Mining Insightful Classification Rules Directly and Efficiently

Hongyan Liu Jian Chen Guoqing Chen
School of Economics and Management Tsinghua University
Beijing, 100084, People's Republic of China
{Liuhy, Chenj, Chengq}@em.tsinghua.edu.cn

ABSTRACT

Classification is one of the important problems in the field of data mining. Many algorithms have been proposed to solve this problem and each has its own drawback. This paper discusses issues about mining classification rules directly and proposes two algorithms, namely UARC and GARC. These algorithms use a more suitable association rule mining technique to find insightful and a complete set of rules directly and accurately. Unlike most other association rule mining algorithms, the algorithms proposed in the paper can find both frequent k-itemset and rules at the same step. After each scan of the database, only rule itemsets and excluded itemsets are saved and used to exclude much more itemsets to generate larger candidate itemsets, which will save much computation time and memory. Upon the information gain criterion, many training cases which satisfy a special condition can be deleted from database, which will lead to fewer I/O times for every remaining scan of database. Finally, a criterion is defined to terminate the whole mining process much earlier and at the same time produce a meaningful rule.

1. INTRODUCTION

Classification is one of the important problems in the emerging field of data mining (also known as KDD: Knowledge Discovery in Database) which is aimed at finding a small set of rules from training data set with predetermined targets [1]. Many algorithms such as decision tree, neural network and statistics methods have been proposed to solve this problem. Compared with other methods, decision tree classifier is relatively better in speed, accuracy and understandability. Many algorithms proposed in the past are only designed for memory-resident data set. Quinlan's C4.5 or its new version C5.0 is one of such well-known methods [2]. In order to deal with the limitation in size of data set for decision tree based classification algorithms, many algorithms have been put forward in recent years [3][4][5], which, however, have some drawbacks as follows. First, in order to find best split, at every node every attribute and every value group (or interval) should be test. With very large data set, this may result in more times of scan of data set and high cost in efficiency. On the other hand, using sampling or partition techniques may result in low accuracy. Second, the classification rules correspond to decision trees without pruning and final generalizing are usually cumbersome, complex and inscrutable. While pruning and final generating rules from trees need extra times to scan the data set. Lastly, as for numeric attributes, the test form is $A \leqslant v$, where A is a attribute and v is a value. This form is not always intuitive and sometimes leads to complicate rules.

Recently, a new algorithm which use association rule mining technique to find classification rules from large amount of data is discussed by Bing Liu et al.[6]. Association rule mining is another important technique in data mining, which aims to find associations among itemsets in transactional database. There are many algorithms proposed in recent years to solve this problem [7][8][9][10][11], among which Apriori given by Agrawal and Srikant (1994) is usually regarded as a typical algorithm. In this algorithm, association rule mining is separated into two steps.: first, generating a complete set of frequent itemsets and then producing association rules from frequent itemsets. Using this algorithm, Being Liu et al. proposed a algorithm named CBA (Classification Based on Association) to find classification rules in two steps which is first generating a complete set of association rules called CARs among training data set and then building a classifier from CARs. As a training data set for classification usually contains a huge number of association rules, so it costs much time to find all of these association rules among which many are not useful, and need a large memory to store them. Furthermore, selecting a subset of these rules to build

0-7803-5731-0/99$10.00 ©1999 IEEE

a classifier is also a time consuming task. Therefore, in this paper, two algorithms are proposed to deal with these problems. The ideas are as follows:

Firstly, in order to prevent from producing a huge number of candidate itemsets, the traditional association rule algorithm is modified. The reason why so many candidate itemsets could be produced is that the minimum support is set to be very low due to the consideration of a complete set of candidate itemsets. However, a classification rule is after all a special kind of a association rule. During the classification rule mining process, what is important is to find rules that are frequent itemsets and at the same time have a high confidence instead of frequent itemsets only with low support. So we combine the two steps of mining association rules to one and focus on finding and storing rule itemsets and excluded itemsets (defined in section 2), both of which are very small sets compared with frequent itemsets. On the other hand, Apriori algorithm reduces the number of candidate itemsets by excluding the itemsets that do not satisfy minimum support to form larger itemsets, while in our algorithms, in addition to this, another constraint is also defined to reduce the candidate itemsets further. That is every itemset that already produced a rule is excluded to make larger candidate itemsets.

Secondly, in order to save time during the scan of data set, based on the notion of information gain [2], some cases which have produced a rule and contain a value of the first split attribute can be deleted from the database. Then the generating method of candidate itemsets changes as well. As a result, the database will become smaller as the times of scan increases.

Thirdly, in order to improve the efficiency of this algorithm and at the same time producing a insightful classification rule, a criterion is defined to terminate the mining process earlier than the usual association rule mining algorithm.

The rest of this paper is organized as follows. Section 2 describes the classification problem using the mining association rule method, and the definition of information gain. Section 3 presents the two algorithms in detail. Section 4 shows some preliminary experiment results. Finally, section 6 contains our concludions.

2. PROBLEM STATEMENT

2.1 Association Rule Mining

Classification rules can be regarded as a special kind of association rules within a data set. The data set for classification usually is separated into two groups, one is for training and the other is for testing. Both of them contain a large amount of cases (tuples) consisting of many attributes and one class label. The task of the classification is to find a set of rules to determine an unseen case's class correctly.

A classification rule is the form of implication like $X \rightarrow C_i$, where X is the combination of attribute values, and C_i is one of g class labels. As for association rules, the rule's form is $X \rightarrow Y$, where X and Y are all combination of attribute values (corresponding to items in transactional database). So the difference is that C_i is a special kind of Y. According to this, we can modify the typical association rule's mining algorithm to mining classification rules directly and efficiently.

Let D be the data set stored in the database, which contains M cases for training and N cases for testing. Each case is described by r distinct attributes and one of g class labels. Let A be the set of r attributes: $A=\{a_1, a_2, \ldots a_r\}$, and G be the set of g class labels: $G=\{C_1, C_2, \ldots, C_g\}$. An item of D is of the form:

$$I_{ij}: a_i = v_j^i$$

where $a_i \in A$, v_j^i is a value of attribute a_i, i=1, 2, ..., r, j=1, 2,..., n_i.

Let I be the set of all items: $I=\{I_{ij} \mid i=1, 2, \ldots, r, j=1, 2, \ldots, n_i\}$, and $X \subseteq I$ be a subset of items. Then a candidate itemset is defined as:

<X, C_i> with two numbers: lcount and wcount

where lcount is the number of cases contain X and wcount is the number of cases containing X and labeled C_i. A case d containing X means $X \subseteq d$. Another concern of this itemset is its size which is defined as the number of items included in X. So if X contains k attribute values, it is called a k-itemset.

The support and confidence of this candidate itemset is defined as follows:

$$\text{support } (< X, C_i >) = \frac{\text{wcount}}{|D|}$$

$$\text{confidence } (< X, C_i >) = \frac{\text{wcount}}{\text{lcount}}$$

where |D| is the number of cases in database D. If its support is greater than a minimum support (called minsup) given by user, then this candidate itemset is a frequent itemset. Furthermore, if the confidence of this frequent itemset is greater than a minimum confidence (called minconf) given by user, this frequent itemset is then called a rule itemset and can be output as a rule:

$$X \rightarrow C_i$$

Otherwise, if the support of a candidate itemset is less than minsup, it is called an excluded itemset that can not be used to produce larger itemsets. Other candidate itemsets except for the excluded itemsets and rule itemsets can be used to produce larger itemsets. The reason why rule itemsets are excluded to produce larger itemsets is that if a rule $X \rightarrow C_i$ holds in D, then $X \cup Y \rightarrow C_i$ also holds, where Y is another subset of items. So one such rule can produce many rules like $X \cup Y \rightarrow C_i$ since Y can be any subset of I. However in fact the rule $X \rightarrow C_i$ is the most meaningful one among those rules.

2.2 Information Gain

The information gain criterion is one of those methods [12][13] used to select best split attributes in decision tree classifiers. We use it in our algorithm to help decide which training cases can be deleted from the database.

Suppose a attribute A has n distinct values that partition the data set T of training cases into subsets T_1, T_2, ..., T_n. For a data set $S \subseteq D$, freq(C_i,S) stands for the number of cases in S that belong to class C_i. |S| denotes the number of cases in data set S. Then info(S) is defined as follows to measure the average amount of information needed to identify the class of a case in S :.

$$\text{info}(S) = -\sum_{j=1}^{g} \frac{freq(C_j, S)}{|S|} \times \log_2\left(\frac{freq(C_j, S)}{|S|}\right)$$

where, g is the number of classes.

After data set T is partitioned in accordance with n values of attribute A, the expected information requirement can be defined as:

$$\text{info}_A(T) = -\sum_{i=1}^{n} \frac{|T_i|}{|T|} \times \text{info}(T_i)$$

The information gained by partitioning T according to attribute A is defines as:

$$\text{gain}(A) = \text{info}(T) - \text{info}_A(T)$$

Among all attributes in data set T, the best split attribute is the one that maximizes the information gain.

3. CLASSIFICATION ALGORITHMS

3.1 UARC: Association Rule Based Classification

UARC is a direct classification rule's mining algorithm, in which a new method is proposed to build candidate itemsets and to prevent both the excluded itemsets and rule itemsets from generating more meaningless candidate itemsets. In addition, a new criterion is defined to terminate the rule mining process earlier. The detail is given as follows.

During rule mining phase, after first scan of data set, a record class list called recClass is created initially to contain the class label attached to each case. The ith entry of the class list corresponds to the ith case in the training database. In the next scan of the database, once a rule is produced, each entry for the cases containing this rule will be set to a value (assume 0) different from every class label. Using this list, we can define a criterion to end the mining process. That is to say, after each scan of database, if each entry of recClass is the same (or almost the same, say 90%, which can be given by users according to the accuracy required and noise contained in this dataset), then stop further scan of database. If the value of each entry rather than 0 is one of the class labels, then a default rule can be produced.

Additionally, after each scan of database, only rule itemsets, excluded itemsets and information about them such as lcount, wcount and the corresponding record numbers will be saved. Other candidate itemsets will be cleared from candidate itemsets list before the next scan of the database starts. During the next scan of the database, new candidate itemsets that do not contain any of excluded itemsets and rule itemsets will be generated for every cases. Figure 1 gives the training process of UARC.

```
1.  initclass(recClass);
2.  rule={1-itemset c| support(c)>=minsup and confidence(c)>=
        minconf};
3.  excluded={1-itemset c| support(c)<minsup};
4.  adjust(recClass);
5.  for(k=2; finish(recClass)==0 && k<=r; k++)
6.    initcand(cand); currentc=0;
7.    for(t=1;t<=M; t++)
8.      Ck=gencand(rule,exclude);
9.      for each itemset c∈Ck
10.       addtocand(c,cand);
11.     end
12.   end
13.   R={c∈cand| support(c)>=minsup and confidence(c)>=
        minconf};
14.   rule=rule∪R;
15.   excluded={ c∈cand | support(c)<minsup};
16.   adjust(recClass);
17. end
18. sort(rule);
```

19. default(rule);
20. output(rule);

Figure 1 Training process of algorithm UARC.

Lines 1 to 4 present the first scan of the database. It initializes the value of recClass to be the class label of each case and produce all 1-itemsets from which rule itemsets and excluded itemsets are selected. According to each rule, its corresponding cases' class label in recClass will be set to zero (line 4).

Lines 5 to 17 are what should be done in subsequent scan of database. During each scan, for each case in D, several itemsets will be generated according to rule itemsets and excluded itemsets (line 8). Each of these itemsets will be added to candidate itemset (line 10) and the corresponding lcount, wcount and each record No. (recno) are saved in structure variable named cand. Then from candidate itemsets cand, rule itemsets and excluded itemsets are selected, and recClass is adjusted. After each pass, finish(recClass) check if every entry (or almost) of recClass has the same value, if so, next scan of database stop. If scan terminated, all of itemsets contained in rule will be sorted according to its confidence and support (line 18). Then produce a default rule according to recClass (line 19), and finally all of rules are output (line 20).

3.2 GARC: Gain Based Association Rule Classification

When using algorithm UARC, in each pass of the algorithm every case is read from the database, while in fact some cases can be deleted from the database permanently. However, which cases can be deleted is a problem of concern that is dealt with upon the information gain criterion. After the first scan of the database, all of candidate itemsets is saved to variable named cand. Using each candidate itemset's lcount and wcount, information gain for each attribute A that can be used to partition the database D into n datasets can be calculated as follows:

$$\text{info}_A(T) = -\sum_{i=1}^{n} \frac{|T_i|}{|T|} \times \text{info}(T_i)$$

$$= -\sum_{i=1}^{n} \frac{|T_i|}{|D|} \times \left(\sum_{k=1}^{g} \frac{freq(C_j,T_i)}{|T_i|} \times \log_2 \frac{freq(C_j,T_i)}{|T_i|}\right)$$

$$= -\sum_{i=1}^{n} \text{support}\ (A = v_i) \times \left(\sum_{k=1}^{g} \text{confidence}\ (i_k) \times \log_2 \text{confidence}\ (i_k)\right)$$

where T_i corresponds to the data set whose attribute A's valve equals v_i, g is the numbers of classes, i_k stands for itemset <A= v_i, C_k>.

With the information gain criterion, a best split attribute (called bestattr) can be selected after first scan of database. Then during the next scan of database, we can delete those cases that produce rules containing attribute specified by bestattr in the previous scan. This is similar to the tree building process: if a rule that contains attribute bestattr is produced, there is no need to deal with its corresponding cases. In addition, there is a little change in the candidate itemset generating method. For each case, we can not produce all of its candidate itemsets and can only produce itemsets include bestattr. The reason is that for the itemsets without attribute bestattr, the lcount and wcount values can not be counted correctly due to the deletion of some cases which contribute to the count of lcount and wcount. For itemsets including attribute bestattr, the values of the attribute bestattr included in itemsets is different from the values contained in datasets that have been deleted, so we can still count these itemsets correctly. This algorithm is shown in figure 2.

1. initclass(recClass);
2. rule={1-itemset c|support(c)>=minsup and confidence(c)>= minconf };
3. excluded={1-itemset c| support(c)<minsup};
4. adjust(recClass);
5. bestattr=gain(cand);
6. getdeleted(delrec, bestattr);
7. for(k=2; finish(recClass)==0 && k<=r; k++)
8. initcand(cand); currentc=0;
9. for(t=1;t<=M; t++)
10. if (t∈delrec) delete t from database;
11. else
12. C_k= gencand(rule, exclude, bestattr);
13. for each itemset c∈C_k
14. addtocand(c,cand);
15. end
16. end
17. R={c∈cand| support(c)>=minsup and confidence(c)>= minconf};
18. rule=rule∪R;
19. excluded={ c∈cand | support(c)<minsup};
20. adjust(recClass);
21. end
22. sort(rule);
23. default(rule);
24. output(rule);

Figure 2 Training process of algorithm GARC.

In this algorithm, lines 5, 6, 10, 12 are different from previous algorithm UARC. gain(cand) is to select bestattr based on

information gain criterion. getdeleted(delrec, bestattr) is to get the record number (will be saved in delrec) that can be deleted according to attribute bestattr. Line 10 is to delete cases included in delrec. Gencand(rule, exclude) differs from that in UARC as stated above.

4. EXPERIMENTAL RESULTS

This section shows an empirical performance evaluation of UARC and GARC. The experiment was divided into two parts. The first part compares UARC, GARC with the typical decision tree classifier C4.5 in terms of accuracy and the number of rules. The second part examines the execution time and scalability of algorithms UARC and GARC.

In order to accomplish the two parts of evaluation, the data set used is the synthetic database proposed in [14]. Each tuple in database has 9 attributes. Ten classification functions were used to produce data distributions of varying complexities in this paper. We use first 5 functions to produce 5 data sets named pred1~pred5, among which pred5 corresponding function 5 is one of the hardest to characterize and results in the highest classification errors.

4.1 Accuracy and Number of Rules

We use a small data set (200 cases) for every function to present the accuracy and number of rules of three algorithms. Since C5.0 is the new version of C4.5 and much more accurate and faster than C4.5, in the experiment we compared our algorithms with C5.0. Table 1 shows the classification accuracy of different algorithms, while table 2 shows the number of rules produced by every algorithm.

Table1 Accuracy of different algorithms

Data Set	C5.0	UARC	GARC
pred1	0.000	0.000	0.000
pred2	0.000	0.000	0.000
pred3	0.060	0.015	0.015
pred4	0.080	0.030	0.060
pred5	0.150	0.115	0.140

Table2 Number of rules produced by different algorithms

Data Set	C5.0	UARC	GARC
pred1	14	16	16
pred2	15	21	15
pred3	20	30	30
pred4	23	30	26
pred5	29	76	38

In the experiment, for a categorical attribute, all its possible values are mapped to a set of consecutive positive integers. For a continuous attribute, its value range is discretized into intervals and each interval is also mapped to a set of consecutive positive integers.

It can be seen from table 1 that both UARC and GARC are more accurate than C5.0 for every data set. Table 2 shows that the number of rules produced by UARC or GARC is more than that produced by C5.0, while those rules that are not included in the result of C5.0 are useful to improve classification accuracy and are just what C5.0 can not obtain. For example, for dataset pred4, one of rules found by both UARC and GARC is:

IF commission≥40000 and commission<50000 THEN group=1

This rule is useful and accurate according to function 4, but C5.0 could not found it.

Table 2 also indicates that UARC generates more rules than GARC for some datasets. The reason is that the set of candidate itemsets becomes small due to the deletion of some cases, which also lead to the lower accuracy of GARC than UARC.

4.2 Speed And Scalability

In this part, we use different size of data set of function 3 to examine the speed and scalability of our two algorithms along the number of training cases. Figure 3 illustrates the execution time used by UARC and GARC as the number of training cases increased from 200 to 100000.

As we can see from figure 3, compared to UARC algorithm, GARC is faster, and the difference between UARC and GARC becomes larger as the number of cases increases. The result also indicates that both two algorithms achieve near-linear execution time on disk resident data.

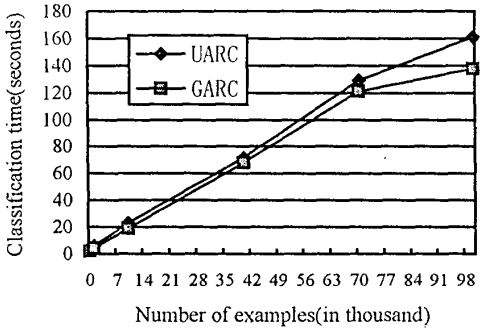

Figure 3 Speed and scalability

5. CONCLUSION

Both classification rule mining and association rule mining are important problems in the field of data mining. Many kinds of classification algorithms have been proposed in the past and in recent years. But each of them has its drawbacks. In this paper, two algorithms are designed to use the basic concept of association rules to find classification rules directly and efficiently. From the empirical evaluation we can see that both two algorithms achieve better classification accuracy and produce complete, useful sets of rules. It is also shown that these two algorithms both have a good scalability, and GARC has a better execution speed than UARC.

ACKNOWLEDGEMENT

This work was partly supported by National Science Foundation of China (grant No. 69674037 and grant No. 79825102) and National Defense Foundation of China.

REFERENCES

1. G. Piatetsky-Shapiro, U. Fayyad, P. Smyth, *From data mining to knowledge discovery: An overview. Advances in Knowledge Discovery and Data Mining*, AAAI/MIT press, 1996,pp.1~35.
2. J. R. Quinlan, *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993.
3. Manish Mehta, Rakesh Agrawal and Jorma Rissanen, *SLIQ: A Fast Scalable Classifier for Data Mining*, Extending database technology, 1996, pp. 18-32.
4. J. Catlett. *Megainduction: Machine Learning on Very Large Databases*. PhD thesis, University of Sydney, 1991.
5. *K. Alsabti, S. Ranka, V. Singh, CLOUDS: A decision Tree classifier for Large Data sets*. In Proc. Of the Fourth Int. Conference on Knowledge Discovery & Data Mining. New York, New York, August 27-31, 1998, pp. 2-8.
6. Bing Liu, Wynne Hsu and Yiming Ma. *Integrity Classification and Association Rule Mining*, In Proc. Of the Fourth Int. Conference on Knowledge Discovery & Data Mining. New York, New York, August 27-31, 1998, page 80-86.
7. R. Agrawal, T. Imielinski, and A. Swami, *Mining association rules between sets of items in large databases.* In: Proceeding of 1993 ACM-SIGMOD Int. Conf. On Management of Data, Washington, D.C., 1993, pp. 207~216.
8. R. Agrawal, R. Srikant. *Fast algorithm for mining association rules*. In Proc. Of the 20th VLDB Conference, Santiago, Chile, 1994, pp. 487~499.
9. J. S. Park, M. S, Chen, and P.S. Yu. *An Effective Hash-based Algorithm for Mining Association rules*. In: Proceedings of ACM-SIGMOD Int. Conf. On Management of Data, Baltimore, MD, 1995, pp. 175~186.
10. H. Toivonen, *Sampling Large Databases for Association Rules*. In 22th International Conference on VLDB, Bombay, India. September 1996, pp. 134~145.
11. S. Brin, R. Motwani, J. Ullman, et al. *Dynamic itemset counting and implication rules for market basket data*. In Proc. Of 1997 ACM-SIGMOD Int. Conf. On Management of Data, Tucson, Arizona, 1997, pp. 255~264.
12. S. M. Weiss and C. A. Kulikowski. *Computer Systems that Learn: Classification and prediction Methods from Statistics, Neural Nets, Machine Learning, and Expert Systems*. Morgan Kaufman, 1991.
13. L. Breiman et. al. *Classification and Regression Trees*. Wadsworth, Belmont, 1984.
14. Rakesh Agrawal, Tomasz Imielinski, Arun Swami, *Database Mining: A Performance Perspective*, IEEE transaction on knowledge and data engineering, December 1993, 5(6), pp. 914~925.