USING HASH BASED APRIORI ALGORITHM TO REDUCE THE CANDIDATE 2-ITEMSETS FOR MINING ASSOCIATION RULE

K.Vanitha and R.Santhis
Department of Computer Studies
Saranathan College of Engineering
Trichy, India
rkvanithamca@gmail.com
Department of Computer Studies
Saranathan College of Engineering
Trichy, India
anandsanthi@rediff.com

Abstract: In this paper we describe an implementation of Hash based Apriori. We analyze, theoretically and experimentally, the principal data structure of our solution. This data structure is the main factor in the efficiency of our implementation. We propose an effective hash-based algorithm for the candidate set generation. Explicitly, the number of candidate 2-itemsets generated by the proposed algorithm is, in orders of magnitude, smaller than that by previous methods, thus resolving the performance bottleneck. Our approach scans the database once utilizing an enhanced version of Apriori algorithm. Note that the generation of smaller candidate sets enables us to effectively trim the transaction database size at a much earlier stage of the iterations, thereby reducing the computational cost for later iterations significantly.

Keywords: Hash based Apriori, Data structure, Candidate 2-itemsets

INTRODUCTION

Data mining is having a vital role in many of the applications like market-basket analysis, in biotechnology field etc. In data mining, frequent itemsets plays an important role which is used to identify the correlations among the fields of database. Association rules are used to identify relationships among a set of items in database. These relationships are not based on inherent properties of the data themselves (as with functional dependencies), but rather based on cooccurrence of the data items.

Association rule and frequent itemset mining became a widely researched area, and hence faster and faster algorithms have been presented. The Apriori algorithm is used for Association Rule Mining [1] given a set of transactions, where each transaction is a set of items, an association rule is an expression X + Y, where X and Y are sets of items. The intuitive meaning of such a rule is that transactions in the database which contain the items in X tend to also contain the items in Y. [2]

Association rule mining is to find out association rules that satisfy the predefined minimum support and confidence from a given database. The problem is usually decomposed into two subproblems. One is to find those itemsets whose occurrences exceed a predefined threshold in the database; those itemsets are called frequent or large itemsets. The second problem is to generate association rules from those large itemsets with the constraints of minimal confidence. Suppose one of the large itemsets is Lk, Lk = {I1, I2, … , Ik}, association rules with this itemsets are generated in the following way: the first rule is {I1, I2, … , Ik-1}{Ik}, by checking the confidence this rule can be determined as interesting or not. Then other rules are generated by deleting the last items in the antecedent and inserting it to the consequent, further the confidences of the new rules are checked to determine the interestingness of them.

Those processes iterated until the antecedent becomes empty. Since the second subproblem is quite straightforward, most of the researches focus on the first subproblem.[3] In this paper, we propose an algorithm, Apriori with hashing technology to store the database in vertical data format. Apriori is the best-known basic algorithm for mining frequent item sets in a set of transactions. We describe implementation of this algorithm that use Hashing to achieve maximum performance, w.r.t. both execution time and memory usage.

THE APRIORI ALGORITHM

One of the first algorithms to evolve for frequent itemset and Association rule mining was Apriori. Two major steps of the Apriori algorithm are the join and prune steps.

The join step is used to construct new candidate sets. A candidate itemset is basically an itemset that could either be
frequent or infrequent with respect to the support threshold. Higher level candidate itemsets ($C_k$) are generated by joining previous level frequent itemsets are $L_{k-1}$ with itself.

The prune step helps in filtering out candidate item-sets whose subsets (prior level) are not frequent. This is based on the anti-monotonic property as a result of which every subset of a frequent item set is also frequent. Thus a candidate item set which is composed of one or more infrequent item sets of a prior level is filtered (pruned) from the process of frequent itemset and association mining.[4]

**Apriori Algorithm**

**Input**

D, a database of transactions

Min_sup, the minimum threshold support

**Output**

$L_k$ Maximal frequent itemsets in D

$C_k$ Set of Candidate k-itemsets.

**Method:**

1. $L_1$ = Frequent items of length 1.
2. For ($k$=1; $L_k$=ф; $k$++) do.
3. $C_{k+1}$ = candidates generated from $L_k$.
4. For each transaction $t$ in database $D$ do.
5. Increment the count of all candidates in $C_{k+1}$ that are contained in $t$.
6. $L_{k+1}$ = candidates in $C_{k+1}$ with minimum support.
7. end do.
8. Return the set $L_k$ as the set of all possible frequent itemsets.

The main notation for association rule mining that is used in Apriori algorithm is the following.

1. A k-itemset is a set of k items.
2. The set $C_k$ is a set of candidate k-itemsets that are potentially frequent.
3. The set $L_k$ is a subset of $C_k$ and is the set of k-itemsets that are frequent.

**HASH BASED APRIORI ALGORITHM**

Our hash based Apriori implementation, uses a data structure that directly represents a hash table. This algorithm proposes overcoming some of the weaknesses of the Apriori algorithm by reducing the number of candidate k-itemsets. In particular the 2-itemsets, since that is the key to improving performance. This algorithm uses a hash based technique to reduce the number of candidate itemsets generated in the first pass. It is claimed that the number of itemsets in $C_2$ generated using hashing can be smalled, so that the scan required to determine $L_2$ is more efficient.

For example, when scanning each transaction in the database to generate the frequent 1-itemsets, $L_1$, from the candidate 1-itemsets in $C_1$, we can generate all of the 2-itemsets for each transaction, hash(i.e) map them into the different buckets of a hash table structure, and increase the corresponding bucket counts. A 2-itemset whose corresponding bucket count in the hash table is below the support threshold cannot be frequent and thus should be removed from the candidate set. Such a hash based apriori may substantially reduce the number of the candidate k-itemsets examined.

**Algorithm:**

1. Scan all the transaction. Create possible 2-itemsets.
2. Let the Hash table of size 8.
3. For each bucket assign a candidate pairs using the ASCII values of the itemsets.
4. Each bucket in the hash table has a count, which is increased by 1 each item an item set is hashed to that bucket.
5. If the bucket count is equal or above the minimum support count, the bit vector is set to 1. Otherwise it is set to 0.
6. The candidate pairs that hash to locations where the bit vector bit is not set are removed.
7. Modify the transaction database to include only these candidate pairs.

In this algorithm, each transaction counting all the 1-itemsets. At the same time all the possible 2-itemsets in the current transaction are hashed to a hash table. It uses a hash table to reduce the number if candidate itemsets. When the support count is established the algorithm determines the frequent itemsets. It generates the candidate itemsets as like the Apriori algorithm.

**METHODOLOGY**

We have implemented Apriori and Hash based Apriori algorithms in Visual Basic. To improve the performance of Apriori algorithm we are using the Hashing Data structure.

We report experimental results on supermarket dataset. We have taken supermarket database as a Text file. In this data set, the average maximal potentially frequent itemset size is set to 16, while the number of transactions in the dataset is set to 25.

Apriori and Hash based Apriori were executed for different minimum support level to generate the candidate 2-itemsets. The performance of Apriori and Hash based Apriori algorithms are evaluated for different minimum support levels.

**RESULTS AND DISCUSSION**

In this section, we present a performance comparison of our development with Apriori. The following table presents the test results of the implementations of Apriori and the Hash based Apriori on the dataset of supermarket for different minimum support level.

<table>
<thead>
<tr>
<th>Minimum support Level</th>
<th>Size of candidate 2-itemsets</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Apriori</td>
</tr>
<tr>
<td>1</td>
<td>136</td>
</tr>
<tr>
<td>2</td>
<td>78</td>
</tr>
<tr>
<td>3</td>
<td>45</td>
</tr>
</tbody>
</table>
As a result, when comparing with Apriori algorithm the size of candidate 2 Itemsets of Hash based Apriori algorithm is reduced. For minimum support level of 1, the size of candidate 2-Itemsets is 136 while using Apriori. But it is reduced to 51 while using Hash based Apriori.

For minimum support level of 7, the size of candidate 2-Itemsets is 21 while using Apriori. While using Apriori with Hashing it is 7. From that the size of c2 is less for Hash based Apriori than Apriori.

Figure I illustrate the results of comparing our implementation of Apriori with Hash based Apriori method. Minimum Support is taken as X-axis and the size of C2 is taken as Y-axis.

Figure I. Memory usage

In these graphs, we see that the memory usage of candidate 2-itemset for both algorithms increases exponentially as the minimum support is reduced. Applying Hashing data structure in apriori reduce the size of candidate 2-items when comparing with apriori. Hence the execution time is reduced. Thus the performance of Apriori with hashing is improved with respect to execution time and memory size.

CONCLUSION

Determining frequent objects is one of the most important fields in data mining. This algorithm can achieve a smaller memory usage than the Apriori algorithm. It is well known that the way candidates are defined has great effect on running time and memory need. We presented experimental results, showing that the proposed algorithm always outperform Apriori. Hash based Apriori is most efficient for generating the frequent itemset than Apriori.

REFERENCES

[1] Analyzing Association Rule Mining and Clustering on Sales Day Data with XLMiner and Weka A. M. Khattak, A. M. Khan, Sungyoung Lee*, and

Young-Koo Lee Department of Computer Engineering, Kyung Hee University, Korea 
{asad.masood, kadil, sylee}@oslab.ac.kr, yklee@khu.ac.kr

{srikant,ragrawal}@almaden.ibm.com

[3] Association Rules Mining: A Recent Overview Sotiris Kotsiantis, Dimitris Kanellopoulos Educational Software Development Laboratory Department of Mathematics, University of Patras, Grecce sotos@math.upatras.gr, dkanellop@teipat.gr
