OCARA AS METHOD OF CLASSIFICATION AND ASSOCIATION RULES OBTAINING

TURIHO Jean Claude

Independent Institute of Lay Adventists of Kigali (INILAK), KIGALI, RWANDA

Email: tulije@gmail.com, jturiho@inilak.ac.rw

ABSTRACT

This article focuses on how to integrate the two major data mining techniques namely, classification and association rules to come up with An Optimal Class Association Rule Algorithm (OCARA) as Method of Classification and Association Rules Obtaining. Classification and association rule mining algorithms are two important aspects of data mining. Classification association rule mining algorithm is a promising approach for it involves the use of association rule mining algorithm to discover classification rules. OCARA inherits the strength of Classification and association rule mining algorithms. Because of this reason, OCARA is a powerful algorithm when compared to either Classification or Association rule mining algorithms.

The reason for OCARA's high accuracy is because of optimal association rule mining algorithm and the rule set is sorted by priority of rules resulting into a more accurate classifier. Therefore, we can confidently say OCARA is an accurate classifier and has better performance and is more efficient when compared with C4.5, CBA, and RMR algorithm.

Keywords: class association rule; association rule; classification; data mining

I INTRODUCTION

Data mining can be described in a lot of different ways. One of the most common definitions is the following, as stated by Frawley et al Another way of explaining data mining is done in Holsheimer: Data mining is the search for relationships and global patterns that exist in large databases, but are 'hidden' among the vast amounts of data, such as a relationship between patient data and their medical diagnosis.

Data mining could also be described as trying to create a simplified model of the complex world described in the database. We may therefore say that data mining is a way of dealing with large amounts of information, and it is helpful

for finding useful information faster than any human. Techniques for data mining find their way into more and more different areas.

Classification and association rule mining algorithms are two important aspects of data mining. Classification association rule mining algorithm is a promising approach for it involves the use of association rule mining algorithm to discover classification rules.

OCARA inherits the strength of Classification and association rule mining algorithms. Because of this reason, OCARA is a powerful algorithm when compared to either Classification or Association rule mining algorithms.

To verify the strength of OCARA, we conducted experiment using eight different data sets of UCI (University of California at Irvine). We compared OCARA with other three popular algorithms (C4.5, CBA, RMR). The end result proved that the support threshold was greatly influenced by the rule accuracy and the rule number. If the support threshold is between 0.02 and 0.03, the accuracy will be much better, as discussed in this article. The support threshold was set as 0.02, and the confidence was set as 0.30 in this article. Therefore, OCARA proved to more efficient when compared with others making it more robust in terms of its accuracy.

II TYPES OF ALGORITHMS USED FOR COMPARISON IN THIS ARTICLE

This article summarizes three algorithms C4.5, CBA and RMR to synthesizing decision trees that has been used in a variety of systems, and it describes new one, OCARA, in detail. Now let have a look for the 3 former algorithms.

2.1 C4.5 Algorithm

C4.5 algorithm is one of two most commonly used systems for induction of decision trees for classification. C4.5 builds decision trees from a set of training data in the same way as ID3, using the concept of information entropy. Training data is a set $S = s_1, s_2,...$ of already classified samples. Each sample $si = x_1, x_2,...$ is a vector where $x_1, x_2,...$ represent attributes or features of the sample. The training data is augmented with a vector $C = c_1, c_2,...$ where $c_1, c_2,...$ represent the class to which each sample belongs. At each node of the tree, C4.5 chooses one attribute of the data that most effectively splits its set of samples into subsets enriched in one class or the other. Its criterion is the normalized information gain (difference in entropy) that results from choosing an attribute for splitting the data. The attribute with the highest normalized information gain is chosen to make the decision. The C4.5 algorithm then recurses on the smaller sublists.

This algorithm has a few base cases.

All the samples in the list belong to the same class. When this happens, it simply creates a leaf node for the decision tree saying to choose that class.

None of the features provide any information gain. In this case, C4.5 creates a decision node higher up the tree using the expected value of the class.

Instance of previously-unseen class encountered. Again, C4.5 creates a decision node higher up the tree using the expected value.

In general, steps in C4.5 algorithm to build decision tree are [48]:

- Choose attribute for root node;

- Create branch for each value of that attribute;

- Split cases according to branches;

- Repeat process for each branch until all cases in the branch have the same class.

Choosing which attribute to be a root is based on highest gain of each attribute [30].

A more interesting problem is that of overfitting. A decision tree that correctly classifies every example in a training set might not be as good a classifier as a smaller tree that does not fit all the training data. In order to avoid this problem, most decision tree algorithms employ a "pruning" method, which means that they grow a large tree and then delete some portion of it. An alternative method is to stop growing the tree once the training set has been sufficiently subdivided (using a "stopping" criterion). That "stopping" criterion has been experimented in past by Quinlan. But Quinlan has adopted the pruning approach for C4.5.

For Quinlan, C4.5's pruning method is based on estimating the error rate of every subtree, and replacing the subtree with a leaf node if the estimated error of the leaf is lower. The idea is as follows: suppose that one could estimate error rate of any node in a decision tree, including leaf nodes. Beginning at the bottom of the tree, if the estimates indicate that the tree will be more accurate when the children of node *n* are deleted and *n* is made a leaf node, then C4.5 will delete *n's* children. If the estimates were perfect, this would always lead to a better decision tree. In practice, although these estimates are very coarse, the method often works quite well. Quinlan illustrates his pruning method with a detailed example from the Congressional voting domain. Before adopting any pruning method though, the reader should look further into literature, since the efficacy of a pruning method varies in different domains [8].

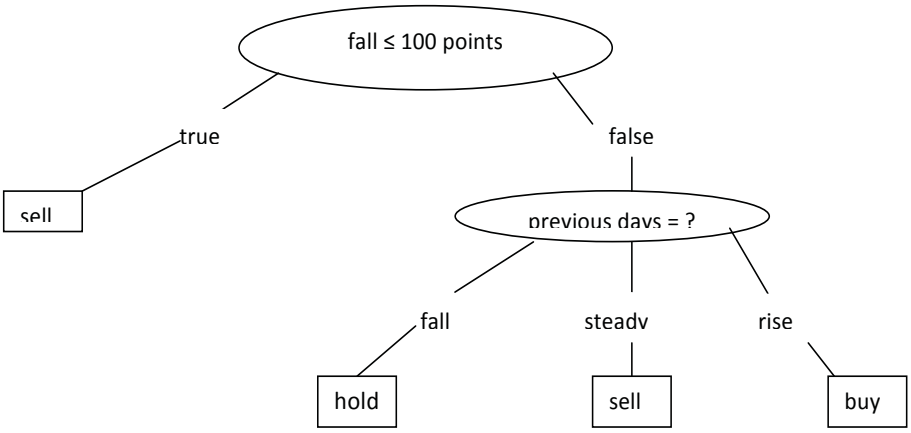

Figure 2.1 A simple decision tree

Input and Output

Input to C4.5 consists of a collection of training cases, each having a tuple of values for a fixed set of attributes (or independent variables) $A = \{A_1, A_2,\ldots, A_k\}$ and a class attribute (or dependent variable). An attribute A_a is described as continuous or discrete according to whether its values are numeric or nominal. The class attribute C is discrete and has values $C_1; C_2; \ldots; C_x$.

The goal is to learn from the training cases a function that maps from the attribute values to a predicted class.

DOM (A1) X DOM (A2) X … X DOM (Ak) -> DOM (C) (2.1)

The distinguishing characteristic of learning systems is the form in which this function is expressed. We focus here on decision trees, a recursive structure that is

- ✓ a leaf node labelled with a class value, or
- ✓ a test node that has two or more outcomes, each linked to a subtree.

To classify a case using a decision tree, imagine a marker that is initially at the top (root) of the tree. If the marker is at a leaf, the label associated with that leaf becomes the predicted class. If the marker is at a test node, the outcome of that test is determined and the marker moved to the top of the subtree for that outcome.

Divide and Conquer

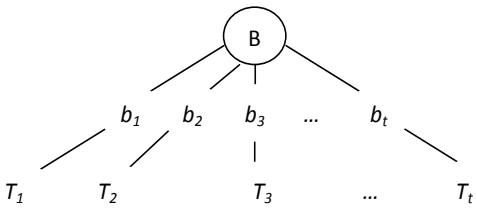

Figure 2.2. The decision tree where Ti is the result of growing a decision tree for the cases in *Si*.

Decision tree learners use a method known as divide and conquer to construct a

Suitable tree from a training set S of cases:

- If all the cases in S belong to the same class (C_j, say), the decision tree is a leaf labelled with C_j.
- Otherwise, let B be some test with outcomes b_1, b_2, ..., b_t that produces a non-trivial partition of S, and denote by S_i the set of cases in S that has outcome
  b_i of B. The decision tree is

Candidate Tests

C4.5 uses tests of three types, each involving only a single attribute A_a. Decision regions in the instance space are thus bounded by hyperplanes, each orthogonal to one of the attribute axes.

- If A_a is a discrete attribute with z values, possible tests are:
  - "A_a = ?" with z outcomes, one for each value of A_a. (This is the default.)
  - "$A_a \in G$?" with $2 \leq g \leq z$ outcomes, where $G = \{G_1;G_2;\ldots;G_g\}$ is a partition of the values of attribute A_a. Tests of this kind are found by a

    greedy search for a partition G that maximizes the value of the splitting criterion (discussed below).
- If A_a has numeric values, the form of the test is "$A_a \leq \theta$ "with outcomes *true* and *false*, where θ is a constant threshold. Possible values of θ are found by sorting the distinct values of A_a that appear in S, then identifying one threshold between each pair of adjacent values. (So, if the cases in S have d distinct values for A_a, d-1 thresholds are considered.)

Selecting Tests

In the divide and conquer algorithm, any test B that partitions S non-trivially will lead to a decision tree, but different Bs give different trees. Most learning systems attempt to keep the tree as small as possible because smaller trees are more easily understood and, by Occam's Razor arguments, are likely to have higher predictive accuracy [39]. Since it is infeasible to guarantee the minimality of the tree [21], C4.5 relies on greedy search, selecting the candidate test that maximizes a heuristic *splitting criterion*.

Two such criteria are used in C4.5, *information gain* and *gain ratio*. Let $RF(C_j; S)$ denote the relative frequency of cases in S that belong to class C_j. The information content of a message that identifies the class of a case in S is then

$$I(S) = -\sum_{j=1}^{x} RF(C_j,S)\log(RF(C_j,S))$$

(2.2)

After S is partitioned into subsets S_1, S_2,..., S_t by a test B, the information gained is then

$$G(S,B) = I(S) - \sum_{i=1}^{t} \frac{|S_i|}{|S|} I(S_i) \quad (2.3)$$

The gain criterion chooses the test B that maximizes $G(S;B)$.

A problem with this criterion is that it favors tests with numerous outcomes for example, $G(S;B)$ is maximized by a test in which each S_i contains a single case. The gain ratio criterion sidesteps this problem by also taking into account the potential information from the partition itself:

$$P(S,B) = -\sum_{i=1}^{t} \frac{|S_i|}{|S|} \log\left(\frac{|S_i|}{|S|}\right) \quad (2.4)$$

Gain ratio then chooses, from among the tests with at least average gain, the test B that maximizes $G(S;B)$ / $P(S;B)$.

Missing Values

Missing attribute values are a common occurrence in data, either through errors made when the values were recorded or because they were judged irrelevant to the particular case. Such lacunae affect both the way that a decision tree is constructed and its use to classify a new case.

When a decision trees is constructed from a set S of training cases, the divide and conquer algorithm selects a test on which to partition S. Let B be a potential test based on attribute A_a with outcomes $b_1, b_2,..., b_t$. Denote by S_0 the subset of cases in S whose values of A_a are unknown, and hence whose outcome of B cannot be determined. As before, let S_i denote those cases with (known) outcome b_i of B. The information gained by B is reduced because we learn nothing about the cases in S_0; so we got a new equation

$$G(S,B) = \frac{|S - S_0|}{|S|} G(S - S_0, B) \quad (2.5)$$

The split information is increased to reflect the additional "outcome" of the test (namely, the fact that it cannot be determined for the cases in S_0). Equation 2 is modified to

$$P(S,B) = -\frac{|S_0|}{|S|} \log\left(\frac{|S_0|}{|S|}\right) - \sum_{i=1}^{t} \frac{|S_i|}{|S|} \log\left(\frac{|S_i|}{|S|}\right) \quad (2.6)$$

Both changes have the effect of reducing the desirability of tests involving attributes with a high proportion of missing values.

When a test B has been chosen, C4.5 does not build a separate decision tree for the cases in S_0. Instead, they are notionally fragmented into fractional cases and added to the subsets corresponding to known outcomes. The cases in S_0 are added to each S_i with weight $|S_i| / |S - S_0|$.

Missing attribute values also complicate the use of the decision tree to classify a case. Instead of a single class, the initial result of the classification is a class probability distribution determined as follows: Let CP $(T; Y)$ denote the result of classifying case Y with decision tree T.

- If T is a leaf, CP (T; Y) is the relative frequency of training cases that reach T.
- If T is a tree whose root is test B, and the outcome of B on case Y is known (b_i, say), then

$$CP(T, Y) = CP(T_i, Y) \quad (2.7)$$

where, as before, T_i is the decision tree for outcome b_i.

- Otherwise, all outcomes of B are explored and combined probabilistically, giving

$$CP(T, Y) = \sum_{i=1}^{t} \frac{|S_i|}{|S - S_0|} CP(T_i, Y) \quad (2.8)$$

Note that the weight of the subtree T_i depends on the proportion of training cases that have outcome b_i, interpreted as the prior probability of that outcome.

When the class probability distribution resulting from classifying case Y with the decision tree has been determined, the class with the highest probability is chosen as the predicted class.

Avoiding Overfitting

The divide and conquer algorithm partitions the data until every leaf contains cases of a single class, or until further partitioning is impossible because two cases have the same values for each attribute but belong to different classes. Consequently, if there are no conflicting cases, the decision tree will correctly classify all training cases. This so-called overfitting is generally thought to lead to a loss of predictive accuracy in most applications [39].

Overfitting can be avoided by a *stopping criterion* that prevents some sets of training cases from being subdivided (usually on the basis of a statistical test of the significance of the best test), or by removing some of the structure of the decision tree after it has been produced. Most authors agree that the latter is preferable since it allows potential interactions among attributes to be explored before deciding whether the result is worth keeping.

Pruning Decision Trees

After a decision tree is produced by the divide and conquer algorithm, C4.5 prunes it in a single bottom-up pass. Let T be a non-leaf decision tree, produced from a training set S, of the form

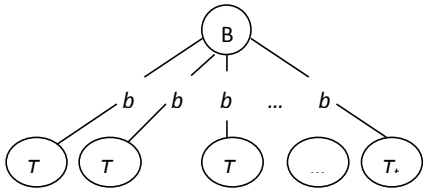

Figure 2.3 Pruning decision tree

where each T_i^* has already been pruned. Further, let T_j^* be the subtree corresponding to the most frequent outcome of B, and let L be a leaf labelled with the most frequent class in S. Let the number of cases in S misclassified by T, T_j^*, and L be E_T, $E_{T_j^*}$ and E_L respectively. C4.5's tree pruning

algorithm considers the three corresponding estimated error rates

- $U_{CF}(E_T, |S|)$,
- $U_{CF}(E_L, |S|)$, and
- $U_{CF}(E_{T_j^*}, |S|)$.

Depending on whichever is lower, C4.5

- leaves T unchanged;
- replaces T by the leaf L; or
- replaces T by its subtree $E_{T_j^*}$.

This form of pruning is computationally efficient and gives quite reasonable results in most applications [30].

2.2 CBA Algorithm

Classification Based on Associations (CBA) is one of classic AC algorithms based on the step-wise a priori algorithm of association rule discovery [2], which discovers frequent itemsets by making multiple passes over the training data set. In the first scan, it counts the support of the one-itemsets and determines whether or not they are frequent, then in each subsequent pass, starting with itemsets found to be frequent in the previous pass produces new candidate itemsets. The generation of rules is straightforward once the frequent itemsets and their support are known. However, the problem of finding frequent itemsets is a harder problem that requires extensive computation capacity especially when the expected number of candidate itemsets is large [14].

Input

Table form dataset (transformed needed) or transaction form dataset.

Output

A complete set of CARs. (Class Association Rule) –done by CBA-RG (rule generator)

A classifier. –done by CBA-CB (classifier builder) [33]

```
1    F1 = {large l - rulesitems};
2    CAR1 = genRules(F1);
3    prCAR1 = pruneRules(CAR1);
4    for ( k = 2; Fk-1 ≠ Ø; k++ ) do
5        Ck = CandidateGen(Fk-1)
6        for each data case d∈D do
7            Cj = ruleSubset(Ck,;d);
8            for each candidate c∈Cj do
9                c.condsupCount++;
10               if d.class = c.class then c.condsupCount++
11           end
12       end
13       Fk ={c ∈Ck | c.condsupCount ≥ minsup};
14       CARk = genRules(Fk);
```

15 $prCAR_k$ = pruneRules(CAR_k);
16 end
17 $CARs = \cup_k CAR_k$
18 $prCARs = \cup_k prCAR_k$

Figure 2.4 The CBA-RG algorithm

1 R = sort(R);
2 **for each rule** $r \in R$ **in sequence do**
3 **temp** = $\emptyset$;
4 for each case $d \in$ D do
5 if satisfies the conditions of r then
6 store d id in *temp* and mark r if it correctly classifies d;
7 if r is marked then
8 insert r at end of C;
9 delete all the cases with ids in *temp* from D;
10 selecting a default class for the current C;
11 Compute the total number of errors of C;
12 end
13 end
14 Find the first p in C with the lowest total number of errors and drop all the rules after p in C;
15 Add the default class associated with p to end of C, and return C (our classifier).

Figure 2.5 The CBA-CB algorithm

2.3 RMR Algorithm

Ranked Multilabel Rule (RMR) algorithm [4] is the one of three popular classification techniques used in this thesis for making a comparison to OCARA in terms of prediction accuracy. RMR generates rules with multiple labels. As CBA algorithm, RMR is also AC algorithm. RMR algorithm uses the fittest class associated with a rule in the prediction step rather than the largest frequency class learned for that rule, is presented. The fittest class is assigned to each rule by a pruning heuristic, which ensures training data objects cannot be shared by rules. This pruning heuristic discards the training data objects for each selected rule and these discarded objects become unavailable for the remaining lower ranked candidate rules. This means even though a training object has been used in learning multiple rules, the pruning method proposed ensures that the final rules in the classifier do not share training objects similar to rule induction approaches. The pruning heuristic may change the class for some

of the rules due to removing the overlapping among rules training data objects.

RMR algorithm has three main phases: rules discovery, repeated learning and classification. In the first phase, RMR scans the training data to discover the potential rules. During rules discovery, RMR utilizes a pruning heuristic that ensures the orders of the classes in the potential rules that share-training objects are updated to minimize rules redundancy. In addition, each evaluated rule's training data objects are removed from the training data. In the second phase, the algorithm proceeds to discover more rules that pass the minsupp and minconf thresholds from the remaining unclassified objects, until no further frequent ruleitems can be found. At that stage, the rules sets derived during each iteration will be merged to form a multi-label classifier. In the third phase, the multi-label classifier is evaluated against a test data set.

In the following lines we present RMR algorithm:

Input: training data *T*, *minsupp* and *minconfthresholds*.

Output: A classifier *R* and a list *Tp* of covered rows.

A rule *r* in *R* has the following properties: *Items*, *class*, *rowIDs* (*tid-list*).

Step 1: Rules Discovery

Scan T for the set S_1 of frequent one-itemset

$R \leftarrow S_1$

$i \leftarrow 1$

while ($S_i \neq 0$)

{ $S_{i+1} \leftarrow$ produce (S_i)

$R \leftarrow R \cup S_1$

$i \leftarrow i+1$}

Step 2: Evaluation step (Pruning) for each candidate rules produced during an iteration

$R' = \{r_1\}$

$T_p = \{r_1.rowIDs\}$

For i = 2 to |R|

$Temp = (r_i.rowIDs - T_p)$

if ($|Temp|>0$)

Update r_i by considering only rowIDs which are in *Temp*

$R' = R' \cup r_i$

$T_p = T_p \cup Temp$

end if

end for

return R', T_p

Step 3: Multi-label rules

Do

i←1

$T' \leftarrow T$

R ←Φ

$T' \leftarrow T' - T_p$

(R', T_p) = *RMRScan* (T', *MinSupp*, *MinConf*) //Repeat step 1 on R' until no further
frequent ruleitem is found

$R \leftarrow R \cup R'$

While ($R' \neq 0$)

Return R

Figure 2.6 The RMR algorithm.

The next table shows the classification accuracy between C4.5, CBA and RMR algorithms.

Table 2.1 Classification accuracy of C4.5, CBA and RMR algorithms

Data-set	Size	ACCURACY (%)		
		C 4.5	CBA	RMR
Glass	214	66.82	69.89	**71.07**
Iris	150	96.00	93.25	93.87
Tic-tac	958	83.71	100.00	100.00
Breast	699	94.66	98.84	95.92
Zoo	101	93.06	95.96	95.15
Heart	270	86.67	83.32	84.74
Diabetes	768	85.82	75.34	78.79
Pima	768	72.78	75.49	78.05
Vote	435	88.27	86.91	88.70
Balance-scale	625	64.32	67.24	77.00
Average		83.21	84.62	86.32

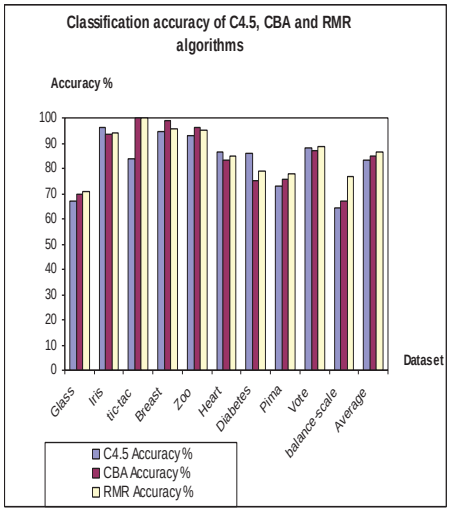

Figure 2.7 Histogram Comparison of. Classification accuracy of C4.5, CBA and RMR algorithms

On average, the RMR algorithm achieved +3.11% and +1.70% above C4.5 and CBA learning methods, respectively.

Thus, RMR generates rules that do not overlap in their training objects and ensures that a training instance is allowed to be covered by only a single rule, solving problems inherited from association rule discovery in AC. Other associative methods apply the largest frequency class associated with the rules in the training data in the prediction step, whereas RMR removes the training objects for each evaluated rule from any lower ranked rules that share these objects with it. This removal results in updating the rank of class labels associated with some of the impacted rules in the classifier.

III BASIC CONCEPTS AND THEORY

3.1 Introduction

Classification association rules mining algorithms have to pre-specify the classification attributes. Assuming

$U = (C, D)$, where U is the training dataset, C is the testing dataset, and D is the classification attributes. The definitions and the theorems are used to prune rules in the OCARA as follows:

3.2 Concepts and Theorems

Definition 1. If the support and interestingness of a rule are not less than given thresholds, this rule is called a strong implication.

Definition 2. Given two rules $P \rightarrow d$ and $Q \rightarrow d$ ($d \in D$), if $P \subset Q$, $Q \subseteq C$, then Q is regarded as more specific than P, and P is regarded as more general than Q.

Definition 3. A rule set is optimal with respect to an interestingness metric if it contains all rules except those with no greater interestingness than one of its more general rules.

Theorem 1. If supp($PX\emptyset d$) = supp($P\emptyset d$), then rule $PX \rightarrow d$ and all its more specific rules will not occur in an optimal rule set defined by any interestingness.

Corollary 1. If supp(P) = supp(PX), then rule $PX \rightarrow d$ for any d and all its more specific rules do not occur in an optimal rule set defined by any interestingness.

Theorem 2. An optimal rule set is a subset of a nonredundant rule set.

Corollary 2. If supp($PX\emptyset d$) = 0, then all more-specific rules of the rule $P \rightarrow d$ do not occur in an optimal rule set defined by any interestingness.

In above theorems and corollaries, $P \subseteq C$, $X \subseteq C$, $d \in D$, "PX" is "$P \cup X$", supp ($P\emptyset d$) is support of $P\emptyset d$, and $\emptyset d$ is the negative of d. Since these theorems and corollaries are similar to the theorems in *ORD*, the proof can refer to paper [26].

Definition 4. A pair of pattern and target set is called a rule set of candidates, denoted by (P, Y) where $P \subseteq C$ and $Y \subseteq D$.

If a class is simply removed from the target set, a candidate rule is removed. If the target set is empty, the candidate stands for no rules. The existence of a candidate relies on two conditions: 1) *Pattern P* is frequent, and 2) *target set Y* is not empty. In addition, if we remove d from (P, D), we may lose rule $P \rightarrow d$ according to Corollary 2, termination statue of target d is defined as follows:

Definition 5. Target $d \in D$ is terminated in candidate (P, D) if supp ($P \neg d$) = 0.

IV OCARA Algorithm

4.1 Introduction

OCARA algorithm (Optimal Class Association Rule Algorithm), according to the definitions and theorems given in previous point, prunes the rules（reference: Prune (l+*1*). The OCARA is described as follows: OCARA algorithm:

Input. training data set U, testing data set T, supportσ and confidenceθ

Output. classifier, accuracy *accut*

The first stage.

At this stage, OCARA obtains the optimal rules set by running *MORS* algorithm as below.

1. Scanning the training data set and then finding out frequency itemsets.
2. Pruning itemsets, then discovering the optimal rules set *ORS*. (The main function for pruning is Prune (l+*1*))

The second stage.

1. Sorting the optimal rules set, then building the classifier. (According to:
   confidence, support, etc)

The third stage.

Using the classifier to classify the training data set, and computing accuracy.

4.2 Discovering the Optimal Rules Set

Here, the MORS algorithm is given. For MORS, confidence is selected as interestingness metric. In addition, the support is local support in this algorithm. The candidate generation and pruning function of MORS are given.

Function. Candidate_gen

1) for each pair of candidates ($P_{l\text{-}1}{}^s$, D_s) and ($P_{l\text{-}1}{}^t$,D_t) in an l-candidate set.

{

insert candidate (P_{l+1} ,D) in the (l+*1*)-candidate set

where $P_{l+1} = P_{l\text{-}1}{}^{st}$ and $D = D_s \cap D_t$

for all $P_l \subset P_{l+1}$

{

if candidate (P_l ,D_l) does not exist

then remove candidate (P_{l+1}, D) and return

else $D = D \cap D_t$

}

}

2) if the target set of ($P_{l+1,}$ D) is empty

then remove the candidate

The Candidate_gen function has a pruning process. The more pruning is given according to Theorem 1 and its corollary. σ is the minimum support in the following function.

Function. Prune ($l+1$)

for each candidate (P, D) in ($l+1$)-candidate set

{

1) for each $d \in D$

{

if supp (Pd) / supp (d) $\leq$ σ then remove d from D //test the frequency

else if supp ($P\emptyset d$) = 0 then mark d terminated

}

2) if D is empty then remove candidate (P, D) and return

3) for each l-level subset $P' \subset P$

{

if supp (P) = supp (P') then empty D //test Corollary 1

else if ($P\emptyset d$) =supp ($P'\emptyset d$) then //test Theorem 1

remove d from D

}

4) if D is empty then remove candidate (P, D)

}

After describing the Candidate_gen function and pruning function, the MORS is given as follows:

Name. MORS

Input. S, the minimum local support σ and the minimum interestingness θ, condition attributes set C and classification attributes set D in U

Output. An optimal rule set ORS defined by σ

Function.

1 let $ORS = \Phi$

2 count support of one-pattern by array

3 build one-candidate sets

4 form and add rules to ORS with σ and θ

5 generate two-candidate sets

6 while new candidate set is not empty

{

count support of patterns for new candidates

prune candidates in the new candidate set

form rules and add optimal rules to *ORS*

generate next-level candidate set

}

7 return the rule set *ORS*

4.3 Sorting Rules

At present, most algorithms that use classification association rule algorithms to mine rules are sorted by descent. They refer to the support when the confidence is the same.

The order of rules will greatly affect the match of rules when it is testing. Moreover, it will affect the accuracy of prediction. Thus, it has to confirm sorting methods of rules. It is defined as follows:

Definition 6. Given two rules *R1* and *R2*, $R1 \succ R2$ (also called *R1*precedes *R2*or *R1*has a higher precedence than *R2*) if :

1. The confidence of *R1*is greater than that of *R2*, or
2. The confidence values of *R1*and *R2* are the same, but the support of *R1*is greater than that of *R2*, or
3. Confidence and support values of *R1*and *R2* are the same, but *R1*has fewer conditions attributes in than that of *R2*, or
4. The confidence, support and condition attributes cardinality values of *R1* and *R2* are the same, but *R1* is associated with more frequent classification attributes than that of *R2*, or
5. All above criteria of *R1* and *R2* are the same, but *R1* was generated earlier than *R2*.

Generally, the support is global support, in order to avoid generating rules too many in the most frequent classes and gaining rules too few in the least frequent classes, the local support is introduced. The local support is defined as follows:

Definition 7. assuming supp (*Pd*) is the global support of $P \rightarrow d$, in that way, supp (*Pd*) / supp (*d*) is local support.

4.4 Matching Rules

Assuming ORS is the optimal rules set by using definition 6 to sort, *T* is the training data sets. The idea of classification prediction is: using the rules in ORS to predict the class of data objects in *T*. The rule in ORS, which matches the condition attributes of testing data object first time, is the rule to predict the testing data object. The prediction is right when the consequence of rule is as same as the classification attribute values of testing data object.

The definition of match is given as follows:

Definition 8. : assuming *Ct* is the condition attributes set of one testing

data object, *Cr* is the condition attributes set of one prediction rule, if $Cr \subseteq Ct$, then the testing data object matches the condition attributes of the rule.

V EXPERIMENTAL RESULTS

5.1 Materials

For carrying out our experiment eight different data sets of UCI (University of California at Irvine) are used namely balance, glass, iris, breast, vote, pima, zoom, and diabete. OCARA is compared with other three popular classification algorithms (C4.5 [39], CBA [33], RMR [48]).

5.2 Description of the Environment

This section presents a brief description of hardware and software requirements to carry out this experiment.

5.2.1 Hardware

The general set up of this experiment requires hardware. All these experiments run in Pentium IV 3.0 GHz, and latest version.

5.2.2 Software

The ordinary software's are used to carry out these experiments. Just we need Win2003.

5.3 Results and Data Analysis

It was proved that the support threshold was greatly influenced by the rule accuracy and the rule number [6]. If the support threshold is between 0.02 and 0.03, the accuracy will be much better, as discussed in [6]. The support threshold was set as 0.02, and the confidence was set as 0.30 in our work.

Table 5.1 The prediction accuracy of four algorithms

Dataset	Size	Accuracy (%)			
		C4.5	CBA	RMR	OCARA
Balance	625	63.57	66.94	74.38	66.46
Glass	214	66.47	69.23	70.89	76.78
Iris	150	96.00	93.25	93.99	96.55
Breast	699	94.66	98.71	95.92	98.90
Vote	435	88.45	86.92	88.98	92.76
Pima	768	73.28	76.38	77.76	79.86
Zoo	101	93.01	95.40	95.30	92.33
Diabetes	768	85.12	75.25	78.46	77.96
Average		82.57	82.76	84.46	85.45

Table 5.1 shows the classification accuracy of C4.5, CBA, RMR, and OCARA for 8 different data sets. The results shown in table 5.1 indicate that

the average of classification accuracy of OCARA is higher than C4.5, CBA, and RMR, and the average accuracy of OCARA is 2.88 percent higher than C4.5, 2.69 percent higher than CBA, and 0.99 percent higher than RMR.

CONCLUSION

This article has successfully proposed and tested a new algorithm, OCARA which is the basic of a method of Classification and Association Rules Obtaining. OCARA is derived from both Classification rules and Association rules and proposed by my study. OCARA is a robust algorithm because it derives its strengths from Classification rules and Association rules algorithms. We can confidently say OCARA is an accurate classifier. When compared with C4.5, CBA, and RMR using 8 different data sets from UCI, OCARA has higher accuracy than C4.5, CBA, and RMR. The average accuracy of OCARA is 2.88 percent higher than C4.5, 2.69 percent higher than CBA, and 0.99 percent higher than RMR.

Therefore, the above experimental results show that OCARA has better performance and is more efficient when compared with other popular algorithms.

However, OCARA has many rules when compared with RMR when the support is lower, note we worked on the method of Classification and Association Rules Obtaining. To overcome this limitation of having many rules, we are encouraging the coming researches to focus on this promising algorithm by improving its efficiency.

REFERENCES

[1] Aasheim Ø. T. and Helge G. S. Rough Sets as a Framework for Data Mining, Trondheim, May 4, 1996.

[2]. Agrawal R., Imielinski T., Swami A. Mining Associations between sets of items in massive databases. In Proc. Of the ACM SIGMOD on Management of Data, 1993.

[3] Agrawal R., Imielinski T., Swami A. Mining association rules between sets of items in large database. In Proc. Of the ACM SIGMOD Conference on Management of Data, May 1993, P. 207-216.

[4]. Agrawal R., Srikant R. Fast algorithms for mining association rule. In: Proceedings of the 20th International Conference on Very Large Data Bases, 1994, 487–499.

[5]. Agrawal R., Mannila H., Srikant R., Toivonen H., Verkamo I. Fast discovery of association rules. In Fayyad U. and et al, editors, Advances in Knowledge Discovery and Data Mining, MIT Press, 1996

[6]. Baralis E., Chiusano S. “Essential Classification Rule Sets,” ACM Trans. Database Systems, vol. 29, no. 4, 2004, 635-674.

[7]. Cendrowska J. PRISM: an algorithm for inducing modular rules [J].In: Int. J. Man-Mach. Stud., 1987, 27(4):349-370

[8] Chan K. C. C., Wong A. K. C. APACS: A System for the Automatic Analysis and Classification of

Conceptual Patterns. In Comput. Intell. 6, 1990, 119-131.

[9]. Clack P., Boswell R. Rule induction with CN2: Some recent improvements. In Y. Kodratoff, editor, Machine Learning – EWSL-91, 1991.

[10] Cook L., Cook J. Ethical Data Mining. Decision Sciences Institute 2002 Annual Meeting Proceedings 2002, 729

[11] Eui-Hong H., George K., Kumar V. Scalable Parallel Data Mining for Association Rules. Transaction on Knowledge and Data Engineering, 12(3): 2000, 728-737.

[12]. Fayyad U.M., Piatestsky-Shaprio Gr., Padhraic S., Uthurusamy R. Editors. Advances in Knowledge Discovery and Data Mining, AAAI Press / MIT Press ISBN 0-262-56097-6

[13]. Frans C., Paul L. The effect of threshold values on association rule based classification accuracy [J]. Data & Knowledge Engineering, 2007, 60:345-360

[14] Han J., Kamber M. Data Mining, Concepts and Techniques, Morgan Kaufmann Publishers ISBN 1-55860-489-8

[15]. Han, J. Pei, Yin Y. Mining frequent patterns without candidate generation. In: Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data, Dallas, Texas, (2000), 1–12.

[16] Heng C. Data Mining. http://cseserv.engr.scu.edu/StudentWebPages/hchhay/hchhay_FinalPaper.htm, 2009-11-05

[17] Houtsma M., Swami A. Set-oriented mining of association rules. Research Report RJ 9567, IBM Almaden Research Center, October 1993.

[18]. Houtsma M., Swami A. Set-oriented mining of association rules in relational databases. In 11th Intl. Conf. data Engineering, 1995.

[19]. Hu H., Li J. Using Association Rules to Make Rule-Based Classifiers Robust [C]. In: Proc. 16th Australasian Database Conf. (ADC), 2005, 47-52

[20] Hu X. Knowledge Discovery in databases: An Attribute-Oriented rough set approach, Regina, Saskatchewan, June, 1995.

Processing Letters 5(1), (1976), 15-17.

[21]. Hyafil L., Rivest R. L. Constructing optimal binary decision trees is NP-complete, Information

[22] Jeffrey D. U. Data Mining Lecture Notes. http://www-db.stanford.edu/~ullman/mining/mining.html, 2009-11-25

[23] Jiawei H., Pei J., Yiwen Y. Mining frequent patterns without candidate generation. ACM SIGMOD Record. In Proceedings of the 2000 ACM SIGMOD international conference on Management of data SIGMOD '00, Volume 29 Issue 2. May 2000

[24] Jong S. P., Ming-Syan Ch.; Yu. P.S. Using a hash-based method with transaction trimming for mining association rules. In Knowledge and Data Engineering, IEEE Transactions on Volume 9, Issue 5, Sept. - Oct. 1997, *813 – 825.*

[25]. Jiuyong L. On Robust Rule-Based Perdiction. IEEE Trans. on Knowledge and Data Engineering, 2006, 18(10)

[26]. Jiuyong L. On Optimal Rule Discovery [J]. IEEE Trans. on Knowledge and Data Engineering, 2006, 18(4):460-471

[27] Juan M. A., Gustavo H. R. An approach to discovering temporal association rules. In Proceedings of the 2000 ACM symposium on Applied computing – Volume 1. March 2000

[28]. Kamal A., Stefanos M., Srikant R. Partial classification using association rules. In David Beckerman, Heikki Mannila, Daryl Pregibon, and Ramasamy Uthurusamy, editors, Proceedings of the third International Conference of knowledge Discovery and Data Mining (KDD-97), AAAI Press, 1997,115.

[29] Keith C. C. Chan, Wai-Ho A. An effective algorithm for mining interesting quantitative association rules. In Proceedings of the 1997 ACM symposium on Applied computing. April 1997

[30]. Kusrini, Hartati. Implementation of C4.5 algorithm to evaluate the cancellation possibility of new student applicants at stmik amikom yogyakarta. In: ICEEI2007, Sri (2007)

[31] Laks V. S. L., Kai-Sang C. L., Raymond T. Ng. The segment support map: scalable mining of frequent itemsets. In ACM SIGKDD Explorations Newsletter, Volume 2 Issue 2. December 2000

[32] Li W., Jiawei H. , Pei J. CMAR: Accurate and Efficient Classification Based on Multiple Class-Association Rules, Proceedings of the 2001 IEEE International Conference on Data Mining, November 29-December 02, 2001, 369-376

[33]. Liu B., Hsu W., Ma Y. Integrating classification and association rule mining [C]. Proceeding of the KDD, New York, 1998, 80-86

[34]. Mannila H., Toivonen H., Verkamo I. Efficient algorithms for discovering association rules. In AAAI Wkshp. Knowledge Discovering in Databases, July 1994.

[35] Mata J., Alvarez J. L., Riquelme J. C. Evolutionary computing and optimization: An evolutionary algorithm to discover numeric association rules. In Proceedings of the 2002 ACM symposium on Applied computing. March 2002

[36] Ming-Syan Ch., Jiawei H., Philip, Yu S. Data Mining: An Overview from a Database Perspective. In IEEE Trans. On Knowledge And Data Engineering. 1996

[37]. Park J.S., Chen M., Yu P.S. An effective hash based algorithm for mining association rules. In ACM SIGMOD Intl. Conf. Management of Data, May 1995.

[38]. Quilan J.R. Induction of decision trees [J]. Machine Learning, 1986 (1):81-106

[39]. Quinlan J. R., Rivest R. L. Inferring decision trees using the minimum description length principle, Information and Computation 80, 227-248

[40]. Quinlan J.R. C4.5. Programs for Machine Learning [M]. Moregan Kaufmann, San Mateo, CA, 1993

[41] Rajesh N., Shekar B. Data mining (DM): poster papers: A relatedness-based data-driven approach to determination of interestingness of

association rules. In Proceedings of the 2005 ACM symposium on applied computing SAC '05. March 2005

[42]. Roberto B., Rakech A, Dimitrios G. Constraint-based rule mining in large, dence database. In Proc. Of the 15th Int'l Conf. on Data Engineering, pages 188-197, 1999.

[43]. Roberto B., Agrawal R. Mining the most interesting rules. In Surajit Chaudhuri and David Madigan, editors, Procecdings of the Fifth ACM SIGKDD International Conference on knowledge Discovery and Data Mining NY., August 15-18 1999.ACM Press, 145-154.

[44]. Sergey B., Rajeev M., Craig'Silverstein. Beyond market baskets: Generalizing associations rules to correlations. SIGMOD Record (ACM Special Interest Group of management of Data), 26(2), 265.

[45] Shafer J. C., Agrawal R., Mehta M. SPRINT: A scalable parallel classifier for data mining. In Proceedings of the 1996 International Conference on Very Large Databases, Mumbai (Bombay), India, September 1996.

[46] Simeon J. S., Djeraba Ch., Osmar R. Z. MDM/KDD2002: multimedia data mining between promises and problems. ACM SIGKDD Explorations Newsletter, Volume 4 Issue 2. December 2002

[47] Srikant R., Agrawal R. Mining quantitative association rules in large relational tables. ACM SIGMOD Record, Proceedings of the 1996 ACM SIGMOD international conference on Management of data SIGMOD 96, Volume 25 Issue 2. June 1996

[48]. Thabtah F.A., Cowling P.I. A greedy classification algorithm based on association rule [J]. Applied Soft Computing, 2007, 7:1102-1111

[49] Wu J. Business Intelligence: The value in mining data. http://www.dmreview.com/master.cfm 6/2003

[50] Zaki M. J., Ogihara M., Parthasarathy S., Li W. Parallel data mining for association rules on shared-memory multi-processors. In Proceedings of the 1996 ACM/IEEE conference on Supercomputing (CDROM) Supercomputing '96. November 1996

[51]. Zaki M.J., Hsiao C.J. Charm. An Efficient Algorithm for Closed Association Rule Mining [C]. In: Proc. SIAM Int. Conf. Data Mining, 2002

[52]. Zaki M.J. "Mining Non-Redundant Association Rules," Data Mining and Knowledge Discovery J., vol. 9, 2004, 223-248.