhi Sonpal[2], Dhwani Mehta[3], Vinaya Sawant[4]

Student, IT Department, D.J. Sanghvi College of Engineering, Mumbai, India [1, 2, 3]

Assistant Professor, IT Department, D.J. Sanghvi College of Engineering, Mumbai, India [4]

Abstract: With the growing amount of data generated from millions of transactions taking place every second all over the world, it has become increasingly necessary to find interesting patterns from this data. Multinational companies, being spread over the globe, have to integrate data from various geographically dispersed sites. This data is generated from varied sources in heterogeneous forms and has to be processed before mining can be carried out on it. Mining association rules from data involves finding correlations between two or more variables in a dataset. Algorithms like Fast Distributed Mining (FDM) and Count Distribution Algorithm (CDA) have been used for association rule mining in a distributed environment. However, these algorithms prove to be inefficient when it comes to dynamically streaming input. Our proposed solution suggests a methodology to implement Association Rule Mining on Distributed Systems using ODAM (Optimized Distributed Association Rule Mining) algorithm on batchwise data. Further, in an effort to aid the user in analysing the output, we display the association rules generated for each item as well as for the entire dataset. Also, the user can view the time series analysis for each association rule.

Keywords: ODAM, Batchwise Data, Apriori, Distributed Association Rule Mining.

I. INTRODUCTION

Distributed Association Rule Mining (DARM) generates association rules from geographically distributed datasets. DARM generates global frequent itemsets by combining and analysing frequent itemsets generated at local sites. However none of the DARM algorithms, Fast Distributed Mining (FDM), Count Distribution Algorithm (CDA) or ODAM (Optimized Distributed Association Rule Mining) can handle dynamically streaming data. They are designed to work in a static environment where the database does not change or get added to.

In a real world scenario, transactional data is always increasing with time. New transactions get added to the database at the rate of 1000 every second. We need a time efficient solution which repeatedly generates rules for the newly added batches of transactions. The changing trends in the association rules as new data comes in should be accurately reflected. Our proposed solution suggests a methodology to implement Association Rule Mining on Distributed Systems using ODAM (Optimized Distributed Association Rule Mining) algorithm on batchwise data. It avoids repeated scanning of the old transactions thus giving a considerable time reduction in generating rules when the data is input batchwise. This makes it ideal for real world applications where it can generate association rules keeping pace with continuously streaming in batches of data. Further, in an effort to aid the user in analyzing the output, we display the association rules generated for each item as well as for the entire dataset. This helps the user to better visualize how a single item is related to all other items in the database. The user can view the changing trend of each association rule over all the previous batches to get an idea of how the support for the rule has changed over time.

II. REVIEW OF LITERATURE

The papers referred by us helped us in formulating an idea of the different Distributed Association Rule Mining (DARM) algorithms.

A. Study of Existing Systems

Association Rule Mining (ARM) is a method of finding interesting patterns for strategic analysis and business decisions from huge datasets[1]. It includes various algorithms – Apriori and Frequent Pattern Mining. In these algorithms for each iteration we find the candidate itemsets and prune the infrequent itemsets. ARM when implemented in distributed environment generates globally frequent patterns for an organization. It includes algorithms like Count Distribution Algorithm (CDA), Fast Distribution Mining (FDM) algorithm and Optimized Distributed Association Rule Mining (ODAM) algorithm [2].

Count Distribution Algorithm (CDA) is a data parallelism algorithm for mining associative rules. It locally computes support counts for each itemsets and generates candidate sets based on the local minimum support count. These candidate sets along with their support counts are broadcasted to all other nodes. Each node computes global frequent itemsets in parallel. At the end of each iteration, all the nodes have same global frequent itemsets [3].

The main feature of CDA is that it uses a simple synchronization scheme as it uses only one set of messages at each iteration[4].

Fast Distribution Mining (FDM) algorithm is a modification of CDA. Here, in each iteration support counts of all itemsets are computed. Considering the local minimum support count for that node, the infrequent items are pruned locally and the candidate set is generated. The node, at which an itemset is frequent, broadcasts requests to all other nodes for collecting their support counts for that itemset. Once the node receives the counts, it computes the global count for that candidate. It compares this count with the global minimum support count to check if the candidate is globally large. At the end of each iteration globally large itemsets found are broadcasted to all the nodes[5].

The three main features of FDM are:

- Using some interesting relationships between locally and globally generated sets, we minimize the number of candidate sets generated and thus reduce the number of messages sent.
- FDM uses both local and global pruning.
- This algorithm requires O(n) for communicating with each other.

If the distribution of the itemsets among the partitions is skewed such that the globally frequent itemsets are confined to a few local sites, then lesser candidate sets are generated; thus improving the performance of the algorithm. Another major improvement in FDM is the usage of relaxation factor. With the help of this, a site broadcasts not only those candidate sets which clearly satisfy the minimum support threshold but also the ones which almost satisfy it. This increases the effectiveness of global pruning[6].

Optimized Distributed Association Rule Mining (ODAM) algorithm involves a central server which receives candidate sets from all the other nodes and generates global frequent itemsets.

In CD and FDM algorithms numerous candidate itemsets are generated, which involves high communication costs. In contrast, ODAM eliminates infrequent items, thereby reducing dataset size significantly; so that we can accumulate more transactions in the main memory. Moreover, the time taken for scanning the database at each iteration is less. As an added advantage, ODAM reduces communication costs and enforces synchronization between local nodes since they communicate only with global server[7].

B. Present Architecture

A large database DB with D transactions is partitioned among n geographically distributed sites as DB_1, DB_2 …. DB_n. The DARM architecture also consists of a global server which communicates with all the local sites. For every iteration, each site uses its local database to compute candidate itemsets which are above the predefined local minimum support count. The local sites send these candidate itemsets along with their support counts to the server.

Based on the counts received, the server calculates the global count for each itemset by methods such as averaging, summation, etc. If the count for the itemset exceeds the global minimum support count, then the itemset is considered globally frequent. The newly computed global frequent itemsets are sent back to all the local sites. The local sites eliminate the infrequent items from the transactions in memory so as to reduce the average transaction length which makes the scanning of transactions faster for future iterations.

C. Drawbacks of Existing System

- Cannot be used with dynamic database.
- Too many or too less rules may be generated based on threshold values.
- Rules generated is not well presented for human analysis
- Takes a lot of time to execute the entire algorithm and generate results.
- The entire algorithm needs to be executed again when new transactions get added.

III. PROPOSED SOLUTION

Our proposed solution involves modifying the ODAM algorithm to make it suitable for batch processing in a time efficient manner.

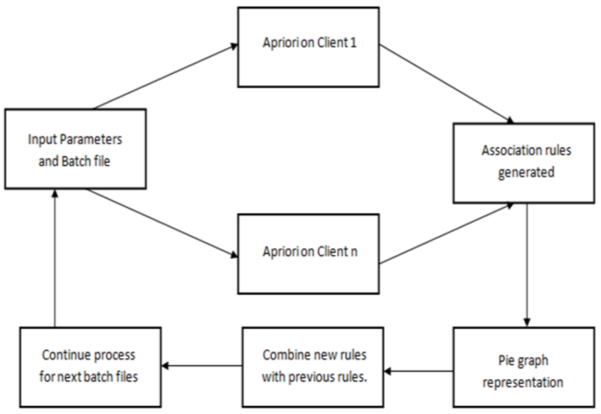

Apriori on Client 1
Input Parameters and Batch file
Association rules generated
Apriori on Client n
Continue process for next batch files
Combine new rules with previous rules.
Pie graph representation

Figure 1: Block Diagram of Proposed Solution

Input Parameters and Batch file: User provides transactional batch file and enters parameters such as batch number, number of items in given transaction file, number of transactions, minimum support count, minimum threshold etc. Thus these parameters can be adjusted as and when required. We use Netbeans Swing GUI to develop this interface.

Apriori on Client: Apriori acts as a base algorithm for most parallel algorithms. This algorithm helps to generate local frequent itemsets. Client connects to the global site using RMI registry[8].

Association rules generated: Frequent itemsets from various clients are combined at a global site. This merging involves averaging support counts from various clients to generate global frequent itemset. Based on global frequent itemsets association rules are generated to show correlations between various itemsets. Further global frequent itemset is used for transaction reduction, which is performed by clients on their respective machines to increase space utilization by eliminating infrequent items from the transactions.

Pie graph representation: To make analysis of the association rules easier, we display the association rules generated for each item as well as for the entire dataset. This helps to understand correlation of an item to various other items in the dataset. Each pie chart displays the support count along with the association rule to allow user understand the strength of the rule generated. For generating pie charts we make use of JFrames.

Combine new rules with previous rules and continues the process next batch file: The new transactions that keep coming are collected over an interval of time to form the next batch file. This newly generated batch file is used to compute new association rule by combining the results of the previous batch with the new one.

Time series analysis: To analyze how the support for a particular rule has changed over the past batches, we display the support counts of that rule over the all the batches till date in the form of a line graph. This helps the user in analyzing the upward of downward trend in the rules and also in predicting the future support expectancy of the rule.

A. Benefits of proposed solution

- The user can view the association rules of the item he is interested in, making analysis simpler and more apt for analysis.
- With pie chart representation user can easily differentiate strong association rules and weak association rules
- The user can view the changing trend of each rule over all the previous batches to get an idea of how the support for the rule has changed over time.
- The UI designed makes the entire process of generating association rule very flexible.
- The recent trends can be combined with old trends in a space and time efficient manner.
- The ODAM algorithm used provides global analysis and supports transaction reduction, which helps saves memory.
- The database scan to be performed is much smaller and thus quicker in batch based computing than static approach.
- Comparing recent trends with old trends is easily achieved and can be used to develop time line analysis of changing business scenario.
- Saves bandwidth costs as only counts of frequent itemset are sent to global site instead of entire transactions.
- After being used to find association rules, the old sets of transactions becomes redundant and are not required to be stored in memory.

B. Proposed Architecture

The architecture consists of a global server which communicates with all the local sites using Remote Method Invocation. The clients will provide the parameters of the batch and the support count based on which the frequent itemsets will be generated. For every iteration, each site uses Apriori algorithm to compute candidate itemsets which are above the predefined local minimum support count. The local sites send these candidate itemsets along with their support counts to the server.
Based on the counts received, the server calculates the global count for each itemset by methods such as averaging. If the count for the itemset exceeds the global minimum support count, then the itemset is considered globally frequent. The newly computed global frequent itemsets are sent back to all the local sites with their support counts. The local sites then performs transaction reduction, where the infrequent transactions from the database are removed so as to reduce the average transaction length which makes the scanning of transactions faster for future iterations and also the saves the memory.
Once all the iterations are done, the association rules are computed on the server side and are displayed using pie charts. One pie chart shows the association rules of all items present in the dataset together and other pie charts are of each item individually which helps to analyze the rules of items the user is interested in.
Moreover, after successively running the algorithm on several batches of data, the time series analysis chart for any rule over all the batches is generated and displayed to the user.

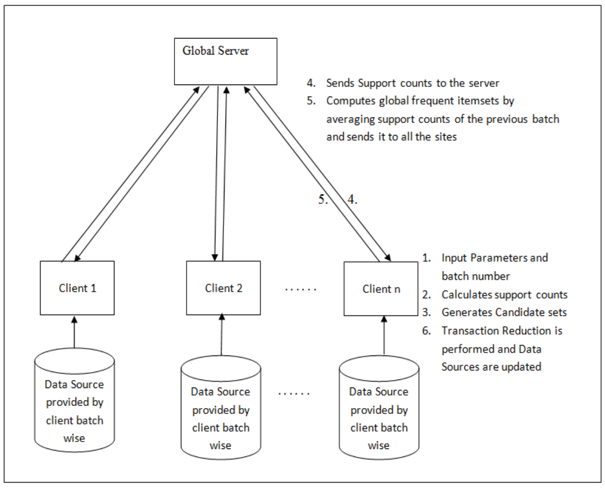

Global Server
4. Sends Support counts to the server
5. Computes global frequent itemsets by averaging support counts of the previous batch and sends it to all the sites
5.
4.
Client 1
Client 2
......
Client n
1. Input Parameters and batch number
2. Calculates support counts
3. Generates Candidate sets
6. Transaction Reduction is performed and Data Sources are updated
Data Source provided by client batch wise
Data Source provided by client batch wise
......
Data Source provided by client batch wise

Figure 2: Proposed Architecture

After the association rules of a batch are generated the user can continue the process by inputting another batch of transaction file and its parameters. For the new batch the same process is repeated, but the computation is done only on the new set of transactions and their support counts are averaged with those of the previous batch, thereby improving the efficiency and saving the time to compute the rules.

IV. MODULE WISE ALGORITHM DESCRIPTION

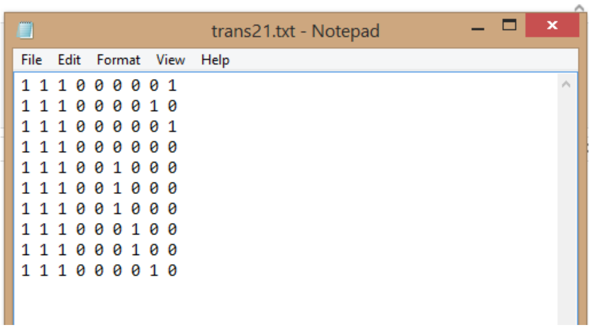
The following section describes the algorithm implemented:

Setting up a distributed environment

- The Java Remote Method Invocation (RMI) system used for communication in our project.
- Start the RMI registry using the command prompt. Commands are:
- rmic updownImpl which generates stub and skeleton files used for communication
- rmiregistry along with the port number to run the RMI registry
- Run the RMI server using java command
- Connect the clients with this system with the help of the IP address of the system

Inputting Parameters

- The clients must input the parameters for each batch of transactions to implement the association rule mining on the dataset
- The parameters on client side are the batch number, name of the transaction file on which the mining has to be implemented, number of transactions & items present the batch and the support count to generate the frequent itemsets
- These input parameters are to be inserted in the UI that appears when the user runs project
- When rules for a batch are computed and if the user wishes to compute for the another batch, the input parameters of the another batch have to inserted in the same manner after choosing the continue option on the UI screen
- On the server side also the user has to enter the parameters, namely, the batch number, the minimum support count, the minimum confidence level and the number of clients present in the system



```
1 1 1 0 0 0 0 0 1
1 1 1 0 0 0 0 1 0
1 1 1 0 0 0 0 0 1
1 1 1 0 0 0 0 0 0
1 1 1 0 0 1 0 0 0
1 1 1 0 0 1 0 0 0
1 1 1 0 0 1 0 0 0
1 1 1 0 0 0 1 0 0
1 1 1 0 0 0 1 0 0
1 1 1 0 0 0 0 1 0
```

Figure 3: Transaction File

This is an example of a batch of data which consists of a set of transactions on which the association rule mining is performed.

Generating Frequent Itemsets

Apriori Algorithm for generating Frequent Itemsets:

- The user input parameters are stored in the variables and the itemset number is incremented
- generateCandidate () method is called with itemset number as the parameter which generates candidate sets based on the minimum threshold value provided by the user
- From the candidate sets the frequent itemsets are computed by calling the calculateFrequentItemsets () method, where itemset number is passed as the parameter
- This process takes place on the client side

Communication

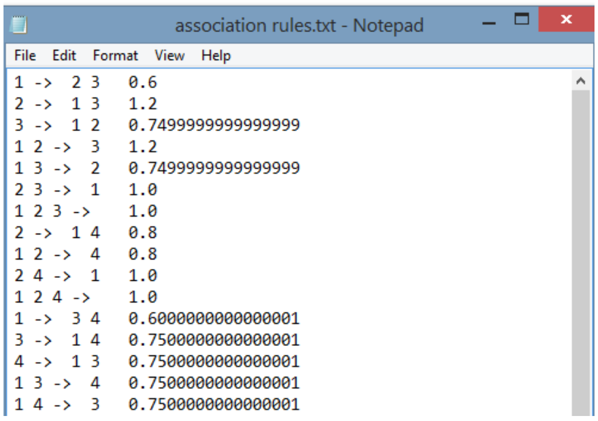
- The frequent itemsets generated are sent to the server with the help of the RMI server
- Naming.lookup() method is invoked to communicate with the server to send the files
- The server generates global frequent itemsets by eliminating infrequent itemsets based on the support count provided to generate the frequent itemsets
- Server sends the frequent itemsets with global counts back to all the clients again using the Naming.lookup () method

Transaction Reduction

- In this module, when the client receives the frequent itemset file back with the support counts, it looks for the transactions which contain only infrequent items.
- When it finds such transactions, it deletes them thus reducing the transaction file size and making the algorithm memory efficient[8].

Generating Association Rules

- When the server receives all the frequent itemset files after all the iterations, it computes all the association rules from the global frequent itemsets based on the support count and the minimum confidence level.
- A text file is created to store the association rules, if any rule is below the confidence level; it is eliminated.
- Once the association rules are generated and stored in the text files, with the help of the RMI server connection, the files are sent to all the clients, so the clients can view the rules on their systems



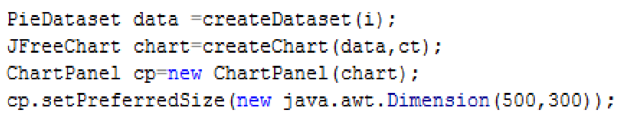
```
1 ->  2 3   0.6
2 ->  1 3   1.2
3 ->  1 2   0.7499999999999999
1 2 ->  3   1.2
1 3 ->  2   0.7499999999999999
2 3 ->  1   1.0
1 2 3 ->    1.0
2 ->  1 4   0.8
1 2 ->  4   0.8
2 4 ->  1   1.0
1 2 4 ->    1.0
1 ->  3 4   0.6000000000000001
3 ->  1 4   0.7500000000000001
4 ->  1 3   0.7500000000000001
1 3 ->  4   0.7500000000000001
1 4 ->  3   0.7500000000000001
```

Figure 4: Association Rules

The above figure is a snapshot of association rules generated and stored in the text file

Displaying association rules

- Once the rules are generated, the pie charts are generated to represent the association rules
- Various classes of Java are used to create the pie charts, snippet of our code to create pie charts is shown in the image below:



```
PieDataset data =createDataset(i);
JFreeChart chart=createChart(data,ct);
ChartPanel cp=new ChartPanel(chart);
cp.setPreferredSize(new java.awt.Dimension(500,300));
```

Figure5: Code to Create Pie Charts

- With the help of these classes and the code to form pie charts from the data provided from the association rules file, different pie charts are generated
- Various pie charts are generated, they include association rules of all the items together and association rules of each item separately for a clear breakdown

Continue the process for the next batch file

- If the user wants to compute the association rules for another batch of transaction, then he/she has to enter the new (incremented) batch number
- Also the name of the new transaction file and its parameters have to be entered by the user in UI to generate the association rules of the new set of the transactions combined with the previous batches
- With the help of the proposed solution, the computation will be done only with the new set of data by simply averaging the support counts with those of the previous batch to improve the efficiency of the data and thereby increasing the dynamicity of the algorithm

Performing Time Series Analysis

- Thereafter, the user can view the time series analysis chart for any rule he chooses. The support counts for that rule are extracted from the previous batch files and displayed to the user in the form of a line graph.
- With the help of JFree classes and the code to form line graphs from the data extracted from the batch files, the time series analysis graph is generated

V. RESULTS AND DISCUSSIONS

Experiments were carried out for testing with following:

- Consider huge dataset with large numbers of transactions and items, divided into n number of smaller batch files which are given to the algorithm at regular intervals.
- Run the batch files on general ODAM algorithm and compare its result with proposed algorithm.

For example, we implemented the existing ODAM algorithm on two clients with 10 transactions initially. These transactions were taken from the TTT database from UCI repository [9]. The execution time came out to be 135 sec on client 1. After this, new batches of 10 transactions were appended to each client. The existing ODAM algorithm took 285 sec to run a total of 20 transactions on client 1.

Total time taken (ODAM) = 135+285
= 420 seconds

On other hand, the proposed solution took 135 sec to execute the same initial 10 transactions. However, when the new 10 transactions got appended, it executed the new transactions in 150sec only at client 1, since the previous transactions were not scanned again.

Total time taken (proposed solution) =135+150
=285 seconds

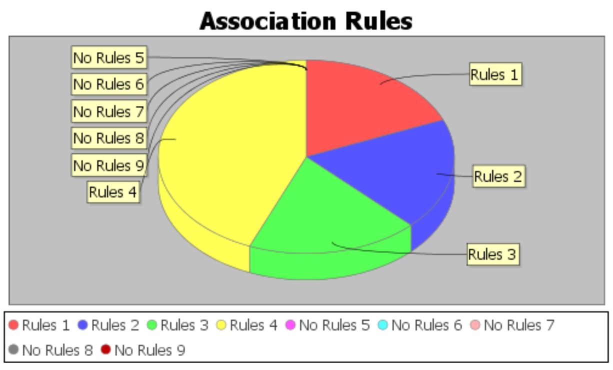
Results

- There was a considerable reduction in time when the proposed solution was run on the batch files instead of the ODAM algorithm. As seen in the above example, the time saved is 132 sec at client 1. Time savings = $\frac{420-285}{420} * 100$
32.14%

The greater the amount of previous transactions which are not rescanned, the more the saving in time. After each batch, the time saved will increase. Therefore the execution time reduces to a considerable amount as a whole.

- Based on the above experiments, similar association rules are generated when implemented using normal ODAM and proposed solution.
- These rules are displayed to the user in the form of pie charts

Association rules for all items together:



Association rules for individual item (Rules for items 1 are displayed below):

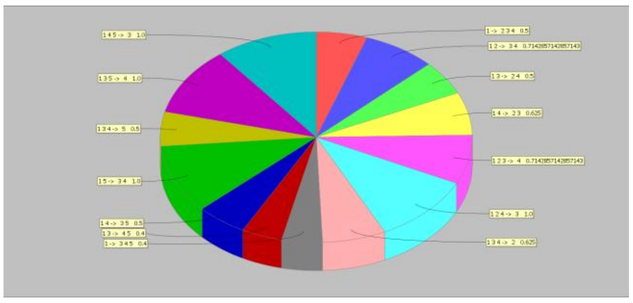


Figure 6: Association rules of item 1

The above figure clearly shows which association rules of item 1 have higher support and also which items it pairs better with.

The below chart shows the time series analysis for the association rule 1 2 4 -> 3. Distributed association rule mining has been carried out successively over five batches of data. The graph shows the changing trend in the given rule over the five batches.

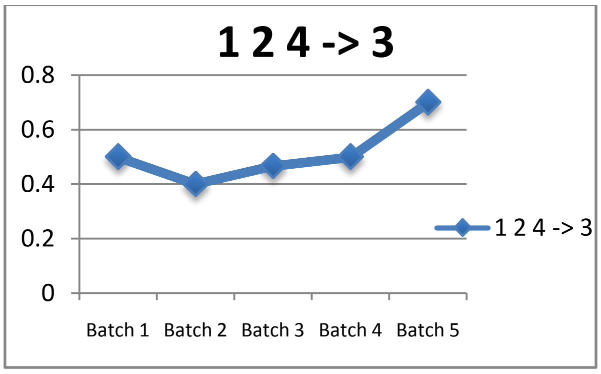


Figure 7: Time Series Analysis

By analysis of the graph we can expect that the support for association rule 1 2 4 ->3 will continue to rise in the next batch, as per the trend observed in the previous batches. Thus marketing and sales strategies involving items 1, 2, 3 and 4 can be planned accordingly.

VI. CONCLUSION

After researching and working for 10 Months, we have implemented Distributed Association Rule Mining on batchwise data. We leveraged the power of the existing algorithm for distributed association rule mining, namely, Optimized Distributed Association Rule Mining (ODAM), and added the functionality of implementing it on dynamic data. The proposed solution gives great results and is bound to improve on the time and space efficiency in processing batchwise data.

The proposed solution further improves upon ODAM by:

- The recent trends can be combined with old trends in a space and time efficient manner.
- With pie chart representation user can easily differentiate strong association rules and weak association rules.

The individual modules are also of extreme importance from the standpoint of distributed data mining:
The transaction Reduction module eliminates infrequent items from the transactions, thereby reducing the average transaction length and increasing space utilization.
The pie chart generation module displays the association rules related to each item which helps the users to view the association rules of the item he is interested in, making analysis simpler.
The time series analysis module allows the user to view the changing trends in the support for a rule over the past batches. This helps in predicting the future support expectancy for that rule.
The code allows for the algorithm to be run on any number of clients connected in a network with minimal tradeoffs in computation time.

VII. FUTURE SCOPE

The following are the points which can be added to our project:

- Implementing it using Hadoop environment can further enhance the speed.
- The algorithm can be further modified to allow dynamic thresholding so that user does not have to worry about deciding the ideal threshold to generate appropriate association rules.
- Automating the batch process iterations after certain time period; eg. - hourly basis, daily basis, etc.
- Modify the algorithm further for dynamic streaming of data from web so it can be used with e-commerce websites.
- Modify the algorithm to work with vertically fragmented database and hybrid-fragmented databases.

REFERENCES

[1] Qiankun Zhao and Sourav S. Bhowmick, "Association Rule Mining: A Survey", Communication of the Association for Information Systems, 2003.
[2] Mafruz Zaman Ashrafi, David Taniar, Kate Smith, "ODAM: An Optimized Distributed Association Rule Mining Algorithm", IEEE DISTRIBUTED SYSTEMS ONLINE 1541-4922 © 2004 Published by the IEEE Computer Society Vol. 5, No. 3; March 2004.
[3] Vinaya Sawant, 2 Ketan Shah, "A Survey of Distributed Association Rule Mining Algorithms", Journal of Emerging Trends in Computing and Information Sciences Vol. 5, No. 5, May 2014.
[4] Byung-Hoon Park and Hillol Kargupta, "Distributed Data Mining: Algorithms, Systems and Applications"
[5] Cornelia Győrödi , "A Comparative Study Of Distributed Algorithms In Mining Association Rules"
[6] David W Cheung, Jiawei Han, Vincent T Ng, Ada W. Fu, Yongjian Fu, "A Fast Distributed Algorithm for Mining Association Rules", International Conference on Parallel and Distributed Systems Proceedings, 1996, p. 31-42.
[7] Vinaya Sawant And Dr. Ketan Shah, "A Review Of Distributed Data Mining Using Agents", International Journal of Advanced Technology & Engineering Research (IJATER)
[8] Jaiwei Han, Micheline Kamber and Jian Pie, "Data Mining Concepts and Techniques"
[9] Bache, K. & Lichman, M. (2013). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.