IJDCST @December Issue- V-1, I-8, SW-2
ISSN-2320-7884 (Online)
ISSN-2321-0257 (Print)

Efficiently Identifying Frequent Item sets using Hash Based Apriori Algorithm

[1]L.Balaji,[2] P.V.Vasavi
[1]Associate Professor,[2]Final M Tech Student
Dept of Computer Science, Rama Chandra College of Engineering

Abstract: 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 rule and frequent item set mining became a widely researched area, and hence faster and faster algorithms have been presented. 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 priori 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

Index terms: Apriori , Candidate –Item set

1. 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 co occurrence 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.

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 sub problems. One is to find those item sets whose occurrences exceed a predefined threshold in the database; those item sets are called frequent or large item sets. The second problem is to generate association rules from those large item sets with the constraints of minimal confidence. Suppose one of the large item sets is Lk, Lk = {I1, I2, … , Ik}, association rules with this item sets 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 rule 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 sub problem is quite straight forward, most of the researches focus on the first sub problem.[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.

II. 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 item set is basically an item set that could either be frequent or infrequent with respect to the support threshold. Higher level candidate item sets (C_i) are generated by joining previous level frequent item sets are L_{i-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.

Apriori Algorithm

Input

D$\rightarrow$ a database of transactions
Min_sup$\rightarrow$ 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;$L_k$!=ϕ;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 asociation 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 candidte 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.

III. 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 C2 generated using hashing can be smalled,so that the scan required to determine L2 is more efficient

For example, when sacnning each transaction in the database to generate the frequent 1-itemsets,L1, from the candidate 1-itemsets in C1, 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 frequnt 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 an 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.

Result & 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 I MEMORY USAGE OF APRIORI AND HASH BASED APRIORI

Miniumm support Level	Size of candidate 2-itemsets	
	Apriori	Hash Based Apriori
1	136	51
2	78	28
3	45	22
4	45	18
5	28	11
6	28	7
7	21	7

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 C_2 is taken as Y-axis.

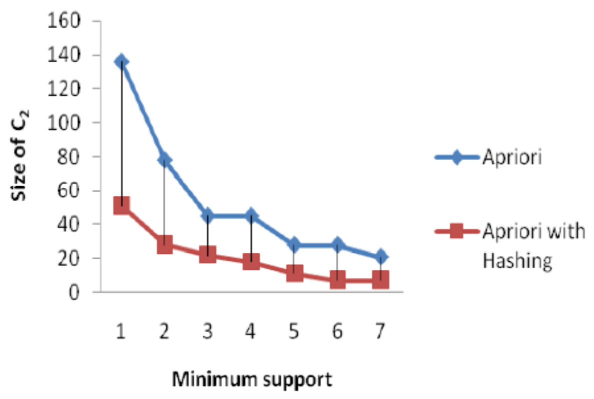

Figure I. Memory usage In these graphs, we see that the memory usage of candidate 2-itemset for both algorithms increases exponentially as the minimum support is reduced. Applying Hashing data structure in apriori reduce the size of candidate 2-itemsets when comparing with apriori. Hence the execution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

IV. 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*

2) Mining Generalized Association Rule Ramakrishnan Srikant* Rakesh AgrawalIBM Almaden Research Center San

Jose, CA 95120 {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, Greecesotos@math.upatras.gr, dkanellop@teipat.gr
4) Datamining Techniques and Trends,N.P.Gopalan,B.Sivalselvan PHI Learning private limited,New Delhi 2009
5) Jiawei Han and Micheline Kamber, “Data Mining Concepts and Techniques”, Second Edition, Morgan Kaufmann Publishers,2006.