Distributed Association Rule Mining with Minimum Communication Overhead

Md. Golam Kaosar, Zhuojia Xu and Xun Yi
School of Engineering and Science, Victoria University, Australia
Victoria University PO Box 14428, Victoria 8001, Australia
md.kaosar@live.vu.edu.au, zhuojia.xu@live.vu.edu.au, xun.yi@vu.edu.au

Abstract

In distributed association rule mining algorithm, one of the major and challenging hindrances is to reduce the communication overhead. Data sites are required to exchange lot of information in the data mining process which may generates massive communication overhead. In this paper we propose an association rule mining algorithm which minimizes the communication overhead among the participating data sites. Instead of transmitting all itemsets and their counts, we propose to transmit a binary vector and count of only frequently large itemsets. Message Passing Interface (MPI) technique is exploited to avoid broadcasting among data sites. Performance study shows that the proposed algorithm performs better than two other well known algorithms known as Fast Distributed Algorithm for Mining Association Rules (FDM) and Count Distribution (CD) in terms of communication overhead.

Keywords: MPI, Data Mining, Association rule mining, AllGather, AllReduce.

1. Introduction

Though information technology (IT) is considered one of the greatest blessings of technology at current era, rapid inflation of information may explode the whole arena of IT if it is not supervised properly. Data mining is one of the means to utilize information by discovering underlying hidden useful knowledge from information. Among different approaches, association rule mining is one of the popular techniques for mining data. In this technique, an interrelation among different items in data is discovered by determining frequent large itemsets which are repeated more than a threshold number of times in the database. Association rule mining can enhance in extracting knowledge in various applications including advertisements, bioinformatics, database marketing, fraud detection, E-commerce, health care, security, sports, telecommunication, web, weather forecasting, financial forecasting, etc. Data mining process can be characterized as centralized and distributed based on the location of data. In case of centralized data mining process, data is resided into a single site whereas in distributed process data is resided into multiple sites. The data may be owned by each site separately or an enormous amount of data may be distributed into multiple data sites.

In distributed ubiquitous computing environment lot of devices, sensors, terminals, equipments, computers, etc. are connected to each other through heterogeneous communication means in which minimization of bandwidth usage is considered as a major concern. To launch data mining applications in such an environment must require an algorithm which minimizes communication overhead.

Association rule mining process can be divided into two major tasks: (a) computation of all frequently large itemsets and (b) generation of all strong association rules which satisfy certain constraints. Since task (b) is considered as straightforward, most research efforts focus on task (a). In this paper we propose a distributed association rule mining algorithm to accomplish task (a) with the objective of minimizing communication overhead.

Significant amount of research work has been performed in association rule mining algorithms. Apriori algorithm proposed by R. Agrawal and R. Srikant (1994) is a classical and popular association rule mining algorithm which is suitable for centralized data mining. Due to the necessity of rapidly growing distributed computing environment, distributed mining algorithms become popular in the market. Distributed data mining algorithm is also necessary to ensure security and privacy in many other circumstances. R. Agrawal and J.C. Shafer (1996) propose a parallel and distributed association rule mining algorithm known as Count Distribution (CD). Main focus of CD is to reduce the communication overhead with the cost of redundant computation in all sites. This model is suitable for a system for which the computational capability dominates the communicational capability. Another algorithm, Fast Distribution Algorithm (FDM) of W. Cheung, J. Han, V. T. Ng,A. W. Fu, Y. Fu (1996), proposed to reduce the number of candidate sets generated in local sites, consequently reduces the communication overhead. It introduces local and
global pruning techniques to eliminate redundant computation. Both of these algorithms are discussed in section 3 of this paper in more detail.

Q. Ding, Q. Ding, W. Perrizzo (2008) propose an efficient algorithm for mining association rules from spatial data for remote sensed imagery (RSI) data. In most association rule mining algorithms, binary relationship (presence or absence) in transactions are considered. But there are some efforts which consider the weight of the transactions too. A weighted association rule mining technique is introduced in K. Sun, F. Bai (2008). Weight of the association rule is measured by introducing a link-based model. A definition of weighted support is introduced and a weighted association rule mining (WARM) algorithm is proposed by F. Tao, F. Murtagh, M. Farid (2003). Both profit and purchased quantity are considered in mining transactional data in S. J. Yen, Y. S. Lee (2007). Different research efforts focus to accomplish different objectives but there is not much research work found which focus to minimize communication overhead in data distributed mining process. Therefore the proposed communication efficient algorithm might lead to bright possibility of deploying data mining applications in ubiquitous computing environment.

The CD algorithm is proposed to reduce the communication overhead without focusing much about the computational overhead. On the other hand DFM focuses on pruning in local and global level to reduce computation in the data sites. Therefore it is speculated that combination of these two algorithms might lead to a simple and efficient solution for association rule mining. Farther contemplation on the algorithms revealed that not all the count values of itemsets are necessary to be transmitted. We introduce an idea of determining the frequent itemsets first without exchanging the counts and then exchanging the counts of only those frequently large itemsets which are locally frequent in at least one site. Not only that, it is also possible to eliminate the redundant computation in each data site. Finally we come up with a new association rule mining algorithm idea, which might perform better than CD and FDM in terms of communication overhead.

Rest of the paper is organized as the following order: Section 2 describes relevant background information in brief. The proposed algorithm is described in section 3 while section 4 illustrates the performance analysis and comparison. Finally section 5 concludes the proposed algorithm.

2. Background

In this section some techniques and algorithms are discussed as background information in brief which are related to the proposed algorithm.

Message Passing Interface (MPI): This is a technique to exchange information among a number of communicating nodes. It is especially suitable for mathematical functions like: summation or accumulation of a particular number which is to be calculated and distributed among nodes. This allows nodes to exchange the information without broadcasting; therefore, it reduces the communication overhead and communication round significantly. Detail of MPI can be found in R. Agrawal and J. Shafer (1996) and Argonne National Laboratory (MPI). Following example illustrates the functionality of MPI in brief.

Let us consider 8 nodes S₁, S₂,…, S₈ have their own count values to be summed and shared among themselves. One straightforward solution is to broadcast everyone’s count to others in the common medium. This practice would require massive data transmission over the medium. Moreover, broadcasting is avoided due to many other reasons if alternatives are feasible. MPI, in this case, provides the best solution which is divided into two sub tasks: ReduceScatter and AllGather. In ReduceScatter, the sum of the counts is accumulated in a single predefined node whereas in AllGather, the sum is transmitted back to all nodes to be shared. Fig.1 shows how ReduceScatter accumulates a distributed value. It requires \( \log_2 N \) (3 in this case) steps to accumulate the result in the converging site.

![Fig.1: Depicts how distributed data is accumulated in a converging node (ReduceScatter).](image-url)

Step 1: Nodes S₁, S₂, S₃ and S₇ transmit their counts to nodes S₂, S₃, S₅ and S₈ respectively.
Step 2: S₂ and S₃ transmit their counts along with the counts of S₁ and S₇ respectively.
Step 3: Now S₁ has counts of S₁, S₂ and S₃, and then transmits all of them to node S₅. Finally node S₅ will have the counts of all nodes. Now node S₅ is capable of calculating the sum of the counts.

Participating sites in the MPI technique can be divided into three categories: edge sites (in this example S₁, S₅, S₆ and S₇) are those which do not receive from others, intermediate sites (in this example S₂, S₃ and S₄) are those sites which receives from other sites and...
converging site (in this example $S_k$) receives all information from others and accumulates.

In the second stage, the sum or unified value of all counts is transmitted back to all the nodes, which works in the reverse manner and known as AllGather.

In MPI, if there are $N$ nodes in the network, then the total number of required transmission is $2(N-1)$, on contrary it is $N(N-1)$ in the case of broadcasting. The number of communication round is $2(\log_2 N)$.

**Association Rule Mining:** Let us consider; in a distributed data mining environment collective database $DB$ is subdivided into $DB_1$, $DB_2$, ..., $DB_N$ in collection of data sites $S_1$, $S_2$, ..., $S_N$ respectively. $I = \{i_1, i_2, \ldots, i_m\}$ is the set of items where each transaction $T \subseteq I$. Typical form of an association rule is $X \Rightarrow Y$, where $X \subseteq I$ and $Y \subseteq I$ and $X \cap Y = \emptyset$. The support $s$ of $X \Rightarrow Y$ is the probability of a transaction in $DB$ containing both $X$ and $Y$. On the other hand confidence $c$ of $X \Rightarrow Y$ is the probability of a transaction containing $X$ will contain $Y$ too. Usually it is the interest of the data miners to find all association rules having support and confidence greater than or equal to minimum threshold value. Let us look at the equations of support and confidence for another instance of an association rule $AB \Rightarrow C$,

$$\text{Support}_{AB \Rightarrow C} = s = \frac{\sum_{i=1}^{m} \text{support \_count}_{ABC(i)}}{\sum_{i=1}^{m} \text{database \_size}(i)}$$

$$\text{Support}_{AB} = \frac{\sum_{i=1}^{m} \text{support \_count}_{AB(i)}}{\sum_{i=1}^{m} \text{database \_size}(i)}$$

$$\text{Confidence}_{AB \Rightarrow C} = c = \frac{\text{Support}_{AB \Rightarrow C}}{\text{Support}_{AB}}$$

More detail on association rule mining process is discussed by J. Han, M. Kamber (2006) and P. N. Tan, M. Steinbach, V. Kumar (2006).

Apriori algorithm proposed by R. Agrawal and R. Srikant (1994) is one of the leading algorithms, which determines all frequently large itemsets along with their support counts from a database efficiently. A brief description of the algorithm is as follows:

Let us say $L_i$ be the frequent $i$-itemset. Apriori algorithm finds $L_k$ from $L_{k-1}$ in two stages: joining and pruning:

(i) Joining: Generates a set of $k$-itemsets $C_k$ known as candidate itemsets by joining $L_{k-1}$ and other possible items in the database.

(ii) Pruning: Any $(k-1)$-itemsets cannot be a subset of a frequent $k$-itemsets which is not frequent. Therefore it should be removed.

**Count Distribution (CD):** In brief, Count Distribution (CD) algorithm works as follows: Each processor or data site generates its local candidate sets based on the global large itemsets of previous iteration using Apriori algorithm. Then it calculates each support count and exchanges with other sites using Message Passing Interface (MPI) technique. Since this protocol exchanges all counts, each site can generate global frequent large itemsets which might be utilized for the following iterations. Due to the use of the same algorithm, all the processors generate same global frequent large itemsets. CD algorithm can be summarized into five major stages:

(i) Each processor generates candidate itemset $C_k$ based on globally frequent large itemset $L_{k-1}$.

(ii) Each processor computes local support count for $C_k$ by passing through the transactions in the database.

(iii) All processors exchange their $C_k$ counts to develop global $C_k$ using MPI technique.

(iv) Each processor computes $L_k$ from $C_k$.

(v) Each processor takes the decision either to continue or to stop. Decision will be the same since they have identical $L_k$.

**Fast Distributed Algorithm (FDM):** Fast Distributed Mining of Association Rules (FDM), was proposed by W. Cheung, J. Han, V. T. Ng, A. W. Fu, Y. Fu (1996). The main idea of this protocol can be summarized as follows:

(i) Computing candidate set: Each site generates candidate set based on globally large ($k$-1)-itemsets and locally large ($k$-1)-itemsets using Apriori algorithm.

(ii) Local pruning: For each item in the candidate set: if the support of the itemset is larger than minimum support, that particular item is added in the locally large k-itemsets.

(iii) Count exchange: Each site broadcasts locally frequent large itemsets to all other sites.

(iv) Globally frequent large itemset computation: Each site computes globally large k-itemsets which is utilized for the following iteration.

**3. Proposed Algorithm**

Let us consider a distributed environment with $N$ number of data sites $S_1$, $S_2$, ..., $S_N$ possessing horizontally partitioned transactional data $DB_1$, $DB_2$, ..., $DB_N$ respectively. All these sites intend to share their data to mine knowledge. Each site agrees to certain threshold values of minimum support ($s$) and confidence ($c$). This proposed algorithm generates all frequently large itemsets having their support values
more than or equal to s. Some of the notations used in the algorithm are included in the following table (Table 1):

<table>
<thead>
<tr>
<th>Notations</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>i-set</td>
<td>An itemset consists of i elements/items</td>
</tr>
<tr>
<td>L_k</td>
<td>Set of all frequent k-itemsets sorted alphabetically. L_k is calculated and maintained in all the sites parallel.</td>
</tr>
<tr>
<td>C_k^i</td>
<td>Set of candidate k-itemsets in site S_i generated from L_k and sorted alphabetically.</td>
</tr>
<tr>
<td>λ_k^i</td>
<td>Set of count values of C_k^i in the order of C_k^i.</td>
</tr>
<tr>
<td>β_k^i</td>
<td>Set of calculated support values of C_k^i in the order of C_k^i.</td>
</tr>
<tr>
<td>V_k^i</td>
<td>Binary vector for C_k^i in the order of C_k^i. If (β_k^i) ≥ s then (V_k^i) = 1 else 0</td>
</tr>
</tbody>
</table>

**Table 1: Notations**

Each site maintains a table known as itemset-table to hold C_k^i, λ_k^i, β_k^i and V_k^i. A typical itemset-table is depicted in fig.3. In every iteration i, the table is updated based on the value of L_{i-1} in all sites.

Exchange of information is performed using MPI technique (discussed in previous section). Let us assume the communication sequence (who transmits to whom) and the converging node are predetermined. In the example provided in fig.1, S_8 sends its count to S_6 and S_6 will send counts of itself and of S_3 to S_5. In that example S_8 is considered as the converging site. Therefore for N nodes S_N can be considered as the converging site.

A certain property of existence of itemsets in database is utilized in the proposed algorithm which is summarized as the following theorem: “If X is a globally frequent large itemset, there exists at least one site where X is locally large”. Proof of the theorem is provided by W. Cheung, J. Han, V. T. Ng, A. W. Fu, Y. Fu (1996). Instead of exchanging the count of all itemsets, this algorithm transmits the information about whether a particular itemset is locally large or not (it is stored in the binary vector V_k^i) in the first attempt. In the following attempt sites only transmit the count of those itemsets which are locally large at least in one site. Thus a significant amount of communication overhead is reduced by avoiding transmitting unnecessary count values of all itemsets.

Major stages of the algorithm can be distinguished as follows:

**Step 1:** Each site S_i exchanges list of its items to all other N-1 sites using MPI technique. Each site computes list of 1-itemsets L_1 and sorts them in alphabetical order.

**Step 2:** Site S_i generates candidate set C_k^i from L_k by appending all non-repeated items with all itemsets in L_{k-1}. As for example let us say L_2 = {a, b, c, d} and L_3 = {ab, cd} then C_k^i = sort ((abc, abd, cda, cdb)) = {abc, abd, acd, cdb} where k=3. Then S_i computes its local λ_k^i, β_k^i and V_k^i and updates the itemset-table.

**Step 3:** If S_i is edge site (as depicted in fig.1), it transmits V_k^i to its pre-assigned sites. If S_i is intermediate sites, it receives all binary vectors from predetermined sites. Once all vectors from all expected sites are received, S_i performs OR operation among all vectors and transmits to its pre-assigned site. If S_i is converging site, it performs OR operation on all the vectors received and returns the resultant vector to all sites.

Let us assume an instance of a binary vector [0 1 1 1 0 1] which represents seven itemsets. The vector also implies that all itemsets are frequent except first and the fifth. As soon as this vector is received in the next node, it performs binary OR operation with its own binary vector and the received one. The result is transmitted to the following node (as illustrated in fig.2). It should be noted that all the nodes are transmitting the same amount of data which is the binary vector of fixed size for a particular iteration. If the number of itemsets are Q in a particular iteration, then the size of the vector would be log_2 Q. At the final stage the converging node comes up with a resultant vector (let us name it V_f), which represents the frequent large itemsets of the following cycle (L_{k+1}). The figure fig.2 displays an example of a typical exchange of binary vector with some sites (S_1, S_3, S_5 and S_8):

**Fig.2:** Computation of L_A from binary vector without exchanging the count values of itemsets.
In the above figure $S_i$, the converging site calculates $V_i = [111001]$ which, implies that first three and the last itemsets are frequent. Therefore $L_k$= {first, second, third and sixth}. Finally the converging site transmits $V_i$ to all sites.

Step 4: Site $S_i$ (for all $i \leq \{1, 2, \ldots, N\}$) removes the entries in the itemset-table for which the value in $V_i$ is 0. Thus $S_i$ generates $L_k$ which would be used to generate $C_{k+1}$ in the following iteration. Thus the pruning (elimination of itemsets which have supports less than s) mechanism happens in step 3 and step 4 together.

Step 5: Site $S_i$ exchanges $C_{k+1}^i$ as the same way it was done in step 4 and 5 except addition operation is done instead of OR operation. In this case an integer vector would propagate instead of the binary vector. Therefore the size of the vector would be higher too. The converging site would receive the total count of all the itemsets which are at locally large at least in one site. Finally converging site returns the count values of all itemsets in $L_k$.

Step 6: Repeat step 2 to step 5 until there exists a large itemset with support $\geq s$.

Following figure (Fig.3) illustrates communication steps of the algorithm among data sites. Fig.3 also shows a typical entry of itemset-table of any site.

Let us assume $s=0.5$ and total number of transactions=100

**Fig.3: An instance of communication among data sites.**

### 4. Performance Analysis

In this section a performance comparison would be presented to compare the number of communication rounds and amount of communication overhead necessary to transmit to compute frequent large itemsets in every iteration. The analytical comparison involves CD, FDM and the proposed algorithm.

Let us consider following parameters:

- $H$= Average number of items in the large $k$-itemsets (number of rows in the itemset-table).
- $L$= Number of Bytes to store count values in itemset-table.
- $N$= Number of data sites.
- $M$= Average size of each item in Byte (number of characters as for example).
- $M'$= Average number of items in each local candidate itemsets.

Communication overhead and number of communication round in each iteration for CD, FDM and proposed algorithm are as follows respectively:

**CD algorithm:** In CD; as discussed in the protocol description, data sites do not transmit the itemsets themselves. Instead, it transmits the counts of the itemsets since all sites have identical set of itemsets. Thus it reduces the amount of overhead to be transmitted in the network.

Therefore total communication overhead in each iteration $P_{CD}= Overhead involved in forward communication (ReduceScatter) + Overhead involved in backward communication (AllGather) (as MPI technique discussed in Section 2)$

\[
P_{CD} = 2 \times H \times L \times (N - 1)
\]

It is also obvious from the protocol that, number of transmission is $O(N)$ and number of communication round is $O(\log_2 N)$.

**FDM algorithm:** In FDM local pruning reduces the number of items significantly. Therefore the overhead of FDM for each cycle is as follows:

\[
P_{FDM-LP} = N(N - 1) \times M \times M' + N(N - 1) \times 0.25 \times H \times L
\]

It is claimed by W. Cheung, J. Han, V. T. Ng, A. W. Fu, Y. Fu (1996) that FDM reduces the number of itemsets by 75% to 90% by local pruning. According to the claim we consider their lower bound of optimization and assume average number of entries in the table be reduced to 0.25H in case of FDM. Therefore number of transmission is $O(N^2)$ and Number of communication round is $O(N)$. 

\[
\text{Proc. of the 8th Australasian Data Mining Conference (AusDM'09)}
\]

Page 21
The proposed algorithm: The proposed algorithm implements the pruning technique from the initial stages of the algorithm by avoiding transmitting the count of any itemset unless it is frequent at least in one site. Therefore pruning reduces the number of items significantly similar to FDM. For the sake of comparison it is assumed that the average number of entries in the table is reduced to 0.25H; as it is in case of FDM. In that case, the overall overhead will be the summation of overhead in transmitting binary vector, overhead of returning resultant vector $V_r$ from converging node and transmission of counts of frequent large itemsets. Therefore,

$$P = (N - 1) \times \left[ \frac{2 \times H}{8} + 0.25 \times H \times L \right]$$

which can be deduced to

$$P = (N - 1) \times H \times \frac{3L}{4}$$

Therefore number of transmission is $O(N)$ and number of communication round is $O(\log_2 N)$.

For the sake of performance comparison different parameters in the above derived performance equations are varied while keeping other parameters constant and assigning reasonable values to some of the parameters. Few performance comparisons are illustrated in Fig.4 and Fig.5.

Fig.4: Average communication overhead in each cycle with H is 10, L is 4 M is 5 and M’ is 3.

In fig.4 it is clearly depicted that as the number of data sites goes up, the overhead for FDM goes up most rapidly since it involves broadcasting. On the other hand CD has better performance since it does not broadcast to exchange the count information. On the other hand the proposed algorithm performs the best since it avoids broadcast and minimizes the size of the frequent large itemset. In addition to that, technique of binary vector exchange enables it to avoid exchanging the counts of non frequent itemsets. Therefore this algorithm minimizes the communication overhead most effectively.

Fig.5: Communication overhead in each cycle with N is 8, L is 4 M is 5 and M’ is 3.

Similarly fig.5 depicts that the proposed algorithm performs the best when average size of k-itemsets is varied keeping the number of sites constant.

5. Conclusion

FDM and CD have their own advantages and disadvantages in different circumstances and in various systems. Though FDM introduces some techniques to minimize candidate itemsets, it overloads the network by broadcasting too much data. On the other hand CD minimizes the communication overhead by avoiding broadcast but it does not consider optimization in frequent large itemsets generation. But avoidance of broadcasting, utilization of pruning technique and transmission of binary vector instead of count values reduces the network overhead significantly in the proposed algorithm. Performance equations show that the proposed algorithm performs better than CD and FDM in terms of communication overhead. Furthermore, centralized pruning and mining and binary vector exchange technique also would reduce some computational overhead in all sites, which is not considered in the performance evaluation. We believe further experiments and implementation of the proposed protocol would strengthen its efficiency and usefulness.

Acknowledgement

This research contribution was supported and funded by Australian Research Council (ARC) Discovery Project (DP0770479) – “Privacy Protection in Distributed Data Mining”.
References


W. Cheung, J. Han, V. T. Ng,A. W. Fu, Y. Fu (1996): A fast distributed algorithm for mining association rules, 4th International Conference on Parallel and Distributed Information Systems, 18-20 pp. 31-42.


