Finding Locally and Periodically Frequent Sets and Periodic Association Rules

A. Kakoti Mahanta[1], F.A. Mazarbhuiya[1], and H.K. Baruah[2]

[1] Department of Computer Science, Gauhati University, Assam, India
anjanagu@yahoo.co.in, fokrul_2005@yahoo.com
[2] Department of Statistics, Gauhati University
hemanta_bh@yahoo.com

Abstract. The problem of finding association rules from a dataset is to find all possible associations that hold among the items, given a minimum support and confidence. This involves finding frequent sets first and then the association rules that hold within the items in the frequent sets. In temporal datasets as the time in which a transaction takes place is important we may find sets of items that are frequent in certain time intervals but not frequent throughout the dataset. These frequent sets may give rise to interesting rules but these can not be discovered if we calculate the supports of the item sets in the usual way. We call here these frequent sets locally frequent. Normally these locally frequent sets are periodic in nature. We propose modification to the Apriori algorithm to compute locally frequent sets and periodic frequent sets and periodic association rules.

1 Introduction

The problem of mining association rules has been defined initially by R. Agarwal et al [1] for application in large super markets. Large supermarkets have large collection of records of daily sales. Analyzing the buying patterns of the buyers will help in taking business decisions. Mining for association rules between items in temporal databases has been described as an important data-mining problem. The market basket transaction is an example of this type.

In this paper we consider datasets, which are temporal and the time in which a transaction has taken place is attached to the transactions. In large volumes of such data, some hidden information or relationship among the items may be there which do not necessarily exist or hold through out the period covered by the dataset. Such information may remain hidden because the support (as defined in [1]) of the association rules is less than the minimum support value provided by the user if the whole dataset is considered. It may be the case that these association rules hold for certain limited time periods and not through out the entire time period covered by the dataset. For finding such association rules we need to find itemsets that are frequent at certain time intervals but not frequent if the whole life span of the database is considered. We call such frequent sets locally frequent. From these locally frequent sets, associations among the items in these sets can be obtained. Since a periodic nature is there in any natural event this kind of association rules normally hold periodically. And if such locally frequent sets are periodic in nature then we call these sets periodic frequent sets and the associated association rules as periodic association rules.

In section 2 we give a brief discussion on the recent works in Temporal Data Mining. In section 3 we describe the terms and notations used in this paper. In section 4, we give the algorithm proposed in this paper for mining locally frequent sets and local association rules. In section 5 we describe a way of extracting periodic frequent sets and the related association rules. We conclude with conclusion and lines for future work in section 6.

2 Recent Works

Given a set I, of items and a large collection D of transactions involving the items, the problem is to find relationships among the items. A transaction t is said to support an item if that item is present in t. A transaction t is said to support an itemset if t supports each of the items present in the itemset. An association rule is an expression of the form $X \Rightarrow Y$ where X and Y are subsets of the item set I. The rule holds with confidence τ if τ% of the transaction in D that supports X also supports Y. The rule has support σ if σ% of the transactions supports $X \cup Y$. A method for the discovery of association rules was given in [1]. This was then followed by subsequent refinements, generalizations, extensions and improvements.

The association rule discovery process is also extended to incorporate temporal aspects. In temporal association rules each rule has associated with it a time interval in which the rule holds. The problems associated are to find valid time periods during which association rules hold, the discovery of possible periodicities that association rules have and the discovery of association rules with temporal features. In [2], [3], [4] and [5], the problem of temporal data mining is addressed and techniques and algorithms have been developed for this. In [2] the problem of mining interesting associations with a high confidence level but with little support is addressed. This problem is caused due to the way in which supports of item sets are calculated. At the denominator we have the total number of transactions in the database. For this reason the support of itemsets that appear in the transactions for a short period, may not satisfy the minimum threshold criterion for becoming frequent. To handle this problem, in this paper for each itemset a lifetime is defined which is the time gap between the first occurrence and the last occurrence of the item in the transactions in the database. Supports of items are calculated only during its life span. Thus each rule has associated with it a time frame corresponding to the lifetime of the items participating in the rule. An algorithm, which is an extension of the A priori algorithm, is then given in order to extract the temporal association rules.

In [7], two algorithms are proposed for the discovery of temporal rules that display regular cyclic variations where the time interval is specified by user to divide the data into disjoint segments like months, weeks, days etc. Similar works were done in [6] and [8] incorporating multiple granularities of time intervals (e.g. first working day of every month) from which both cyclic and user defined calendar patterns can be achieved.

Our approach is different from the above approaches. We are considering the fact that some items are seasonal or appear frequently in the transactions for certain short time intervals only. They appear in the transactions for a short time and then disappear for a long time. After this they may again reappear for a certain period and this process may repeat. For these item sets if their life span is considered as defined in [2] in calculating their support values, they may not turn out to be frequent. These items may lead to interesting association rules with large confidence values. In this paper we calculate the support values of these sets locally in a particular season or in the time interval in which it is appearing frequently and if they are frequent in the time interval under consideration then we call these sets locally frequent sets. The large time gap in which they do not appear is not counted. This new method will be able to extract all frequent sets discovered by the algorithm described in [2] and some more sets that are frequent according to the way in which we are defining locally frequent sets. We also define periodic frequent sets and periodic association rules. As mentioned in the previous paragraph similarly works were also done in [7], [6] and [8]. But in all these methods the user should have prior information about the periods, which are required to be given as input. Also for different item sets the periods or cycles may differ. Our algorithm finds the periods for itemsets in a natural way without having any prior information about these.

3 Terms, Notations and Symbols Used

Let T = $<t_o, t_1, \ldots\ldots\ldots\ldots\ldots>$ be a sequence of time stamps over which a linear ordering $<$ is defined where $t_i < t_j$ means t_i denotes a time which is earlier than t_j. Let I denote a finite set of items and the transaction database D is a collection of transactions where each transaction has a part which is a subset of the itemset I and the other part is a time-stamp indicating the time in which the transaction had taken place. We assume that D is ordered in the ascending order of the time stamps. For time intervals we always consider closed intervals of the form $[t_1, t_2]$ where t_1 and t_2 are time stamps. We say that a transaction is in the time interval $[t_1, t_2]$ if the time stamp of the transaction say t is such that $t_1 \leq t \leq t_2$. We define the local support of an itemset in a time interval $[t_1, t_2]$ as the ratio of the number of transactions in the time interval $[t_1, t_2]$ containing the itemset to the total number of transactions in $[t_1, t_2]$ for the whole database D. We use the notation $Sup_{[t_1,t_2]}$(X) to denote the support of the itemset X in the time interval $[t_1, t_2]$. Given a threshold σ we say that an itemset X is frequent in $[t_1,t_2]$ if $Sup_{[t_1,t_2]}$(X) $\geq (\sigma/100)*$ tc where tc denotes the total number of transactions in D that are in the time interval $[t_1,t_2]$. We say that an association rule $X \Rightarrow Y$, where X and Y are itemsets holds in the time interval $[t_1, t_2]$ if and only if given threshold τ,

$$Sup_{[t_1,t_2]}(X \cup Y) / Sup_{[t_1,t_2]}(X) \geq \tau / 100.0$$

and $X \cup Y$ is frequent in $[t_1, t_2]$. In this case we say that the confidence of the rule is τ.

For each locally frequent itemset we keep a list of time intervals in which the set is frequent where each interval is represented as [*start, end*] where *start* and *end* gives the starting and ending time of the time-interval. *end* – *start* gives the length of the time interval. Given two intervals [$start_1$,end_1] and [$start_2$, end_2] if the intervals are non-overlapping and $start_2 > end_1$ then $start_2 - end_1$ gives the distance between the time intervals.

4 Algorithm Proposed

4.1 Generating Locally Frequent Sets

While constructing locally frequent sets, with each locally frequent set a list of time-intervals is maintained in which the set is frequent. Two thresholds minthd1 and minthd2 are used and these are given as input. During execution, while making a pass through the database, if for a particular itemset the time gap between its current time-stamp and the time when it was last seen is less than the value of minthd1 then the current transaction is included in the current time-interval under consideration; otherwise a new time-interval is started with the current time-stamp as the starting point. The support count of the item set in the previous time interval is checked to see whether it is frequent in that interval or not and if it is then it is added to the list maintained for that set. Also for the locally frequent sets a minimum period length is given by the user as minthd2 and time intervals of length greater than or equal to this value are only kept. If minthd2 is not used than an item appearing once in the whole database will also become locally frequent.

Procedure to compute L_1, the set of all locally frequent item sets of size 1

For each item while going through the database we always keep a time-stamp called *lastseen* that corresponds to the time when the item was last seen. When an item is found in a transaction and the time-stamp is *tm* and the time gap between *lastseen* and *tm* is greater than the current minimum threshold given, then a new time interval is started by setting *start* of the new time interval as *tm* and *end* of the previous time interval as *lastseen*. The previous time interval is added to the list maintained for that item provided that the duration of the interval and the support of the itemset in that interval are both greater than the minimum thresholds specified for each. Otherwise *lastseen* is set to *tm*, the counters maintained for counting transactions are increased appropriately and the process is continued. Following is the algorithm to compute L_1, the list of locally frequent sets of size 1. Suppose the number of items in the dataset under consideration is n and we assume an ordering among the items.

Algorithm 4.1

```
  $C_1 = \{(i_k, tp[k]) : k = 1,2,.....,n\}$
    where $i_k$ is the k-th item and tp[k] points to a list of time intervals initially empty.
  for k = 1 to n do
      set lastseen[k], itemcount[k]and transcount[k]  to zero
  for each transaction t in the database with time stamp tm do
   {for k = 1 to n do
      { if $\{i_k\} \subseteq t$ then
```

```
                { if(lastseen[k] == 0)
                   {lastseen[k] = firstseen[k] = tm;
                    itemcount[k] = transcount[k] = 1;
                   }
                 else  if (tm – lastseen[k] < minthd1)
                       {lastseen[k]=tm; itemcount[k]++;
                        transcount[k]++;
                       }
                   else
                     {if ((|lastseen[k]–firstseen[k]|≥ minthd2)
                        &&(itemcount[k]/transcount[k]*100 ≥ σ))
                           add(firstseen[k], lastseen[k]) to tp[k];
                      itemcount[k] = transcount[k] = 1;
                      lastseen[k] = firstseen[k] = tm;
                    }
                }
            else transcount[k]++;
        }   // end of k-loop //
      } // end of do loop //
   for k = 1 to n do
    {if (|lastseen[k]–firstseen[k]|≥ mintdh2)and (itemcount[k] / transcount[k] * 100 ≥ σ)
                add  (firstseen[k], lastseen[k] ) to tp[k];
       if(tp[k] != 0) add {$i_k$, tp[k]} to $L_1$
    }
```

Two support counts are kept, *itemcount* and *transcount*. If the count percentage of an item in a time interval is greater than the minimum threshold then only the set is considered as a locally frequent set.

After this Apriori candidate generation algorithm is used to find candidate frequent set of size 2. With each candidate frequent set of size two we associate a list of time intervals that are obtained in the pruning phase. In the generation phase this list is empty. If all subsets of a candidate set are found in the previous level then this set is constructed. The process is that when the first subset appearing in the previous level is found then that list is taken as the list of time intervals associated with the set. When subsequent subsets are found then the list is reconstructed by taking all possible pair wise intersection of subsets one from each list. Sets for which this list is empty are further pruned. Using this concept we describe below the modified A priori algorithm for the problem under consideration.

Algorithm 4.2

Modified A priori

```
 Initialize
 k = 1;
 $C_1$ = all item sets of size 1
 $L_1$ = {frequent item sets of size 1 where with each itemset {$i_k$} a list tp[k] is maintained  which gives all time  intervals in which the set is frequent}
```

```
        $L_1$ is computed using algorithm 1.1 */
  for(k = 2; $L_{k-1} \neq \phi$ ; k++) do
      { $C_k$ = apriorigen($L_{k-1}$)
      /* same as the candidate generation method of the A priori algorithm setting tp[i] to
        zero for all i*/
      prune($C_k$);
      drop all lists of time intervals maintained with the sets in $C_k$
      Compute $L_k$ from $C_k$.
  //$L_k$ can be computed from $C_k$ using the same procedure used for computing $L_1$ //
      k = k + 1
    }
  Answer = $\bigcup_k L_k$

  Prune($C_k$)
  {Let m be the number of sets in $C_k$ and let the sets be $s_1, s_2, \ldots, s_m$. Initialize the pointers tp[i] pointing to the list of time-intervals maintained with each set $s_i$ to null
  for i = 1 to m do
     {for each (k-1) subset d of $s_i$ do
        {if d $\notin L_{k-1}$ then {$C_k = C_k$ - {$s_i$, tp[i]}; break;}
        else
         {if (tp[i]==null) then set tp[i] to point to the list of intervals maintained for d
          else
             {take all possible pair-wise intersection of time intervals one from each
              list, one list maintained with tp[i] and the other maintained with d and
              take this as the list for tp[i]
             delete all time intervals whose size is less than the value of minthd2
             if tp[i] is empty then {$C_k = C_k$ - {$s_i$,tp[i]}; break; }
            }
         }
       }
     }
   }
```

4.2 Generating Association Rules

If an itemset is frequent in a time-interval [t_1, t_2] then all its subsets are also frequent in the time-interval [t_1, t_2]. But to generate the association rules as defined in section 3, we need the supports of the subsets in [t_1, t_2] which may not be available after application of the algorithm as defined in 4.1. For this one more scan of the whole database will be needed. For each association rule we attach an interval in which the association rule holds.

5 Extracting Periodic Patterns

In the algorithm proposed in section 4 for each locally frequent set, a list of time intervals is maintained in which the set is frequent. In this list if we find that the distance between the intervals is almost equal (up to a small variation) then we call these frequent sets periodic frequent sets. Association rules that hold periodically are called periodic association rules.

Now if the time-stamps of the transactions are the calendar dates more interesting patterns may be seen. In a market-basket database we normally will see that the sales of cold drinks go up in summer. So in each year in summer cold drink will become frequent. But this periodicity is of a special type. If we consider only the month and day associated with the transactions, then the periods in which an item set is frequent will have large overlapping.

6 Conclusion and Lines for Future Work

An algorithm for finding frequent sets that are frequent in certain time periods called locally frequent sets in the paper is given. The technique used is similar to the A priori algorithm. From these locally frequent sets interesting rules may follow. Some of these locally frequent sets may be periodic in nature. We call these sets periodic frequent sets and the associated rules as periodic association rule and suggest methods of extracting these.

For further improvement, instead of maintaining the time intervals as lists, other suitable data structures such as a balanced binary search tree could be used. Further, we are in the process of implementation the algorithm and are planning to test the algorithm using some standard data generators and real life data sets available in http://www.ics.uci.edu/ and http://www.almaden.ibm.com/.

References

1. Agrawal,R.,Imielinski,T.and Swami, A.; Mining association rules between sets of items in large databases; Proceedings of ACM SIGMOD ’93, Washington(1993).
2. Ale, Juan M and Rossi, G.H.; An approach to discovering temporal association rules; Proceedings of 2000 ACM symposium on Applied Computing (2000).
3. Chen, X. and Petrounias, I.; A framework for Temporal Data Mining; DEXA’98, Austria. Springer-Verlag, Lecture Notes in Computer Science 1460 (1998) 796-805
4. Chen, X. and Petrounias, I.; Language support for Temporal Data Mining; Proceedings of , PKDD ’98, Springer Verlag, Berlin (1998) 282-290
5. Chen, X., Petrounias, I. and Healthfield, H.; Discovering temporal Association rules in temporal databases; Proceedings of IADT’98, 312-319
6. Li, Y., Ning, P., Wang, X. S. and Jajodia, S.; Discovering Calendar-based Temporal Association Rules, In Proc. of the 8th Int’l Symposium on Temporal Representation and Reasonong (2001)
7. Ozden, B., Ramaswamy, S. and Silberschatz, A.; Cyclic Association Rules, Proc. of the 14th Int’l Conference on Data Engineering, USA (1998) 412-421.
8. Zimbrao, G., Moreira de Souza, J., Teixeira de Almeida V. and Araujo da Silva, W.; An Algorithm to Discover Calendar-based Temporal Association Rules with Item’s Lifespan Restriction, Proc. of the 8th ACM SIGKDD 2002.