Journal of Computer Science 2 (1): 76-81, 2006
ISSN 1549-3636
©) 2006 Science Publications

# Mining Level-Crossing Association Rules from Large Databases

[1]R.S. Thakur, [2]R.C. Jain and [3]K.R. Pardasani
[1]Department of Master in computer Application, Govt. Geetanjali Girls' College, Bhopal (M.P.) India
[2]Department of Computer Application, SATI, Vidisha (M.P.) India
[3]Department of Mathematics, Maulana Azad National Institute of Technology,
Bhopal (M.P.) India

**Abstract:** Existing algorithms for mining association rule at multiple concept level, restricted mining strong association among the concept at same level of a hierarchy. However mining level-crossing association rule at multiple concept level may lead to the discovery of mining strong association among at different level of hierarchy. In this study, a top-down progressive deepening method is developed for mining level-crossing association rules in large transaction databases by extension of some existing multiple-level association rule mining techniques. This method is using concept of reduced support and refine the transaction table at each level.

**Key words:** Mining algorithms, mining association rules, level-crossing association rules

## INTRODUCTION

Studies on mining association rules have evolved from techniques for discovery of functional dependencies[1], strong rules[2], classification rules[3,4], causal rules[5], clustering[6], etc. to disk-based, efficient methods for mining association rules in large sets of transaction data[7-10]. However, previous work has been focused on mining association rules at a single concept level as well as multiple-level. There are applications, which need to find "level-crossing" associations at multiple concept levels. For example, besides finding 80% of customers that purchase milk may also purchase bread, it could be informative to also show that 75% of people buy wheat bread if they buy 2% milk or 70% of people buy milk if they buy wheat bread. The association relationship in the latter statement is expressed at a lower level but often carries more specific and concrete information than that in the former. This requires progressively deepening the knowledge mining process for finding refined knowledge from data. The necessity for mining multiple level (level-crossing) association rules or using taxonomy information at mining association rules has also been observed by other researchers[8,11].

To confine the association rules discovered to be strong ones, that is, the patterns which occur relatively frequently and the rules which demonstrate relatively strong implication relationships, the concepts of *minimum support* and *minimum confidence* have been introduced[7,8]. For mining level-crossing association rules at multiple concept level, concept taxonomy should be provided for generalizing primitive level concepts to high level ones.

In many applications, the taxonomy information is either stored implicitly in the database, such as "Wonder wheat bread is a wheat bread which is in turn a bread', or computed elsewhere[3]. Thus, data items can be easily generalized to multiple concept levels.

In this study, a top-down progressive deepening method is developed by extension of some existing algorithms for mining single and multilevel association rules. The method first finds large data items at the top-most level and then progressively deepens the mining process into their large descendants at lower concept levels. At each lower level, find level-crossing association rule among frequent item at same level and frequent itemsets of all upper levels. Due to pruning uninteresting data items at each level generation of candidate sets is getting minimum at each lower concept levels.

## MULTIPLE LEVEL ASSOCIATION RULES

We assume that the database contain 1) an item data set which contain the description of each item in $I$ in the form of ( $A_i$, *description*$_i$), where $A_i \in I$ and 2) a transaction data set, $\mathcal{T}$, which consists of a set of transaction ($T_i$ {$A_p$….., $A_q$}), where $T_i$ is a transaction identifier and $A_i \in I$ (for i = p……q).

To find relatively frequent occurring patterns and reasonably strong rule implications, a user or an expert may specify two thresholds: *minimum support*, σ' and *minimum confidence* , φ'. Notice that for finding level-crossing association rules, different

**Corresponding Author:** R.S. Thakur, Department of Master in computer Application, Govt. Geetanjali Girls' College, Bhopal (M.P.) India

minimum support and/or minimum confidence can be specified at different levels.

**Definition 1:** A pattern *A* is large in set *S* at level *l* if the support of *A* is no less then its corresponding minimum support threshold $\sigma'_l$. A rule "*A → B/S*" is strong if , for a set *S*, each ancestor (i.e. the corresponding high-level item) of every item in *A* and *B*, if any , is frequent at its corresponding level "*A ∧ B/S* " is frequent (at the current level ) and the confidence of "*A → B/S* " is no less then minimum confidence threshold at the current level.

The definition implies a filtering process which confines the pattern to be examined at lower level to be only those with large support at their corresponding high level. Based on this definition, the idea of mining level-crossing association rules is illustrated below.

Table 1: A sales transaction table

| transaction_id | bar_code_set |
|---|---|
| 351428 | {17325, 92108, 55349, ….} |
| 982510 | {92458, 77451, 60395, … } |
|  | { …, …} |

**Example 1:** Let the query be to find level-crossing association at concept of multiple-level in the database in Table 1 for the purchase patterns related to *category*, *content* and *brand* of the food which can only be stored for less than three weeks.

Table 2: A sales_item (description) relation

| Bar_code | category | brand | content | size | storage_pd | price |
|---|---|---|---|---|---|---|
| 17325 | milk | foremost | 2% | 1(ga.) | 14(day) | $3.89 |
| ….. | ….. | …… | ….. | …… | ……. |  |
|  | ….. |  |  |  |  |  |

Table 3: A generalized sales_item description table

| GID | bar_code_set | category | content | brand |
|---|---|---|---|---|
| 112 | {17325, 31414, 91265 } | Milk | 2% | foremost |
| ….. | {……………} | …… | ……. | ……… |

The relevant part of the sales item description relation in Table 2 is fetched and generalized into a generalized sales_item description table, as shown in Table 3, in which is tuple represent a generalized item which is the merge of a group of tuples which share the same values in the interested attributes. For example, the tuple with the same *category*, *content* and *brand* in Table 1 are merged into one, with their *bar codes* replaced by a bar-code set. Each group is then treated as an atomic item in the generation of the lowest level association rules. For example, the association rule generated regarding to milk will be only in relevance to (at the low concept levels) *brand* (such as Dairyland) and *content* (such as 2%) but not to *size*, *producer*, etc.

The taxonomy information is provided implicitly in Table 3. Let *category* (such as "milk") represent the first-level concept, *content* (such as "2%") for the second level one and *brand* (such as "Foremost") for the third level one. The table implies a concept tree like Fig. 1. The process of mining level-crossing association rules is actually will be starting from level second, but first discover large patterns at the top-most concept level similar to Hen and Fu[11]. Let the minimum support at this level be 5% and the minimum confidence be 50%. One may find the large 1-itemset: "bread (25%), meat(10%), milk (20%), vegetable(30%)".

At the second level, only the transactions which contain the large items at the first level are examined. Let the minimum support at this level be 2% and the minimum confidence be 40%. One may find the frequent 1-itemsets: "lettuce (10%), wheat bread(15%), white bread(10%), 2% milk(10%), …." , then level-crossing large 2-itemsets will be : "〈 milk, wheat bread (6%)〉,〈 bread, 2% milk(4%)〉,….." and strong level-crossing association rule: "milk →wheat bread(60%), bread → 2% milk, ", etc.

The process repeats at even lower concept level until no large patterns can be found.

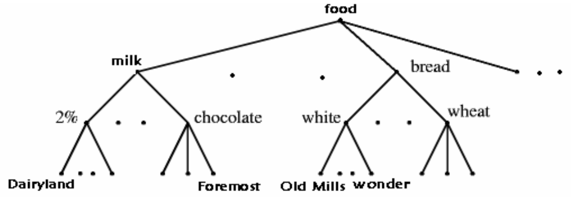

Fig. 1: A taxonomy for the relevant data items

## METHOD FOR MINING LEVEL-CROSSING ASSOCIATION RULES

A method for mining "level-crossing" association rules is introduced in this section, which uses a hierarchy information encoded transaction table[11]. This is based on the following consideration. First, a data mining query is usually in relevance to only a portion of the transaction database, such as *food* instead of all the items. It is beneficial to first collect the relevant set of data and then work repeatedly on the task-relevant set. Second, encoding can be performed during the collection of task-relevant data and thus there is no extra "encoding pass" required. Third, an encoding string, which represents a position in a hierarchy, required less bits than the corresponding object-identifier or bar-code.

To simply our discussion, an abstract example, which simulates the real life example of Example 1, is analyzed as follows:

**Example 2:** As stated above, the taxonomy information for each (grouped) item in Example 1 is encoded as a sequence of digits in the transaction table $\mathcal{T}$[1] (Table 4). For example, the item '2% Foremost milk' is encoded as '112' in which the digit, '1', represents 'milk' at level-1, the second, '1', for '2%(milk)' at level-2 and the third, '2', for the brand 'Foremost' at level-3. Similar to Agrawal and Srikant[8], repeated items (i.e., items with the same encoding) at any level will be treated as one item in one transaction.

Table 4: Encoded transaction table: $\mathcal{T}$[1]

| TID | Items |
|---|---|
| $T_1$ | {111, 121, 211, 221} |
| $T_2$ | {111, 211, 222, 323} |
| $T_3$ | {112, 122, 221, 411} |
| $T_4$ | {111, 121} |
| $T_5$ | {111,122,211,221, 413} |
| $T_6$ | {113, 323, 524} |
| $T_7$ | {131, 231} |
| $T_8$ | {323, 411, 524, 713} |

The derivation of the large itemsets at level 1 proceeds as follows. Let the minimum support be 4 transactions (i.e., *minsup*[1] = 4). (Notice since the total number of transactions is fixed, the support is expressed in an absolute value rather then a relative percentage for simplicity). The level-1 derivation of large itemset as done[11] i.e., large 1-itemset table $\mathcal{L}$[1,1] can be derived by scanning $\mathcal{T}$[1] and $\mathcal{L}$[1,1] is then used to filter out (1) any item which is not large in a transaction and (2) the transactions in $\mathcal{T}$[1] which contain only small items. This results in a filtered transaction table $\mathcal{T}$[2] of Fig. 2. Now large 2-itemset table $\mathcal{L}$[1,2] can be derived by scanning $\mathcal{T}$[2].

Level-1 minsup=4

Level-1 large 1-itemsets: $\mathcal{L}$ [1,1]

| Itemset | Support |
|---|---|
| {1**} | 7 |
| {2**} | 5 |

Filtered transaction table: $\mathcal{T}$[2]

| TID | Items |
|---|---|
| $T_1$ | {111,121,211,221} |
| $T_2$ | {111,211,222} |
| $T_3$ | {112,122,221} |
| $T_4$ | {111,121} |
| $T_5$ | {111,122,211,221} |
| $T_6$ | {113} |
| $T_7$ | {131,231} |

Level-1 Large 2-Itemsets: $\mathcal{L}$[1,2]

| Itemset | Support |
|---|---|
| {1**,2**} | 4 |

Fig 2: Large itemsets at level 1 and filtered transaction table :$\mathcal{T}$[2]

According to the definition of ML-association rules, only the descendants of the large item at level-1 (i.e., in $\mathcal{L}$ [1,1]) are considered as candidate in the level-2 large 1-itemsets. Let *minsup*[2] = 3.

The derivation of level-2 large item sets generates the same large 1-itemsets $\mathcal{L}$ [2,1] (can be derived from the filtered transaction table $\mathcal{T}$[2] by accumulating the support count and removing those whose is smaller then the minimum support, which results in $\mathcal{L}$ [2,1]. $\mathcal{L}$ [2,1] is then used filter out any item which is not large in a transaction and the transaction in $\mathcal{T}$[2] which contain only small items. This results in a filtered transaction table $\mathcal{T}$[3] i.e. pruning of infrequent items at each level ) as shown in Fig. 3. However, the candidate items are not confined to pairing only those in $\mathcal{L}$ [2,1] because the item in $\mathcal{L}$ [2,1] can be paired with those in $\mathcal{L}$ [1,1] as well, such as {11*, 1**} (for potential association like "milk → 2% milk"), or {11*, 2**}(for potential association like "2% milk → bread").These candidate large 2-itemsets will be checked against $\mathcal{T}$[3] to find large items (for the level-mixed nodes, the minimum support at lower level, i.e., minsup[2], can be used as a default). Such a process generate the large 2-itemsets table $\mathcal{L}$ [2,2] as shown in Fig. 3.

Level-2 minsup=3

Level-2 large 1-itemsets: $\mathcal{L}$ [2,1]

| Itemset | Support |
|---|---|
| {11*} | 6 |
| {12*} | 4 |
| {21*} | 3 |
| {22*} | 4 |

Filtered transaction table: $\mathcal{T}$[3]

| TID | Items |
|---|---|
| $T_1$ | {111,121,211,221} |
| $T_2$ | {111,211,222} |
| $T_3$ | {112,122,221} |
| $T_4$ | {111,121} |
| $T_5$ | {111,122,211,221} |
| $T_6$ | {113} |

Level-2 large 2-itemsets: $\mathcal{L}$ [2,2]

| Itemset | Support |
|---|---|
| {11*, 12*} | 4 |
| {11*, 21*} | 3 |
| {11*, 22*} | 4 |
| {12*, 22*} | 3 |
| {21*, 22*} | 3 |
| {11*, 2**} | 4 |
| {12*, 2**} | 3 |
| {21*, 1**} | 3 |
| {22*, 1**} | 4 |

Level-2 large 3-itemsets: $\mathcal{L}$ [2,3]

| Itemset | Support |
|---|---|
| {11*, 12,*, 22*} | 4 |
| {21*, 22,*, 1**} | 3 |

Fig 3: Large itemsets at level 2 and filtered transaction table :$\mathcal{T}$[3]

Notice that the table does not include the 2-item pair formed by an item with its own ancestor such as 〈{11*, 1**}, 5 〉 since its support must be the same as its corresponding large 1-itemset in $\mathcal{L}$ [2,1], i.e., 〈{11*}, 5〉, based on the set containment relationship: any transaction that contains {11*} must contain {1**} as well.

Similarly, the level-2 large 3-itemsets $\mathcal{L}$ [2,3] can be computed, with the results shown in Fig. 3 also, the entries which pair with there own ancestors are not listed here since it is contained implicitly in their corresponding 2-itemsets. For example, 〈{11*, 12*},4〉 in $\mathcal{L}$ [2,2] implies〈{11*,12*,1**},4〉 in $\mathcal{L}$[2,3].

Level-3 minsup=3

Level-3 large 1-itemsets: $\mathcal{L}$ [3,1]

| Itemset | Support |
|---|---|
| {111} | 4 |
| {211} | 4 |
| {221} | 3 |

Filtered transaction table: $\mathcal{T}$[4]

| TID | Items |
|---|---|
| $T_1$ | {111,211,221} |
| $T_2$ | {111,211} |
| $T_3$ | {221} |
| $T_4$ | {111} |
| $T_5$ | {111,211,221} |

Level-3 large 2-itemsets: $\mathcal{L}$ [3,2]

| Itemset | Support |
|---|---|
| {111, 211} | 3 |
| {111, 21*} | 3 |
| {111, 2**} | 3 |
| {11*, 211} | 3 |
| {1**, 211} | 3 |

Fig 4: Large itemsets at level 3 and filtered transaction table :$\mathcal{T}$[4]

Finally, the large 1-itemsets table at level–3, $\mathcal{L}$[3,1], should be the same as Fig. 3 (can be derived from the filtered transaction table $\mathcal{T}$[3] and generate transaction table $\mathcal{T}$[4] by filtering table $\mathcal{T}$[3] ). The large 2-itemset table includes more itemsets since these items can be paired with higher level large items, which leads to the large 2-itemsets $\mathcal{L}$ [3,2] and large 3-itemsets $\mathcal{L}$ [3,3] as shown in Fig. 4. similarly, the itemsets {111, 11*} and {111, 1**} have the same support as {111} in $\mathcal{L}$ [3,1] and are thus not include in $\mathcal{L}$ [3,2].

Since the large $k$-itemset ($k$ > 1) tables do not explicitly include the pair of items with their own ancestors, attention should be paid to include them at the generation of association rules. However, since the existence of a special item always indicates the existence of an item in that class, such as "2% milk → milk (100%)", such trivial rules should be eliminated. Thus, only nontrivial implications, such as "milk → 2% milk (70%)", will be considered in the rule generation.

The above discussion leads to the following algorithm for mining strong level-crossing association rules.

**Algorithm 1:** *Find large item sets for mining strong level-crossing association rules in a transaction database.*

**Input:** (1) $\mathcal{T}$ [l], a hierarchy-information-encoded and task-relevant set of transaction database, in the format of 〈TID, *Itemset*〉, in which each item in the *Itemset* contains encoded concept hierarchy information and (2) the minimum support threshold (*minsup*[*l*] ) for each concept level *l*.

**Output:** level-crossing large item sets.

**Method:** A top-down, progressively deepening process which collects large item sets with level-crossing at different concept levels as follows:

Starting at level 1, derive for each level *l*, the large *k*-items sets, $\mathcal{L}$ [*l* ,*k*] , for each *k* and the large item set, $\mathcal{L}\mathcal{L}$ [*l*] ( for all *k*'s ), as follows:

```
$l$ := 1; $Temp$:= 0; $\mathcal{L}[l, Temp]$ := 0;
for ($l$ := 1; $\mathcal{L}[l,1] \neq 0$ and $l < max\_level$;
$l$++) do
  { $\mathcal{L}[l,1]$ := large_1_itemsets($\mathcal{T}[l]$, $l$ );
    $\mathcal{L}[l, Temp] := \mathcal{L}[l, Temp] \bigcup \mathcal{L}[l,1]$;

   $\mathcal{T}[l+1]$ := filtered_t_table($\mathcal{T}[l]$, $\mathcal{L}[l,1]$);

   for ($k$ := 2; $\mathcal{L}[l, k-1] \neq 0$; $k$++) do
   { if $l = 1$ then
        {$C_k$ := get_candidate_set ($\mathcal{L}[l,k-1]$ );}
     else { if $k = 2$ then
            {$C_k$ := get_crosslevel_candidate_set
                ($\mathcal{L}[l,Temp]$ );}
else{$C_k$:=get_crosslevel_candidate_set
                ($\mathcal{L}[l,k-1]$ );}
        }
  foreach transaction $t \in \mathcal{T}[l+1]$ do
            { $C_t$ := get_Subsets($C_k$, t);
              foreach candidate c ∈ $C_t$ do
              c.support++;
        }
  $\mathcal{L}[l,k]$:= {c∈ $C_k$ |c.support ≥ minsup[$l$]}
  }
        $\mathcal{L}\mathcal{L}[l] := \bigcup_K \mathcal{L}[l,k]$ ;
     }
```

**Procedure** ***filtered_t_table***($\mathcal{T}$ [$l$]: transaction table at level $l$)

```
{
  foreach transaction  $t \in \mathcal{T}[l]$ do
    { for all item set $i \in t$
        if $(i \in t) \wedge (i \notin \mathcal{L}[l, 1])$
              Delete  i  from t ;
          Add  t  to $\mathcal{T}[l+1]$ ;
      }
  }
```

**Procedure** ***get_crosslevel_candidate_set***($\mathcal{L}$ [$l$ ,$k$-1] : frequent ($k$-1)-itemsets at level $l$ )

```
{   foreach itemset $l_1 \in \mathcal{L}[l, k-1]$
  foreach itemset $l_2 \in \mathcal{L}[l, k-1]$
     if $(l_1[1] = l_2[1]) \wedge (l_1[2] = l_2[2]) \wedge \ldots$
           $\wedge (l_1[k-2] = l_2[k-2])$
             $\wedge (l_1[k-1] < l_2[k-1])$  then
          c = $l_1$ join $l_2$;
  if  has_ancestor_itemset_pair( c )  then
         delete c ;
  if  has_infrequent_subset( c, $\mathcal{L}[l, k-1]$ )  then
         delete c ;
  else  add c to $C_k$ ;
}
```

**Procedure** ***has_ancestor_itemset_pair***(c: candidate set of cross level )

```
{   foreach itemset  $i \in c$
          foreach itemset  $j \in c$
          if  i  is ancestor of  j  then
                 return True;
          return False;
}
```

**Explanation of algorithm 1:** According to Algorithm 1, the discovery of large support items at each level $l$ proceeds as follows. At level-1, the large itemsets derived as done in [11] i.e., 1-itemsets $\mathcal{L}$ [$l$,1] is derived from $\mathcal{T}$[1] by "*large*_1_*itemsets* ($\mathcal{T}$[1] , $l$ )", at any other level $l$, $\mathcal{L}$ [$l$,1] is derived from $\mathcal{T}$ [$l$] by " *large*_1_*itemsets*($\mathcal{T}$[1] , $l$ )", after scanning the transaction table, filter out those items whose support is smaller then *minsup*[$l$]. The filtered transaction table $\mathcal{T}$[2] is derived by "*filtered_t_table* ($\mathcal{T}$[1], $\mathcal{L}$ [1,1] )", which uses $\mathcal{L}$ [1,1] as a filter to filter out any item which is not large at level-1 and the transactions which contain no large items.

For $k > 1$ itemset table at level-1 is derived as done in the apriori candidate generation algorithm [8], i.e., first compute the candidate set from $\mathcal{L}$ [$l$, $k$-1] then count support of each item of candidate set in $\mathcal{T}$ [$l$ + 1] and collect only those itemsets into $\mathcal{L}$ [$l$, $k$] which has support count no less then *minsup*[$l$].

At each level $l > 1$ for $k = 2$ compute the candidate set from $\mathcal{L}$ [$l$, *Tamp*] (is a union of large 1-itemset of all previous levels) by procedure *get_crosslevel_candidate_set*($\mathcal{L}$ [$l$, *Tamp*] ) but for $k > 2$ , procedure *get_crosslevel_candidate_set* ($\mathcal{L}$ [$l$, $k$-1] ) is used. The procedure *has_ancestor_itemset_pair*(c) is used for removing those candidate set which has a item is ancestor of other items in c and procedure h*as_infrequent_subset*( c, $\mathcal{L}$ [$l$ ,$k$-1] ) work done as in the apriori candidate generation algorithm [8], i.e. remove those candidate set which has infrequent subset.

The large itemsets at level $l$, $\mathcal{L}\,\mathcal{L}$ [$l$], is the union of $\mathcal{L}$ [$l$, $k$] for all the $k$'s. After finding the large itemsets, the set of association rules for each level $l$ can be derived from the large itemsets $\mathcal{L}\,\mathcal{L}$ [$l$] based on the minimum confidence at this level, *minconf*[$l$].

## CONCLUSION

We have extended the scope of the study of mining association rules among from concept at the same level of a hierarchy to concept of different level of hierarchy in multiple concept level and studied new method for mining level-crossing association rules from large transaction databases. A top-down progressive deepening technique is design for mining level-crossing association rules, which extends the existing single and multilevel association rule mining algorithms and explore techniques for sharing data structure and intermediate results across level. Deriving a new filtered transaction tables at each processing level, this method will do less processing work and generate minimum candidate sets.

## REFERENCES

1. Mannila, H. and K.J. Raiha, 1987. Dependency inference. In Proc. Intl. Conf. Very Large Data Bases, Brighton, England, pp: 155-158.
2. Piatetsky-Shapiro, G., 1991. Discovery, analysis and presentation of strong rules. In G. Piatetsky-Shapiro and W. J. Frawley, (Eds.), Knowledge Discovery in Databases, AAAI/MIT Press, pp: 229-238.
3. Han, J., Y. Cai and N. Cercone, 1993. Data-driven discovery of quantitative rules in relational databases. IEEE Trans. Knowledge and Data Engineering, 5: 29-40.

4. Quinlan, J.R., 1992. C4-5: Programs for Machine Learning. Morgan Kaufmann.
5. Michalski, R.S. and G. Tecuci, 1994. Machine Learning, A Multistrategy Approach. Vol. 4. Morgan Kaufmann.
6. Fisher, D., 1987. Improving inference through conceptual clustering. In Proc. AAAI Conf., Seattle, Washington, pp: 461-465.
7. Agrawal, R., T. Imielinski and A. Swami, 1993. Mining association rules between sets of items in large databases. In Proc. ACM-SIGMOD Intl. Conf. Management of Data, Washington, D.C., pp: 207-216.
8. Agrawal, R. and R. Srikant, 1994. Fast algorithms for mining association rules. In Proc. Intl. Conf. Very Large Data Bases, Santiago, Chile, pp: 487-499.
9. Agrawal, R. and R. Srikant, 1995. Mining sequential patterns. In Proc. Intl. Conf. Data Engineering, Taipei, Taiwan.
10. Park, J.S., M.S. Chen and P.S. Yu, 1995. An effective hash-based algorithm for mining association rules. In Proc. ACM-SIGMOD Intl. Conf. Management of Data, San Jose, CA.
11. Hen, J. and Y. Fu. 1999. Mining multiple-level association rules in large databases. In IEEE Trans. Knowledge and Data Engineering, vol. 11: 5.