Rule Induction for Classification Using Multi-objective Genetic Programming

A.P. Reynolds and B. de la Iglesia

School of Computing Sciences,
University of East Anglia,
Norwich

Abstract. Multi-objective metaheuristics have previously been applied to partial classification, where the objective is to produce simple, easy to understand rules that describe subsets of a class of interest. While this provides a useful aid in descriptive data mining, it is difficult to see how the rules produced can be combined usefully to make a predictive classifier. This paper describes how, by using a more complex representation of the rules, it is possible to produce effective classifiers for two class problems. Furthermore, through the use of multi-objective genetic programming, the user can be provided with a selection of classifiers providing different trade-offs between the misclassification costs and the overall model complexity.

1 Introduction

Earlier work by the authors [1,2,3,4] described the application of multi-objective metaheuristics to the problem of partial classification [5]. This problem is the search for simple rules, that represent 'strong' or 'interesting' descriptions of a specified class, or subsets of the specified class, even when that class has few representative cases in the data. These rules are of the form

- `if age` $\geq$ `28 and firstDegree = mathematics and attendance` $\geq$ `90% then result = distinction`

where the antecedent is a conjunction of simple attribute tests and the consequent, describing the class of interest, is the same for all rules generated.

Such simple rules may have high confidence, in that the rule produces few false positives. They may have high coverage, in that they describe a high proportion of the class of interest. Multi-objective metaheuristics can be used to produce different trade-offs between confidence and coverage. However, this simple rule representation is insufficiently descriptive to produce an individual rule with both high confidence and coverage.

In other work, Ghosh and Nath [6] used a multi-objective genetic algorithm for association rule mining, optimizing the accuracy, comprehensibility and interestingness of the rules produced. Association rules are similar to that shown above, but with tests usually limited to equalities and with an unconstrained consequent that may be any conjunction of such tests.

S. Obayashi et al. (Eds.): EMO 2007, LNCS 4403, pp. 516–530, 2007.

Both partial classification and association rule mining fall primarily into the category of *descriptive* data mining. A natural question is, can this work with simple descriptive rules be extended or modified to create *understandable* and highly *predictive* models that can classify previously unseen records? There are two approaches to take to this task: select a subset of the simple rules created to act as a classifier or increase the expressiveness of the rule representation.

Ishibuchi et al. [7,8] take the first of these approaches. In their work, a multi-objective algorithm is used to select a small subset of association rules produced by another algorithm, minimizing rule set complexity and error rate. However, this approach has a number of disadvantages:

- A good rule set may contain individuals that are far from the Pareto-front, according to the objectives of rule confidence and support [8]. Hence a very large set of simple rules must be created. For example, from a small training set of 342 records, 17070 classification rules were extracted, from which the multi-objective metaheuristic selected no more than 25 [8].
- Ideally, a record should assigned to a class if *any* of the rules in the rule set make this prediction. Then each rule provides a useful description of a subset of the data. However, in practice rules may make conflicting predictions. To handle conflicts, Ishibuchi et al. essentially create a decision list rather than a simple rule set [9], with rules lower on the list being used only when none of the higher rules apply. Converting such a list to a simple rule set reveals added complexity hidden in the decision list representation.
- A set of rules may not be the simplest way in which to represent a model of the data. This is illustrated by the example given in section 2.

In this paper we take the alternative approach of using a more expressive rule representation, specifically by using expression trees. While there is much literature on the use of genetic programming to optimize trees for the purposes of classification, this mostly concentrates on the optimization of decision trees, e.g. [10,11,12,13]. In particular, Mugambi and Hunter [14] apply multi-objective genetic programming to decision tree induction, optimizing both tree accuracy and tree simplicity. However, decision trees are different to the expression trees developed in this paper, with internal nodes that define partitions of the data and leaf nodes that indicate class membership. While rules may easily be extracted from such decision trees, we concentrate in this paper on the direct production and optimization of rules.

The work of Setzkorn and Paton [15,16] is perhaps more relevant to this paper, applying multi-objective genetic programming directly to fuzzy rule induction. However, internal nodes are restricted to two fuzzy forms of the boolean 'and' operation and the algorithm optimizes sets of these rules.

Section 2 provides details of the expression tree representation used. This representation is manipulated by a multi-objective metaheuristic to produce models with different trade-offs between model complexity and model accuracy (section 3). Section 4 describes the experiments performed and presents the results of using this approach. Finally, section 5 presents some conclusions and section 6 describes areas of further research.

2 Rule Representation and Manipulation

2.1 Attribute Tests

The algorithm described in this paper manipulates rules of the form

$$antecedent \rightarrow consequent,$$

where both antecedent and consequent are constructed from *attribute tests*. Three different types of attribute test (AT) are used:

Value: e.g. `colour = red`,
Inequality: e.g. `colour` $\neq$ `green`,
Binary partition: e.g. `age` $\geq$ `42` or `height` $\leq$ `156`.

Value and inequality tests are used exclusively on categorical fields, while binary partition tests can only be used with a numeric field. A more detailed description of ATs and some alternative AT types may be found in previous work [4].

2.2 Attribute Test Representation

Values occurring in each field are stored in reference arrays. The index values, rather than the values from the database, are used in the representation of the ATs as shown in figure 1. Each AT type is represented and mutated as follows:

Value/Inequality: Represented by the categorical field number and the category index. A mutation changes the category index to random value.
Binary partition: Represented by the numeric field number, the index of the bound value and a flag indicating the type of bound. A mutation changes the index of the bound by up to 20% of the number of values that occur in the database, while ensuring that the AT does not become trivial or impossible to satisfy. The type of the bound is not changed.

The algorithm manipulates rule antecedents constructed from combinations of different ATs. The consequent is fixed, representing the *class of interest*. Any unseen record that matches the rule antecedent is predicted to belong to the class of interest. However, in contrast to previous work, any record that does not match the rule antecedent is predicted to belong to some other class.

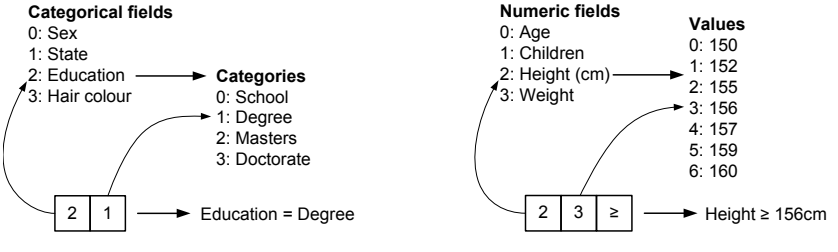


Fig. 1. Representation of a categorical value AT and a binary partition AT

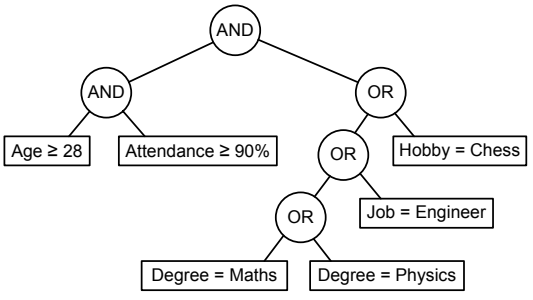


Fig. 2. A binary boolean expression tree

2.3 Rule Trees

ATs are combined in expression trees that represent the rule antecedent, as shown in figure 2. Leaf nodes contain ATs, while internal nodes contain a boolean operator. These operators have been restricted to be either 'or' or 'and' in the experiments reported here, though it is easy to include additional boolean operators if desired. Notice that the tree contains only 6 ATs, one per leaf node, rather than the 12 required to represent this antecedent as a set of simple rules. In order to simplify the genetic operators used, binary trees are used. This increases the number of internal nodes, but leaves the number of leaf nodes unaltered. Note that such trees may easily be converted into rule sets if this is how the client wishes to view the models produced.

2.4 Genetic Operators

Initialization: The population is initialized with randomly generated balanced trees of depth two, where the root node is considered to be at depth zero.

Mutation: During mutation, there is a 50% probability that a random AT is mutated, a 25% probability that an AT and its parent node is removed and a 25% probability that a random AT with a new internal node is added.

Crossover: Subtree crossover [17] proceeds by selecting a node at random in each tree and swapping the subtrees headed by these nodes. As is commonplace in genetic programming, a choice between crossover and mutation is made when creating new solutions, rather than both being applied probabilistically.

2.5 Bloat and Rule Simplification

It is well established that solutions generated during genetic programming tend to suffer from *bloat*, i.e. they grow excessively, often without any great improvements in fitness [17,18]. Such solutions usually contain redundant sections, referred to as *introns*. It has been suggested that the occurrence of such introns

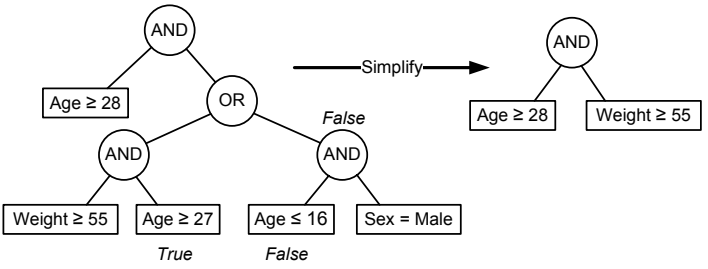


Fig. 3. Simplifying the right hand subtree by assuming a value of 'true' for the left hand subtree

should not be hindered [19], as they protect solutions from the more destructive effects of crossover. However, bloat leads to an increase in evaluation times and may also interfere with finding better solutions, since time is spent manipulating the introns rather than useful code[20]. Langdon and Poli [18] suggest that "since no clear benefits offset these detrimental effects, practical solutions to the code bloat phenomenon are necessary to make GP and related search techniques feasible for real-world applications".

In this paper, bloat is counteracted in three ways. Firstly, although the simplicity of a rule is already considered as an objective of the problem, counteracting bloat provides an additional reason for using this objective. Using rule simplicity in this way has been found to be effective in reducing code bloat in the literature [20]. Secondly, rule simplification is performed, removing redundant sections from rules. Finally, since these measures alone are insufficient to eliminate bloat, a simple limit on rule size is imposed. If, after simplification, a rule exceeds this AT limit, ATs and their parent nodes are removed until the constraint is satisfied. In this paper, this limit has been set to 20 ATs, in order to demonstrate the effect of rule size on misclassification costs. In practice, this limit is likely to be set to a smaller value, since 20 AT rules are too large for easy human comprehension and smaller rules can be evaluated more quickly.

Figure 3 illustrates the rule simplification performed. Here, the right hand subtree need only be evaluated if the left hand subtree evaluates to 'true'. Therefore, assuming that the left hand subtree is 'true', we determine which nodes in the right hand subtree must be 'false' or must be 'true', simplifying as shown. Similarly, if the root node of a tree contains the boolean operation 'or', the right hand subtree need only be evaluated if the left evaluates to false. Note that such simplifications can still be made if the left hand subtree has more than one node. All simplifications of this form are made at every internal node in the rule tree.

Note that this does not ensure that the rule is as simple as possible. For example, we do not currently use the distributivity law to simplify rules. Similarly, given three colours, red, green and blue, the rule antecedent `colour` $\neq$ `red and colour` $\neq$ `blue` could be simplified to `colour = green`.

3 Rule Evaluation

If the understandability of the rule is not a concern, then the overall aim is to produce a rule that, when applied to *previously unseen* data, minimizes the expected costs of misclassification. In practice, we minimize total misclassification costs on the training data and rule complexity.

There are two reasons for minimizing rule complexity:

- The more complex a rule is permitted to be, the more likely it is that *overfitting* [9] will occur, making accuracy on training data an unreliable measure of accuracy on unseen data.
- Simple rules are easier to understand. If part of the aim of classification is the extraction of knowledge, for example when attempting to discover patterns in scientific data, it has been argued [21] that the classifier *must* be comprehensible to a human expert. Also, a client is more likely to use a classifier if he understands it.

3.1 Misclassification Costs

Rule antecedents generated by our algorithm describe the class of interest, with any record not matching the antecedent assumed to be not in the class. A 'false positive' occurs when the rule predicts that a record belongs to the class of interest when it does not, and a 'false negative' occurs when the rule predicts that a records does not belong to the class of interest when it does.

Using equal false positive and false negative costs results in the minimization of the simple error rate. This commonly used measure of performance is not always appropriate, for example when a false positive results in additional labour while a false negative results in injury or death. Also, if the class of interest is small, minimizing the simple error rate may merely result in the rule that predicts that all records are not in the class. If the class is truly of special interest, the false negative cost must be increased to discover of patterns of interest.

In the experiments reported, we have also used the balanced error rate:

$$BER = \frac{1}{2}\left(\frac{\text{No. of false positives}}{\text{Total number of positives}} + \frac{\text{No. of false negatives}}{\text{Total number of negatives}}\right)$$

Other experiments were performed with a false positive cost of 1 and a false negative cost of 10.

3.2 Measuring Rule Complexity

In most of our experiments, the complexity of a rule is given by the number of ATs in the rule tree. This simplifies the comparison of results obtained with different parameter settings and ensures that the complement of the rule, describing records that *do not* belong to the selected class, has the same complexity. (Note that the same could not be said if the categorical inequality AT type was not

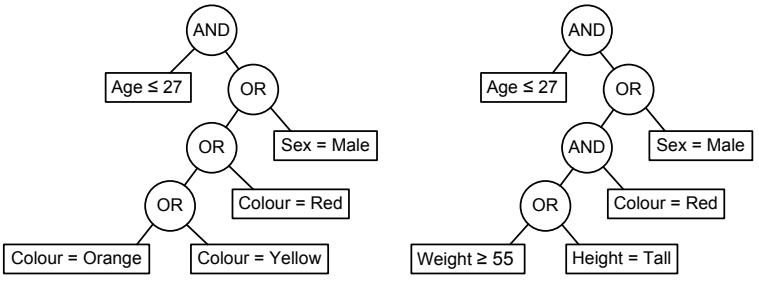


Fig. 4. Rules of the same size need not be equally understandable to the human reader

used.) While counting the ATs has the advantage of simplicity, it may not accurately portray the ease with which a rule can be understood. In figure 4 the first tree is easier to comprehend than the second, due to the repeated use of the both the same operator and the same attribute in the right hand subtree.

In practice, rule complexity also depends upon the client and upon his or her preferences regarding rule presentation. For example, the client may prefer to see the second rule in figure 4 presented as the following rule set.

- `if age ≤ 27 and sex = male then...`
- `if age ≤ 27 and colour = red and weight ≥ 55 then...`
- `if age ≤ 27 and colour = red and height = tall then...`

In this case, the rule complexity may be given as eight ATs rather than the five in the original rule tree. Fortunately, the algorithm used can easily be adapted to use this and other measures of rule complexity and an example is given at the end of section 4.

4 Experimentation and Results

4.1 Data

Rules were extracted from five datasets from the UCI machine learning repository [22]: the Adult, Forest Cover Type, Contraception, Breast Cancer (Wisconsin) and the Pima Indians Diabetes datasets. Any records containing missing data were removed prior to applying the algorithm. Table 1 describes the datasets in more detail.

The Adult dataset and on 10,000 records selected at random from the Cover Type dataset were used to tune algorithm parameters (section 4.2), before running the algorithm on all five of the datasets.

4.2 Algorithm and Parameter Tuning

The multi-objective metaheuristic selected for the optimization of rule trees was NSGA II [23,24], using an external store to hold the best solutions found. This

Table 1. Datasets used and classes of interest

Name	Records	Fields: Numeric	Fields: Categorical	Classes	Class of Interest	Class Prevalence
Adult	45222	6	8	2	Salary > $50,000	24.8%
Cover type	581012	10	2	7	Spruce-fir	36.5%
Contraception	1473	5	4	3	No contraceptive use	42.7%
Breast cancer	683	10	0	2	Malignant	35.0%
Pima Indians	532	7	0	2	Test positive	33.3%

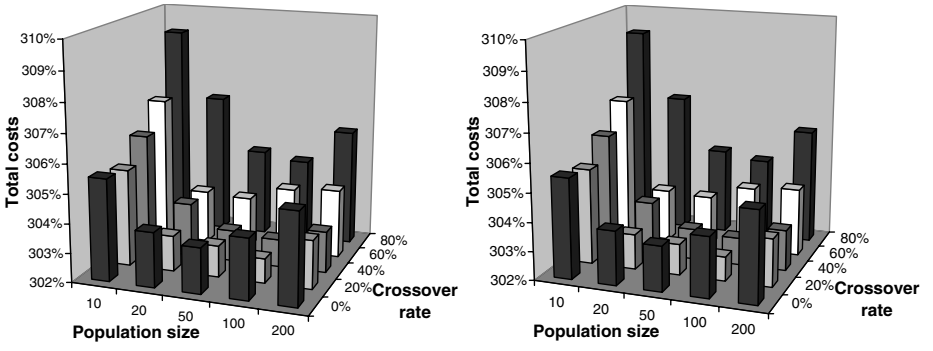


Fig. 5. Comparison of different crossover rates and population sizes for the Adult and Cover Type datasets, using equal false positive and false negative costs

algorithm has been shown to be an effective multi-objective optimizer, both in general and when optimizing rules [1,2,3,4]. Parameter tuning was performed on the Adult dataset and on 10,000 records selected at random from the Cover Type dataset, minimizing the simple error rate and the number of ATs.

The number of different parameters and potential variations of the algorithm made the cost of exhaustive experimentation with parameter settings prohibitive. Instead, effort was focused on finding the best values for crossover rate and population size only. Experiments were performed with six population sizes, 10, 20, 50, 100, 200 and 500, and six crossover rates, 0%, 20%, 40%, 60%, 80% and 100%. Each experiment consisted of 30 runs of the algorithm, with 200,000 rule evaluations per run. Results were compared by summing the error rates of the best rule at each level of rule complexity, up to 20 ATs. This is equivalent to comparing on the dominated area in the objective space [25]. Mean results are shown in figure 5. Results with a population size of 500 or a crossover rate of 100% are omitted since these were considerably worse than the results displayed. Similar graphs were obtained when minimizing the balanced error rate and when the false negative cost was increased to ten times the false positive cost. In each case, best performance was obtained using a crossover rate of 20% or 40% and a population size of 100 for the Adult dataset and 50 for the Cover Type dataset.

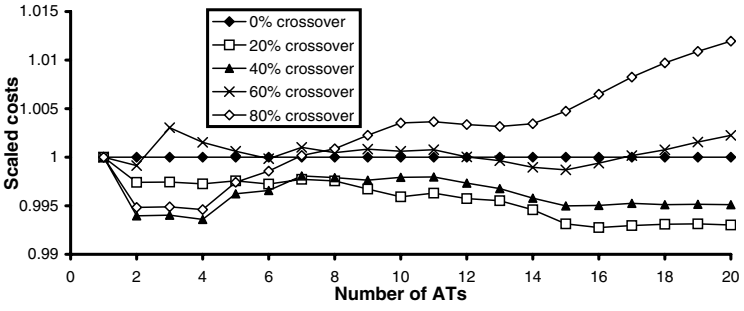


Fig. 6. Comparison of performance at different crossover rates, using a population of 100 rules, the adult dataset and simple error rate. Results are scaled with respect to the performance at 0% crossover, 100% mutation.

Figure 6 shows how performance varies with the crossover rate, at different levels of rule complexity. While a certain amount of crossover is required to produce good results, too much crossover (and hence too little mutation) results in degraded performance for large rules. There are two possible reasons for this:

- Crossover applied to large rules results in major, disruptive changes when subtle modifications may be more appropriate. Adapting the crossover operator to be biased towards smaller changes is a matter for further research.
- A loss of diversity in the population, early in the search process, requires the use of mutation to reintroduce useful ATs and subtrees. This would explain the very poor performance obtained when using crossover only.

4.3 Training, Validation, Selection and Testing

Evaluating the performance of a new classification algorithm often consists of two stages:

Training: The classifier is first trained using a set of training data to create a model to be used for prediction.

Testing: The overall aim is to achieve high performance on *unseen* data, which is not necessarily implied by high performance on the training data. Therefore a set of test data is used to evaluate the model.

Suppose our algorithm is applied to data provided by a hypothetical client. First the algorithm produces a range of rules from the training data with differing trade-offs between misclassification cost and rule complexity. In order to give the client some idea as to how well the rules produced generalize, the rules are re-evaluated on new validation data. At this point, the client selects a rule. To compare with other algorithms that produce just one model, we assume that the client selects the rule with minimum misclassification costs on the validation data, though in practice the client may elect to choose a simpler rule. To ensure a fair comparison, this rule must be re-evaluated again on further test data.

Table 2. Misclassification costs on test data and run times

Dataset	Cost	Select best Mean	StdDev	ATs	Select 5AT Mean	StdDev	Time (s)
Adult	Simple	14.42%	0.10%	19.6	15.64%	0.12%	1032
	Balanced	17.82%	0.14%	18.0	19.42%	0.07%	957
	1-10	12.45%	0.13%	15.8	12.90%	0%	898
Cover type	Simple	21.00%	0.19%	19.5	23.35%	0.08%	8217
	Balanced	21.71%	0.31%	19.5	23.57%	0.16%	9168
	1-10	10.58%	0.17%	19.2	11.20%	0.09%	7988
Contraceptive	Simple	29.45%	3.48%	8.8	29.46%	3.53%	55
	Balanced	31.97%	3.78%	7.9	32.22%	4.18%	55
Breast cancer	Simple	4.14%	2.25%	4.2	4.10%	2.28%	37
	Balanced	4.97%	2.53%	4.8	5.09%	2.61%	30
Pima Indians	Simple	24.38%	4.47%	4.9	24.48%	4.65%	47
	Balanced	26.35%	6.25%	8.0	26.84%	6.33%	41

Both the Adult dataset and the Cover Type dataset were partitioned once only, since the Adult dataset is already split into training and test data (30162 and 15060 records) and the size of the Cover Type dataset limits the amount of experimentation that can be performed. The Adult training set provided was split again at random, with approximately 80% forming the new training set and 20% providing a validation set. The Cover Type data was split 50–25–25 at random, with 50% forming the training set. Each experiment consisted of 30 runs of the algorithm, using a crossover rate of 30% and a population size of 100 for the Adult dataset and 50 for the Cover Type data.

The smaller datasets were split into ten approximately equal parts to be used as the test sets. For each test set, the remaining 90% of the records were split at random into two roughly equal parts to be used as training and validation sets for two experiments, resulting in 20 experiments for each dataset. This use of ten fold cross validation allows for fair comparison with both the results of Ishibuchi et al. [7,8] and with the results of 33 classifiers provided by Lim et al. [26]. The algorithm was run 30 times for each experiment, resulting in 600 runs. The parameter settings found to perform well for the Adult data (a crossover rate of 30% and a population of 100) were used in these experiments.

4.4 Results

Table 2 shows the mean quality and size of the rule selected by the client if he selects the best rule according to error rate on the validation data, breaking ties on the training error rate. Mean error rates for the best 5 AT rule and the approximate run time for one run on a 2GHz processor are also given.

The results obtained when the client chooses not to sacrifice rule accuracy can be roughly compared with those provided in the UCI machine learning repository [22] for the Adult dataset and those provided by Lim et al. [26] for the Breast

Cancer and Pima Indians datasets. In the first case, the UCI repository provides results for 16 algorithms, with error rates of between 14.05% and 21.42% and our algorithm ranks 3rd out of 17 algorithms. The 33 algorithms evaluated by Lim et al. have error rates varying between 2.78% and 8.48% for the Breast cancer dataset and between 22.1% and 31.0% for the Pima Indians dataset. Our algorithm ranks 17th and 22nd out of 34 algorithms respectively.

The trade-off between misclassification costs, on both the training and validation data, and rule simplicity for the Adult dataset is shown in figure 7. Here, the error bars give the standard deviation at each level of rule complexity.

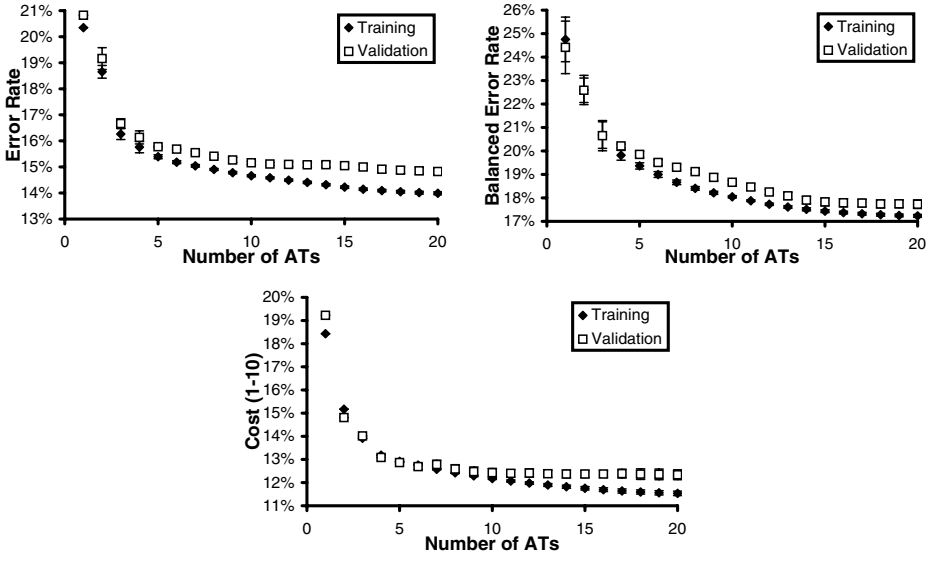
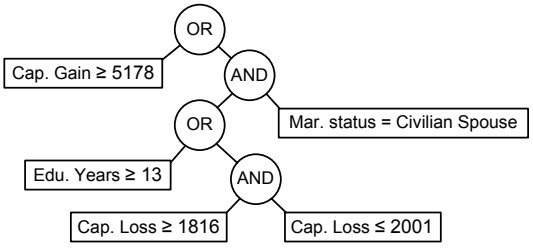


Fig. 7. Rule quality for the adult dataset. Misclassification costs are given by the simple error rate, the balanced error rate and by setting the false negative cost to be ten times the false positive cost respectively.

Typical 5 AT rules, with associated confusion matrices, are shown in figures 8 to 10. The first rule is fairly accurate when predicting that a record belongs to the class of interest, but it identifies little more than half of this class, since more emphasis is placed on accuracy on the larger set of uninteresting records. Subsequent rules have increases in the number of false positives and decreases in the number of false negatives, as would be expected. The rules become less restrictive as illustrated by the greater use of the boolean 'or' operation in figure 10, describing more of the class of interest but with decreased confidence.

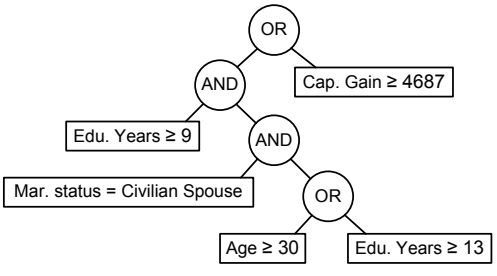
Figure 11 shows the results obtained when optimizing the simple error rate for the Breast cancer and Pima Indians diabetes datasets, allowing comparison with the results obtained by Ishibuchi et al. [7,8]. The results obtained by the



Confusion matrix

		Predicted -	Predicted +
Actual	-	10732	628
	+	1710	1990

Fig. 8. A typical five AT rule antecedent, predicting a high salary, produced when minimizing the simple error rate, with its confusion matrix on test data

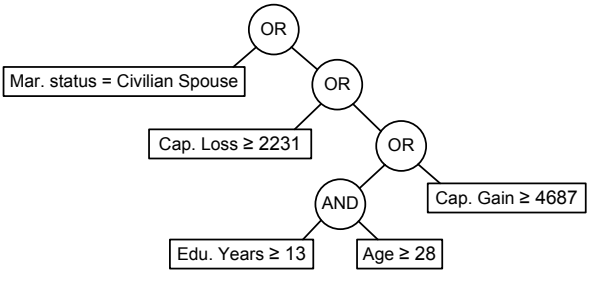


Confusion matrix

		Predicted -	Predicted +
Actual	-	8631	2729
	+	549	3151

Fig. 9. A five AT rule antecedent produced when minimizing the balanced error rate, with its confusion matrix on test data

two approaches on the Breast cancer data are broadly similar. Results for the Pima Indians diabetes datasets are similar in testing — once ten or more rules are permitted in the rule set, Ishibuchi et al. achieve error rates of approximately 24.8% — but on the training data Ishibuchi et al. only reach an error rate of approximately 16% even when 20 rules of up to 3 tests each are permitted. Note that both approaches result in rules that generalize poorly, as indicated by the large disparity between training and testing error rates, though even the best classifier evaluated by Lim et al. [26] has an error rate of 22.1% on test data.



Confusion matrix

		Predicted -	Predicted +
Actual	-	6524	4836
	+	140	3560

Fig. 10. A five AT rule antecedent produced when the cost of a false negative is ten times that for a false positive

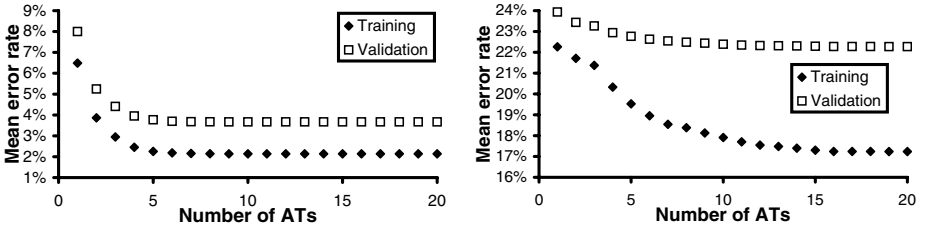


Fig. 11. Error rates for the Breast cancer and Pima Indians datasets

Finally, the algorithm was modified to suit a user with a preference for viewing rule sets, by changing the rule complexity objective to the number of ATs after conversion to such a rule set. Applying this modified algorithm to the Adult dataset and selecting an 8 AT rule set produced the following rules, with an error rate of 15.01% in training and 15.19% in testing:

- `If cap. gain` $\geq$ `5178 then salary` $\geq$ `$50,000`
- `If cap. loss` $\geq$ `2392 then salary` $\geq$ `$50,000`
- `If mar. status = civilian spouse and cap. loss` $\geq$ `1762 and cap. loss` $\leq$ `1980 then salary` $\geq$ `$50,000`
- `If mar. status = civilian spouse and edu. years` $\geq$ `13 and hours per week` $\geq$ `31 then salary` $\geq$ `$50,000`
- `Otherwise salary` $<$ `$50,000`

5 Conclusions

The results presented illustrate that the approach can produce reasonable results on two class problems, though there is room for improvement. However, the algorithm provides additional benefits. The most obvious is that the client can be provided with a range of models with different trade-offs between rule complexity and misclassification costs. This allows the client to select a rule that is accurate enough while also being comprehensible.

The overall approach is also flexible. Rules may be presented to the client in a number of ways and the measure of rule complexity can easily be adapted to match the method of rule presentation and the client's concept of rule comprehensibility. Different measures of misclassification cost can easily be used. There is no restriction on the data types of the fields of the dataset and no need to discretize numeric fields as required by other algorithms.

While the efficiency of the algorithm could be improved, the approach appears to be able to handle larger datasets than that of Ishibuchi et al. While no timings are given that would allow an effective comparison, Ishibuchi et al. report that their approach has a large computational load [7] and the largest dataset to which their code has been applied — the Pima Indians diabetes data — contains only 768 records. This is understandable, as even when applied to small datasets, 17070 simple rules are generated from which the genetic algorithm must select a set. Many of these simple rules are unlikely to be useful in a full classifier.

6 Further Research

Three or more classes: While the algorithm has been shown to be effective for two class problems an obvious improvement would be to extend the approach to handle three or more classes.

Three objectives: In practice, the client may only be able to approximate the costs of false negatives and false positives. In this case, the problem may be modeled as having three objectives to be minimized: the rule complexity, the number of false positives and the number of false negatives.

Diversity management: Preliminary investigations revealed a major loss of diversity in the population of rules early in the search process, even when no simplification routines were applied. If techniques can be found that provide better management of population diversity, the algorithm may be able to produce better results in a fraction of the time.

Efficiency improvements: There is scope for a number of efficiency improvements. For example, early in the search it makes little sense to evaluate the rules on the entire dataset, when evaluation on just a sample will provide enough information to guide the search at this stage.

Rule types: Other AT types (see previous work [2,4]) and operators may be used to further enhance the expressiveness of the rules produced.

References

1. B. de la Iglesia, M. S. Philpott, A. J. Bagnall, and V. J. Rayward-Smith. Data Mining Rules Using Multi-Objective Evolutionary Algorithms. In *Proceedings of 2003 IEEE Congress on Evolutionary Computation*, pages 1552–1559, 2003.
2. Beatriz de la Iglesia, Alan Reynolds, and Vic J Rayward-Smith. Developments on a Multi-Objective Metaheuristic (MOMH) Algorithm for Finding Interesting Sets of Classification Rules. In *Evolutionary Multi-Criterion Optimization: Third International Conference, EMO 2005*, volume 3410 of *Lecture Notes in Computer Science*, pages 826–840, March 2005.
3. B. de la Iglesia, G. Richards, M. S. Philpott, and V. J. Rayward-Smith. The application and effectiveness of a multi-objective metaheuristic algorithm for partial classification. *European Journal of Operational Research*, 169(3):898–917, 2006.
4. Alan Reynolds and Beatriz de la Iglesia. Rule induction using multi-objective metaheuristics: Encouraging rule diversity. In *Proceedings of the 2006 IEEE World Congress on Computational Intelligence*, pages 6375–6382. IEEE, 2006.
5. K. Ali, S. Manganaris, and R. Srikant. Partial Classification using Association Rules. In *Proceedings of the Third International Conference on Knowledge Discovery and Data Mining*, pages 115–118. The AAAI Press, 1997.
6. Ashish Ghosh and Bhabesh Nath. Multi-objective rule mining using genetic algorithms. *Information Sciences*, 163:123–133, 2004.
7. Hisao Ishibuchi and Yusuke Nojima. Accuracy-Complexity Tradeoff Analysis by Multiobjective Rule Selection. In *Proceedings of the ICDM 2005 Workshop on Computational Intelligence in Data Mining*, pages 39–48, 2005.
8. Hisao Ishibuchi, Isao Kuwajima, and Yusuke Nojima. Multiobjective Association Rule Mining. In *Proceedings of the PPSN Workshop on Multiobjective Problem Solving from Nature*, 2006.

9. Ian H. Witten and Eibe Frank. *Data Mining: Practical Machine Learning Tools and Techniques*. Elsevier, second edition, 2005.
10. T. Tanigawa and Q. Zhao. A Study on Efficient Generation of Decision Trees Using Genetic Programming. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2000)*, pages 1047–1052. Morgan Kaufmann, 2000.
11. R. K. DeLisle and S. L. Dixon. Induction of Decision Trees via Evolutionary Programming. *Journal of Chemical Information and Modelling*, 44(3):862–870, 2004.
12. J. Eggermont, J. N. Kok, and W. A. Kosters. Detecting and Pruning Introns for Faster Decision Tree Evolution. In *Parallel Problem Solving from Nature VIII*, volume 3242, pages 1071–1080. Springer-Verlag, 2004.
13. M. C. J. Bot. Improving Induction of Linear Classification Trees with Genetic Programming. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2000)*, pages 403–410. Morgan Kaufmann, 2000.
14. E. M. Mugambi and A. Hunter. Multi-Objective Genetic Programming Optimization of Decision Trees for Classifying Medical Data. In *Proceedings of the 7th International Conference on Knowledge-Based Intelligent Information and Engineering Systems (KES 2003)*, volume 2773 of *Lecture Notes in Computer Science*, pages 293–299, 2003.
15. C. Setzkorn and R. C. Paton. MERBIS - A Multi-Objective Evolutionary Rule Base Induction System. Technical Report ULCS-03-016, Department of Computer Science, University of Liverpool, 2003.
16. C. Setzkorn and R. C. Paton. MERBIS - A Self-Adaptive Multi-Objective Evolutionary Rule Base Induction System. Technical Report ULCS-03-021, Department of Computer Science, University of Liverpool, 2003.
17. John Koza. *Genetic programming: on the programming of computers by means of natural selection*. MIT Press, 1992.
18. William B. Langdon and Riccardo Poli. *Foundations of Genetic Programming*. Springer, 1998.
19. P. J. Angeline. Genetic Programming and Emergent Intelligence. In K. E. Kinnear, editor, *Advances in Genetic Programming*, pages 75–97. MIT Press, 1994.
20. A. Ekárt and S. Z. Németh. Selection Based on the Pareto Nondomination Criterion for Controlling Code Growth in Genetic Programming. *Genetic Programming and Evolvable Machines*, 2:61–73, 2001.
21. J. R. Quinlan. Simplifying decision trees. *International Journal of Man-Machine Studies*, 27(3):221–234, 1987.
22. C. J Merz and P. M. Murphy. UCI repository of machine learning databases. Univ. California, Irvine, 1998.
23. Kalyanmoy Deb. *Multi-Objective Optimization using Evolutionary Algorithms*. John Wiley & Sons Ltd, 2001.
24. Kalyanmoy Deb, Samir Agrawal, Amrit Pratab, and T. Meyarivan. A Fast Elitist Non-Dominated Sorting Genetic Algorithm for Multi-Objective Optimization: NSGA-II. In *Proceedings of PPSN VI*, volume 1917 of *Lecture Notes in Computer Science*, pages 849–858. Springer, 2000.
25. Eckart Zitzler and Lothar Thiele. Multiobjective Optimization Using Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach. *IEEE Transactions on Evolutionary Computation*, 3(4):257–271, 1999.
26. 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*, 40:203–229, 2000.