MAC: A Multiclass Associative Classification Algorithm

Neda Abdelhamid*‡, Aladdin Ayesh*§, Fadi Thabtah†¶, Samad Ahmadi*|| and Wael Hadi†**,

*Informatics Department
De Montfort University, Leicester, LE1 9BH
†Management Information System Department
Philadelphia University, Amman, Jordan
‡p09050665@myemail.dmu.ac.uk
§aayesh@dmu.ac.uk
¶ffayez@philadelphia.edu.jo
||sahmadi@dmu.ac.uk
**whadi@philadelphia.edu.jo

Abstract. Associative classification (AC) is a data mining approach that uses association rule discovery methods to build classification systems (classifiers). Several research studies reveal that AC normally generates higher accurate classifiers than classic classification data mining approaches such as rule induction, probabilistic and decision trees. This paper proposes a new multiclass AC algorithm called MAC. The proposed algorithm employs a novel method for building the classifier that normally reduces the resulting classifier size in order to enable end-user to more understand and maintain it. Experimentations against 19 different data sets from the UCI data repository and using different common AC and traditional learning approaches have been conducted with reference to classification accuracy and the number of rules derived. The results show that the proposed algorithm is able to derive higher predictive classifiers than rule induction (RIPPER) and decision tree (C4.5) algorithms and very competitive to a known AC algorithm named MCAR. Furthermore, MAC is also able to produce less number of rules than MCAR in normal circumstances (standard support and confidence thresholds) and in sever circumstances (low support and confidence thresholds) and for most of the data sets considered in the experiments.

Keywords: Associative classification; associative rule; data mining; rule learning.

1. Introduction

In the last few years, the numbers of offline and online data sets stored in different domain applications have been significantly growing. Often, these data sets contain important information that can be utilised by end-user in their business related activities. The process of discovering and extracting the concealed useful information from these data sets manually by domain experts is extremely difficult and requires care, experience and time. Therefore, employing automated scientific techniques to discover the useful information from the large data sets may grant companies to make appropriate decisions that work for improving their competitive advantages.

Data mining is one of the important computing tools that can discover, and derive the concealed useful information automatically from different kinds of data sets (Ikonomakis et al., 2005). One crucial task of data mining about forecasting a particular class or category of unseen data case is called classification. Particularly, the goal of classification is to construct models (classifiers) from labelled training data sets in order to predict the class value of previously unseen data case as accurately as possible (Thabtah, 2006). Another known data mining task is association rule discovery (Agrawal and Srikant, 1994). Association rule is about finding hidden correlations among different variables in a data set and generating them in simple “If-Then” rules. Associative classification (AC) is a data mining approach that integrates classification and association rule discovery (Li et al., 2001). Precisely, AC employs association rule techniques to find the rules, and then chooses a subset of them to build a classifier.

In the last decade, different research studies on AC mining have resulted in the disseminations of various algorithms including CBA (Liu et al., 1998), CMAR (Li et al., 2001), CPAR (Yin and Han, 2003), MCAR (Thabtah et al., 2005), CACA (Tang and Liao, 2007), BCAR (Yoon and Lee, 2008), LCA (Thabtah et al., 2010), and others. These studies have revealed that AC is able to construct more accurate classifiers than traditional classification data mining approaches including rule induction.
N. Abdelhamid et al.

(Cohen, 1995) and decision tree (Quinlan, 1993). Nevertheless, the numbers of rules discovered by the AC algorithms are normally huge. One primary reason of the large size classifiers resulting from these algorithms is inherited from association rule discovery since all correlations among the attribute values and the class attribute are tested during the training phase. This paper proposes a new algorithm called Multiclass Associative Classification (MAC) that reduces the number of rules discovered without drastically impacting the predictive accuracy of the classifiers. In other words, and during constructing the classifier, we would like to minimise the number of rules selected. This can help decision makers especially in understanding, controlling and maintaining the final set of rules mainly when making a prediction decision.

For instance, we run experiments on two data sets “Led” and “Diabetes” from the UCI data repository (Merz and Murphy, 1996) using an AC algorithm called MCAR (Thabtah et al., 2005) and the proposed algorithm. The “Led” and “Diabetes” data sets contain 3,200 and 768 data cases respectively. Further, we utilised a minimum support (minsupp) and a minimum confidence (minconf) of 2% and 40% respectively. The number of rules generated using MCAR and the proposed algorithm from these data sets are (209-Led, 95-Diabetes) and (83-Led, 69-Diabetes) respectively with predictive accuracies of (Led—77.68%), (Diabetes—71.90%), (Led—73.00%, Diabetes—74.20%) respectively. It is obvious from the results that MCAR’s classifier size on the “Led” data set is almost two and a half times larger than that of MAC without achieving any accuracy improvement rather it achieved 1.10% less accuracy than MAC on this data set. Meaning the additional 126 rules of MAC’s classifier are in fact useless when it comes to predicting the class of unseen data and can be a burden on decision maker particularly in understanding and maintaining the classifier. On the other hand, and for the “Diabetes” data set, MCAR produced 28 rules more than MAC that contributed positively on the classifier predictive accuracy achieving a +3.4% accuracy over that of the proposed method. These results if limited indicate that in some cases reducing the classifier size by pruning redundant rules may positively affect the accuracy of the classifiers. Though, in some other cases it may slightly decrease the predictive accuracy but more importantly the end-user ends up with moderate size classifiers which he/she can easily interpret and control more than large size ones. Overall, the research question that this paper is trying to answer is: Does rule pruning or reducing the size of the classifier improve the accuracy of associative classifiers?

The AC problem, solution strategy and literature review are introduced in Sec. 2. The proposed algorithm main steps are presented in Sec. 3, and the experimental results are given in Sec. 4. Finally, conclusions are depicted in Sec. 5.

2. Associative Classification Data Mining

We follow (Thabtah, 2006) in the definition of the AC problem in data mining. The training data set D has n distinct attributes A1, A2, . . . , An and C is a list of class labels. The number of cases in D is denoted |D|. An attribute may be categorical (where each attribute takes a value from a known set of possible values) or continuous where each attribute takes a value from an infinite set, e.g. (real or integer). For categorical attributes, all possible values are mapped to a set of positive integers. In the case of continuous attributes, any discretisation method can be utilised.

Definition 1: An AttributeValue can be described as an attribute name Ai and its value ai, denoted (Ai, ai).

Definition 2: The jth row or a training case in D can be described as a list of attribute values (A1, a1), . . . , (Ak, ak), plus a class denoted by cj.

Definition 3: An AttributeValueSet set can be described as a set of disjoint attribute values contained in a training case, denoted {(A1, a1), . . . , (Ak, ak)}.

Definition 4: A ruleitem r is of the form ⟨antecedent, c⟩, where antecedent is an AttributeValueSet and c ∈ C is a class.

Definition 5: The actual occurrence (actoccr) of a ruleitem r in T is the number of cases in D that match r’s antecedent.

Definition 6: The support count (suppcount) of ruleitem r = ⟨antecedent, c⟩ is the number of cases in D that matches r’s antecedent, and belongs to a class c.

Definition 7: A ruleitem r passes the minsupp threshold if, suppcount(r)/|D| ≥ minsupp. Such a ruleitem is said to be a frequent ruleitem.

Definition 8: A ruleitem r passes the minimum confidence (minconf) threshold if suppcount(r)/actoccr(r) ≥ minconf.

Definition 9: A rule is represented as: Antecedent → c, where antecedent is an AttributeValueSet and the consequent is a class.

2.1. Solution strategy

Normally, the majority of AC algorithms operate in two steps, step one involves rules discovery and production,
and in step 2, a classifier is built from the discovered rules found in step 1. To explain the discovery of rules and building the classifier, consider the training data set shown in Table 1, which represents three attributes (Attr1, Attr2) and the class attribute (Class). Assume that the \( \minsupp \) and \( \minconf \) have been set to 20% and 50%, respectively. A typical AC algorithm such as MCAR (Thabtah et al., 2005) firstly discovers all frequent ruleitems which hold enough supports (Table 2). Once all frequent ruleitems are found, then MCAR transforms the subset of which hold enough confidence values into rules and then ranks them according to certain thresholds (Thabtah, 2006). The bold rows within Table 2 are the rules, and from those the classifier is derived. A rule is considered part of the classifier if it covers certain number of cases in the training data set. So, a subset of the discovered rules is chosen to form the classifier which in turn is evaluated against an independent data set known by the test data set to obtain its effectiveness.

### 2.2. Literature review

In the last few years, several AC algorithms have been proposed like CBA (Liu et al., 1998), CPAR (Yin and Han, 2003), CAAR (Xu et al., 2004), 2-PS (Qian et al., 2005), MCAR (Thabtah et al., 2005), CACA (Tang and Liao, 2007), ACCF (Li et al., 2008), BCAR (Yoon and Lee, 2008), CBAR (Niu et al., 2009), LCA (Thabtah et al., 2010) and others. These algorithms employ different strategies to find and extract the rules, sort the rules, prune the redundant rules, build the classifier and predict test data cases. Hereunder we shed the light on some of these algorithms.

One of the initial algorithms that showed the use of association rule discovery in classification is CBA (Liu et al., 1998). Specifically, CBA implements the known \textit{A priori} algorithm (Agrawal and Srikant, 1994) to discover the frequent ruleitems and this step is called candidate generation. These frequent ruleitems (\langle attributes, values, class \rangle) must pass the \( \minsupp \) threshold. Then, the frequent ruleitems are used to derive the complete set of rules, which in turn is used to build the classifier. Experiments against the 26 UCI data sets showed that CBA outperformed decision trees with regards to classification accuracy.

A greedy AC algorithm called CPAR was proposed in Yin and Han (2003). It utilises the measure of FOIL-gain in building the rules. In particular, CPAR looks for the largest attribute value gain among the available attributes in the training data set to add it in the rule’s antecedent. Once the attribute value is determined, the weights of the positive examples associated with it will be deteriorated by a multiplying factor, and the process is repeated until all positive cases in the training data set are covered. One common disadvantage of CPAR is that searching for the best attribute value in terms of gain is time consuming especially for large data sets since the gain for every possible value is required to be computed in order to identify the best value. It has been claimed that CPAR produces more accurate classifiers than CBA algorithm in the same study.

Qian et al. (2005) proposed an association rule-based approach suitable for textual data collections called 2-PS. This approach firstly builds a classifier through two phases in which the first phase aims to removing redundant rules, i.e. rules found within every class are deleted by a sentence-level constraint. In the second phase, all remaining rules are compared according to the available class labels and then merged together. Lastly, when it comes to forecasting a new document, the multiple label rules are taken into account based on a sorting procedure to select the most applicable one to the test document. Experimental results against unstructured text data set showed that the 2-PS algorithm achieved good accuracy.

A new CBA-like algorithm called CARGBA was developed in Kundu et al. (2008). This algorithm works in two phases in which in the first phase, it utilises the

<table>
<thead>
<tr>
<th>Table 1. Training data set.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Instance number</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>5</td>
</tr>
<tr>
<td>6</td>
</tr>
<tr>
<td>7</td>
</tr>
<tr>
<td>8</td>
</tr>
<tr>
<td>9</td>
</tr>
<tr>
<td>10</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table 2. Frequent items derived by MCAR from Table 1.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Frequent attribute value</td>
</tr>
<tr>
<td>--------------------------</td>
</tr>
<tr>
<td>(a1),c2</td>
</tr>
<tr>
<td>(a1), c1</td>
</tr>
<tr>
<td>(b1),c2</td>
</tr>
<tr>
<td>(b2), c1</td>
</tr>
<tr>
<td>(a1,b1),c2</td>
</tr>
<tr>
<td>(a1,b2), c1</td>
</tr>
</tbody>
</table>
A priori candidate generation function similar to CBA to produce the rules. Then, it derives a number of specific rules that survive the confidence threshold and without enforcing the support pruning. In the second phase, CARGBA ranks the rules according to CBA rule ranking method, and it inserts a rule into the classifier if in covers at least one training case. Further, it uses Correlation Coefficient to prune redundant rules. This algorithm has been tested against a limited number of data sets from the UCI data collection and compared with three known classification methods (C4.5, CBA, CMAR). The results showed limited but direct evidence that CARGBA is competitive with other classification approaches with respect to accuracy.

Lukasz and Krzysztof (2006) have slightly extended CBA algorithm to deal with classifying image data. Particularly, they altered the support threshold to cope with the image features (colour and texture). Further, the training strategy, rule sorting and classification steps are adopted from CBA approach. Experimental results obtained against 400 photographs showed that the proposed method scales well in terms of classification accuracy and the numbers of generated rules when compared with previous image classification algorithms.

In Tang and Liao (2007), an AC algorithm called CACA was proposed that transforms the training data set into vertical data representation in order to efficiently improve the searching for rules. In particular, it counts the frequency of every attribute value in the training data set and then sorts these values in descending order. Any attribute which fails to satisfy the minsupp threshold gets removed during this step. For the remaining attributes, CACA intersects their appearances in the training data set to reduce the search space of frequent ruleitems. In addition, CACA classifies the unseen data like CBA. Experimental results suggested that CACA performs better with reference to accuracy and computation time than MCAR algorithm against some UCI data sets.

A few years ago, Li et al. (2008) extended an association rule discovery algorithm named Charm to handle classification benchmarks and they named the new algorithm ACCF. In particular, ACCF employed the concept of closed itemsets of Charm to cut down the number of rules produced so that decision makers can control the classifier and edit the rules. Experimental results against eighteen different data sets from the UCI data repository (Merz and Murphy, 1996) showed that ACCF produced slightly better classifier with respect to accuracy as well as size than CBA.

In Niu et al. (2009), an algorithm called ACCR was proposed, which extends the A priori algorithm to generate classification rules. ACCR main idea lies in solving the rare items problem in which when the user sets the support threshold too high, some good quality rules are discarded, on the other hand, when the support is set too low, many redundant rules get generated which consequently consumes more processing time and storage. The authors of ACCR developed a measure of rule goodness called “compactness” that stores ruleitems with low support but high confidence to ensure that high quality rules are not removed. The compactness is computed as:

\[
\text{Compactness}(I) = \sum_{i=1}^{m} \frac{\text{Lift}(R_i)}{m}, \quad (1)
\]

\[
\text{Lift}(X \rightarrow Y) = \frac{\text{Conf}(X \rightarrow Y)}{\text{Sup}(Y)} = \frac{\text{Sup}(X \cup Y)}{\text{Sup}(X)\text{Sup}(Y)}, \quad (2)
\]

where the “lift” is the degree of independence between antecedent (X) and consequent (Y) of the measured rule X → Y. If the value is close to one, the relationship between antecedent and consequent should be relatively small. The experimental results showed that the ACCR algorithm has better accuracy if contrasted with CBA against the UCI data sets.

Thabtah et al. (2010) developed an AC algorithm called LCA that improved the performance of CBA algorithm especially in the training phase. LCA minimises the number of candidate itemsets joining which usually consumes time and memory resources by only joining candidate itemsets that have common class label.

An AC algorithm called GA-ACR that adopts GA search strategy to build the classifier was proposed in Chien and Chen (2010). GA is a common searching strategy in Artificial Intelligence (AI) based on the Darwinian natural selection and mutation in biological production. Normally, a GA method starts with an initial population of objects, and it tests the fitness of the objects in the population until a stopping criteria is met. During testing, it performs selection, crossover, and mutation operations on objects. In GA-ACR algorithm, the input is different continuous attributes some of which are technical indicators (the relevant difference between two items). Then the algorithm discovers the relation sets among the items in the form of relation (item,operator,item) in which there are three different items (constant, technical indicator, attribute), and the operator are restricted to (,). A conjunction of the relation sets is the rule antecedent. The GA-ACR algorithm cuts down the search space by
providing a relation pruning method that indicates which pairs of items can be compared for which attributes in a relation. During the rule discovery step, a rule is encoded in multi-level structure and represented as a chromosome. The first level contains the number of items encoded and the value of the gene corresponds to the relation type of the item. The algorithm produces the genes for the first level and then the second level and considers discarding irrelevant relations. It should be noted that only the first level genes are applied in the crossover to prevent producing useless rules, though, mutation is applied to genes in the first and second level. All rules produced must pass the minsupp and the minconf thresholds, and then sorted according to CBA rule sorting procedure.

Limited experimentations on stock data collection gathered from 10 different companies have been carried out with reference to accuracy. The results pointed out that the GA-ACR algorithm outperforms a simple data distribution algorithm. No comparisons of the GA algorithm and other AC algorithms are conducted in order to generalise the performance of the algorithm.

Wang et al. (2011) proposed an AC called ADA that constructs rules from both the input training data set as well as the classified resources such as the training data set, current classification rules, and test cases. Meaning the classifier gets amended on the fly after the classified resources reach to a certain amount. The authors have used a co-training method (Mei et al., 2006) to accomplish the task of updating the classifier by refining the new discovered knowledge by the existing classification rules. The co-training method used in ADA has been adopted from the semi-supervised learning of pattern context where the labelled training documents are exercised to figure out the class labels of the unlabelled cases. More details can be found in Mei et al. (2006). Overall, ADA can be considered a semi-incremental AC algorithm since few training cases or users set of frequent patterns (keywords) are only necessary to build the associative classifier instead of the complete training cases. Then, the classified cases as well as the classification rules are employed to update the classifier by adding or removing rules.

3. The Proposed Algorithm (MAC)

The proposed algorithm utilises AC learning strategy to generate the rules. It comprises two main steps: Rules discovery and classifier building. In the first step, MAC iterates over the input training data set in which the rules is found and extracted. Then in the second step it tests the discovered rules on the training data set in order to select one subset to represent the classifier. The general description of the MAC learning algorithm is depicted in Fig. 1, and details are given in the next subsections.

We assume that the input attributes are categorical or continuous attributes. For continuous attributes any discretisation measure is employed before the training phase.

### 3.1. Rule discovery and classifier construction

#### 3.1.1. Rule discovery and sorting methods

MAC uses fast training method that employs simple intersection among attribute value TIDs to discover the rules. The TID of an attribute value holds the row numbers that contain the attribute values in the training data set. The proposed algorithm discovers the frequent attribute value of size 1 (F1) after iterating over the training data set. Then, it intersects the TIDs of the disjoint attribute values in F1 to discover the candidate attribute values of size 2, and after determining F2, the possible remaining frequent attribute values of size 3 are obtained from intersecting the TIDs of the disjoint attribute values of F2, and so forth. The TIDs of the attribute value comprises useful information that are utilised to locate values easily in the training data set especially in computing the support and confidence for rules. For example, the frequent ruleitems (size 1) \((a_1, c_2)\) and \((b_1, c_2)\) that are shown in Table 2 are used to produce the candidate ruleitem (size 2) \((a_1, b_1, c_2)\) by simply intersecting their TIDs, i.e. \((1,2,6,10)\) and \((1,2,6)\) within the training data set (Table 1). The result of the above intersection is the set \((1,2,6)\) which its length, i.e. 3, denotes the support value of the new ruleitem \((a_1, b_1, c_2)\). Now, since this attribute value support is...
larger than the \textit{minsupp} threshold, i.e. 20\%, \{(a1, b1), c2\} will become frequent.

The abovementioned training approach is called vertical mining and has been used successfully in association rule discovery and classification (i.e. Thabtah \textit{et al.}, 2005). This approach transforms the original training data set into a table that contains the locations (TIDs) of each attribute value in the training data set, and then it employs simple intersections among these TIDs to discover frequent rule items and produce the rules. Since this approach iterates over the training data set once it is highly effective according to several experimental studies in the literature of data mining and machine learning communities especially with regards to processing time and memory usage. More details on the advantage of vertical mining over traditional approaches are given in Zaki and Gouda (2003).

When frequent attribute values are identified, MAC generates any of which as a single label rule when it passes the \textit{minconf} threshold. Once the complete set of rules is derived the sorting procedure is invoked (shown in Fig. 2) to ensure that rules with high confidence are given higher priority to be selected as part of the classifier. The rule sorting procedure is adopted from Thabtah \textit{et al.} (2005) since it reduces rule random selection which positively affect the classification accuracy of the generated classifier.

3.1.2. Classifier construction

After rules are sorted then a subset of which gets chosen to comprise the classifier. Figure 3 illustrates how the classifier is build by MAC. Particularly, and for each training case, MAC iterates over the set of discovered rules (top-down fashion) and marks the first rule that matches the training case as a classifier rule. The same process is repeated until all training cases are utilised. Finally, MAC outputs all marked rules to form the classifier. The remaining unmarked rules are discarded by the proposed algorithm since some higher ranked rules have used their training cases during building the classifier and therefore these unmarked rules become redundant and useless.

The rule pruning of the proposed algorithm differs from other pruning procedure in AC such as MCAR in that it does not require the similarity of the class labels of both the selected rule and the training case as a condition of rule significance rather it only considers the matching between the rule body and the training case. This indeed reduces overfitting since most of current AC algorithms including CBA and MCAR mark a rule as a classifier rule if it matches the training case and has the same class as the training case. This may result in more accurate prediction on the training data set but not necessarily on new unseen test cases. The class matching of the candidate rule and the training case does not necessarily give an additional sign of rule goodness besides the matching condition between this rule body and the training case attribute values. In other words, the performance of the classification model is not yet generalised since it has not been tested on an independent test cases to measure its predictive power. We argue that the similarity test between the candidate rule class and the training case class has limited effect on the predictive power of the resulting classifiers during the prediction step. Later in Sec. 5, we show the main results obtained with reference to classification accuracy on different UCI data sets for both rule pruning procedures (the one which looks at the class (MCAR) and the one that marks the applicable rule without checking the class (MAC)).
3.1.3. **Prediction procedure**

In classifying a test case, the prediction procedure of the MAC algorithm works as follow:

It groups the rules which match the test case body into groups according to their class, and then assigns the test case the class of the group which has largest number of rules. Unlike traditional AC methods like CBA which utilise a single rule for prediction, MAC makes the prediction decision based on multiple rules, which is considered by previous research studies on AC methods an advantage since multiple high confidence and support rules contributed to the prediction decision. Lastly, in cases when no rules in the classifier are applicable to the test case, the default class (majority class in the training dataset) will be assigned to that case.

### 3.2. **MAC versus other AC algorithms**

In the research literature of single label AC mining, there are several different algorithms developed most of which are CBA-based. In other words, the majority of current AC algorithms are developed to enhance either the predicative power or the efficiency of CBA algorithm. The main distinctions between the proposed algorithm and these algorithms are the following:

- **MAC** utilises vertical mining approach for finding the knowledge from data sets whereas the majority of current AC algorithms employ level-wise search in discovering the rules (*A priori* approach). The main advantage of using vertical mining is that the data set is scanned only once, and then simple intersections among the TIDs of ruleitems of size one are required to derive the remaining ruleitems which are the input for rule production step. To the knowledge of the authors, there is only two single label AC algorithm that employs vertical mining called MCAR and CACA.

- The methodology of constructing the classifier from the discovered set of rules in the proposed algorithm is novel since MAC does not consider the similarity between the candidate rule class and that of the training case during the selection of the classifier rules. This reduces overfitting of the resulting classifier as well as its size. On the other hand, other AC algorithms such as CBA and MCAR insert the rule into the classifier particularly if the candidate rule class is identical to that of the training case.

### 4. **Experimental Results**

We have evaluated the proposed algorithm against nineteen different data sets from the UCI data repository (Merz and Murphy, 1996). Ten-fold cross-validation was utilised to evaluate the classification models and to produce error rates in the experiments. Ten-fold-cross-validation is a common evaluation measure for computing the error rate for the classifiers against data in data mining and machine learning. It initially divides the input data arbitrary into ten parts in which nine parts are used to learn the rules and the remaining hold out part is utilised to test the rules predictive quality. The procedure is then invoked repeatedly 10 times on the input data and the derived results (error rates) of all runs are then averaged. The following three known classification data mining approaches: decision tree (C4.5) (Quinlan, 1993), rule induction (RIPPER) (Cohen, 1995), and associative classification (MCAR) have been contrasted with MAC with reference to error rate. This has been performed to evaluate the predictive power of MAC.

Our selection of the above classification approaches is based on the different learning strategies they employ in discovering and producing the knowledge. For example, C4.5 uses divide and conquer to build a decision tree classifier. On the other hand, RIPPER uses search heuristics to produce “IF-THEN” rules. Lastly AC adopts association rule discovery strategy to search for correlations (rules) among the attribute values and the class attribute in the training data set.

We used a Pentium IV 2.0 GH machine to run the experiments. The experiments of RIPPER and C4.5 were conducted using the Weka software system (www.weka.com), which is an open java source code for the common data mining and machine learning algorithms. Lastly, MCAR and the proposed algorithm were implemented in Java.

Several researchers in association rule discovery and AC (i.e. Liu et al., 1998; Li et al., 2001; Thabtah, 2006), have pointed out that the support threshold often controls the numbers of rules generated and the training time consumed during the rule discovery and production steps. Thus, we have followed other data mining scholars (i.e. Liu et al., 1998), in setting the support threshold to 2% in the experiments of MCAR and the proposed algorithm. The confidence threshold, however, has less impact on the general performance of AC algorithms and we set it to 40% for MCAR and MAC.

Table 3 contains the data sets we consider and the error rate figures of the contrasted algorithms. This table also shows the characteristics of each data set utilised in the experiments including the data set name, the data set size (number of training cases), and the number of class labels per data set. It should be noted that there are small (Contact-lenses), medium (Tic-tac) and large
After analysing Table 3 we found out that the proposed algorithm scales well if compared to common classification data mining algorithms. In particular, MAC has achieved on average $0.94\%$ and $0.72\%$ higher accuracy than rule induction (RIPPER) and decision tree (C4.5) algorithms. On the other hand, the original MCAR algorithm has outperformed the proposed method against the UCI data sets on average by $1.15\%$. These error rate results have a good indication that AC data mining approach usually produces more accurate classifiers than rule induction and decision tree approaches.

A deeper examination on the numbers of rules produced by both the proposed algorithm and MCAR were performed against the UCI data sets. We have considered two scenarios, one using standard support and confidence (MinSupp 2%, MinConf 40%), and one with lower support and confidence (MinSupp 1%, MinConf 10%) since we would like to identify the behaviour of both algorithms in normal and sever cases.

The number of rules produced against the data sets by both MCAR and MAC for the above scenarios are shown in Figs. 4 and 5 respectively. It is clear from Fig. 4 that the proposed algorithm often generates less number of rules than MAC. Specifically, and in normal circumstances, MAC produced on average 9.79 less number of rules than MCAR algorithm on the data sets we consider. Furthermore, in sever cases when the support and confidence are set to low values, the generated number of rules of MCAR becomes even larger, and precisely it derived on average 12.89 more rules than MAC. This indicates that in both normal and sever situations MAC normally extracts smaller classifiers than MCAR. In other words, holding a large number of rules in some cases particularly to predict a limited number of test cases is impractical. There should be a trade-off between the classification accuracy and the size of the resulting classifiers especially when slightly more error can be tolerated in exchange for a more concise set of rules. Moreover, larger classifiers make it difficult for a human to be able to interpret, and maintain. Thus, even though the proposed method achieved slightly higher error rate against the UCI data sets than MCAR it indeed derived a moderate sized classifiers.

### Table 3. Error rate of C4.5, RIPPER, MCAR and MAC.

<table>
<thead>
<tr>
<th>Data set</th>
<th>Size</th>
<th>Num. of classes</th>
<th>RIPPER</th>
<th>CBA</th>
<th>MCAR</th>
<th>MAC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Austra</td>
<td>690</td>
<td>2</td>
<td>14.79</td>
<td>14.64</td>
<td>14.04</td>
<td>16.39</td>
</tr>
<tr>
<td>Balance-scale</td>
<td>625</td>
<td>3</td>
<td>25.44</td>
<td>34.34</td>
<td>14.3</td>
<td>14.92</td>
</tr>
<tr>
<td>Breast</td>
<td>699</td>
<td>2</td>
<td>4.58</td>
<td>4.16</td>
<td>4.27</td>
<td>5.53</td>
</tr>
<tr>
<td>Cleve</td>
<td>303</td>
<td>2</td>
<td>22.45</td>
<td>16.87</td>
<td>18.54</td>
<td>20.62</td>
</tr>
<tr>
<td>Contact</td>
<td>24</td>
<td>3</td>
<td>25.00</td>
<td>20.00</td>
<td>34.68</td>
<td>34.58</td>
</tr>
<tr>
<td>Diabetes</td>
<td>768</td>
<td>2</td>
<td>23.96</td>
<td>24.66</td>
<td>22.31</td>
<td>25.79</td>
</tr>
<tr>
<td>Glass</td>
<td>214</td>
<td>7</td>
<td>31.31</td>
<td>30.11</td>
<td>24.76</td>
<td>25.51</td>
</tr>
<tr>
<td>Heart-s</td>
<td>294</td>
<td>2</td>
<td>21.77</td>
<td>20.80</td>
<td>18.8</td>
<td>22.75</td>
</tr>
<tr>
<td>Iris</td>
<td>150</td>
<td>3</td>
<td>5.34</td>
<td>6.75</td>
<td>7.06</td>
<td>5.73</td>
</tr>
<tr>
<td>Labour</td>
<td>57</td>
<td>2</td>
<td>22.81</td>
<td>5.01</td>
<td>16.49</td>
<td>16.49</td>
</tr>
<tr>
<td>Lymph</td>
<td>148</td>
<td>4</td>
<td>22.98</td>
<td>23.62</td>
<td>26.08</td>
<td>29.05</td>
</tr>
<tr>
<td>Mushroom</td>
<td>8124</td>
<td>2</td>
<td>0.10</td>
<td>8.71</td>
<td>0.26</td>
<td>0.11</td>
</tr>
<tr>
<td>Pima</td>
<td>768</td>
<td>2</td>
<td>26.70</td>
<td>24.51</td>
<td>22.39</td>
<td>25.71</td>
</tr>
<tr>
<td>Primary-tumor</td>
<td>339</td>
<td>23</td>
<td>65.20</td>
<td>74.89</td>
<td>58.31</td>
<td>60.38</td>
</tr>
<tr>
<td>Tic-tac</td>
<td>958</td>
<td>2</td>
<td>3.03</td>
<td>0.00</td>
<td>1.02</td>
<td>0.29</td>
</tr>
<tr>
<td>Vote</td>
<td>435</td>
<td>2</td>
<td>11.73</td>
<td>12.65</td>
<td>13.09</td>
<td>12</td>
</tr>
<tr>
<td>Wine</td>
<td>178</td>
<td>3</td>
<td>5.62</td>
<td>7.31</td>
<td>1.67</td>
<td>4.26</td>
</tr>
<tr>
<td>Zoo</td>
<td>101</td>
<td>7</td>
<td>6.94</td>
<td>14.86</td>
<td>4.04</td>
<td>13.46</td>
</tr>
</tbody>
</table>

(Mushroom) data sets so that we can derive a general conclusions on the performance of the proposed algorithm. After analysing Table 3 we found out that the proposed algorithm scales well if compared to common classification data mining algorithms. In particular, MAC has achieved on average $+0.94\%$ and $+0.72\%$ higher accuracy than rule induction (RIPPER) and decision tree (C4.5) algorithms. On the other hand, the original MCAR algorithm has outperformed the proposed method against the UCI data sets on average by $1.15\%$. These error rate results have a good indication that AC data mining approach usually produces more accurate classifiers than rule induction and decision tree approaches.
5. Conclusions

A novel algorithm for multi-class classification based on association rule called MAC has been proposed in this paper. MAC employs a new methodology in construction the classifier which often results in generating moderate size classifiers. Furthermore, the proposed algorithm uses a fast intersection method based on vertical data representation to learn the rules. Experimentations against 19 different data sets from the UCI data repository using different classification algorithms have been performed. The bases of the experiments are two common data mining measures namely error rate and the number of rules generated. The results obtained reveal that the proposed algorithm outperformed both C4.5 and RIPPER algorithms with respect to error rate, and it scales well if compared with a known AC algorithm called MCAR. Moreover, MCADM usually generates less number of rules than MCAR on the data sets we consider. Meaning although MAC achieved on average 1.15% less accuracy than MCAR algorithm on the 19 data sets it extracts on average 9.79 less number of rules. This indeed enables decision makers of controlling and understanding the proposed algorithm classifiers more than that of MCAR. In near future, we would like to extend MAC to handle multi-label data sets and generated multiple labels classifiers.

References


Niu, Q, S Xia and L Zhang (2009). Association classification based on compactness of rules, WKDD,


