Proceedings of the 5th WSEAS Int. Conf. on Artificial Intelligence, Knowledge Engineering and Data Bases, Madrid, Spain, February 15-17, 2006 (pp384-387)

Generating Classification Association Rules with Modified Apriori Algorithm

B. TUNC and H. DAG
Computational Science and Engineering Department,
Istanbul Technical University (ITU),
Maslak, 34469
TURKEY

Abstract: - Association rule mining is a useful and widely used method to extract patterns from large sets of data especially if we deal with basket type data such as customer buying attitudes. It finds all possible relations between data fields, and this property is functional in many research domains; however it happens to be useless in domains like medicine and health as it produce lots of ineffectual rules concerning irrelevant data fields. Thus it is not likely to classify a disease easily by using classic rule mining algorithms. Classification algorithms, on the other hand, only generate decision trees or classifiers according to pre-determined target; therefore, they need to be tuned to produce human readable rules that can be used in decision support. In this study, an integrated approach was proposed to produce association rules that can be used as classifiers. *Apriori* algorithm was used as a base model and modified the algorithm to be able to generate human readable classification association rules. Some experiments with real medical data sets were conducted to compare our results with the results of other well known algorithms like C4.5 and Ripper.

Key-Words: Data mining, Association rules, Classification, Medical data

1 Introduction

Keeping customers' (in general meaning) data in electronic media and using them as a part of decision making process has been improved fast since critical enhancements in data acquisition and manipulation took place in last decade. With this enormously large date warehouses it has been possible to generate ideas and knowledge from raw data with help of several data mining approaches. Lots of data mining algorithms have been developed with different approaches such as classification and association rule induction [7]. Classification rule mining aims to produce paths to classify a given instance. It is a target driven approach since it tries to put an instance in one of the predefined classes. In association rule mining, it is meant to find all possible rules in the data set which satisfy some user-defined parameters. Association rule mining does not require and accept targets to generate rules.

In domains like medicine and health, it is required to produce classification rules to determine an instance's category i.e. a patient's diagnosis [4]. Several works showed that neither classification rule mining nor association rule mining are not satisfactory to be used in such domains [9]. Association rule mining finds all possible relations between data fields, and this property is functional in many research domains; however it happens to be useless in domains like medicine and health as it produce lots of ineffectual rules concerning irrelevant data fields. Thus it is not likely to classify a disease easily by using classic rule mining algorithms. Classification algorithms, on the other hand, only generate decision trees or classifiers according to pre-determined target; therefore, they need to be tuned to produce human readable rules that can be used in decision support. For these reasons integration of two approaches has attracted various authors now that it allows using both the power of association rule mining and the briefness of classification.

Most of the studies relevant to rule mining have been some how related to the *Apriori* algorithm developed by Agrawal et al. [1]. It is one of the most widely known and used algorithm in the data mining literature. Our effort is to develop a partly-new algorithm AClass (stands for **A**priori for **Class**ification) to generate Classification Association Rules (CARs). It is again depended on *Apriori* algorithm and that is why the term "partly-new" was preferred to use.

AClass algorithm integrates classification and association rule mining as decomposed in Section 2. The details about algorithm and its logic will be presented in Section 3. In this study, it is aimed to work mainly in medical domain; hence, 3 health data sets were used to conduct performance analyses with our algorithm. Results showed that our algorithm is competitive to other integrated algorithms. Performance results will be presented in Section 4. Section 5 will discuss our future plans and give a brief conclusion.

2 Problem Decomposition

Considering data representation in medical domain, it is suggested to have a data set which includes number of transactions about patients' measured values. Let **D** be the set of all transactions, **A** be the set of all attributes i.e. fields in data set. Then the notation for association rules is used as follows: $X \Rightarrow Y$ where $X \subset A$, $Y \subset A$ and $X \cap Y = f$. $X \Rightarrow Y$ holds with confidence **c** if **c**% of transactions in **D** that contain X also contain Y. $X \Rightarrow Y$ has a support value of **s** if **s**% of transactions in **D** contain $X \cup Y$. Let **C** be the set of all possible classes which are used to categorize transactions in **D**, where $C \subset A$. Then the rule $X \Rightarrow Y$ is said to be classification association rule if $Y \subset C$. It can be stressed that Y needs to consist of only one $A_i \in A$ since any transaction can not have multiple class labels. Original *Apriori* algorithm is more general in that it allows a consequent to have more than one item; thus, some modifications had to been done to restrict consequents to be size of one as mentioned above.

In real life problems, attributes come in variety of formats like categorical or continues values whereas AClass algorithm only accepts 0 and 1 values. Therefore Discretization methods were used to handle different formats. For categorical attributes, new attributes, which will have a value of 1 if the original attribute has that value and 0 if it does not, for each value of the original attribute were added. Intervals for continues attributes were used and same logic was applied as with categorical attributes. Intervals can be determined by several methods. It is important to take account the information loss and rule accumulation factors while deciding. Further details can be obtained from [2].

It is possible to use two stage approach to generate CARs. In this approach all possible CARs are first generated, and then some purification strategies are applied to build a classifier. Nevertheless, AClass prunes the rules just before they are generated. After each frequent itemset (the term 'itemset' is employed to indicate set of attributes as it is used in most of other studies) that satisfies user-defined support value is created, rule production immediately takes place. If a CAR is generated with sufficient confidence value then that frequent itemset is removed from hash-tree to prevent any superset of that set is considered as a new frequent itemset. Other run-time pruning strategies can be found at [3].

3 AClass Algorithm

AClass algorithm generates pruned CARs in one stage instead of first producing and then pruning; even so, sections describing these two tasks were separated to provide better understanding. At the end of main process a classifier is created and the only task remaining is the ordering rules space. Rules are ordered according to their confidence, support, and length. Details about these stages will be explained in following three sections.

3.1 Generating CARs

AClass algorithm uses almost same logic as *Apriori* to produce frequent itemsets and rules except some extra controlling tasks to determine whether a frequent itemset can offspring a CAR or not. After each frequent itemset is generated, it is controlled to certify that it contains one of the class attribute. In case of being unable to find any class attribute in a frequent itemset, this itemset is not sent to rule production procedure.

Attribute names are treated as integer values to simplify operations. They are in ascending order and class attributes are placed at the end. Since itemsets also contains attribute names in ascending order it is easy to check whether an itemset includes a class attribute as class attribute will always be last item.

Hash-tree data structure is used to store itemsets and with a subset function, support of each itemset is counted by traveling through hash-tree. At each step, candidates are generated by merging prefound frequent itemsets at previous step, and then their support values are counted. Any itemset having a support value below the user-defined threshold is removed from hash-tree. Inasmuch as next group of itemsets will be formed from previous itemsets placed in the hash-tree, it is prevented that any supersets of itemsets with low support will appear in new candidate group. These steps are related to following simple logic. Let (X,Y) be an itemset with a support value S_1, and (X,Y,Z) with S_2. Since (X,Y,Z) is a superset of (X,Y), their support values always hold the inequality $S_1 \geq S_2$. That is if S_1 is below the threshold, S_2 has to be below, too.

Although we only generate rules from itemsets which include one of the class attributes, we can not remove an itemset with no class attribute for the reason that an itemset containing class attribute may be formed at next step from this one. Let $Z \in C$, and $X,Y \notin C$. We can merge two itemset (X,Z) and (X,Y) to create a new itemset (X,Y,Z) which also contains class attribute Z.

3.2 Pruning CARs

AClass purifies rule space as it generates new rules. Its main pruning strategy depends on near-equivalence of support values. An itemset's support will be nearly equal to one of its subsets, and never greater than. This fact introduces the approach of omitting rules with supersets in their antecedent. For example, suppose a CAR $X \Rightarrow Z$ was already generated with a sufficient support and confidence value. It is obvious that rule $X \Rightarrow Z$, will comprise more transactions in **D** than $X, Y \Rightarrow Z$ by the idea mention above. That is there is no need to include rule $X, Y \Rightarrow Z$ in classifier since the rule $X \Rightarrow Z$ can already be applied to all transactions that will be covered by latter rule. However it is more likely that confidence of the rule $X, Y \Rightarrow Z$ will be greater than the first one. This is a trade-off between effects of support and confidence in total accuracy of classifier. This problem was overcome by introducing a new confidence threshold that will determine whether a superset will be omitted or not. If a rule is generated with confidence above our threshold which is usually greater than the original user-defined minimum confidence value, its constructor itemset will be removed from hash-tree to omit new rules with supersets in their antecedent. Otherwise, that is confidence value is below our new threshold, itemset will remain in hash-tree.

3.3 Ordering CARs and Building Classifier

There are several methods applied in literature to order rule space such as CSA, WRA, Laplace Accuracy, and c^2 testing [5]. In this work, CSA (confidence, support, and size of antecedent) approach was used. The first factor that determines rules' order is their confidence value: the higher rule's confidence is, the lower its order number is. If we have two rules having exactly same confidence then we look for their supports. Again higher support is preferred. Finally, for the rules with same confidence and support, the smaller rule is placed before the longer one.

Algorithm does not involve any further pruning or manipulating tasks after ordering is completed. With ordered rule space and a default class, which is the most seen class in data set, a classifier is presented at the end. Classifier is applied to test data line by line. That is the first rule in classifier is tried to determine a transaction's class and if it is not applicable, the second one is attempted, then the third one, and finally default class is assigned to transaction.

A pseudo code for AClass algorithm is presented on Figure 1. FI_k denotes Frequent Itemsets with size k. $PreCan_k$ is the set of candidates with size k which are not pruned; Can_k is the set of final candidates. **D** is the set of training data. **C** is the set of all class attributes and c is a class attribute in **C**. can.sup denotes the support of a single candidate in Can_k. Treashold is our new control value to decide to remove a candidate after successful rule generation.

```
Create $FI_1$;
For (k=2; $FI_{k-1} \neq f$; k++) {
    //Generate non-pruned candidates from
    //previous frequent itemsets
    $PreCan_k$ = CandidateGenerate ($FI_{k-1}$);
    //Prune Pre-Candidates to create candidates
    $Can_k$ = CandidatePrune ($PreCan_k$);

    ForAll Transaction in D {
        Count candidates and calculate support;
    } End ForAll

    ForAll Candidates can $\in Can_k$ {
        IF (can.sup >= MinSup) {
            Add can to $FI_k$;
            IF (c $\in$ C | c $\in$ can) {
                //Generate rule from candidate and return
                //confidence value
                Conf = GenerateClassAssRule (can);
                IF (Conf >= Treashold) {
                    Remove can from Hash-Tree;
                }
            }
        }
        Else {
            Remove can from Hash-Tree;
        } End IF
    } End ForAll
} End For

//Build classifier
OrderRules (Rule Space);
PrintClassifier (Rule Space);
```

Fig. 1: Algorithm AClass

4 Experimental Results

For performance testing several health related data sets from UCI ML repository were used. All experiments are performed on a 2.8GHz Pentium-4 PC with 1GB main memory on which Linux Suse distribution is running. Each algorithm is obtained from its author and used with default configurations.

AClass algorithm was tested against C4.5 [10], Ripper [6], and Apriori-TFPC [5]. The first two algorithms are well known classification systems which are also qualified to generate classification rules. The last one is very similar to AClass algorithm as it is also based on Apriori except data structure used, pruning and ordering steps applied. All approaches are capable to generate rules that can classify transactions.

No time analysis but only accuracy experiments were conducted Accuracy results are shown on Table. 1. Details about the data sets used are given on Table 2. 50/50 training/test data approach was used for accuracy testing. Each data set was divided into two parts. The first part was used for training the algorithm and the second part for testing rules generated. AClass and Apriori-TFPC algorithms were run with support and confidence values of %20, %90 respectively. Other algorithms are used with their default configurations. Only for 'allbp' data set, it was needed to decrease support value to %5 and confidence value to %70 since one of the classes was dominant and it was the only way to produce rules for other classes.

Data sets were cleaned and discretized before using. For algorithms C4.5 and Ripper, which are capable of using continuous values, both discretized and non-discretized data sets were used and the best results were presented.

Table.1: Accuracy results for each algorithm

Data Set	Apriori-TFPC	C4.5	Ripper	AClass
Allbp	98.15	89.4	91.8	98.15
wdbc	87.67	94.7	93.68	93.33
breast	97.94	97.7	95.91	97.95
AVGR.	***94.59***	***93.93***	***93.8***	***96.48***

Table.2: Data sets used in accuracy tests

Data Set	
allbp	Thyroid disease records. After cleaned it consists 1946 instances and 13 attributes including class attribute. 50 attributes after discretization.
wdbc	Wisconsin diagnostic breast cancer records with 569 instances and 31 attributes including class attribute. 130 attributes after discretization.
breast	Wisconsin breast cancer records with 683 instances and 10 attributes including class attribute. 30 attributes after discretization.

5 Conclusion and Future Work

AClass algorithm which depends on well known algorithm *Apriori* is intended to generate classification association rules and, as accuracy results show, it achieves high accuracy points especially when compared with ordinary classification approaches. Association rule mining is a powerful technique that can handle all kind of pattern in data. However for domains like medicine and health, it is important to be able to classify transactions. AClass algorithm is an integrated approach and generates association rules which are capable of classification with high accuracy levels.

It will be also interesting to improve time consumption value and scalability of algorithm in the future works. Testing AClass against other well known approaches such as CMAR [8], CBA [9] with more data sets is also planned. At this stage, only real life data from medical domain were used to analyze its efficiency in this domain. It is also possible to use synthetic data sets to conduct time performance experiments. Parallelization of algorithm to be used on very large data sets is one of our main goals.

Acknowledge: Authors wish to acknowledge the financial support from the State Planning Organization of Turkey. They also appreciate technical support from ITU Informatics Institute, High Performance Computing Center (HPCC) where all experiments were conducted.

References:

[1] R. Agrawal, R. Srikant, Fast algorithms for mining association rules, *IBM Research Report RJ9839*, IBM Almaden Research Center, San Jose, CA, 1993.

[2] R. Agrawal, R. Srikant, Mining quantitative association rules in large relational tables, *In Proc. of the ACM SIGMOD International Conference on Management of Data*, 1996, pp.1-12.

[3] R. J. Bayardo, Brute-force mining of high confidence classification rules, *In Proc. of International Conference on Knowledge Discovery and Data Mining*, 1997, pp.123-126.

[4] Krzysztof J. Cios, William Moore, Uniqueness of medical data mining, *Artificial Intelligence in Medicine*, 26(1-2), 2002, pp. 1-24.

[5] Coenen, Leng, An evaluation of approaches to classification rule selection, *In Proc. of the IEEE ICDM*, 2004, pp.359-362.

[6] W. Cohen, Fast effective rule induction, *In Proc. of the ICML*, 1995, pp.115-123.

[7] Alex A. Freitas, Simon H. Lavington, *Mining Very Large Databases with Parallel Processing*, Kluwer Academic Publishers, 2000.

[8] W. Li, J. Han, J. Pei, CMAR: Accurate and efficient classification based on multiple class-association rules, *In Proc. of the ICDM*, 2001, pp.369-376.

[9] B. Lui, W. Hsu, Y. Ma, Integrating classification and association rule mining, *In Proc. of the KDD*, 1998, pp.80-86.

[10] J. R. Quinlan, *C4.5: Programs for Machine Learning*, Morgan Kaufmann, 1993.