Hierarchical Online Mining for Associative Rules

Naresh Jotwani

Dhirubhai Ambani Institute of Information & Communication Technology
Gandhinagar 382009
INDIA
naresh_jotwani@da-iict.org

Abstract

Mining for associative rules in transaction databases has been a very active area of research in recent years; purchases made by customers at a store provide one example of such transactions. The typical multi-pass associative rule mining algorithm finds large itemsets in the first phase, using a specified minimum support level, and then discovers associative rules in the second phase. In this paper, we propose and discuss an online algorithm which is based on hierarchical classification of items. The proposed algorithm carries out the first phase efficiently in one pass, with tight bounds on the computational effort required, and modest memory requirements. The algorithm is thus capable of online mining of associative rules from transaction streams. We also discuss in the paper some practical issues related to applications of associative rule mining, and the consequent need to broaden the definition of an associative rule.

1. Introduction

One widely discussed paradigm of data mining is the mining of associative rules in transaction databases, an example of such transactions being the purchases made by customers from a traditional or web-based store [1]. In such a case, the associative rules mined convey information of the type: A customer buying set of items X will with a high probability also buy set of items Y. An associative rule of this type is denoted by $X \Rightarrow Y$, with disjoint sets of items X and Y being termed *antecedent* and *consequent* respectively. This type of information about customer behaviour of a product of service can be immensely useful in management decisions related, for example, to product marketing, pricing and promotion.

The task of associative rule mining is typically carried out in two phases: (i) Finding the sets of all items in the transaction database satisfying a user-specified *minimum support*, and (ii) For each set of items discovered in the first set, say Z, finding appropriate complementary subsets X and Y of Z such that the rule $X \Rightarrow Y$ has a user-specified *minimum confidence*.

Implicit behind algorithms proposed for associative rule mining are the following assumptions: (i) The total number of items from which a customer makes a selection is large, since a typical store will have thousands or tens of thousands of items on offer, and (ii) The transaction database is available and analyzed offline, so that the algorithm can make multiple passes over the database. However, it is these required multiple passes over transaction databases which limit the efficiency of currently proposed associative rule mining algorithms.

In this paper, we propose a *hierarchical* and *online* approach to mining associative rules in a transaction database. By *hierarchical mining*, we mean that items occurring in a transaction are classified into various categories, sub-categories, and so on. In a typical store, for example, a category may be 'Toiletries', while a sub-category may be 'Soaps'. Clearly, the associative rules of possible interest to management may be of two types. One type of associative rule may associate individual items, for example: Every customer buying a large bar of soap brand A will buy with a high probability a specific type of canned soup of brand B. However, a much more potentially useful type of association will be: Every customer of soap will buy with a high probability some brand of canned food.

International Conference on Management of Data
COMAD 2005b Hyderabad, India, December 20-22, 2005.

Note that, in the latter case, categories or sub-categories of items are being associated rather than individual items.

Arguably, in fact, the second type of association is more likely to be of interest to management, while most reported algorithms provide no means of dealing with categories and sub-categories of products. This is in fact a serious limitation of many reported methods which needs to be addressed so that more meaningful associations can be discovered. Even if strong associations exist between categories or sub-categories in the buying patterns of customers, most of the currently reported algorithms will not be able to discover them. Instead, the algorithms proposed so far will report a large number of associations between individual items, and the more meaningful pattern or patterns will be lost amongst the many individual associations reported. In this paper, we show an approach which can yield an association pertaining to any level in the hierarchy into which products are categorized.

By *online mining*, we mean that the algorithm needs to make only one pass over the transaction data. In some cases, as discussed below, the hierarchical mining algorithm may make a subsequent pass over the transactions; but even in such cases, the user has a choice of avoiding the second pass, and instead looking for category-wise associations in upcoming transactions rather than those already processed. If patterns of associations are persistent, and a sufficiently large number of transactions are examined, then clearly the same associations will be discovered independent of the time-period spanned by the transactions examined. In this important sense, therefore, the approach presented in this paper satisfies the requirements of associative rule mining from online data streams.

1.1 Overview

In section 2 below, we formally define the problem of associative rule mining, both in its original version and the proposed hierarchical version. We also present a very brief survey of other associative rule mining algorithms which have been proposed so far. In section 3, we describe the algorithm which satisfies the requirements of hierarchical and online mining and point out some implications of the algorithm. In section 4, we discuss some practical aspects of data mining, and of the proposed algorithm, including a typical application which is different from the department store paradigm on which the original problem formulation is based.

Finally, in section 5, we present our conclusions and some ideas for further work.

2. Problem Definition and Related Work

Set $I = \{i_1, i_2, \ldots, i_N\}$ represents the set of *items* available for purchase. The transaction database TD contains sales transactions. Each transaction record $T \in TD$ contains a subset of I, being the items purchased in the particular transaction, and therefore we may write in brief $T \subseteq I$.

An *associative rule* is denoted by $X \Rightarrow Y$, where $X,Y \subseteq I$ and X,Y are disjoint; the rule is an expression indicating that, for a transaction $T \in TD$, if $X \subseteq T$ holds, then $Y \subseteq T$ will also hold with a high degree of probability. X and Y are termed respectively the *antecedent* and the *consequent* of the associative rule.

S*upport* for associative rule $X \Rightarrow Y$ is defined as the fraction of transactions in TD which contain items $X \cup Y$; *confidence* of the rule is defined as the fraction of transactions in TD containing X which also contain Y. We shall denote the support and confidence of rule $X \Rightarrow Y$ as $supp[X \Rightarrow Y]$ and $conf[X \Rightarrow Y]$ respectively. As convenient notation, we may sometimes write XY to represent $X \cup Y$, where $X,Y \subseteq I$.

The problem of associative rule mining is defined as: Find all associative rules of the type $X \Rightarrow Y$ which have a specified minimum support and confidence in TD.

One issue must be correctly accounted for by any mining algorithm: So-called *redundant rules* must be removed from the final output of the algorithm [7]. For example, if we assume that $X \Rightarrow YZ$ is a rule with the requisite support and confidence in TD, then rules $X \Rightarrow Y$, $X \Rightarrow Z$, $XZ \Rightarrow Y$ and $XY \Rightarrow Z$ will also necessarily have the requisite support and confidence. Which of these is the correct rule to be included in the output? To attempt to answer this question, we may need to make an assumption of the following type:

Assumption of independence of consequents: If $\{y_1\}$ and $\{y_2\}$ are both consequents of antecedent X, then:

$$conf[X \Rightarrow \{y_1,y_2\}] \leq conf[X \Rightarrow \{y_1\}] * conf[X \Rightarrow \{y_2\}].$$

Justification for this assumption is as follows: Suppose, with antecedent X, the inequality is not satisfied. Then the presence of y_1, say, in the transaction results in an increased probability of y_2 being also present. But in that case $X \cup \{y_1\}$ should properly be discovered to be an antecedent for $\{y_2\}$, and the rule having only X as

antecedent is redundant. Therefore, if the algorithm correctly discovers maximal antecedents of the type $X \cup \{y_1\}$, the assumption of independence of consequents is justified.

Another possible form of redundancy arises if both the rules $X \Rightarrow Y$ and $Y \Rightarrow X$ have the required minimum support and confidence. In such cases, the correct output would be a symmetric associative rule of the type $X \Leftrightarrow Y$. The issue of finding *essential* rules, as opposed to redundant rules, has been discussed in [7].

As mentioned in the previous section, reported algorithms for associative rule mining work in two phases:

(i) Find the sets of items with specified minimum support, and

(ii) From the result of the first phase, find associative rules with specified minimum confidence.

The importance of the first phase of the reported algorithms is seen at once from the fact that $N = |I|$ is assumed to be a fairly large number, in thousands or tens of thousands. Since a transaction T is any element of $\mathcal{P}(I)$, and $|\mathcal{P}(I)| >> |TD|$, most elements of $\mathcal{P}(I)$ will be absent from TD. Minimum support is in fact a mechanism to weed out from further consideration transactions which are unlikely to yield useful associations.

When we classify the items into a hierarchical scheme, however, the overall picture changes substantially. For the purposes of this discussion, let us assume that, at each level of classification, a fixed number of M classes or sub-classes are present. M will in practice be a fairly small number; for the purposes of this paper, as an example, we assume M = 12. Thus there are M primary classes $C_1, C_2 \ldots C_M$ into which the products in set I are classified. Further, the products in primary class *k* are classified into sub-classes $C_{k1}, C_{k2} \ldots C_{kM}$, and so on down the hierarchy. We can thus use notation such as:

$$I = C_1 \cup C_2 \cup \ldots \cup C_M$$

and

$$C_k = C_{k1} \cup C_{k2} \cup \ldots \cup C_{kM}.$$

Let us assume arbitrarily a total of four levels of classification, representing a balanced and regular M-ary classification tree of height four. The total number of items which can be accommodated is then equal to M^4. For M=12, this represents a total of about 20K items, which is of the order of the typical value usually assumed for $|I|$. In terms of our notation, any item $i \in I$ can now be represented as C_{jklm}, where *j*, *k*, *l* and *m* are indices to the four successive levels of classification of item *i*; C_{jkl}, for example, represents the third or lowest level sub-class containing the item. From a practical point of view, we should note that commercial product codes do usually follow hierarchical classification.

The associative rules being mined for with the hierarchical classification of items now have the following form: In the rule $X \Rightarrow Y$, X and Y are disjoint subsets of some C_α, which is a class or sub-class as defined above. At the root node of the hierarchy, we naturally have $C_\alpha = I$.

The key difference this classification makes to the first phase of the algorithm is this: Since M is a fairly small number, it is feasible and efficient to store the occurrence counts of all 2^M classes, sub-classes, or items in a sub-class, using an integer array, a simple and efficient structure which serves our requirements. Of course it is not feasible to track simultaneously the occurrence counts of all classes, sub-classes or items, since the number $2^{|I|}$ is unmanageably large. What is proposed here instead is that the user tracks at a given time certain selected classes, sub-classes, or items in a sub-class. If K is number of classes or sub-classes being tracked at a given time, the number of arrays of size 2^M required is K, and the memory requirements are very reasonable.

The processing of each transaction $T \in TD$ requires incrementing integer values in these K integer arrays, and therefore the first phase of the algorithm can be implemented efficiently online.

Thus, instead of using minimum support to weed out itemsets in the first phase, the proposed algorithm, basically a version of 'divide-and-conquer' strategy, accumulates in the first phase *all* itemset counts in the selected K classes and sub-classes.

In fact there is another assumption often made which may not be justified in practice. In test results usually reported, it is assumed that $|T|$, the number of items in transaction T, is Poisson distributed with mean $\mu = 4$ or 6. The processing time of the reported algorithms depends critically on μ, but it is not clear whether Poisson distribution with a fairly small mean represents typical customer transactions in a large store.

In the algorithm proposed in this paper, no assumption is made about $|T|$, since a version of 'divide-and-conquer' strategy is employed to deal with the combinatorial explosion implicit in the fact that T is in general any subset of I. At the same time, simple and efficient counting makes it possible to implement the algorithm as part of an online system.

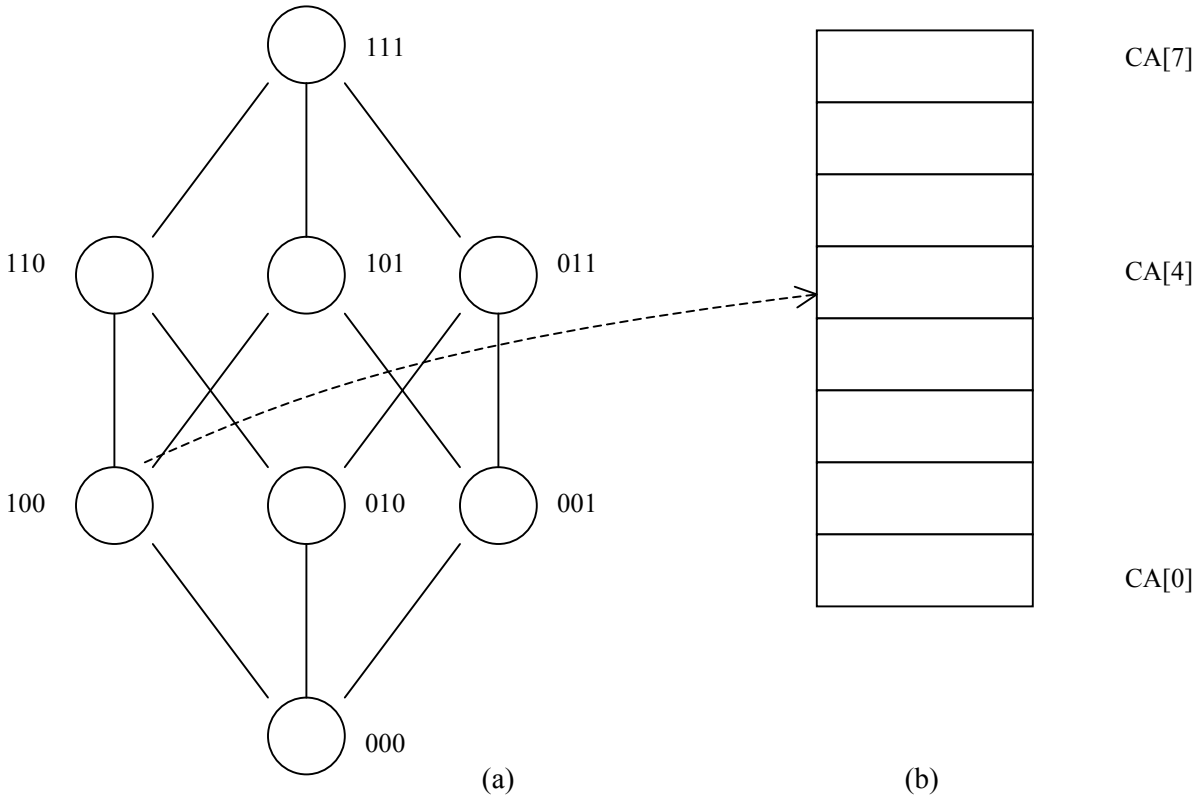

Figure 1. All possible subsets of a set of three elements are shown in the lattice of part (a); bit patterns shown represent subsets in a natural way. Array CA of part (b) has one element corresponding to each subset in (a). One such correspondence is shown using a dashed line and arrow.

The problem of mining for associative rules in transaction databases was initially defined and developed in its present form in [1,2]. Many other researchers have since reported related work, see for example the work reported in [3,5,6,7], or survey articles such as [4,8]. The stress in most of the reported work is usually on the first phase of the algorithm. However no reported algorithm can accomplish the first phase in one pass, since the support level for a set of items can be known only after an initial pass over the entire transaction database.

A version of online associative rule mining is described in [5], which makes use of a definition of online processing less strict than that used in the present work. Both the phases of the algorithm are discussed in [7], which also addresses explicitly the question of redundant rules. Multiple-level association rules are discussed in [3], but the approach adopted therein is very different from the one in the present work. As we shall see, the approach proposed in this paper can be combined with the second phase of the algorithm described in [7].

3. Description of the Algorithm

Figure 1 illustrates how the count data corresponding to any class or sub-class can be stored in an integer array. Here, for the purpose of illustration, we have taken M = 3. Elements in an array of size $2^3 = 8$ can in this case keep track of the number of occurrences of the corresponding subsets of class or sub-class elements. The bit-pattern corresponding to any subset yields in a natural manner the index of the corresponding array element. In practice, of course, we will have a value of M larger than 3; for M = 12, for example, the array CA will have 4K elements. With this representation, to test whether the subset corresponding to array index `i` is itself a subset of the one corresponding to array index `j`, we simply test for the condition `i||j == j`.

Since we are not weeding out subsets of items having less than a minimum support, the first phase of the algorithm can be implemented in one pass in a simple and efficient manner.

Since we may be tracking up to K different classes or sub-classes at one time, in the algorithm there are in fact K integer arrays of size 2^M each, denoted by `CA[i]`, i=1,2,…,K. For each transaction T and class or sub-class C being tracked, the function named `constructBitmap(T,C)` in the algorithm below returns the M-bit vector representing the sub-classes or items of C which are present in T.

```
//
// Hierarchical online associative rule mining – Phase I
//
// Important variables:
// M    : Degree of the classification tree
// TD   : Transaction database or stream
// SC   : The set of classes or sub-classes being tracked
// K    : |SC|
// C    : One class or sub-class in SC
// CA[i]: Count array corresponding to the i-th class
//        or sub-class being tracked, of size 2^M
// bmap : Unsigned integer
//
// function constructBitmap(T,C): Returns a bitmap
// corresponding to set of sub-classes of C present in T
//
// function findSubset(bmap,j): Returns the j-th subset
// of bmap as a bitmap, for 1 < j < 2^|bmap|-2
//
// Collect raw counts for each class or sub-class
//
for each transaction T in TD do
   for i = 1 to K do {
      C = i-th class or sub-class in SC
      bmap = constructBitmap(T,C);
      if (bmap) {
         CA[i][bmap]++;
         b = number of 1 bits in bmap;
         for j = 1 to 2^b-2 do {
            subset = findSubset(bmap,j);
            CA[i][subset]++;
         }
      }
   }
//
// Note: TD is not required for further processing
//
// End of Phase I. Array CA[i] has counts corresponding to the
// i-th class or sub-class in the set of classes being tracked.
//
// The arrays CA[i], i=1,2..K, are used in Phase II.
//
```

Figure 2. Phase I of the online associative rule mining algorithm.

With this background, the algorithm for the first phase is shown in Figure 2. The innermost loop in the algorithm shown, consisting of the `for` loop on loop variable `j`, adjusts the counts to the basic condition of associative rule mining: When a transaction T contains a subset X of items (or of classes or sub-classes, as in our case), all possible subsets of X except the null set must also be counted as having occurred in the transaction.

For the functions `constructBitmap(T,C)` and `findSubset(bmap,j)` in the above algorithm, implementation details are not shown. If we recall that commercial product codes do follow a hierarchical coding scheme, it becomes fairly clear that `constructBitmap(T,C)` should admit of a simple implementation.

The function `findSubset(bmap,j)` is in the innermost loop. This function involves bit-wise operations on bit strings of length M, and its running time is linear in M. For each transaction T and class or sub-class C, the total running time of the innermost loop is thus linear in $M*2^b$, where b is the number of sub-classes of C which are present in T. Function `findSubset` involves bit operations on operators of length M bits, and can therefore be coded efficiently. In the present algorithm, this innermost loop provides the subset counts which in earlier reported algorithms have been computed using two or more passes over TD.

In the second phase of this algorithm, as in other reported algorithms, associative rules are to be discovered from the counts accumulated in arrays `CA[i]`, i=1,2,…,K. In [7], for example, a structure described as 'adjacency lattice' is used to discover associative rules, avoiding redundancies to a certain extent. We propose that, under the hierarchical classification as defined above, the second phase of the mining algorithm should follow the basic strategy of [7], though there exists further possibility of removal of redundant rules.

Of course the present algorithm allows rules to be discovered only amongst or within the K classes or sub-classes of items being tracked. In practice, it is the user who will determine which classes or sub-classes are to be mined for associations at a given time. A second pass over TD is required only if the user decides to mine previously stored data for a different set of classes or sub-classes.

In this context, we may note that, as illustrated in Figure 1, the integer arrays `CA[i]` represent, in the terminology of [7], the K different adjacency lattices[1] of the classes or sub-classes being tracked. We have in fact accumulated into these K adjacency lattices all itemset counts with 'primary threshold' of zero. A suitable version of the second phase algorithm of [7] is therefore to be used in conjunction with the first phase algorithm described here. The second phase algorithm of [7] does not ensure that all redundant rules are removed, and some further work in that direction is likely to yield useful results. The assumption made above about the independence of consequents, and further reasoning along similar lines, is likely to prove useful towards achieving this objective.

4. Discussion

For discovered associative rules to play a significant role in decision support, attributes of transactions other than item sets must also be considered. Therefore any mining algorithm should be capable of taking into account such other attributes of transactions, examples of these being: the total monetary value of a transaction, the date and possibly time, whether any special promotional schemes were availed, and so on. The user may decide, for example, to omit from analysis transactions with monetary value less than a certain minimum; or the user may wish to analyze only the transactions of a set of selected customers.

To achieve this with the algorithm proposed in this paper, what is required is simply this: The function `constructBitmap` should take into account such specified additional conditions, for each transaction processed. If the conditions are satisfied, the return value `bmap` is used exactly as described; otherwise, the return value is set to zero and thus the transaction is ignored. SQL provides one possible mechanism for specifying such additional conditions.

As mentioned above, a second or subsequent pass over TD is required only if it is decided to mine previously stored data for a different set of classes or sub-classes, and possibly different additional conditions of the type outlined above. If associative rules are assumed to be persistent over the previously stored transactions and the online transaction stream, clearly the user may analyze either set of transactions to mine for associative rules. To have a basis for decision support, in practice the user will of course be interested to

[1] In fact the arrays represent Boolean algebras, but the additional properties are not being used in the algorithm presented.

discover associative rules which are persistent over the relevant period of time.

There are types of business in which a transaction does not really follow the department store paradigm, with large $|I|$, which is the basis of the original problem formulation [1]. Telephone company call data records provide one example of a different type of transaction; banks and airlines also have product offerings and transactions which are of a different kind from those in a store. In telephone call records, for example, we may decide to analyze attributes such as: whether the user has pre-paid for the service, time-period of the call e.g. peak, off-peak or night, local or long-distance call, and call duration. Relevant information about a call can be encoded using, say, sixteen bits. Therefore such transaction data can be mined online for associative rules, with $M = 16$ and without any further classification.

In general, the specific mining algorithm used should depend both on the type of transactions generated and on the type of associations which the user needs to discover in the transactions. The algorithm presented in this paper is capable of being tailored to a range of different needs.

5. Conclusions and Future Work

In any large organization, the possibility of online mining of transaction data for associative rules will remain attractive. In this paper, we have explored a hierarchical approach to such online data mining. We have shown how the practical requirements can be met using a simple and efficient strategy with tight bounds on the computational effort required. In practice, it is possible to attach pre-qualifying conditions, of the type discussed above, with the data mining operation Also, as discussed above, there exist types of business in which the equivalent of *item set* in each transaction can be encoded using no more than, say, sixteen bits; call data records of phone companies are transactions of this type. In such cases, the online approach outlined in this paper can be used without any need for classification.

Further work in this direction will focus on two areas. Firstly, we hope to generalize the concept of an associative rule in the presence of hierarchical classification. In particular, we hope to explore the possibility of discovering rules such as $XY \Rightarrow Z$ where X, Y and Z may occur in different classes or sub-classes. In general, the idea would be to explore fully the rule mining capability of the hierarchical approach. Also, in the second phase of the algorithm, we hope to focus on extracting only the essential rules from the associative rules discovered. Experimental work is being planned towards this end, and there is also a need to further explore reasoning of the type which leads to the assumption of independence of consequents.

References

[1] Agrawal R, Imielinski T and Swami A, Mining association rules between sets of items in very large databases, Proceeding of the ACM SIGMOD Conference, 1993, pp.207-216.

[2] Agrawal R and Srikant R, Fast Algorithms for Mining Association Rules Large Databases, Proceeding of the 20th VLDB Conference, 1994, pp. 478-499.

[3] Han J and Fu Y, Discovery of Multiple-Level Association Rules from Large Databases, Proceeding of the 21st VLDB Conference, 1995, pp. 420-431.

[4] Chen M S, Han J and Yu P S, Data Mining: An Overview from Database Perspective, IEEE Trans. on Knowledge & Data Engineering, Vol 8, No 6, 1996, pp. 866-883.

[5] Hidler, C, Online Association Rule Mining, Proceeding of the ACM SIGMOD Conference, 1999, pp.145-156.

[6] Han J, Pei J and Yin Y, Mining frequent patterns without candidate generation, Proceeding of the ACM SIGMOD Conference, 2000, pp.1-12.

[7] Aggarwal C C and Yu P S, A new approach to online generation of association rules, IEEE Trans. on Knowledge and Data Engineering, Vol 13, No 4, 2001, pp.527-540.

[8] Zhao, Q and Bhowmick, S S, Association Rule Mining: A Survey, TR No 2003116, CAIS, Nanyang Technological University, Singapore, 2003.