MMAC: A New Multi-class, Multi-label Associative Classification Approach

Fadi A. Thabtah
Modelling Optimisation Scheduling And Intelligent Computing Research Centre
Fabdelja@bradford.ac.uk

Peter Cowling
Modelling Optimisation Scheduling And Intelligent Computing Research Centre
P.i.Cowling@bradford.ac.uk

Yonghong Peng
Department of Computing, University of Bradford, BD7 1DP, UK
Y.h.Peng@bradford.ac.uk

Abstract

Building fast and accurate classifiers for large-scale databases is an important task in data mining. There is growing evidence that integrating classification and association rule mining together can produce more efficient and accurate classifiers than traditional classification techniques. In this paper, the problem of producing rules with multiple labels is investigated. We propose a new associative classification approach called multi-class, multi-label associative classification (MMAC). This paper also presents three measures for evaluating the accuracy of data mining classification approaches to a wide range of traditional and multi-label classification problems. Results for 28 different datasets show that the MMAC approach is an accurate and effective classification technique, highly competitive and scalable in comparison with other classification approaches.

1. Introduction

Classification is a well-known task in data mining that aims to predict the class of an unseen instance as accurately as possible. While single label classification, which assigns each rule in the classifier the most obvious label, has been widely studied [9, 11, 13, 18], little work has been done on multi-label classification. Most of the work to date on multi-label classification is related to text categorisation [10, 15]. There are many approaches for building single class classifiers from data, such as divide-and-conquer [14] and separate-and-conquer [8]. Most traditional learning techniques derived from these approaches, such as decision trees [7, 13], and statistical and covering algorithms [11], are unable to treat problems with multiple labels.

The most common multi-label classification approach is one-versus-the rest (OvR) [17], which constructs a set of binary classifiers obtained by training on each possible class versus all the rest. OvR approach performs the winner-take-all strategy that assigns a real value for each class to indicate the class membership.

Another known approach in multi-label classification is one-versus-one (OvO) [15], which constructs a classifier that has been trained on each possible pair of classes. For K classes, this results in (K-1) K/2 binary classifiers, which may be problematic if K is large. On the other hand, the OvR approach has been criticised for training on several separate classification problems, since each class can easily be separated from the rest, and therefore problems a rise, like contradictory decisions, i.e. whenever two or more rules predict the test instance, and no decision, i.e. whenever none of the resulting rules can predict the test instance [6].

Another important task in data mining is the discovery of all association rules in data. Classification and association rule discovery are similar, except that there is only one target to predict in classification, i.e., the class, while association rule can predict any attribute in the data. In recent years, a new approach that integrates association rule with classification, named associative classification, has been proposed [9, 12]. A few accurate classifiers that use associative classification have been presented in the past few years, such as CBA [12], CMAR [9], and CPAR [18].

In existing associative classification techniques, only one class label is associated with each rule derived, and thus rules are not suitable for the prediction of multiple labels. However, multi-label classification may often be useful in practise. Consider for example, a document which has two class labels “Health” and “Government”, and assume that the document is associated 50 times with the “Health” label and 48 times with the “Government” label, and the number of times the document appears in the training data is 98. A traditional associative technique like CBA generates the rule associated with the “Health” label simply because it has a larger representation, and discards the other rule. However, it is very useful to generate the other rule, since it brings up useful knowledge having a large representation in the training data, and thus could take a role in classification. In this paper, a novel approach for multi-class and multi-label classification, named multi-class, multi-label associative classification (MMAC), is

introduced. It assumes that for each instance that passes certain thresholds, there is a rule associated with not only the most obvious class, but with the second, third,…, k_{th} possible class label. Three evaluation methods are presented in this research paper in order to evaluate classifiers derived by MMAC on different application themes, and compare them to other approaches.

The multi-label classification problem is introduced in Section 2. Basic concepts of association rule and associative classification are discussed in Section 3. The MMAC approach and our methods for evaluation of traditional and multi-label classifiers are presented in Section 4, and the experimental results are given in Section 5. Finally the conclusions are presented in Section 6.

2. Multi-label Classification

Most of the research conducted on classification in data mining has been devoted to single label problems. A traditional classification problem can be defined as follows: let D denote the domain of possible training instances and Y be a list of class labels, let H denote the set of classifiers for $D \rightarrow Y$, each instance $d \in D$ is assigned a single class y that belongs to Y. The goal is to find a classifier $h \in H$ that maximises the probability that $h(d) = y$ for each test case (d, y). In multi-label problems, however, each instance $d \in D$ can be assigned multiple labels $y1, y2, \ldots, y_k$ for $y_i \in y$, and is represented as a pair $(d, (y1, y2, \ldots, y_k))$ where $(y1, y2, \ldots, y_k)$ is a list of ranked class labels from y associated with the instance d in the training data.

3. Classification Based on Association Rule

3.1 Frequent Items, Support and Confidence

Let T be the training data with n attributes $A_1, A_2, \ldots, A_n$ and C is a list of class labels. A particular value for attribute A_i will be denoted a_i, and the class labels of C are denoted c_j.

Definition 1: An item is defined by the association of an attribute and its value (A_i, a_i), or a combination of between 1 and n different attributes values, e.g. $< (A_1, a_1)>$, $< (A_1, a_1), (A_2, a_2)>$, $(A_1, a_1), (A_2, a_2), (A_3, a3)>$, … etc.

Definition 2: A rule r for multi-label classification is represented in the form:

$$(A_{i1}, a_{i1}) \wedge (A_{i2}, a_{i2}) \wedge \ldots \wedge (A_{1m}, a_{im}) \rightarrow c_{i1} \vee c_{i2} \vee \ldots \vee c_{im}$$

where the condition of the rule is an item and the consequent is a list of ranked class labels.

Definition 3: The actual occurrence (*ActOccr*) of a rule r in T is the number of cases in T that match r's condition.

Definition 4: The support count (*SuppCount*) of r is the number of cases in T that matches r's condition, and belong to a class c_i. When the item is associated with multiple labels, there should be a different *SuppCount* for each label.

Definition 5: A rule r passes the minimum support threshold (*MinSupp*) if for r, the $SuppCount(r)/|T| \geq MinSupp$, where $|T|$ is the number of instances in T.

Definition 6: A rule r passes the minimum confidence threshold (*MinConf*) if $SuppCount(r)/ActOccr(r) \geq MinConf$.

Definition 7: Any item in T that passes the *MinSupp* is said to be a frequent item.

3.2 Associative Classification

Generally, in association rule mining, any item that passes *MinSupp* is known as a frequent item. If the frequent item consists of only a single value, i.e. items < (A1, x1)>, < (A1, x2)> and < (A2, y1)> in Table 1, it is said to be a frequent single item. The frequent single items are inputs to the process of finding possible frequent pairs of items, the frequent pairs of items are input to discover frequent triples of items, and so on. Associative classification techniques generate frequent items by making multiple passes over the training data. In the first pass, they count the support of single items and determine whether it is frequent, and then in each subsequent pass, they start with items found to be frequent in the previous pass in order to produce new possible frequent items.

After frequent items have been discovered, associative classification methods derive a complete set of class-association-rules (CAR) for those frequent items that pass *MinConf*. These kinds of techniques are often called confidence-based methods, since they generate only the most obvious class per association rule. One of the first algorithms to bring up the idea of using an association rule for classification was proposed in [12]. It has been named CBA. It consists of two main phases; phase one implements the famous Apriori algorithm [2] in order to discover frequent items. Phase two involves building the classifier. Experimental results indicated that CBA produced classifiers which are competitive to popular learning methods like decision trees [13].

Table 1. Training data 1

RowIds	A1	A2	Single Class
1	x1	y1	c1
2	x1	y2	c2
3	x1	y1	c2
4	x1	y2	c1
5	x2	y1	c2
6	x2	y1	c1
7	x2	y3	c2
8	x1	y3	c1
9	x2	y4	c1
10	x3	y1	c1

4. MMAC

Our proposed algorithm consists of three phases: rules generation, recursive learning and classification. In the first phase, it scans the training data to discover and generate a complete CAR. In the second phase, MMAC proceeds to discover more rules that pass the *MinSupp* and *MinConf* thresholds from the remaining unclassified instances, until no further frequent items can be found. In the third phase, the rules sets derived at each iteration will be merged to form a global multi-class label classifier that will then tested against test data. Figure 1 represents a general description of our proposed method, which we will explain in more detail below. Training attributes can be categorical, i.e. attributes with limited distinct values, or continuous, i.e., real and integer attributes. For categorical attributes, we assume that all possible values are mapped to a set of positive integers. At the present time, our method does not treat continuous attributes.

Input: Training data, confidence and support (σ) thresholds
Output: A set of multi-label rules and the classification accuracy
Phase 1:
- ❖ Scan the training data *T* with *n* columns to discover frequent items
- ❖ Produce rules set_i by converting any frequent item that passes *MinConf* into a rule.
- ❖ Rank the rules set according to (confidence, support, …, etc).
- ❖ Evaluate the rules set_i in order to remove redundant rules.

Phase 2:
- ❖ Discard instances P_i associated with rules set_i
- ❖ Generate new training data $T' \leftarrow T - P_i$
- ❖ Repeat phase 1 on T' until no further frequent item is found

Phase 3:
- ❖ Merge rules sets generated at each iteration to produce a multi-label classifier
- ❖ Classify test objects

Figure 1. MMAC algorithm

4.1 Building the Classifier

4.1.1 Frequent Items Discovery and Rules Generation. To increase the efficiency of frequent items discovery and rules generation, MMAC employs a new technique based on an intersection method that has been presented in [21]. We have extended their method to accomplish classification. Our method scans the training data once to count the occurrences of single items, from which it determines those that pass *MinSupp* and *MinConf* thresholds, and stores them along with their occurrences (rowIds) inside fast access data structures. Then, by intersecting the rowIds of the frequent single items discovered so far, we can easily obtain the possible remaining frequent items that involve more than one attribute. The rowIds for frequent single items are useful information, and can be used to locate items easily in the training data in order to obtain support and confidence values for rules involving more than one item.

To clarify the picture, consider for instance frequent single items A and B, if we intersect the rowIds sets of A and B, then the resulting set should represent the tuples where A and B happen to be together in the training data, and therefore the classes associated with A^B can be easily located, in which the support and confidence can be accessed and calculated, which they will be used to decide whether or not A^B is a frequent item and a candidate rule in the classifier. Since the training data have been scanned once to discover and generate the rules, this approach is highly effective in runtime and storage because it does not rely on the traditional approach of discovering frequent items [1], which requires multiple scans.

Once an item has been identified as a frequent item, MMAC checks whether or not it passes the *MinConf* threshold. If the item confidence is larger than *MinConf*, then it will be generated as a candidate rule in the classifier. Otherwise, the item will be discarded. Thus, all items that survive *MinConf* are generated as candidate rules in the classifier.

4.1.2 Ranking of Rules and Pruning. In order to ensure a subset of effective rules form the classifier, a detailed ranking technique, which is shown in Figure 2, is presented. It reduces the need for random selection and aims to ensure that high confidence general and detailed

Definition 8: Given two rules, r_a and r_b, r_a precedes r_b if :
1. The confidence of r_a is greater than that of r_b
2. The confidence values of r_a and r_b are the same, but the support of r_a is greater than that of r_b
3. The confidence and support values of r_a and r_b are the same, but r_a has larger *ActOccr* than r_b in the training data
4. Both confidence and support and *ActOccr* values of r_a and r_b are the same, but r_a has fewer conditions in its left hand side (LHS) than that of r_b
5. All above criteria are identical for r_a and r_b, but r_a was generated before r_b

Figure 2. Rules ranking technique of MMAC

rules are kept for classification. Pruning takes place by discarding any item that has a support value less than the *MinSupp*, and a confidence value less than the *MinConf* threshold. Another pruning of the rules occurs in the rule evaluation which we will discuss in the next subsection.

4.1.3 Rules Evaluation. A rule *r* is said to be significant if and only if it covers at least one training instance. After a set of rules is generated and ranked, an evaluation step takes place to test each rule in order to remove redundant rules. If a rule correctly classifies at least a single instance, then it will be marked as a survivor, and a good candidate rule. In addition, all instances correctly classified by it will be deleted from the training data. In the case that a rule has not classified any training instance, it will then be removed from the rules set.

4.1.4 Recursive Learning. For given training instances *D*, other associative classification algorithms like CBA and CPAR derive a single label rules set, and form a default class for the remaining unclassified instances in *D*. On the other hand, the MMAC derives more than one rules set, and merges them to form a multi-label classifier. For *D*, the proposed method produces the first rules set in which each rule is associated with the most obvious class label. Once this rules set is generated, all training instances associated with it will be discarded. The remaining unclassified instances will then become new training data, say D', and the MMAC checks whether there are still any more frequent items remaining undiscovered in D' (rules derived from *D* which may be associated with more than one class label). If so, a new set of rules will be generated from D', and the remaining unclassified instances in D' will form new training data, and so forth. The algorithm proceeds with learning until no more frequent items could be discovered. At that stage, any remaining unclassified instance will form a default class.

This process results in learning from several subsets of the original training data and generating few rules sets. Consider for example the training data shown in Table 2. Assume that the *MinSupp* and *MinConf* have been set to 20% and 40%, respectively. At the first iteration, MMAC derives a set of rules that covers the instances that are not underlined in Table 2, which eventually will be discarded at the end of the iteration. The remaining unclassified instances which are underlined will represent the new training data for iteration two, in which two more rules will be learned and produced to form the second rules set. When the learning process is finished, a merging of the rules sets which have been produced at iterations 1 and 2 will be performed to obtain a global multi-label classifier. In many cases, a rule will be presented in more than one rules set and is associated with different class labels like item <(A1, x1)> which has two representations in Table 2, one with class label "c1" in rules set 1, and one with class label "c2" in rules set 2. A good question will be how one can rank the class labels in a rule to represent this item.

Table 2. Training data

RowId	A1	A2	Class
1	x1	y1	c1
2	x1	y2	c2
3	x1	y1	c2
4	x1	y2	c1
5	x2	y1	c2
6	x2	y1	c1
7	x2	y3	c2
8	x1	y3	c1
9	x2	y4	c1

4.1.5 Ranking of Class Labels. Definition 9: A class label $l_1 < l_2$, also known as l_1, precedes l_2 in a rule *r* if it has a larger representation in the training data. Consider, for example, an item < (A, a)(B, b)> which is associated with three labels ("c1", "c2", "c3"). Assume that it has 100 representation in the training data in which it is associated with labels "c1", "c2" and "c3", 50, 30, and 20 times. Moreover, assume that this item has passed *MinSupp* and *MinConf* thresholds when associated with "c1", "c2" and "c3". MMAC ranks these labels based on their number of occurrences ("c1"<"c2"<"c3"), and a rule will be presented for this instance in the following form: $(A,a) \wedge (B,b) \rightarrow c1 \vee c2 \vee c3$.

4.2 Classification

In classification, let *R* be the set of generated rules and *T* the training data. The basic idea of the proposed method is to choose a set of high confidence rules in *R* to cover *T*. In classifying a test object, the first rule in the set of rules that matches the test object condition classifies it. This process ensures that only the highest ranked rules classify test objects.

4.3 Comparison of MMAC and CBA

CBA and MMAC were applied on the training data shown in Table 1 by using a *MinSupp* of 20% and *MinConf* of 40% to illustrate the effectiveness of the rules sets derived by both algorithms. Table 3a represents the classifier generated by CBA, which consists of two rules and covers 8 training instances, which are (1, 2, 3, 4, 5, 6, 8, 10). The remaining two instances form a default class rule that covers 20% of the entire data.

Table 3b represents the classifier produced by MMAC on the same training data, in which more rules have been discovered, i.e. two more rules than the CBA classifier. The rules extracted will be then ranked and merged to form a multi-class label classifier in which some of its rules are associated with a list of ranked class labels. In this particular example, there is only one rule derived by our proposed algorithm from Table 1 that has

Table 3a. CBA classifier

RuleId	Frequent Item	Support	Confidence	Class Label
1	x1	3/10	3/5	C1
3	y1	3/10	3/5	C1
default				C2

Table 3b. MMAC classifier

RuleId	Frequent Item	Support	Confidence	Class Label
1a	x1	3/10	3/5	C1
1b	x1	2/10	2/5	C2
2	x2	2/10	2/4	C2
3	y1	3/10	3/5	C1
default				C1

multiple labels which is $(A1, x1) \rightarrow c1 \vee c2$. The MMAC classifier covers nine training instances, and the remaining one forms the default class. Unlike the CBA algorithm that was unable to produce rules with multiple labels, our proposed method generates rules that can predict multiple labels. Moreover, the default rule of MMAC classifier covers only 10% of the training data, and therefore it has less impact on the classification of unseen data that may significantly affect the accuracy in the classification, and could lead to deterioration in the overall error rate.

Generally, the main differences between MMAC and other associative algorithms are the following:

- MMAC presents not only a single class classifier but also a multi-label one, in which each instance is associated with its ranked list of classes.
- Other associative classification techniques often use multiple passes to discover frequent items. Alternatively, MMAC uses a new technique for discovering the rules, which requires only one scan.
- MMAC introduces a detailed rule ranking technique that minimises randomisation when a choice point among two or more rules occurs in the rules ranking process.
- The proposed method presents a recursive learning phase that discovers more rules, and minimises the role of the default class in classifying test objects.
- Other associative techniques discover frequent items in one phase, and generate the rules in a separate phase. The proposed method discovers and generates rules in one phase.

4.4 Evaluation Measures

Since multi-label classification has been investigated mostly in text categorisation, there is very little work conducted on developing evaluation measures for its classifiers. There are no standard evaluation techniques applicable to the multi-label classification problems. Moreover, the right measure is often problematic and depends heavily on the features of the conducted problem, such as those used in [3]. In this section, we introduce three evaluation measures suitable for the majority of binary, multi-class and multi-label classification problems.

4.4.1 Top-label. This evaluation measure takes into consideration only the top-ranked class label and ignores any other labels associated with an instance. For traditional classification task where there is only one class label to assign to the test object, and given an instance and its associated class label <*d*, *y*>, a classifier *H* predicts a list of ranked class labels $Y_j = \left\langle Y_j^{\,1}, Y_j^{\,2}, ..., Y_j^{\,k} \right\rangle$, if the predicted first class label matches the true class label *y* of the instance, i.e. $Y_j^{\,1} = y$, then the classification is correct. The top-label method estimates how many times the top-ranked class label is the correct class label. So, for a set of single-class instances $I = < (x_1, y_1), (x_2, y_2), \ldots, (x_m, y_m)>$, the top-label is $1/m \sum_{j=1}^{m} \left(Y_j^{\,1} = y_j\right)$, where *m* represents the number of instances.

4.4.2 Any-label. This evaluation technique measures how many times any of the predicted labels of an instance matches the actual class label in all cases of that instance in the test data. If any of the predicted class labels of an instance *d* matches the true class label *y*, i.e. $Y_j^{\,i} = y$, then the classification is correct. For a set of single-class instances $I = < (x_1, y_1), (x_2, y_2), \ldots, (x_m, y_m)>$, the any-label is $1/m \sum_{j=1}^{m} \left(Y_j^{\,i} = y_j\right)$, where *m* represents the number of instances.

4.4.3 Label-weight. This technique enables each predicted label for an instance to play a role in classifying a test case based on its ranking, and therefore it could be considered as a multi-label evaluation measure. An instance may belong to several class labels, each one associated with it by a number of occurrences in the training data. Each class label can be assigned a weight according to how many times that label has been associated with the instance. Let rule r_j be associated with a list of ranked labels $Y_j = \left\langle Y_j^{\,1}, Y_j^{\,2}, ..., Y_j^{\,k} \right\rangle$, and denote w_j^k as the set of weights for Y_j where $\sum_{j=1}^{k} w_j^{\,k} = 1$. A classifier *H* is defined as $D \rightarrow Y$ such that it assigns a weight of the correct class label to an instance as $H(d) = W^i$, where $d \in D$, and $W^i \in W_j^{\,k}$. For a set of single-class instances $I = < (x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)>$, the label-weight

is $\frac{1}{m}\sum_{k=1}^{m}\left(w^{i} * \langle \delta(H(d_i), y_i \rangle\right)$, where $\delta(x,y) = \begin{cases} 1 & if \quad x = y \\ 0 & if \quad x \neq y \end{cases}$.

For example, if an item (A ,a) is associated with class labels "c1", "c2" and "c3", 7, 5 and 3 times, respectively, in the training data. Each class label will be assigned a weight, i.e. 7/15, 5/15, and 3/15, respectively, for labels "c1", "c2" and "c3". This technique assigns the predicted class label weight to the case if the predicted class label matches the case class label. For instance if label "c2" of item (A, a) matches a case in the test data that has "c2" as its class, then the case will be considered a hit, and 5/15 will be assigned to the case.

5. Experimental Results

We investigated our approach against 19 different datasets from [20] as well as a different datasets for forecasting the behaviour of an optimisation heuristic within a hyperheuristic framework [5, 16]. Stratified ten-fold cross-validation was used to derive the classifiers and error rates in the experiments. Cross-validation is a standard evaluation measure for calculating error rate on data in machine learning. Three popular classification techniques a decision tree rule (PART), RIPPER and CBA have been compared to MMAC in terms of classification accuracy, in order to evaluate the predictive power of the proposed method.

The choice of such learning methods is based on the different strategies they use to generate the rules. Since the chosen techniques are only suitable for traditional classification problems where there is only one class assigned to each training instance, we therefore used classification accuracy derived by only the top-label evaluation measure for fair comparison.

All experiments were conducted on a Pentium IV 1.6 GH PC. The experiments of PART and RIPPER were conducted using the *Weka* software system [20]. *Weka* stands for Waikato Environment for Knowledge Analysis. It is an open java source code for the machine teaching community that includes implementations of different methods for several different data mining tasks such as classification, clustering, association rule and regression. CBA experiments were conducted using a VC++ implementation version provided by [19]. Finally, MMAC was implemented using Java.

We have evaluated 19 selected datasets from *Weka* data collection [20], in which, a few of them (6) were reduced by ignoring their integer and/or real attributes. Several tests using ten-fold cross-validation have been performed to ensure that the removal of any real/integer attributes from some of the datasets does not significantly affect the classification accuracy. To do so, we only considered datasets where the error rate was not more than 6% worse than the error rate obtained on the same dataset before the removal of any real/integer attributes. Thus, the ignored attributes do not impact on the error rate too significantly.

Many studies have shown that the support threshold plays a major role in the overall classification accuracy of the set of rules produced by existing associative classification techniques [9, 12]. Moreover, the support value has a larger impact on the number of rules produced in the classifier and the processing time and storage needed during the algorithm rules discovery and generation. From our experiments, we noticed that the support rates that ranged between 2% to 5% usually achieve the best balance between accuracy rates and the size of the resulted classifiers. Moreover, the classifiers derived when the support was set to 2% and 3% achieved high accuracy, and most often better than that of decision trees rule (PART), RIPPER and CBA. Thus, the *MinSupp* was set to 3% in the experiments. The confidence threshold, on the other hand, is less complex and does not have a large effect on the behaviour of any associative classification method as support value, and thus it has been set to 30%.

Table 4 represents the classification rate of the classifiers generated by PART, RIPPER, CBA and MMAC against 19 benchmark problems from *Weka* data

Table 4. Classification accuracy of PART, RIPPER, CBA and MMAC

Dataset	PART	RIPPER	CBA	MMAC
Tic-Tac	92.58	97.54	98.60	99.29
Contact-lenses	83.33	75.00	66.67	79.69
Led7	73.56	69.34	72.39	73.20
Breast-cancer	71.32	70.97	68.18	72.10
Weather	57.14	64.28	85.00	71.66
Heart-c	81.18	79.53	78.54	81.51
Heart-s	78.57	78.23	71.20	82.45
Lymph	76.35	77.70	74.43	82.20
Mushroom	99.81	99.90	98.92	99.78
primary-tumor	39.52	36.28	36.49	43.92
Vote	87.81	87.35	87.39	89.21
CRX	84.92	84.92	86.75	86.47
Sick	93.90	93.84	93.88	93.78
Balance-scale	77.28	71.68	74.58	86.10
Autos	61.64	56.09	35.79	67.47
Breast-w	93.84	95.42	94.68	97.26
Hypothyroid	92.28	92.28	92.29	92.23
zoo	91.08	85.14	83.18	96.15
kr-vs-kp	71.93	70.24	42.95	68.75

collection. The accuracy of MMAC has been derived using the top-label evaluation measure. Our algorithm outperforms the rule learning methods in terms of accuracy rate, and the won-loss-tied records of MMAC against PART, RIPPER and CBA 13-6-0, 15-4-0 and 15-4-0, respectively.

The evaluation measures of MMAC have been compared on 9 solution runs produced by the Peckish hyperheuristic [5] with regard to accuracy, and number of rules produced. Figures 3a and 3b represent the relative prediction accuracy that indicates the difference of the classification accuracy of MMAC evaluation measures with respect to those derived by CBA and PART, respectively. In other words, how much better or worse MMAC measures perform with respect to CBA and PART learning methods. The relative prediction accuracy numbers shown in Figures 3a and 3b are conducted using the formula $\frac{(Accuracy_{MMAC} - Accuracy_{PART})}{Accuracy_{PART}}$ and $\frac{(Accuracy_{MMAC} - Accuracy_{CBA})}{Accuracy_{CBA}}$ respectively. After analysing the charts, we found out that there is consistency between the top-label and label-weight measures, since both of them consider only one class in the prediction. The top-label takes into account the top-ranked class, and the label-weight considers only the weight for the predicted class that matches the test case. Thus, both of these evaluation measures are applicable to traditional single-class classification problems. On the other hand, the any-label measure considers any class in the set of the predicted classes as a hit whenever it matches the predicted class regardless of its weight or rank. Is should be noted that, the relative accuracy of MMAC evaluation methods against dataset number 8 in Figure 3a and 3b, is negative since CBA and PART achieved a higher classification rate against this particular dataset.

A comparison of the knowledge representation produced by our method, PART and CBA has been conducted to evaluate the effectiveness of the set of rules derived. Figure 4 represents the classifiers generated form the hyperheuristic datasets. Analysis of the rules sets indicated that MMAC derives a few more rules than PART and CBA for the majority of the datasets. In particular, the proposed method produced more rules than PART and CBA on 8 and 7 datasets, respectively. A possible reason for extracting more rules is based on the recursive learning phase that MMAC employs to discover more hidden information that most of the associative classification techniques discard, since they only extract the highest confidence rule for each frequent item that survives *MinConf*.

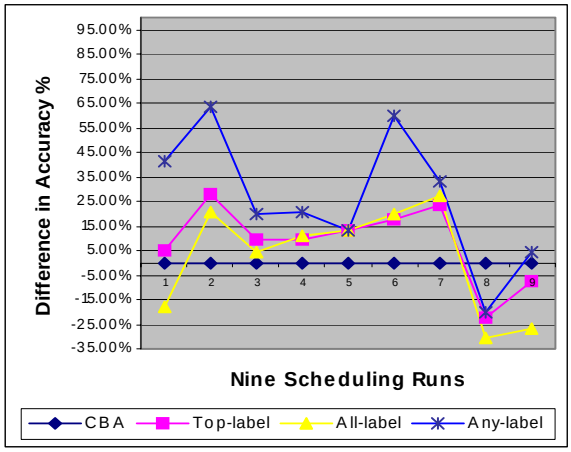

Figure 3a. Difference of accuracy between MMAC evaluation measures and CBA algorithm.

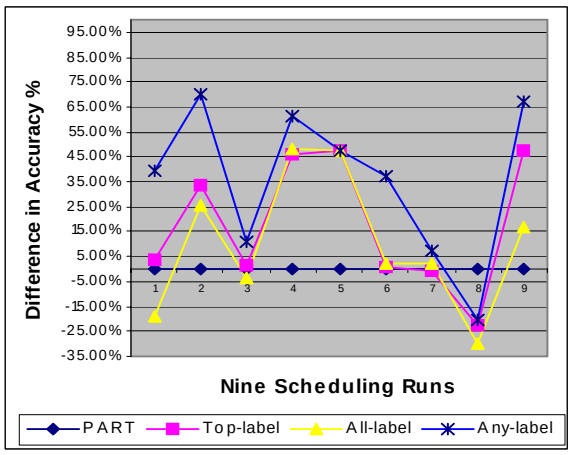

Figure 3b. Difference of accuracy between MMAC evaluation measures and PART algorithm.

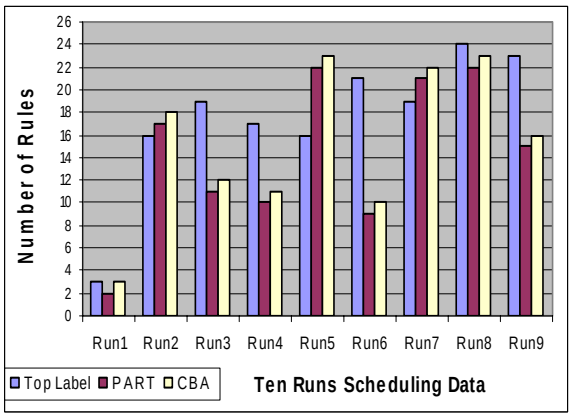

Figure 4. Classifier sizes of MMAC (top-label), PART and CBA algorithms against the scheduling data.

6. Conclusions

A new approach for multi-class, and multi-label classification has been proposed that has many distinguishing features over traditional and associative classification methods in that it (1) produces classifiers that contain rules with multiple labels, (2) presents three evaluation measures for evaluating accuracy rate, (3) employs a new method of discovering the rules that require only one scan over the training data, (4) introduces a ranking technique which prunes redundant rules, and ensures only high effective ones are used for classification, and (5) integrates frequent items set discovery and rules generation in one phase to conserve less storage and runtime. Performance studies on 19 datasets from *Weka* data collection and 9 hyperheuristic scheduling runs indicated that our proposed approach is effective, consistent and has a higher classification rate than the-state-of-the-art decision tree rule (PART), CBA and RIPPER algorithms. In further work, we anticipate extending the method to treat continuous data and creating a hyperheuristic approach to learn "on the fly" which low-level heuristic method is the most effective.

References

[1] R. Agrawal, T. Amielinski and A. Swami. Mining association rule between sets of items in large databases. *In Proceeding of the 1993 ACM SIGMOD International Conference on Management of Data*, Washington, DC, May 26-28 1993, pp. 207-216.

[2] R. Agrawal and R. Srikant. Fast algorithms for mining association rule. *In Proceeding of the 20th International Conference on Very Large Data Bases,* 1994, pp. 487 – 499.

[3] M. Boutell, X. Shen, J. Luo and C. Brown. Multi-label semantic scene classification. Technical report 813, Department of Computer Science, University of Rochester, Rochester , NY 14627 & Electronic Imaging Products R & D, Eastern Kodak Company, September 2003.

[4] A. Clare and R.D. King. Knowledge discovery in multi-label phenotype data. In L. De Raedt and A. Siebes, editors, *PKDD01*, volume 2168 of Lecture Notes in Artificial Intelligence, Springer - Verlag, 2001, pp. 42-53.

[5] P. Cowling and K. Chakhlevitch. Hyperheuristics for Managing a Large Collection of Low Level Heuristics to Schedule Personnel. *In Proceeding of 2003 IEEE conference on Evolutionary Computation*, Canberra, Australia, 8-12 Dec 2003.

[6] R. Duda, P. Hart, and D. Strok. *Pattern classification.* Wiley, 2001.

[7] E. Frank and I. Witten. Generating accurate rule sets without global optimisation. In Shavlik, J., ed., *Machine Learning: In Proceedings of the Fifteenth International Conference*, Madison, Wisconsin. Morgan Kaufmann Publishers, San Francisco, CA, pp. 144-151.

[8] J. Furnkranz. Separate-and-conquer rule learning. *Technical Report* TR-96-25, Austrian Research Institute for Artificial Intelligence, Vienna, 1996.

[9] W. Li, J. Han and J. Pei. CMAR: Accurate and efficient classification based on multiple class association rule. In *ICDM'01*, San Jose, CA, Nov. 2001, pp. 369-376.

[10] T. Joachims. Text categorisation with Support Vector Machines: Learning with many relevant features. *In Proceeding Tenth European Conference on Machine Learning*, 1998, pp. 137-142.

[11] T. S. Lim, W. Y. Loh and Y. S. Shih. A comparison of prediction accuracy, complexity and training time of thirty-three old and new classification algorithms. *Machine Learning*, 39, 2000.

[12] B. Liu, W. Hsu and Y. Ma. Integrating Classification and association rule mining. *In KDD '98*, New York, NY, Aug. 1998.

[13] J.R. Quinlan. C4.5: *Programs for Machine Learning*. San Mateo, CA: Morgan Kaufmann, San Francisco, 1993.

[14] J.R. Quinlan. Generating production rules from decision trees. *In Proceeding of the 10th International Joint Conferences on Artificial Intelligence*, Morgan Kaufmann, San Francisco, 1987, pp. 304-307.

[15] R. Schapire and Y. Singer, "BoosTexter*:* A boosting-based system for text categorization," *Machine Learning*, vol. 39, no. 2/3, 2000, pp. 135-168.

[16] F. Thabtah, P. Cowling and Y. Peng. Comparison of Classification techniques for a personnel scheduling problem. *In Proceeding of the 2004 International Business Information Management Conference, Amman, July 2004*.

[17]Y. Yang. An evaluation of statistical approaches to text categorisation. Technical Report CMU-CS-97-127, Carnegie Mellon University, April 1997.

[18] X. Yin and J. Han. CPAR: Classification based on predictive association rule. In *SDM 2003*, San Francisco, CA, May 2003.

[19]CBA:http://www.comp.nus.edu.sg/~dm2/ p_download.html

[20] Weka: Data Mining Software in Java: http://www.cs.waikato.ac.nz/ml/weka.

[21] M. J. Zaki, S. Parthasarathy, M. Ogihara, and W. Li. New algorithms for fast discovery of association rules. *In Proceedings of the 3rd KDD Conference*, Aug. 1997, pp.283-286.