ISSN:0975-9646

A Survey on Association Rule Mining Algorithm and Architecture for Distributed Processing

[1]. Imran Qureshi [2]. Jammi Ashok [3] Vinaysagar Anchuri

[1]Associate Professor, [2]Head of CSE Dept, [3]Assistant Professor
[1,2,3]Department of Computer Science and Engineering,
Guru Nanak Institute of Technology, Hyderabad, AP-INDIA.

Abstract-Association rule mining is a data mining technique used to uncover previously unknown hidden patterns or rules from huge databases usually tera and peta bytes of data. There are many popular algorithms for mining various association rules like Apriori, portioning, dynamic item set counting etc. But the main drawback of these algorithms is their sequential nature. Processing large databases in sequential order has many disadvantages like time consuming, scalability and performance issues. In order to avoid the above said problems we look for parallel or distributed association rule mining for providing scalability and better performance.

Keywords: Association, mining, frequent item sets, Apriori algorithm

I INTRODUCTION

Association rule mining is used to find relationships in a given data set. Many organizations are showing their interest to discover such relationships in their databases which helps them to take strategic business decisions to improve their performance. One classic example where association rules mining is market basket analysis which is used to analyze customer buying habits by finding association between between different items placed in their basket. By doing so they get an insight into which items are bought frequently altogether. By this relationship they can make some decisions like arranging the store in such a manner that all frequent items bought together are placed in opposite racks or by given some promotional offers like if X is bought then there is 5% discount on Y. In this way sales will be increased. Market basket analysis is just one example where association rule mining is used it can also be used different purposes like marketing, advertising, floor placement and inventory control. Although they have been used for other purposes as well including predicting faults in telecommunication networks. Association rules are used to show the relationships between data items Computer antivirus [support =2%, confidence=60%]

The above rule says that if a computer is purchased then there is a 60% possibility that he may purchase antivirus also. And among whole transactions 2% of transactions are such that computer and antivirus are purchased altogether. Support and confidence are two measures of rule interestingness. Support has to be set by the user and care has to be taken that it should not be too large or too small.

II BASIC CONCEPT

The most common approach to finding association rules is to break up the problem into two parts:

1. Find all frequent item sets
2. Generate strong association rules from frequent items.

Finding all frequent item sets is a difficult task where as generating strong association rules are less costly

A large frequent item set is an item set whose number of occurrences is above threshold, s. Once the large item sets have been found, we know that any interesting association rule X→Y, must have X U Y in this set of frequent item sets. Note that the subset of any large item sets is also large. Finding large item sets generally is quite easy but very costly. The naive approach would be to count all item sets that appear in any transaction. Given a set of items of size m, there are 2m subsets. Since we are not interested in empty set, the potential number of large item sets is then 2m-1. Because of the exclusive growth of this number, the challenge of solving association rule problem is often viewed as how to efficiently determine all large items sets. When m=5, there are potentially 31 item sets when m=30 this becomes 1073741823. Most association rule mining algorithms are based on smart ways to reduce the number of item sets to be counted. These potentially large item sets are called candidates, and the set of all counted i.e. large item sets are called candidate item set (c).one performance measure used for association rule algorithm is the size of C. Another problem to be solved by association rule algorithm is what data structure is to be used during the counting process. A trie or hash tree is common.

III ALGORITHM TO FIND FREQUENT ITEM SETS USING SUPPORT FUNCTION

Input:
D database of transaction
I items
L large item sets
S support
α confidence
Output
R Association rules satisfying s and α
R=Ø;
For each l ἐ L do
For each x ϵ l such those x# Ø do
If support (l)/support(x)>= α then
R=RU {x→ (1-x)};

IV COMPARATIVE ANALYSIS OF BASIC ASSOCIATION RULE MINING ALGORITHMS

IV.I Apriori Algorithm

The apriori algorithm is the most well known association rule algorithm and is used in most commercial products. It uses the following property which we call the large item set property:

Any subset of a large item set must be large

The large item sets are also said to be downward closed because if an item set satisfies the minimum support requirements, so do all of its subsets. The basic idea of the apriori algorithm is to generate candidate item sets of a particular size and then scan the database to count these to see if they are large. During scan i, candidates of size i, Ci are counted. Only those candidates that are large are used to generate candidates for next pass. That is Li are used to generate Ci+1. An item set is considered as a candidate only if all its subsets also are large. To generate candidates of size i+1, joins are made of large item sets found in previous pass. An algorithm called Apriori-Gen is used to generate the candidate item sets for each pass after the first. All singleton item sets are used as candidates in the first pass. Here the set of large item sets of previous pass, Li-1 is joined with itself to determine the candidates. Individual item sets must have alto be combined.erl but one item in common in order to be combined.

Algorithm for apriori algorithm

Pass 1

1. *Generate the candidate item sets in C_1*
2. *Save the frequent item sets in L_1*

Pass k

1. *Generate the candidate item sets in C_k from the frequent item sets in L_{k-1}*
    1. *Join L_{k-1} p with L_{k-1}q, as follows:*
        insert into C_k
        select $p.item_1$, $p.item_2$, . . . , $p.item_{k-1}$, $q.item_{k-1}$
        from L_{k-1} p, L_{k-1}q
        where $p.item_1 = q.item_1$, . . . $p.item_{k-2} = q.item_{k-2}$, $p.item_{k-1} < q.item_{k-1}$
    2. *Generate all (k-1)-subsets from the candidate item sets in C_k*
    3. *Prune all candidate item sets from C_k where some (k-1)-subset of the candidate item set is not in the frequent item set L_{k-1}*
2. *Scan the transaction database to determine the support for each candidate item set in C_k*
3. *Save the frequent item sets in L_k*

Let us see one working example

Original table:

Transaction ID	Items Bought
T1	{M, O, N, K, E, Y }
T2	{D, O, N, K, E, Y }
T3	{M, A, K, E}
T4	{M, U, C, K, Y }
T5	{C, O, O, K, I, E}

Step 1: Count the number of transactions in which each item occurs, Note 'O=Onion' is bought 4 times in total, but, it occurs in just 3 transactions.

Item	No of transactions
M	3
O	3
N	2
K	5
E	4
Y	3
D	1
A	1
U	1
C	2
I	1

Step 2: Now remember we said the item is said frequently bought if it is bought at least 3 times. So in this step we remove all the items that are bought less than 3 times from the above table and we are left with

Item	Number of transactions
M	3
O	3
K	5
E	4
Y	3

This is the single items that are bought frequently. Now let's say we want to find a pair of items that are bought frequently. We continue from the above table (Table in step 2)

Step 3: We start making pairs from the first item, like MO, MK, ME, MY and then we start with the second item like OK, OE, and OY. We did not do OM because we already did MO when we were making pairs with M and buying a Mango and Onion together is same as buying Onion and Mango together. After making all the pairs we get,

Item pairs
MO
MK
ME
MY
OK
OE
OY
KE
KY
EY

Step 4: Now we count how many times each pair is bought together. For example M and O is just bought together in {M,O,N,K,E,Y}While M and K is bought together 3 times in {M,O,N,K,E,Y}, {M,A,K,E} AND {M,U,C, K, Y}

After doing that for all the pairs we get

Item Pairs	Number of transactions
MO	1
MK	3
ME	2
MY	2
OK	3
OE	3
OY	2
KE	4
KY	3
EY	2

Step 5: Golden rule to the rescue. Remove all the item pairs with number of transactions less than three and we are left with

Item Pairs	Number of transactions
MK	3
OK	3
OE	3
KE	4
KY	3

These are the pairs of items frequently bought together.
Now let's say we want to find a set of three items that are brought together.
We use the above table (table in step 5) and make a set of 3 items.

Step 6: To make the set of three items we need one more rule (it's termed as self-join),
It simply means, from the Item pairs in the above table, we find two pairs with the same first Alphabet, so we get

- OK and OE, this gives OKE
- KE and KY, this gives KEY

Then we find how many times O,K,E are bought together in the original table and same for K,E,Y and we get the following table

Item Set	Number of transactions
OKE	3
KEY	2

While we are on this, suppose you have sets of 3 items say ABC, ABD, ACD, ACE, BCD and you want to generate item sets of 4 items you look for two sets having the same first two alphabets.

- ABC and ABD -> ABCD
- ACD and ACE -> ACDE

And so on … In general you have to look for sets having just the last alphabet/item different.

Step 7: So we again apply the golden rule, that is, the item set must be bought together at least 3 times which leaves us with just OKE, Since KEY are bought together just two times.
Thus the set of three items that are bought together most frequently are O,K,E.
Next step is to generate strong association rules from frequent item sets which satisfy minimum support threshold and confidence threshold
Confidence (A→B) =prob (B/A) =support (AUB)/support (A)
Algorithm for finding strong association rule
For each frequent item set, l , generate all non empty subsets of f.
For every non empty subset s of l do
Output rules s→ (l-s) if support (f)/support(s)>=min_confidence

The Apriori algorithm takes advantage of the fact that any subset of a frequent item set is also a frequent item set. The algorithm can therefore, reduce the number of candidates being considered by only exploring the item sets whose support count is greater than the minimum support count. All infrequent item sets can be pruned if it has an infrequent subset.
Drawbacks of apriori algorithm are transaction database are memory resident and requires many database scans

IV.II AIS Algorithm

Candidate item sets are generated and counted on-the-fly as the database is scanned
For each transaction, it is determined which of the large item sets of the previous pass are contained in this transaction
New candidate item sets are generated by extending these large item sets with other items in this transaction.
The disadvantage of the AIS algorithm is that it results in unnecessarily generating and counting too many candidate item sets that turn out to be small.

IV.III SETM Algorithm

Candidate item sets are generated on-the-fly as the database is scanned, but counted at the end of the pass
New candidate item sets are generated the same way as in AIS algorithm, but the TID of the generating transaction is saved with the candidate item set in a sequential structure.
At the end of the pass, the support count of candidate item sets is determined by aggregating this sequential structure.
The SETM algorithm has the same disadvantage of the AIS algorithm. Another disadvantage is that for each candidate item set, there are as many entries as its support value.

IV.IV AprioriTid Algorithm

The database is not used at all for counting the support of candidate item sets after the first pass.
The candidate item sets are generated the same way as in Apriori algorithm.
Another set C' is generated of which each member has the TID of each transaction and the large item sets present in this transaction. This set is used to count the support of each candidate item set
The advantage is that the number of entries in C' may be smaller than the number of transactions in the database, especially in the later passes.

IV.V AprioriHybrid Algorithm

Apriori does better than AprioriTid in the earlier passes. However, AprioriTid does better than Apriori in the later passes. Hence, a hybrid algorithm can be designed that uses Apriori in the initial passes and switches to AprioriTid when it expects that the set C' will fit in memory.
The main drawback of these association rule mining algorithms are they are sequential and are not scalable.

V PROBLEM STATEMENT

The association rule mining algorithms require more number of database scans which is a major drawback if the size of database is large in order to overcome this problem

focus has to be shifted to parallel association rule mining which overcomes the scalability problem as well as performance issues in sequential association rule mining.

VI METHODOLOGY

Parallel processing means doing multiple things at a single time. Generally there are two types of parallelism techniques they are data parallelism and task parallelism. Data parallelism focuses on distributing the data across different parallel computing nodes. Task parallelism is used when you have multiple tasks to be done. Task parallelism divides the tasks among multiple processors. Examples for task parallelism is pipelining, image processing and graphic processing

Let us see one example how parallelism speeds up the execution time. Assume we have to process 1 GB of data to be processed on a single processor. Assume time taken to process 1GB data is 100 seconds. Now let us assume that you have 5 processors working in parallel and you have divided data into blocks each of 205MB. Assume time taken to complete one block of data is 10 seconds so the total time taken to complete the task is 10 seconds which is a significant improvement over sequential manner.

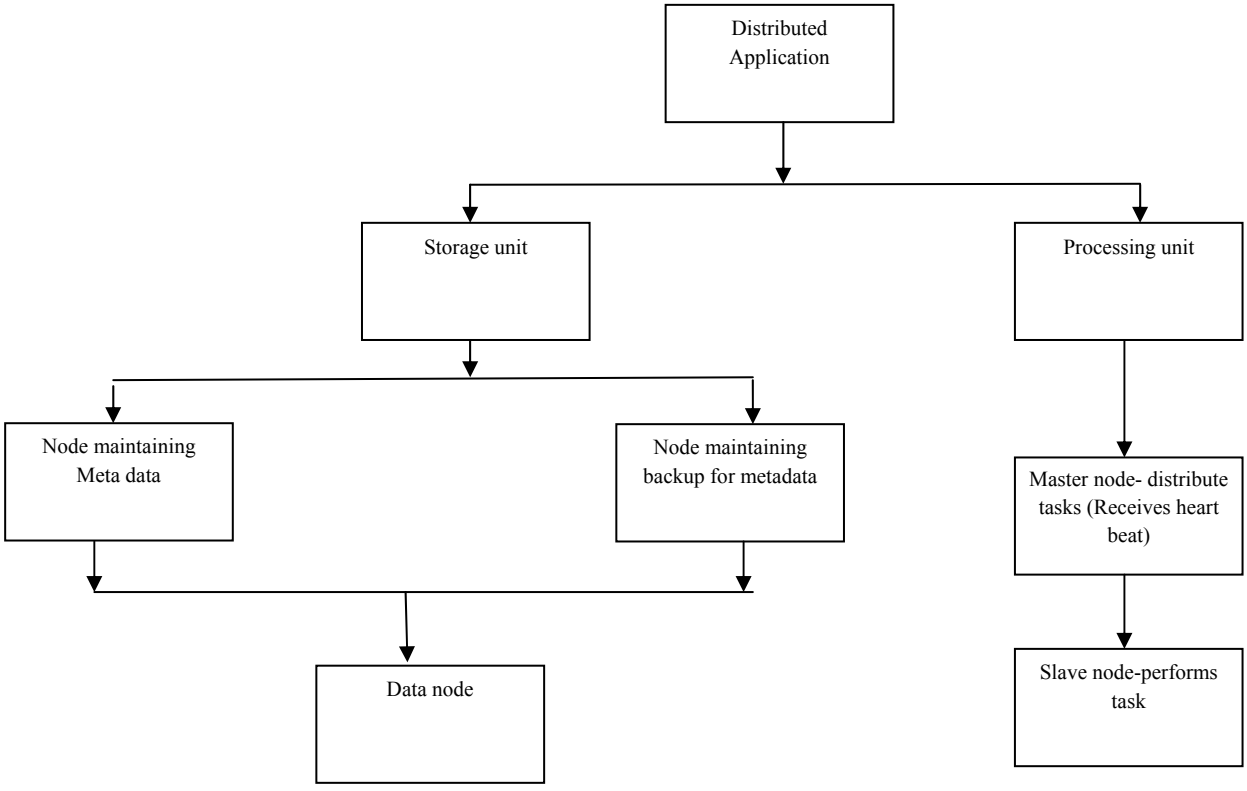
Traditional parallelism approach was to move data to the process which results in communication overhead. That is in this approach 205 MB of data (one block) is moved over network to some location where t has to be processed. This results in communication bottleneck. The new approach to overcome above problem is "process moves to the data". That is algorithm which is to process the data will be moved over network to some location where data has to be processed. The size of the process is very small compared to data.

VII DISTRIBUTED ARCHITECTURE FOR ASSOCIATION RULE MINING

The distributed architecture mainly consists of two units responsible for storage and processing respectively. This architecture is a master and slave architecture where master assigns some works to the slaves and in turn slaves do the job and reports to the master again. The storage unit is again divided into two parts responsible for keeping metadata i.e. the location where the data blocks are stored which is useful for fault tolerance and keeping backup of metadata respectively. Let us see now one working example. Let us assume we want to process 1GB of data in distributed environment.

The task is given to processing unit which reduces or partitions the data into block each of 64MB. Approximately 16 blocks, these blocks are distributed to various nodes to be processed. This distribution has to be done once over network. Now 1GB data is divided into 16 blocks each at different location. Now for suppose we want to find out frequent patterns in this 1GB data. So we have to load the association rule mining algorithm to the processing unit, the algorithm will run at 16 different nodes at the same time and thus the efficiency is increased to a greater extent. Let us assume if data explodes over night massively still we can achieve the performance by adding more nodes. Adding nodes is less costly compared to adding servers. Thus scalability is achieved using distributed architecture

Parallel processing architecture:



Fault Tolerance

Fault tolerance is achieved using simple technique called replication factor of 3. That is each block is replicated three times and is placed at different locations. For suppose let us assume one slave has stopped responding in this case the data at that slave node is lost. At regular intervals of time master checks for heartbeats of slaves, if master is unable to find heartbeat of a slave it immediately contacts its Meta data and finds where the replication is located and immediately instructs the slave where replica is available to process the damaged block also. In this way my system is totally fault tolerance

VIII CONCLUSION

I have described a novel architecture for distributed association rule mining. I have discussed various components and their responsibilities for handling the task. Future work can be done on how to recover the slave nodes which has stopped responding.

REFERENCES

[1] Data Mining Introductory and advanced topics by Margaret H.Dunham
[2] Data Mining concepts and Techniques by jiawei Han and Michelin kamber second edition
[3] M Zaki "Parallel and distributed association mining: A survey"
[4] http://www.saedsayad.com/association_rules.htm
[5] http://nikhilvithlani.blogspot.in/2012/03/apriori-algorithm-for-data-mining-made.html
[6] ttp://www.ijcta.com/documents/volumes/vol4issue2/ijcta2013040226.pdf
[7] Hadoop Architecture from Google

AUTHOR'S BIOGRAPHY

Imran Qureshi is presently working as Associate Professor in CSE department at Guru Nanak Institute of Technology, Hyderabad, and A.P. He received his B. Tech Degree in Information Technology from Mother Theresa Engineering College and M.Tech. with specialization in Computer Science from Al-Habeeb Engineering College, Hyderabad. His main research interest includes Data mining, Data Structures and algorithms and Distributed operating systems. He is Pursuing his PhD from Jawaharlal Nehru Technological University, Hyderabad.

Prof Jammi. Ashok is presently working as Professor and Head of CSE department at Guru Nanak Institute of Technology, Hyderabad, A.P. He received his B.E. Degree from Electronics and Communication Engineering from Osmania University and M.E. with specialization in Computer Technology from SRTMU, Nanded. His main research interest includes pattern Recognition, Neural Networks, Data mining and Artificial Intelligence. He has been involved in the organization of a number of conferences and workshops. He published more than 55 papers in International journals and conferences. He has submitted his PhD thesis in Anna University in 2012.

Vinaysagar Anchuri is presently working as Assistant Professor in Dept. of CSE at Guru Nanak Institute of Technology, Hyderabad, AP. He took his B. Tech degree from Christu Jyothi Institute of Technology and Science, Jangaon, Warangal in Computer Science and Engineering and M. Tech from Adam's Engineering College, Polancha, Khammam. His main interest includes Data Mining, Software Engineering and Semantic Web.