RULES-6: a simple rule induction algorithm for handling large data sets

D T Pham* and **A A Afify**
Intelligent Systems Laboratory, Manufacturing Engineering Centre, School of Engineering, Cardiff University, Cardiff, UK

The manuscript was received on 7 September 2004 and was accepted after revision for publication on 6 July 2005.

DOI: 10.1243/095440605X31931

Abstract: RULES-3 Plus is a member of the RULES family of simple inductive learning algorithms with successful engineering applications. However, it requires modification in order to be a practical tool for problems involving large data sets. In particular, efficient mechanisms are needed for handling continuous attributes and noisy data. This article presents a new rule induction algorithm called RULES-6, which is derived from the RULES-3 Plus algorithm. The algorithm employs a fast and noise-tolerant search method for extracting IF-THEN rules from examples. It also uses simple and effective methods for rule evaluation and handling of continuous attributes. A detailed empirical evaluation of the algorithm is reported in this paper. The results presented demonstrate the strong performance of the algorithm.

Keywords: classification, inductive learning, rule induction, discretization, noise handling

1 INTRODUCTION

Classification has numerous applications in mechanical engineering [**1–3**]. An automatic classifier is constructed by employing a set of precategorized examples to develop a model that can handle new examples from the same population. Inductive learning techniques are particularly suited to the construction of classifiers. They are simple and fast. Another advantage is that they generate models that are easy to understand. Finally, inductive learning classifiers are more accurate compared with other classification techniques.

Inductive learning techniques can be divided into two main categories, namely, decision tree induction and rule induction. There are a variety of algorithms for building decision trees. The most popular are CART [**4**], ID3 and its descendants C4.5 and C5.0 [**5–9**], SLIQ [**10**], SPRINT [**11**], and PUBLIC [**12**]. These learning systems are categorized as 'divide-and-conquer' inductive systems [**6**]. The knowledge induced by these systems is represented as decision trees. A decision tree consists of internal nodes and leaf nodes. Each internal node represents a test on an attribute and each outgoing branch corresponds to a possible result of this test. Each leaf node represents a classification to be assigned to an example. To classify a new example, a path from the root of the decision tree to a leaf node is identified on the basis of the values of the attributes of the example. The class at the leaf node represents the predicted class for that example. Decision trees are generated from training data in a top–down, general-to-specific direction. The initial state of a decision tree is the root node that is assigned all the examples from the training set. If all examples belong to the same class, then no further decisions need to be made to partition the examples, and the solution is complete. If examples at this node belong to two or more classes, then a test is made at the node, which will result in a split. The process is recursively repeated for each of the new intermediate nodes until a completely discriminating tree is obtained.

As with decision tree learning, there are many rule induction algorithms. Among them are AQ [**13–15**], CN2 [**16, 17**], RIPPER [**18**], and SLIPPER [**19**], which can all be categorized as 'separate-and-conquer' inductive systems. In contrast to decision tree

Corresponding author: Intelligent Systems Laboratory, Manufacturing Engineering Centre, Cardiff University, Queen's Building, The Parade, Newport Road, Cardiff CF24 3AA, UK.

learning, rule induction directly generates IF-THEN rules. Rule induction systems produce either an unordered set of IF-THEN rules or an ordered set of IF-THEN rules, also known as decision lists [**20**], both including a default rule. To classify an instance in the case of ordered rules, the ordered list of rules is examined to find the first whose antecedent is satisfied by the instance. The predicted class is then the one nominated by this rule. If no rule antecedent is satisfied, the instance is predicted to belong to the default class. In the case of unordered rules, it is possible for some instances to be covered by more than one rule. To classify a new instance in this case, some conflict resolution approach must be employed. The general operation of separate-and-conquer rule induction algorithms is the same. They create the rule set one rule at a time. After a rule is generated, the instances covered by it are removed from the training data set and the same induction procedure is applied to the remaining data set until all the instances are covered by at least one rule in the rule set.

RULES (RULe Extraction System) is a family of inductive learning algorithms that follow the separate-and-conquer rule induction approach. The first three algorithms in the RULES family (RULES-1, 2, and 3) were developed by Pham and Aksoy [**21–23**]. Later, the rule forming procedure of RULES-3 was improved by Pham and Dimov [**24**] and the new algorithm was called RULES-3 Plus. Compared with its immediate predecessor RULES-3, RULES-3 Plus has two new, strong features. First, it employs a more efficient search procedure instead of the exhaustive search conducted in RULES-3. Second, it incorporates a metric called the H measure [**25**] for selecting and sorting candidate rules according to their generality and accuracy. RULES-3 does not employ any measure for assessing the information content of rules. The first incremental learning algorithm in the RULES family was RULES-4 [**26**]. RULES-4 allows the stored knowledge to be updated and refined rapidly when new examples are available. RULES-4 employs a *short term memory* (STM) to store training examples when they become available. The STM has a user-specified size. When the STM is full, RULES-4 invokes RULES-3 Plus to generate rules. Pham *et al.* [**27**] described another algorithm, also based on RULES-3 Plus, called RULES-5, which can effectively handle problems involving continuous attributes. As with RULES-3 Plus, RULES-5 employs the H measure for evaluating the quality of rules.

RULES-3 Plus has been employed for the extraction of classification rules for solving different engineering problems, e.g. the recognition of design form features in CAD models for computer-aided process planning [**28**], the mapping of manufacturing information to design features [**28**], and the classification of defects in automated visual inspection [**29**]. RULES-3 Plus still suffers from problems that limit its efficiency and widespread use. One of the main problems is that RULES-3 Plus learns a complete and consistent rule set that tries to cover all of the positive and none of the negative training instances – instances of the target class (the class of the training instance under consideration) in the training set are called positive instances and instances in the training set that do not belong to the target class are called negative instances. In the case of noisy data, this leads to the generation of over-specific rules that overfit the training data. A second problem is that the H measure is computationally complex and does not lead to the highest level of predictive accuracy and generality. Finally, continuous-valued attributes are discretized using a simplistic equal-width approach before data are passed to the learning system. This discretization method is arbitrary and does not seek to discover any information inherent in the data, thereby hampering the ability of RULES-3 Plus to learn.

This paper presents RULES-6, a new rule induction algorithm which addresses the weaknesses of the RULES-3 Plus algorithm. In particular, it employs a new noise-tolerant search method which relaxes the consistency constraint and uses search-space pruning rules which significantly reduce the search time. It also adopts a simple metric for rule evaluation and a more robust method for handling continuous attributes. These enhancements enable the efficient generation of accurate and compact rule sets.

The paper is organized as follows. Section 2 briefly reviews RULES-3 Plus. Section 3 gives a detailed description of the RULES-6 algorithm. Section 4 discusses the evaluation of the performance of RULES-6 using real data. Section 5 concludes the paper and provides suggestions for future work.

2 RULES-3 PLUS ALGORITHM

The RULES-3 Plus algorithm works in an iterative fashion. In each iteration, it takes a seed example not covered by previously created rules to form a new rule. Having found a rule, RULES-3 Plus removes those examples that the rule covers from the training set, by marking them, and appends a rule at the end of its rule set. The algorithm stops when all examples in the training set are covered. This produces an unordered set of complete and consistent rules. It should be noted that only the examples covered by previously formed rules are marked in order to stop RULES-3 Plus from repeatedly finding the same rule. However, these examples are used to guide the specialization process and to assess the accuracy and generality of each newly formed rule. Considering the whole set of examples when forming rules, RULES-3 Plus is less prone to the *fragmentation* problem (i.e. the amount of available data reducing as induction progresses) [**30, 31**] and the *small disjuncts* problem (i.e. rules covering few training examples having a high error rate) [**32–36**]. As a result, a compact rule set can be obtained.

To form a rule, RULES-3 Plus performs a general-to-specific beam search for the most general and consistent rule. It starts with the most general rule and specializes it in steps considering only conditions extractable from the selected seed example. The aim of specialization is to construct a rule that covers the seed example and as many positive examples as possible while excluding all negative examples. The result is a rule that is consistent and as general as possible.

To assess the information content of each newly formed rule, RULES-3 Plus uses a metric called the *H* measure. As mentioned previously, this is a computationally complex measure that does not discriminate well between rules of different qualities. The algorithm deals with attributes having continuous values by dividing the range of each attribute into a fixed number of intervals using the equal-width discretization method. With this method, the number of intervals for each attribute is specified by the user. From the given set of examples, RULES-3 Plus constructs a new set for which the values of all continuous attributes are represented by appropriate intervals. Induction is then carried out with the new set of examples, the intervals being treated as any other value.

A pseudo-code description of the RULES-3 Plus algorithm and a simple example clearly illustrating its operation can be found in reference [**24**].

3 RULES-6 ALGORITHM

RULES-6 stands for 'RULe Extraction System – Version 6'. A pseudo-code description of RULES-6 is given in Fig. 1. Like its predecessors in the RULES family, RULES-6 extracts rules by processing one example at a time. The algorithm first selects a seed example, the first example in the training set not covered by previously created rules, and then calls the *Induce-One-Rule* procedure to extract a rule that covers that example. Following this, all covered examples are marked, the learned rule is added to the rule set, and the process repeated until all examples in the training set have been covered. The Induce-One-Rule procedure is outlined in Fig. 2.

```
Procedure Induce_Rules (TrainingSet, BeamWidth)
RuleSet = ∅
While all the examples in the TrainingSet are not covered Do
  Take a seed example s that has not yet been covered.
  Rule = Induce_One_Rule (s, TrainingSet, BeamWidth)
  Mark the examples covered by Rule as covered.
  RuleSet = RuleSet ∪ {Rule}
End While
Return RuleSet
End
```

Fig. 1 A pseudo-code description of RULES-6

The Induce-One-Rule procedure searches for rules by carrying out a pruned general-to-specific search. The search aims to generate rules which cover as many examples from the target class and as few examples from the other classes as possible, while ensuring that the seed example remains covered. As a consequence, simpler rules that are not consistent, but are more accurate for unseen data, can be learned. This contrasts with the rule forming procedure of the RULES-3 Plus algorithm, which restricts its search to only those rules that are completely consistent with the training data, leading to overfitting if the data is noisy.

A beam search is employed to find the best rule. This is done by using two rule lists named *PartialRules* and *NewPartialRules*. PartialRules, which is the same size as the beam width w, stores the w best partial rules during the specialization process. Only the rules in this list are considered for further specialization. NewPartialRules is used to save valid partial rules obtained by specializing the rules in Partial Rules. The learning of single rules starts with the most general rule whose body is empty (step 1 in Fig. 2) and specializes it by incrementally adding conditions to its body (step 3 in Fig. 2). Possible conditions are attribute-value pairs of the selected seed example. In the case of nominal attributes, conditions of the form $[A_i = v_{is}]$ are created, where v_{is} is the value of A_i in the selected seed example s. In the case of continuous attributes, a prior discretization method is used to split the range of each attribute into a number of smaller intervals that are then regarded as nominal values. For each condition, a new rule is formed by appending a condition to the current rule that differs from the conditions already included in the rule. The score of each new rule is computed and the rule with the best accuracy is remembered (step 4 in Fig. 2). The new rule is then inserted into the NewPartialRules list (step 9 in Fig. 2) unless one of the conditional tests (step 5, 6, or 7 in Fig. 2) prevents this because it is deemed that no improved rule will be obtained from the new rule. In the latter case, the new rule is regarded

```
Procedure Induce_One_Rule (s: Seed example, Instances: Training set, w: Beam width)
PartialRules = NewPartialRules = ∅
BestRule = most general rule (the rule with no conditions)
PartialRules = PartialRules ∪ {BestRule}                                          (step 1)
While PartialRules ≠ ∅ Do                                                         (step 2)
  For each Rule ∈ PartialRules Do
  {First, generate all specialisations of the current rule, save useful ones and determine all the
   InvalidValues according to one of the conditional tests in steps (5), (6) or (7)}
   For each nominal attribute A_i that does not appear in Rule Do
    If v_is ∈ Rule.ValidValues, where v_is is the value of A_i in s Then
      NewRule = Rule ∧ [A_i = v_is]                                               (step 3)
      If NewRule.Score > BestRule.Score Then                                      (step 4)
        BestRule = NewRule
      If Covered_Positives (NewRule) ≤ MinPositives OR                            (step 5)
        Covered_Negatives (Rule) – Covered_Negatives (NewRule) ≤ MinNegatives OR  (step 6)
        Consistency (NewRule) = 100% Then                                         (step 7)
        Parent (NewRule).InvalidValues = Parent (NewRule).InvalidValues + {v_is}  (step 8)
      Else
        NewPartialRules = NewPartialRules ∪ {NewRule}                             (step 9)
   End For
  End For
  Empty PartialRules
  For each Rule ∈ NewPartialRules Do
  {Next, delete partial rules that cannot lead to an improved rules and determine all the InvalidValues
   according to the conditional test in step (10)}
    If Rule.OptimisticScore ≤ BestRule.Score Then                                 (step 10)
      NewPartialRules = NewPartialRules – {Rule}                                  (step 11)
      Parent (Rule).InvalidValues = Parent (Rule).InvalidValues + Last_Value_Added (Rule)   (step 12)
  End For
  For each Rule ∈ NewPartialRules Do
  {Finally, remove from the ValidValues set of each rule all the values that will lead to unnecessary
   construction of useless specialisations at subsequent specialisation steps}
    Rule.ValidValues = Rule.ValidValues – Parent (Rule).InvalidValues             (step 13)
  End For
  If w > 1 Then
   Remove from NewPartialRules all duplicate rules
  Select w best rules from NewPartialRules and insert into PartialRules           (step 14)
  Remove all rules from NewPartialRules
End While
Return BestRule
End
```

Fig. 2 A pseudo-code description of the *Induce_One_Rule* procedure. *PartialRules*, a list of rules to be specialized and *NewPartialRules*, a new list of rules to be used for further specializations

as ineffective and additional specializations will not improve the values for the quality measure. If the new rule is discarded, the last attribute value used to form it is added to the set of attribute values (*InvalidValues*) of its immediate parent, the current rule, so as to ensure that it will be removed from the other specializations of the same parent rule (step 8 in Fig. 2). Thus, the NewPartialRules list only contains useful rules that can be employed for further specialization. This process is repeated until there are no remaining rules to be specialized in the PartialRules list.

Another test that allows sections of the search space to be pruned away is now applied to each rule in the NewPartialRules list after the best rule overall in the current specialization step is identified. Rules that satisfy the conditional test at step 10 are removed from the NewPartialRules list (step 11 in Fig. 2), because they will not lead to improved rules. The last attribute values used to generate these rules are added to the InvalidValues of their parents (step 12 in Fig. 2). All InvalidValues are then deleted from the corresponding set of *ValidValues* for each rule in the NewPartialRules list (step 13 in Fig. 2). Such values cannot lead to a viable specialization from any point in the search space that can be reached via identical sets of specializations and thus excluding them will prevent the unnecessary construction of ineffective specializations at subsequent specialization steps.

After eliminating all duplicate rules, the best w rules from the NewPartialRules list are chosen to replace all rules in the PartialRules list (step 14 in Fig. 2). The comparison between rules is based on the quality measure defined in section 3.1. If two rules have an equal quality measure, the simpler rule, in other words, the one with fewer conditions, is selected. If both the quality measure and the number of conditions of the rules are the same, the more general rule that covers more instances is chosen.

The specialization process is then repeated until the PartialRules list becomes empty (step 2 in Fig. 2) owing to the tests at steps 5, 6, 7, and 10. It should be noted that the PartialRules and NewPartialRules lists are reused after each iteration. During specialization, the best rule obtained is stored and returned at the end of the procedure. In RULES-3 Plus, the specialization process stops once a consistent rule that covers the seed example has been formed and this rule is taken as the best one. It should be noted that consistent rules having a very low coverage might be discovered in the early stages of the rule generation process and stopping the search process once a consistent rule has been found might lead to the generation of non-optimal rules. However, if the search process continues, more general rules might be created.

The following sections discuss the key ideas underlying RULES-6 in greater detail.

3.1 Rule quality metric

Given that the rule induction process could be conceived as a search process, a metric is needed to estimate the quality of rules found in the search space and to direct the search towards the best rule. The rule quality measure is a key element in rule induction. In real-world applications, a typical objective of a learning system is to find rules that optimize a rule quality criterion that takes both training accuracy and rule coverage into account so that the rules learned are both accurate and reliable.

A quality measure must be estimated from the available data. All common measures are based on the number of positive and negative instances covered by a rule. Several different metrics are used in the existing algorithms. These include *purity* (utilized in GREEDY [**30**] and SWAP-1 [**37**]), *information content* (employed in PRISM [**38**]), *entropy* (adopted in the original version of the CN2 algorithm [**16**]), the metric applied in RIPPER [**18**], and *accuracy* (used in I-REP [**39**] and PROLOG [**40**]). The problem of the first four measures is that they attain their optimal values when no negative instances are covered. For example, a rule r_1 that only covers one positive instance scores more highly than a rule r_2 covering 999 positive instances and one negative instance. Also, they do not aim to cover many positive instances. For example, a rule r_3 that covers 100 positive and 10 negative examples is deemed of identical value to another rule r_4 that covers 10 000 positive and 1000 negative examples. As a result, these metrics tend to select very specific rules covering only a small number of instances. This is undesirable as rules covering few instances are unreliable, especially where there is noise in the data. The accuracy of these rules on the training data does not adequately reflect their true predictive accuracy on new test data. The problem of the accuracy measure, as pointed out by Cohen [**18**], is that this measure sometimes does not lead to a satisfactory behaviour. For example, it favours a rule r_5 that covers 2000 positive and 1000 negative examples over a rule r_6 that covers 1000 positive and only one negative example.

One of the popular metrics that penalize rules with low coverage is the *Laplace accuracy estimate* (used in CN2 [**17**]), COVER [**41, 42**], and several other algorithms). The Laplace formula is given by

$$\text{LaplaceAccuracy}(n_{\text{class}}, n_{\text{covered}}, k) = \frac{n_{\text{class}} + 1}{n_{\text{covered}} + k} \quad (1)$$

where k is the number of classes, n_{class} is the number of positive instances covered by the rule, and n_{covered} is the number of instances covered by the rule. The Laplace function balances accuracy against generality. In general, it prefers rules that cover more positive instances over rules that cover fewer instances and also prefers rules with a lower proportion of the cover that is negative over those for which that proportion is higher. Segal [**43**] showed that the Laplace estimate has the desirable property of taking into account both accuracy and coverage when estimating rule accuracy. However, it has a problem when learning rules with less than 50 per cent training accuracy. The Laplace estimate does not satisfy the requirement that the rule quality value should rise with increased coverage. Cestnik [**44**] also conducted experiments in four medical domains and his results indicated that the Laplace accuracy estimate was often unrealistic, especially in multi-class decision problems. This occurred because of the assumption that underlies Laplace accuracy estimate, namely, the *a priori* distribution is uniform.

A more general version of the Laplace estimate, called the *m-probability-estimate*, has been developed by Cestnik [**44**] and is defined as follows

$$m\text{Accuracy}(n_{\text{class}}, n_{\text{covered}}, k) = \frac{n_{\text{class}} + mP_0(C_t)}{n_{\text{covered}} + m} \qquad (2)$$

where $P_0(C_t)$ is the *a priori* probability of the target class C_t and m is a domain dependent parameter. The value of m is related to the amount of noise in the domain. m can be small if little noise is expected and should increase if the amount of noise is substantial. The Laplace estimate can be obtained from the m-probability-estimate when m is set to k, the total number of classes, and $P_0(C_t)$ is assumed to be uniform. It should be noted that the m-probability-estimate generalizes the Laplace estimate so that rules that cover no instances will be evaluated with the *a priori* probability instead of the value $1/k$, which is more flexible and convenient.

The performance of the H measure and the seven quality measures mentioned earlier when used with the RULES-6 algorithm was evaluated empirically [**45**]. The evaluation was carried out on a large number of data sets and the results showed that the m-probability-estimate outperformed the other measures. Therefore, RULES-6 employs the m-probability-estimate (equation (2)) to select the best rule (step 4 in Fig. 2) and to decide on the best specializations to retain (step 14 in Fig. 2) after each specialization step. In RULES-6, the m value is set to k and the *a priori* probability $P_0(C_t)$ is assumed to be equal to the training accuracy of the empty rule that predicts the target class, namely

$$P_0(C_t) = \frac{n_{C_t}}{N} \qquad (3)$$

where n_{C_t} is the number of instances in the target class C_t and N is the total number of instances in the training data set. This version of the Laplace accuracy estimate is a good choice because it has a strong theoretical background [**46**] and it meets the requirements of a good estimation function.

For the examples mentioned earlier, if it is assumed that k equals 2 and that both the total numbers of positive and negative instances are equal, the rule r_1 that only covers one positive instance scores 0.667 and the rule r_2 that covers 999 positive instances and one negative instance scores 0.998. Scores of the rules r_3, r_4, r_5, and r_6 with positive and negative coverages of (100, 10), (10 000, 1000), (2000, 1000), and (1000, 1) are 0.902, 0.909, 0.667, and 0.998, respectively. Therefore, rules r_2, r_4, and r_6 are considered better than rules r_1, r_3, and r_5, which seems intuitively correct. This indicates that the m-probability-estimate prefers rules that cover many positive instances and few negative instances, thus being biased towards finding general rather than more specific rules.

3.2 Search-space pruning rules

The size of the search space for inducing one rule grows exponentially with both the number of attributes used to describe each instance and the number of values allowed for each attribute. Moreover, the iterative nature of rule induction algorithms suggests that the computational requirements would be high on large data sets even with the reduced search spaces considered by algorithms such as RULES-6. Therefore, an efficient search method is essential in order for a learning algorithm to handle large data sets.

The search space can be efficiently organized by taking advantage of a naturally occurring structure over the hypothesis space that exists for any classification learning problem – a general-to-specific partial ordering of hypotheses [**47**]. This structure implies that all specializations of a rule cover a monotonically decreasing number of positive and negative instances. This organization property provides a powerful source of constraints on the search performed by the RULES-6 algorithm. RULES-6 constrains the search space by employing the four pruning rules listed in Table 1. These pruning rules remove portions of the search space that do not maximize the quality measure, thus significantly speeding up the search process. Although the rules

Table 1 Search-space pruning rules employed by RULES-6

(1) If Covered_Positives $(r) \leqslant$ MinPositives Then Prune (r)
(2) If Covered_Negatives (r) − Covered_Negatives $(r') \leqslant$ MinNegatives Then Prune (r')
(3) If Consistency $(r) = 100\%$ Then Prune (r)
(4) If Optimistic_Score $(r) \leqslant$ Score (BestRule) Then Prune (r)

r' is any specialization of rule r and Prune (r) indicates that the children of r should not be searched.

removed by the pruning rules are relatively poor rules, pruning rules improve performance without affecting the quality of the learned rules.

The effectiveness of these pruning rules depends upon how efficiently they can be implemented and upon the regularity of the data to which the search is applied. The remainder of this section describes the pruning rules in detail.

The pruning rules in Table 1 are derived from the following ideas. As the aim of specialization is to find a rule that maximizes the quality measure, further specialization of a rule can be stopped the moment it becomes clear that additional specialization will not improve the quality measure for the rule. Furthermore, in order to reduce the number of specialization steps and thus speed up the learning process, a rule ought to be an improvement over its parent. If this is not the case, the rule should not be further specialized. Finally, as only one solution is sought, further specialization of a rule can be terminated when it cannot improve on the current best rule.

The first pruning rule (step 5 in Fig. 2) is used to stop further specialization when the number of positive instances covered by a rule is below a threshold (*MinPositives)* and thus can be viewed as implementing a form of prepruning. Such specializations are deemed ineffective as the goal is to find rules that cover as many positive instances as possible. In RULES-6, MinPositives is a user-specified parameter. The value of this parameter should be kept low, especially in domains that are free of noise, to avoid generating over-simplified rule sets. An appropriate empirical range for this parameter is between 1 and 5. This pruning rule requires almost no additional overhead to employ, as the number of positive instances covered by a rule must, in any case, be determined to calculate its accuracy. Section 4.1 gives empirical evidence that this pruning rule reduces the learning time of RULES-6 without decreasing the accuracy of its rule sets.

The second pruning rule (step 6 in Fig. 2) discards descendants of a rule that does not exclude at least some new negative instances. A rule that does not remove any new negative instances is deemed ineffective as either it excludes positive instances only or it keeps the covered instances unchanged. With greater values of the minimum number of removed negative instances (*MinNegatives*), this pruning rule ensures that each specialization step changes a rule significantly. As a result, part of the search space can be eliminated in the early stages of the rule specialization process, which speeds up the execution of the algorithm. In RULES-6, MinNegatives is a parameter of the algorithm that can be specified by the user. An appropriate empirical range for this parameter is between 1 and 5. As is the case for the first pruning rule; no additional overhead is required to employ this pruning rule as the number of negative instances covered by a rule must be determined for the quality measure. Section 4.1 shows that this pruning rule improves the quality of the generated rules and speeds up the execution of the algorithm.

The third pruning rule (step 7 in Fig. 2) avoids expanding rules that have become consistent. The reason is that any further specialization will only decrease the number of positive instances covered by these rules and therefore yield lower values for the quality measure. It should be noted that the best overall rule will still be returned because the best rule is retained after each specialization step. Again, no additional overhead is required to use this pruning rule. Section 4.1 demonstrates the effectiveness of this pruning rule.

The fourth pruning rule (step 10 in Fig. 2) removes all specializations of a rule if its optimistic value of the quality measure cannot improve on the current best rule. The optimistic value can be determined by observing that the specialization of a rule can only make it become more specific and thereby decrease the number of instances that it covers. As the quality measure is highest when positive cover is maximized and negative cover is minimized, a simple optimistic value is obtained by determining the quality measure of a rule with the same positive cover as the current rule but with a negative cover of zero. If this value is lower than that for the current best rule, specialization is terminated because none of the rule specializations can improve on the current best value. Section 4.1 confirms that this pruning rule improves the quality of the rule sets of RULES-6 substantially and reduces its learning time.

Clearly, the effectiveness of this pruning rule depends on the value of the current best rule. In general, the larger the best rule value, the greater the search space that can be removed. As a result, the implementation of the fourth pruning rule is delayed until the best overall rule in the current specialization step is determined. In this way, the performance of this rule is maximized. It should be noted that when all rules at a certain specialization level satisfy any of the aforementioned pruning rules, the PartialRules set becomes empty. This terminates the search for the best rule (step 2 in Fig. 2).

The pruning rules discussed earlier only remove specializations of rules that are guaranteed not to be a solution. The effect of these rules can be maximized on the basis of the following ideas. If it can be determined that an attribute value used to specialize a certain rule in the search space cannot lead to a solution, then it follows that no solution can result from the application of such an attribute value to the other specializations of this rule. This can be justified as follows. Consider the rule $r = A \wedge C \rightarrow$ *target class*, where condition C is used to specialize a conjunct A. Let $p = A \wedge B \wedge C \rightarrow$ *target class* be another rule where conditions B and C are successively used to specialize the same conjunct A. If rule r resulting from the application of condition C to conjunct A does not cover any positive instances, then it follows that rule r will not be considered for further specialization according to the first pruning rule. Furthermore, as conjunct $A \wedge B$ is a specialization of conjunct A, it must cover fewer instances than those covered by A. As a result, rule p resulting from the application of condition C to conjunct $A \wedge B$ will also not cover any positive instances and consequently rule p should also be excluded from further specialization.

A similar argument demonstrates that if the application of condition C to conjunct A causes rule r to be discarded according to any of the other pruning rules in Table 1, then it follows that rule p should also be discarded as it covers a subset of instances covered by rule r.

The aforementioned idea can be implemented by maintaining and manipulating for each rule, r, a separate list containing all possible attribute values, *r.ValidValues*, that can be applied in the search space below r. Initially, the list contains all the nominal attribute values. A rule is specialized only by appending values to it, provided that the attributes of such values do not already appear in the rule. Each value that is appended to a rule is removed from its r.ValidValues list. When the rule is specialized, the values are examined to determine if any values can be pruned away. Any values that can be pruned away are deleted from r.ValidValues to prevent the unnecessary construction of ineffective specializations at subsequent specialization steps. New rules are then created for each of the attribute values remaining in r.ValidValues.

3.3 Discretization method

As most real-world applications of classification learning involve continuous-valued attributes, properly handling these attributes is important. Discretization of the data is one possibility. The usual approach to discretization of continuous-valued attributes is to perform this discretization prior to the learning process [**48–52**]. First, all continuous attributes in the data are discretized to obtain a discrete data set. Then, learning algorithms are applied to this discretized data set.

Several prior discretization methods have been developed. The *equal-width* method proposed by Wong and Chiu [**53**] and used in the RULES-3 Plus algorithm is perhaps the simplest discretization procedure. It simply involves dividing the range of a continuous variable into l equal intervals, where l is a user-defined parameter. As the equal-width approach considers neither the distribution of the values of the continuous attribute nor the dependency between the class label and the continuous attribute, it is likely that classification information will be lost as a result of combining values that are closely associated with different classes into the same interval. Furthermore, the number of intervals has a strong impact on performance. If too many intervals are specified, the learned model will be complex. If too few intervals are specified, information that can be used to distinguish instances will be lost.

The *1R Discretizer* method introduced by Holte [**54**] is a simple example of supervised discretization. This method first sorts the values of a continuous attribute in ascending order and then puts instances having equal values or having the same class label into one interval. Adjacent intervals can then be merged if they share the same majority class label. To avoid too many intervals being generated, each interval must include at least a prespecified number of instances.

A number of entropy-based discretization algorithms have been developed [**55, 56**]. They are mostly inspired by Quinlan's decision tree induction algorithms ID3 and C4.5 [**6, 7**]. The method of Fayyad and Irani [**56**] is similar to that of Catlett [**55**], but employs an elegant test based on the minimum description length (MDL) principle to determine a stopping criterion for the recursive discretization strategy. Moreover, the method takes advantage of the fact [**57**] that the optimal cut points when discretizing a continuous attribute using an average class entropy evaluation function can only be selected from a set called the boundary points. This can be used to improve the efficiency of the discretization algorithm, where the algorithm need only examine the boundary points of each continuous attribute rather than all its distinct values.

The aforementioned methods are heuristic techniques and, therefore, they cannot guarantee finding the optimal discretization. However, their efficiency makes them attractive choices in practical applications. In recent years, several optimization techniques for discretization of continuous-valued attributes have been developed [**58–63**]. The optimum discretization method of Cai [**63**] is an efficient algorithm using an evaluation function based on the MDL principle. The optimal number of intervals is obtained by searching only the boundary points' search space of each continuous attribute and selecting those points that optimize the evaluation metric.

The experimental results of many studies [63–66] have indicated that the choice of a discretization method depends on both the data to be discretized and the learning algorithm. The performance of the four discretization methods mentioned earlier when used with the RULES-6 algorithm was evaluated empirically [45]. The evaluation was carried out on a large number of data sets and the results showed that the performance of the RULES-6 algorithm significantly improved when continuous-valued attributes were discretized using the entropy method. As a result, this discretization method is adopted for use with RULES-6.

4 EMPIRICAL EVALUATION OF RULES-6

This section reports on a series of experiments to assess the performance of the RULES-6 algorithm. First, an empirical evaluation of the pruning rules of RULES-6 was presented. Experiments were conducted to explore the relative contribution of each of these rules to the performance of the algorithm. Second, RULES-6 was compared with its immediate predecessor RULES-3 Plus and with the well-known inductive learner C5.0, which is probably the best performing induction algorithm commercially available.

Three criteria were used to evaluate the performance of the tested algorithms, namely, classification accuracy, rule set complexity, and execution time. Classification accuracy is obviously the most important criterion in most induction tasks. It is defined as the percentage of instances from the test set that were correctly classified when the rules developed from the corresponding training set were applied. The complexity of a rule set is commonly measured by the total number of rules or total number of conditions in that rule set. The assessment of the execution time is more difficult because of uncertainties in computer systems. The time measures considered here are the total CPU time in seconds and the number of rules evaluated during the search process. All execution times were obtained on a Pentium IV computer with a 2.4 GHz processor, 512 MB of memory, and Windows NT 4.0 operating system.

In order to draw reliable conclusions about the behaviour of the learning algorithms, 40 data sets (listed in Table 2) were considered. All data sets were obtained from the University of California at Irvine (UCI) repository of machine learning databases [67]. These data sets are commonly used to evaluate machine learning algorithms. They are representative of many different types of classification learning problems. They differ in the number of learning instances that are available, the degree of noise in these instances, the number of classes and the proportion of instances belonging to each class, the number of nominal and continuous-valued attributes used to describe the instances, and the application area from which the data were obtained.

In the experiments conducted in this study, the hold-out approach was used to partition the data into training and test data [68, 69]. For large data sets with more than 1000 instances, each set was randomly divided once into a training set with two-thirds of the data and a test set with the remaining one-third. For small data sets with fewer than 1000 instances, the previously described procedure was repeated 10 times and the results were averaged.

4.1 Evaluation of the search-space pruning rules

To evaluate the relative effectiveness of each of the search-space pruning rules given in Table 1, the RULES-6 algorithm was employed to find rule sets using first none of the pruning rules and then using each individual rule. Results are also reported for the case where all pruning rules were employed. In this experiment, the data sets from Table 2 were used and the beam width, MinPositives, and MinNegatives parameters were set to 8, 2, and 1, respectively.

Table 3 presents the number of rules explored for each search method. Also given is the percentage by which the number of rules examined is reduced by the addition of each pruning rule. This equals $(a-b)/b^{*}100$, where a is the number of rules explored when no pruning rules were applied and b is the number of rules considered using the added pruning rule(s). A rule is deemed to have been examined if it is generated at step 3 of the RULES-6 algorithm (Fig. 2).

As can be seen, the addition of each pruning rule reduced the number of rules that RULES-6 had to process for all the data sets. Further, in many cases, the magnitude of this reduction was very large. For example, for the *Sonar* data set, the number of rules to explore dropped by 99.4 per cent (from 282 406 to 1823) when all the four rules were applied. It is also notable that the second pruning rule has the largest impact on the number of rules examined. The order of importance of the remaining pruning rules appeared to be as follows: fourth pruning rule (most important), first pruning rule, and third pruning rule (least important).

Table 4 shows the execution time in CPU seconds for each additional pruning rule. The percentage reduction in the execution time for each search method is also indicated. For all the data sets, the addition of the pruning rules resulted in a decrease in computation time. It is worth noting the

Table 2 Summary of the data sets used in the experiments [**67**]

Data set name	Number of instances	Number of nominal attributes	Number of continuous attributes	Number of classes
Abalone	4177	1	7	29
Adult	48 842	8	6	2
Anneal	898	32	6	6
Australian	690	8	6	2
Auto	205	10	15	6
Balance-scale	625	0	4	3
Breast	699	0	10	2
Breast-cancer	286	9	0	2
Car	1728	6	0	4
Chess	3196	36	0	2
Cleve	303	7	6	2
Crx	690	9	6	2
Diabetes	768	0	8	2
German	1000	13	7	2
German organization	1000	12	12	2
Glass2	163	0	9	2
Heart-disease	270	0	13	2
Heart-Hungarian	294	5	8	2
Hepatitis	155	13	6	2
Horse-colic	368	15	7	2
Hypothyroid	3163	18	7	2
Ionosphere	351	0	34	2
Iris	150	0	4	3
Lymphography	148	15	3	4
Monk1	556	6	0	2
Monk2	601	6	0	2
Monk3	554	6	0	2
Mushroom	8124	22	0	2
Promoter	106	57	0	2
Satimage	6435	0	36	6
Segment	2310	0	19	7
Shuttle	58 000	0	9	7
Sick-euthyroid	3163	18	7	2
Sonar	208	0	60	2
Soybean-large	683	35	0	19
Splice	3190	61	0	3
Tic-tac-toe	958	9	0	2
Tokyo	961	0	46	2
Vehicle	699	0	18	4
Vote	435	16	0	2

computation times for the *Anneal, Chess, Hypothyroid, Ionosphere, Promoter, Sick-euthyroid, Sonar, Soybean-large*, and *Tokyo* data sets.

Table 5 presents the complexity of the rule sets generated with each of the pruning rules. The percentage reduction in the number of conditions obtained with each experiment is also given. Table 6 shows the classification accuracies obtained with each of the pruning rules. The percentage increase in the classification accuracy achieved with each condition is also given. A number of results are notable. First, the application of the third and fourth pruning rules resulted in a minor increase in the complexity of the rule sets. Second, the other pruning rules caused a large reduction in the complexity of the rule sets and this reduction resulted in an improved classification accuracy in most cases. For example, for the *Satimage* data set, the number of conditions dropped from 1909 to 664, whereas the accuracy increased from 82.0 to 82.9 per cent when all the pruning rules were employed.

4.2 Comparison with RULES-3 Plus

This section describes an empirical study to compare RULES-6 with RULES-3 Plus. Table 7 lists the results obtained. As can be seen from the table, the performance obtained by RULES-6 was impressive. There were considerable improvements in classification accuracy for 25 data sets. For data sets *Abalone, German-organization, Glass2, Heart-Hungarian, Hepatitis, Horse-colic, Lymphography, Shuttle, Sick-euthyroid*, and *Vehicle*, the improvements were more obvious. The accuracy degraded for 11 data sets. For the remaining four data sets, equivalent results were obtained. It can also be seen from the table that RULES-6 produced much more compact rule sets than RULES-3 Plus. The total number of rules decreased by 88.5 per cent (from 11 654 to 1342). Also, the total number of conditions dropped by 95.7 per cent (from 101 787 to 4361). The reduction in the number of rules and number of conditions for the *Adult* data set is

Table 3 Total number of rules explored for each search method

Data set name	RULES-6 with no pruning rules	RULES-6 with rule (1) added		RULES-6 with rule (2) added		RULES-6 with rule (3) added		RULES-6 with rule (4) added		RULES-6 with all pruning rules	
	Number	Number	%Reduction	Number	%Reduction	Number	%Reduction	Number	%Reduction	Number	%Reduction
Abalone	4037	1871	53.7	3531	12.5	3827	5.2	2754	31.8	1480	63.3
Adult	100 371	48 658	51.5	52 434	47.8	77 211	23.1	64 975	35.3	28 847	71.3
Anneal	93 968	71 415	24.0	4628	95.1	70 857	24.6	45 469	51.6	3730	96.0
Australian	29 049	10 930	62.4	10 282	64.6	22 041	24.1	15 934	45.1	5100	82.4
Auto	36 911	19 426	47.4	6447	82.5	24 191	34.5	13 184	64.3	3042	91.8
Balance-scale	165	165	0.0	165	0.0	165	0.0	158	4.2	156	5.5
Breast	3317	1594	51.9	868	73.8	2128	35.8	1072	67.7	368	88.9
Breast-cancer	6876	2043	70.3	3684	46.4	5370	21.9	4265	38.0	1703	75.2
Car	3069	1784	41.9	2957	3.6	2996	2.4	2675	12.8	1626	47.0
Chess	211 449	133 326	36.9	24 219	88.5	177 159	16.2	97 481	53.9	17 995	91.5
Cleve	14 096	4991	64.6	3441	75.6	10 123	28.2	6305	55.3	1503	89.3
Crx	37 410	13 202	64.7	13 012	65.2	28 485	23.9	18 881	49.5	7645	79.6
Diabetes	1576	1576	0.0	368	76.6	1543	2.1	1504	4.6	352	77.7
German	147 009	55 774	62.1	39 255	73.3	111 978	23.8	75 181	48.9	16 664	88.7
German organization	161 919	89 224	44.9	30 875	80.9	148 951	8.0	123 428	23.8	19 347	88.1
Glass2	1063	772	27.4	61	94.3	927	12.8	844	20.6	44	95.9
Heart	9963	6549	34.3	1613	83.8	8691	12.8	6555	34.2	1207	87.9
Heart-Hungarian	8465	3551	58.1	1708	79.8	7018	17.1	5059	40.2	708	91.6
Hepatitis	22 871	5795	74.7	1848	91.9	12 589	45.0	5800	74.6	991	95.7
Horse-colic	82 712	16 981	79.5	17 234	79.2	43 134	47.9	24 156	70.8	7750	90.6
Hypothyroid	57 127	32 371	43.3	4567	92.0	45 854	19.7	36 516	36.1	2171	96.2
Ionosphere	79 545	34 172	57.0	5374	93.2	45 737	42.5	17 863	77.5	2284	97.1
Iris	60	60	0.0	50	16.7	39	35.0	21	65.0	20	66.7
Lymphography	15 218	5256	65.5	2650	82.6	8976	41.0	4674	69.3	1547	89.8
Monk1	1239	1063	14.2	1188	4.1	1175	5.2	867	30.0	836	32.5
Monk2	3328	2608	21.6	2998	9.9	3238	2.7	3011	9.5	2397	28.0
Monk3	644	563	12.6	569	11.6	546	15.2	335	48.0	320	50.3
Mushroom	39 089	38 217	2.2	3461	91.1	18 859	51.8	8435	78.4	4805	87.7
Promoter	138 504	5722	95.9	4790	96.5	17 774	87.2	4936	96.4	1707	98.8
Satimage	779 106	178 512	77.1	355 265	54.4	538 056	30.9	193 923	75.1	75 755	90.3
Segment	73 604	27 024	63.3	18 724	74.6	39 480	46.4	17 829	75.8	5335	92.8
Shuttle	11 554	9272	19.8	4392	62.0	5508	52.3	3470	70.0	2963	74.4
Sick-euthyroid	63 398	37 454	40.9	5147	91.9	54 771	13.6	48 945	22.8	3507	94.5
Sonar	282 406	165 063	41.6	2550	99.1	260 104	7.9	211 040	25.3	1823	99.4
Soybean-large	153 796	92 987	39.5	16 729	89.1	91 716	40.4	35 515	76.9	6974	95.5
Splice	674 654	253 896	62.4	205 482	69.5	465 279	31.0	206 965	69.3	119 491	82.3
Tic-tac-toe	4163	2981	28.4	2781	33.2	3535	15.1	2287	45.1	1879	54.9
Tokyo	229 045	83 443	63.6	17 278	92.5	148 181	35.3	54 917	76.0	6498	97.2
Vehicle	50 669	24 287	52.1	18 623	63.2	40 784	19.5	29 222	42.3	9203	81.8
Vote	15 239	3116	79.6	2879	81.1	9546	37.4	3201	79	751	95.1

Table 4 Execution times taken for each search method

Data set name	RULES-6 with no pruning rules	RULES-6 with rule (1) added		RULES-6 with rule (2) added		RULES-6 with rule (3) added		RULES-6 with rule (4) added		RULES-6 with all pruning rules	
	Time (s)	Time (s)	%Reduction	Time (s)	%Reduction	Time (s)	%Reduction	Time (s)	%Reduction	Time (s)	%Reduction
Abalone	5	2	60.0	4	20.0	5	0.0	3	40.0	2	60.0
Adult	2192	1137	48.1	1225	44.1	1500	31.6	1360	38.0	747	65.9
Anneal	39	29	25.6	3	92.3	29	25.6	19	51.3	2	94.9
Australian	6	2	66.7	2	66.7	5	16.7	3	50.0	1	83.3
Auto	3	2	33.3	1	66.7	2	33.3	1	66.7	1	66.7
Balance-scale	0	0	0.0	0	0.0	0	0.0	0	0.0	0	0.0
Breast	1	0	100.0	0	100.0	0	100.0	0	100.0	0	100.0
Breast-cancer	1	0	100.0	0	100.0	1	0.0	0	100.0	0	100.0
Car	1	1	0.0	1	0.0	0	100.0	1	0.0	1	0.0
Chess	390	253	35.1	58	85.1	329	15.6	182	53.3	37	90.5
Cleve	1	1	0.0	1	0.0	1	0.0	1	0.0	0	100.0
Crx	7	3	57.1	4	42.9	6	14.3	4	42.9	2	71.4
Diabetes	1	1	0.0	0	100.0	1	0.0	0	100.0	0	100.0
German	37	16	56.8	15	59.5	29	21.6	20	45.9	6	83.8
German organization	50	30	40.0	12	76.0	50	0.0	39	22.0	7	86.0
Glass2	0	0	0.0	0	0.0	0	0.0	0	0.0	0	0.0
Heart	1	1	0.0	0	100.0	1	0.0	1	0.0	0	100.0
Heart-Hungarian	1	0	100.0	0	100.0	1	0.0	1	0.0	0	100.0
Hepatitis	2	1	50.0	1	50.0	1	50.0	0	100.0	0	100.0
Horse-colic	11	3	72.7	4	63.6	7	36.4	8	27.3	1	90.9
Hypothyroid	106	63	40.6	11	89.6	87	17.9	70	34.0	5	95.3
Ionosphere	16	7	56.3	2	87.5	8	50.0	3	81.3	1	93.8
Iris	0	0	0.0	0	0.0	0	0.0	0	0.0	0	0.0
Lymphography	1	0	100.0	0	100.0	1	0.0	0	100.0	0	100.0
Monk1	0	0	0.0	0	0.0	0	0.0	0	0.0	0	0.0
Monk2	1	0	100.0	1	0.0	0	100.0	1	0.0	0	100.0
Monk3	0	0	0.0	0	0.0	0	0.0	0	0.0	0	0.0
Mushroom	152	152	0.0	34	77.6	84	44.7	37	75.7	26	82.9
Promoter	24	1	95.8	1	95.8	2	91.7	1	95.8	0	100.0
Satimage	2301	622	73.0	1441	37.4	1587	31.0	682	70.4	275	88.0
Segment	45	19	57.8	19	57.8	25	44.4	13	71.1	5	88.9
Shuttle	319	219	31.3	243	23.8	157	50.8	101	68.3	90	71.8
Sick-euthyroid	108	69	36.1	10	90.7	96	11.1	88	18.5	7	93.5
Sonar	78	46	41.0	1	98.7	73	6.4	59	24.4	0	100.0
Soybean-large	36	23	36.1	6	83.3	22	38.9	9	75.0	3	91.7
Splice	1700	516	69.6	373	78.1	1173	31.0	379	77.7	237	86.1
Tic-tac-toe	1	1	0.0	1	0.0	1	0.0	1	0.0	1	0.0
Tokyo	71	26	63.4	7	90.1	42	40.8	18	74.6	2	97.2
Vehicle	12	6	50.0	5	58.3	10	16.7	7	41.7	3	75.0
Vote	2	1	50.0	1	50.0	1	50.0	1	50.0	1	50.0

Table 5 Total number of conditions generated for each search method

Data set name	RULES-6 with no runing rules	RULES-6 with rule (1) added		RULES-6 with rule (2) added		RULES-6 with rule (3) added		RULES-6 with rule (4) added		RULES-6 with all pruning rules	
	Number	Number	%Reduction	Number	%Reduction	Number	%Reduction	Number	%Reduction	Number	%Reduction
Abalone	76	51	32.9	75	1.3	76	0.0	76	0.0	50	34.2
Adult	636	410	35.5	637	−0.2	640	−0.6	640	−0.6	410	35.5
Anneal	46	40	13.0	44	4.3	50	−8.7	50	−8.7	42	8.7
Australian	216	120	44.4	184	14.8	215	0.5	215	0.5	109	49.5
Auto	62	46	25.8	63	−1.6	66	−6.5	68	−9.7	46	25.8
Balance-scale	29	29	0.0	29	0.0	29	0.0	29	0.0	29	0.0
Breast	30	20	33.3	24	20.0	30	0.0	31	−3.3	21	30.0
Breast-cancer	110	60	45.5	104	5.5	110	0.0	110	0.0	62	43.6
Car	179	137	23.5	179	0.0	179	0.0	179	0.0	137	23.5
Chess	222	168	24.3	186	16.2	236	−6.3	240	−8.1	148	33.3
Cleve	94	55	41.5	74	21.3	94	0.0	94	0.0	50	46.8
Crx	214	126	41.1	186	13.1	222	−3.7	222	−3.7	140	34.6
Diabetes	25	25	0.0	25	0.0	25	0.0	25	0.0	25	0.0
German	494	304	38.5	507	−2.6	527	−6.7	517	−4.7	312	36.8
German organization	469	318	32.2	448	4.5	486	−3.6	476	−1.5	319	32.0
Glass2	10	9	10.0	10	0.0	10	0.0	10	0.0	9	10.0
Heart	69	56	18.8	61	11.6	69	0.0	69	0.0	56	18.8
Heart-Hungarian	52	31	40.4	43	17.3	52	0.0	52	0.0	28	46.2
Hepatitis	54	26	51.9	31	42.6	53	1.9	53	1.9	25	53.7
Horse-colic	196	113	42.3	169	13.8	208	−6.1	212	−8.2	125	36.2
Hypothyroid	97	74	23.7	77	20.6	104	−7.2	104	−7.2	45	53.6
Ionosphere	54	36	33.3	47	13.0	49	9.3	43	20.4	30	44.4
Iris	5	5	0.0	5	0.0	5	0.0	5	0.0	5	0.0
Lymphography	38	26	31.6	46	−21.1	38	0.0	38	0.0	33	13.2
Monk1	61	61	0.0	61	0.0	61	0.0	61	0.0	61	0.0
Monk2	208	235	−13.0	208	0.0	208	0.0	208	0.0	235	−13.0
Monk3	23	23	0.0	23	0.0	23	0.0	23	0.0	23	0.0
Mushroom	50	50	0.0	40	20.0	73	−46.0	62	−24.0	67	−34.0
Promoter	23	13	43.5	20	13.0	23	0.0	23	0.0	14	39.1
Satimage	1909	640	66.5	1250	34.5	1317	31.0	1296	32.1	664	65.2
Segment	177	117	33.9	177	0.0	177	0.0	180	−1.7	113	36.2
Shuttle	118	108	8.5	110	6.8	119	−0.8	119	−0.8	108	8.5
Sick-euthyroid	115	75	34.8	86	25.2	115	0.0	115	0.0	73	36.5
Sonar	82	50	39.0	52	36.6	83	−1.2	83	−1.2	44	46.3
Soybean-large	111	85	23.4	105	5.4	111	0.0	111	0.0	84	24.3
Splice	906	458	49.4	564	37.7	625	31.0	587	35.2	474	47.7
Tic-tac-toe	76	71	6.6	80	−5.3	76	0.0	76	0.0	71	6.6
Tokyo	112	70	37.5	79	29.5	117	−4.5	115	−2.7	48	57.1
Vehicle	225	149	33.8	220	2.2	225	0.0	225	0.0	140	37.8
Vote	55	21	61.8	47	14.5	55	0.0	55	0.0	22	60.0

Table 6 Classification accuracies obtained with each search method

	RULES-6 with no pruning rules	RULES-6 with rule (1) added		RULES-6 with rule (2) added		RULES-6 with rule (3) added		RULES-6 with rule (4) added		RULES-6 with all pruning rules	
Data set name	Accuracy (%)	Accuracy(%)	%Increase	Accuracy(%)	%Increase	Accuracy(%)	%Increase	Accuracy(%)	%Increase	Accuracy(%)	%Increase
Abalone	25.5	25.3	−0.8	25.5	0.0	25.5	0.0	25.5	0.0	25.5	0.0
Adult	83.4	83.1	−0.4	83.4	0.0	83.4	0.0	83.4	0.0	83.2	−0.3
Anneal	94.0	93.0	−1.1	93.7	−0.4	94.0	0.0	94.0	0.0	94.0	0.0
Australian	84.3	83.5	−1.0	83.0	−1.5	83.9	−0.5	83.9	−0.5	85.7	1.5
Auto	63.1	65.2	3.4	63.8	1.1	65.2	3.4	62.3	−1.2	62.3	−1.2
Balance-scale	64.6	64.6	0.0	64.6	0.0	64.6	0.0	64.6	0.0	64.6	0.0
Breast	93.1	94.0	0.9	94.4	1.4	93.1	0.0	92.7	−0.5	93.1	0.0
Breast-cancer	72.6	74.7	2.9	71.6	−1.4	72.6	0.0	72.6	0.0	74.7	2.9
Car	83.7	84.2	0.6	83.7	0.0	83.7	0.0	83.7	0.0	84.2	0.6
Chess	98.0	98.2	0.2	97.9	−0.1	97.7	−0.3	97.7	−0.3	98.6	0.6
Cleve	81.2	80.2	−1.2	81.2	0.0	81.2	0.0	81.2	0.0	81.2	0.0
Crx	80.0	71.5	−10.6	81.0	1.3	80.0	0.0	80.0	0.0	81.0	1.3
Diabetes	71.5	71.5	0.0	71.5	0.0	71.5	0.0	71.5	0.0	71.5	0.0
German	72.5	74.9	3.3	71.0	−2.1	73.1	0.8	73.1	0.8	74.0	2.1
German organization	74.6	72.8	−2.4	72.5	−2.8	74.3	−0.4	73.7	−1.2	75.4	1.2
Glass2	74.5	74.5	0.0	74.5	0.0	74.5	0.0	74.5	0.0	74.5	0.0
Heart	83.3	83.3	0.0	83.3	0.0	83.3	0.0	83.3	0.0	83.3	0.0
Heart-Hungarian	79.6	78.6	−1.3	80.6	1.3	79.6	0.0	79.6	0.0	79.6	0.0
Hepatitis	82.7	82.7	0.0	84.6	2.3	82.7	0.0	82.7	0.0	84.6	2.3
Horse-colic	76.5	75.0	−1.9	83.8	9.6	76.5	0.0	76.5	0.0	75.0	−1.9
Hypothyroid	96.9	97.0	0.1	96.9	0.0	96.9	0.0	96.9	0.0	96.9	0.0
Ionosphere	90.6	91.5	0.9	93.2	2.8	88.9	−1.9	89.7	−0.9	91.5	0.9
Iris	96.0	96.0	0.0	96.0	0.0	96.0	0.0	96.0	0.0	96.0	0.0
Lymphography	84.0	80.0	−4.8	82.0	−2.4	84.0	0.0	84.0	0.0	88.0	4.8
Monk1	100.0	100.0	0.0	100.0	0.0	100.0	0.0	100.0	0.0	100.0	0.0
Monk2	84.8	85.4	0.7	84.8	0.0	84.8	0.0	84.8	0.0	85.4	0.7
Monk3	95.1	95.1	0.0	95.1	0.0	95.1	0.0	95.1	0.0	95.1	0.0
Mushroom	100.0	100.0	0.0	100.0	0.0	100.0	0.0	99.9	−0.1	100.0	0.0
Promoter	92.3	88.6	−4.0	85.7	−7.1	85.7	−7.1	85.7	−7.1	85.7	−7.1
Satimage	82.0	83.0	1.2	84.5	3.0	84.8	3.4	84.4	2.9	82.9	1.0
Segment	91.7	90.9	−0.8	91.2	−0.6	91.7	0.0	91.7	0.0	90.5	−1.3
Shuttle	99.6	99.7	0.1	99.6	0.0	99.6	0.0	99.6	0.0	99.7	0.1
Sick-euthyroid	97.2	93.2	−4.1	97.3	0.1	97.2	0.0	97.2	0.0	97.0	−0.2
Sonar	67.1	67.1	0.0	68.6	2.1	67.1	0.0	67.1	0.0	68.6	2.1
Soybean-large	86.8	82.0	−5.6	90.4	4.0	86.8	0.0	86.8	0.0	86.8	0.0
Splice	92.0	93.3	1.4	92.8	0.8	92.5	0.5	92.1	0.1	92.4	0.4
Tic-tac-toe	98.8	98.8	0.0	98.4	−0.3	98.8	0.0	98.8	0.0	98.8	0.0
Tokyo	91.3	91.3	0.0	90.0	−1.4	91.7	0.5	91.5	0.2	91.0	−0.2
Vehicle	69.5	67.4	−3.1	68.1	−2.0	69.5	0.0	69.5	0.0	68.1	−2.0
Vote	95.6	95.6	0.0	94.8	−0.8	95.6	0.0	95.6	0.0	95.6	0.0

Table 7 Results for RULES-3 Plus and RULES-6

Data set name	RULES-3 Plus Accuracy (%)	RULES-3 Plus Number of rules	RULES-3 Plus Number of conditions	RULES-3 Plus Number of rules explored	RULES-3 Plus CPU time (s)	RULES-6 Accuracy (%)	RULES-6 Number rules	RULES-6 Number of conditions	RULES-6 Number of rules explored	RULES-6 CPU time (s)
Abalone	18.5	313	1947	26 853	28	**25.3**	**21**	**49**	**1012**	**1**
Adult	77.5	6686	70 144	1986 685	29 938	**83.1**	**118**	**395**	**16 193**	**415**
Anneal	**99.7**	37	119	10 119	3	93.3	**16**	**45**	**1912**	**1**
Australian	83.9	148	807	26 301	4	**85.2**	**29**	**115**	**2892**	**0**
Auto	62.3	48	94	5534	0	62.3	**14**	**44**	**1582**	0
Balance-scale	**77.0**	213	691	3341	1	64.6	**11**	**29**	**155**	**0**
Breast	**95.7**	40	94	2023	1	92.3	**10**	**20**	**257**	**0**
Breast-cancer	68.4	86	284	5674	1	**72.6**	**26**	**74**	**1311**	**0**
Car	**88.4**	165	801	7826	2	84.2	**44**	**137**	**1374**	**1**
Chess	**99.0**	108	2164	176 109	347	98.5	**31**	**141**	**8728**	**19**
Cleve	77.7	33	73	2214	0	**82.2**	**17**	**48**	**913**	0
Crx	**80.0**	142	863	30 277	4	79.5	**34**	**119**	**3624**	**1**
Diabetes	66.8	190	739	12 399	2	**71.5**	**12**	**25**	**305**	**0**
German	70.9	247	1043	57 120	13	**75.7**	**77**	**289**	**8402**	**3**
German organization	66.4	252	1381	90 770	29	**76.6**	**58**	**286**	**9684**	**3**
Glass2	69.1	46	154	2894	0	**78.2**	**5**	**8**	**43**	0
Heart-disease	81.1	60	158	4985	1	**83.3**	**16**	**52**	**725**	**0**
Heart-Hungarian	72.4	48	196	5611	1	**79.6**	**11**	**28**	**396**	**0**
Hepatitis	61.5	25	47	2023	0	**82.7**	**11**	**29**	**519**	0
Horse-colic	75.0	91	223	12 526	1	**80.9**	**31**	**105**	**3902**	1
Hypothyroid	94.9	138	1743	88 221	164	**95.5**	**17**	**44**	**1000**	**2**
Ionosphere	92.3	48	94	7654	2	**94.0**	**15**	**38**	**1588**	**1**
Iris	94.0	13	25	122	0	**96.0**	**4**	**5**	**20**	0
Lymphography	80.0	26	56	2431	0	**86.0**	**15**	**37**	**882**	0
Monk1	100.0	22	61	759	0	100.0	22	61	**652**	0
Monk2	**98.8**	262	1504	13 709	1	83.6	**47**	**174**	**1572**	**0**
Monk3	95.1	12	23	270	0	95.1	12	23	**263**	0
Mushroom	100.0	**25**	**37**	**1556**	**5**	100.0	28	83	2779	14
Promoter	74.3	14	26	3481	1	**77.1**	**9**	**14**	**1146**	**0**
Satimage	82.0	915	7993	798 350	1943	**82.8**	**196**	**666**	**42 926**	**155**
Segment	**90.5**	172	1198	51 880	35	89.6	**42**	**112**	**3136**	**3**
Shuttle	91.7	63	289	4689	87	**99.7**	**55**	**108**	**1927**	**60**
Sick-euthyroid	89.4	195	3119	154 065	291	**97.2**	**22**	**66**	**1678**	**3**
Sonar	68.6	37	67	9293	1	**70.0**	**13**	**39**	**921**	**0**
Soybean-large	**93.9**	76	542	46 253	13	82.0	**29**	**82**	**3953**	**1**
Splice	91.8	239	1127	209 203	340	**92.7**	**135**	**474**	**66 354**	**118**
Tic-tac-toe	94.7	89	374	7970	1	**97.8**	**29**	**101**	**1757**	**0**
Tokyo	**91.3**	83	478	48 551	27	89.4	**19**	**49**	**3673**	**2**
Vehicle	59.6	214	875	42 013	7	**68.1**	**31**	**125**	**4032**	**1**
Vote	**97.0**	33	134	4999	0	95.6	**10**	**22**	**464**	0
Total	3271.0	11 654	10 1787	3966 753	33 294	**3343.9**	**1342**	**4361**	**204 652**	**805**

Bold figures indicate the best results.

particularly notable. The fewer and more general rules created by the RULES-6 algorithm made it much faster than RULES-3 Plus as indicated in Table 7. The total number of evaluations fell by 94.8 per cent (from 3966 753 to 204 652) and this was accompanied by a total reduction of 97.6 per cent in the execution time from 33 294 to 805 s. These results confirm that RULES-6 is more robust to noise and more accurate than RULES-3 Plus.

4.3 Comparison with C5.0

RULES-6 was compared with C5.0 for the 40 data sets listed in Table 2. C5.0 has a facility to generate a set of pruned production rules from a decision tree. Table 8 presents the results for each algorithm on each data set. In each case, the accuracy of the test data and the complexity of the resulting rule sets are given. The number of rules was taken as a measure of the complexity of the rule set. A complexity of 1 was assigned to the default rule.

It is clear from Table 8 that the accuracy obtained by RULES-6 was, in total, higher than that of C5.0. In addition, RULES-6 achieved higher accuracies for 19 out of 40 data sets, whereas C5.0 yielded better accuracies for 17 out of 40 data sets. Both algorithms achieved similar accuracies for the remaining four data sets. It is also clear from the table that, in total, RULES-6 created fewer rules than C5.0.

Table 8 Results for RULES-6 and C5.0

Data set name	C5.0 Accuracy (%)	C5.0 Number of Rules	RULES-6 Accuracy (%)	RULES-6 Number of Rules
Abalone	23.4	522	**25.3**	**21**
Adult	**86.4**	**100**	83.1	118
Anneal	93.3	**11**	93.3	16
Australian	**87.4**	**20**	85.2	29
Auto	62.3	23	62.3	**14**
Balance-scale	**81.3**	19	64.6	**11**
Breast	**95.0**	**9**	92.3	10
Breast-cancer	**75.8**	**17**	72.6	26
Car	**91.8**	58	84.2	**44**
Chess	97.2	**21**	**98.5**	31
Cleve	77.2	**13**	**82.2**	17
Crx	**84.5**	**23**	79.5	34
Diabetes	70.7	14	**71.5**	**12**
German	72.7	**15**	**75.7**	77
German organization	71.8	**17**	**76.6**	58
Glass2	69.1	9	**78.2**	**5**
Heart	78.9	**12**	**83.3**	16
Heart-Hungarian	74.5	**7**	**79.6**	11
Hepatitis	76.9	**5**	**82.7**	11
Horse-colic	**83.8**	**10**	80.9	31
Hypothyroid	94.8	**5**	**95.5**	17
Ionosphere	89.7	**6**	**94.0**	15
Iris	92.0	5	**96.0**	**4**
Lymphography	76.0	**7**	**86.0**	15
Monk1	100.0	**17**	100.0	22
Monk2	65.7	**1**	**83.6**	47
Monk3	**100.0**	**6**	95.1	12
Mushroom	99.8	**10**	**100.0**	28
Promoter	74.3	**7**	**77.1**	9
Satimage	**86.9**	**118**	82.8	196
Segment	**93.4**	**24**	89.6	42
Shuttle	**99.9**	**12**	99.7	55
Sick-euthyroid	90.4	**8**	**97.2**	22
Sonar	**74.3**	**11**	70.0	13
Soybean-large	**93.4**	32	82.0	**29**
Splice	92.7	**60**	92.7	135
Tic-tac-toe	92.2	34	**97.8**	**29**
Tokyo	**92.3**	**8**	89.4	19
Vehicle	**69.9**	46	68.1	**31**
Vote	**97.0**	**5**	95.6	10
Total	3328.9	1347	**3343.9**	**1342**

Bold figures indicate the best results.

However, with RULES-6, the number of rules was lower for 10 data sets but higher for 30 data sets. The smaller number of rules produced by C5.0 can be attributed to the rule set (decision tree) pruning techniques employed. Research is ongoing to develop pruning techniques for the RULES-6 algorithm. Overall, RULES-6 is very competitive when compared with C5.0.

5 CONCLUSIONS AND FUTURE WORK

RULES-6 is an improved version of the RULES-3 Plus algorithm. The innovation in RULES-6 is that it has the ability to handle noise in the data, which is achieved by employing a search method that tolerates inconsistency in the rule specialization process. This makes the rule sets extracted by RULES-6 both more accurate and substantially simpler than those produced using RULES-3 Plus. RULES-6 also employs appropriate search-space pruning rules to avoid useless specializations and to terminate search during rule construction, which substantially increases the efficiency of the learning process. Finally, RULES-6 adopts a very simple criterion for evaluating the quality of rules and a robust method for handling attributes with continuous values, which further improves the performance of the algorithm. The new features of RULES-6 make it not only more robust and effective but also more efficient, thus enhancing the usefulness of the algorithm for applications involving very large data sets.

More work could be carried out to improve the performance of the RULES-6 algorithm. Additional rule-space pruning strategies could be considered to increase the speed of the learning algorithm further. Postpruning techniques could also be used to reduce the error and complexity of the learned rule set in a postprocessing phase. Finally, a method for discretization of continuous-valued attributes during the learning process could be considered. Incorporating discretization into the learning process has the advantage of taking into account the bias inherent in the learning system as well as the interactions between the different attributes.

ACKNOWLEDGEMENTS

This work was carried out within the ERDF (Objective One) projects ‘Innovation in Manufacturing’, ‘Innovative Technologies for Effective Enterprises’, and ‘Supporting Innovative Product Engineering and Responsive Manufacturing’ (SUPERMAN) and within the project ‘Innovative Production Machines and Systems’ (I*PROMS).

REFERENCES

1 **Braha, D.** *Data mining for design and manufacturing: methods and applications*, 2001 (Kluwer Academic Publishers, Boston).

2 **Monostori, L.** AI and machine learning techniques for managing complexity, changes and uncertainties in manufacturing. In Proceedings of the 15th Triennial World Congress, Barcelona, Spain, 2002, pp. 119–130.

3 **Pham, D. T.** and **Afify, A. A.** Machine learning techniques and their applications in manufacturing. *Proc. Instn Mech. Engrs, Part B: J. Engineering Manufacture*, 2005, **219**(B5), 395–412.

4 **Breiman, L., Friedman, J. H., Olshen, R. A.,** and **Stone, C. J.** *Classification and regression trees*, 1984 (Belmont, Wadsworth).

5 **Quinlan, J. R.** Learning efficient classification procedures and their application to chess endgames. In *Machine learning: an artificial intelligence approach* (Eds R. S. Michalski, J. G. Carbonell, and T. M. Mitchell), 1983, Vol. I, pp. 463–482 (Tioga Publishing Co., Palo Alto, CA).
6 **Quinlan, J. R.** Induction of decision trees. *Mach. Learn.*, 1986, **1**, 81–106.
7 **Quinlan, J. R.** *C4.5: Programs for machine learning*, 1993 (Morgan Kaufmann, San Mateo, CA).
8 **ISL.** *Clementine data mining package*, 1998 (SPSS UK Ltd., Surrey, UK).
9 **RuleQuest.** Data Mining Tools C5.0. Pty Ltd, 30 Athena Avenue, St Ives NSW 2075, Australia, available from: http://www.rulequest.com/see5-info.html, accessed 1 February 2003.
10 **Mehta, M., Agrawal, R.,** and **Rissanen, J.** SLIQ: a fast scalable classifier for data mining. In Proceedings of the fifth International Conference on *Extending database technology*, Avignon, France, 1996, pp. 18–32.
11 **Shafer, J., Agrawal, R.,** and **Mehta, M.** SPRINT: a scalable parallel classifier for data mining. In Proceedings of the 22nd International Conference on *Very large data bases (VLDB)*, Mumbai (Bombay), India, 1996, pp. 544–555.
12 **Rastogi, R.** and **Shim, K.** PUBLIC: a decision tree classifier that integrates building and pruning. In Proceedings of the 24th International Conference on *Very large data bases (VLDB)*, New York, USA, 1998, pp. 404–415.
13 **Michalski, R. S.** On the quasi-minimal solution of the general covering problem. In Proceedings of the fifth International Symposium on *Information processing (FCIP 69)*, Bled, Yugoslavia, 1969, **A3** (Switching Circuits), pp. 125–128.
14 **Michalski, R. S., Mozetic, I., Hong, J.,** and **Lavrac, N.** The multi-purpose incremental learning system AQ15 and its testing application to three medical domains. *American association of artificial intelligence*, 1986, pp. 1041–1045 (Morgan Kaufmann, Los Altos, CA).
15 **Michalski, R. S.** and **Kaufman, K. A.** The AQ19 system for machine learning and pattern discovery: a general description and user guide. Reports of the Machine Learning and Inference Laboratory, MLI 01-2, George Mason University, Fairfax, VA, USA, 2001.
16 **Clark, P.** and **Niblett, T.** The CN2 induction algorithm. *Mach. Learn.*, 1989, **3**(4), 261–284.
17 **Clark, P.** and **Boswell, R.** Rule induction with CN2: some recent improvements. In Proceedings of the fifth European Conference on *Artificial intelligence*, Porto, Portugal, 1991, pp. 151–163.
18 **Cohen, W. W.** Fast effective rule induction. In Proceedings of the 12th International Conference on *Machine learning*, Lake Tahoe City, California, USA, 1995, pp. 115–123 (Morgan Kaufmann, San Francisco, CA).
19 **Cohen, W. W.** and **Singer, Y.** A simple, fast and effective rule learner. In Proceedings of the 16th National Conference on *Artificial intelligence*, Menlo Park, CA, 1999, pp. 335–342 (AAAI/MIT Press, Menlo Park, CA).
20 **Rivest, R.** Learning decision lists. *Mach. Learn.*, 1987, **2**, 229–246.
21 **Pham, D. T.** and **Aksoy, M. S.** An algorithm for automatic rule induction. *Artificial Intelligence in Engineering*, 1993, **8**(4), 227–282.
22 **Pham, D. T.** and **Aksoy, M. S.** RULES: a simple rule extraction system. *Expert Syst. Appl.*, 1995, **8**(1), 59–65.
23 **Pham, D. T.** and **Aksoy, M. S.** A new algorithm for inductive learning. *J. Syst. Eng.*, 1995, **5**, 115–122.
24 **Pham, D. T.** and **Dimov, S. S.** An efficient algorithm for automatic knowledge acquisition. *Pattern Recog.*, 1997, **30**(7), 1137–1143.
25 **Lee, C.** Generating classification rules from databases. In Proceedings of the Ninth Conference on *Application of artificial intelligence in engineering*, PA, USA, 1994, pp. 205–212.
26 **Pham, D. T.** and **Dimov, S. S.** An algorithm for incremental inductive learning. *Proc. Instn. Mech. Engrs, Part B: J. Engineering Manufacture*, 1997, **211**, 239–249.
27 **Pham, D. T., Bigot, S.,** and **Dimov, S. S.** RULES-5: a rule induction algorithm for problems involving continuous attributes. *Proc. Instn Mech. Engrs, Part C: J. Mechanical Engineering Science*, 2003, **217**(C12), pp. 1273–1286.
28 **Pham, D. T.** and **Dimov, S. S.** An approach to concurrent engineering. *Proc. Instn Mech. Engrs, Part B: J. Engineering Manufacture*, 1998, **212**(B1), 13–27.
29 **Jennings, N. R.** Automated visual inspection of engine valve stem seals. Internal Report, University of Wales Cardiff, Cardiff, UK, 1996.
30 **Pagallo, G.** and **Haussler, D.** Boolean feature discovery in empirical learning. *Mach. Learn.*, 1990, **3**, 71–99.
31 **Domingos, P.** *A unified approach to concept learning*. PhD Thesis, University of California, Irvine, 1997.
32 **Holte, R. C., Acker, L. E.,** and **Porter, B. W.** Concept learning and the problem of small disjuncts. In Proceedings of the 11th International Joint Conference on *Artificial intelligence*, Detroit, Michigan, USA, 1989, pp. 813–818.
33 **Weiss, G. M.** Learning with rare cases and small disjuncts. In Proceedings of the 12th International Conference on *Machine learning*, Lake Tahoe City, California, USA, 1995, pp. 558-565 (Morgan Kaufmann, San Francisco, CA).
34 **Weiss, G. M.** and **Hirsh, H.** The problem with noise and small disjuncts. In Proceedings of the 15th International Conference on *Machine learning*, Madison, Wisconsin, USA, 1998, pp. 574–578 (Morgan Kaufmann, San Francisco, CA).
35 **Frayman, Y., Ting, K. M.,** and **Wang, L.** A fuzzy neural network for data mining: dealing with the problem of small disjuncts. In IEEE International Joint Conference on *Neural networks (IJCNN-99)*, Washington, DC, 1999, **4**, pp. 2490–2493.
36 **Weiss, G. M.** and **Hirsh, H.** A quantitative study of small disjuncts. In Proceedings of the 17th National Conference on *Artificial intelligence*, Austin, Texas, 2000, pp. 665–670.
37 **Weiss, S.** and **Indurkhya, N.** Reduced complexity rule induction. In Proceedings of the 12th International Joint Conference on *Artificial intelligence*, Sydney, Australia, 1991 pp. 678–684 (Morgan Kaufmann, San Francisco, CA).
38 **Cendrowska, J.** PRISM: an algorithm for inducing modular rules. *Int. J. Man–Mach Studies*, 1987, **27**, 349–370.
39 **Fürnkranz, J.** and **Widmer, G.** Incremental reduced error pruning. In Proceedings of the 11th International Conference on *Artificial intelligence*, Amsterdam, The Netherlands, 1994, pp. 453–457.

40 **Muggleton, S.** *Foundations of inductive logic programming*, 1995 (Prentice Hall, Englewood Cliffs, NJ).
41 **Webb, G.** Systematic search for categorical attribute-value data-driven machine learning. In *AI '94–Proceedings of the sixth Australian joint conference on artificial intelligence* (Eds C. Rowles, H. Liu, and N. Foo), Melbourne, 1993, pp. 342–347 (World Scientific, Chicago).
42 **Webb, G.** OPUS: an efficient admissible algorithm for unordered search. *J. Artif. Intell. Res.*, 1995, **3**, 431–465.
43 **Segal, R. B.** *Machine learning as massive search*. PhD Thesis, Department of Computer Science and Engineering, University of Washington, 1997.
44 **Cestnik, B.** Estimating probabilities: a crucial task in machine learning. In Proceedings of the Third European Conference on *Artificial intelligence (ECAI-90)*, Stockholm, Sweden, 1990, pp. 147–149 (Pitman, London).
45 **Afify, A. A.** *Design and analysis of scalable rule induction systems*. PhD Thesis, University of Wales Cardiff, School of Engineering, Systems Engineering Division, Cardiff, UK, 2004.
46 **Good, I. J.** *The estimation of probabilities: an essay on modern Bayesian methods*, 1965 (MIT Press, Cambridge, MA).
47 **Mitchell, T. M.** *Machine learning*, 1997 (McGraw Hill, New York).
48 **Ho, K. M.** and **Scott, P. D.** Zeta: a global method for discretization of continuous variables. *IEEE Trans. Knowl. Data Eng.*, 1997, **9**(5), 718–730.
49 **Zighed, D. A., Rakotomalala, R.,** and **Feschet, F.** Optimal multiple intervals discretization of continuous attributes for supervised learning. In Proceedings of the Third International Conference on *Knowledge discovery and data mining (KDD-97)*, Newport Beach, California, USA, 1997, pp. 295–298 (AAAI Press, Menlo Park, CA).
50 **Frank, E.** and **Witten, I. H.** Making better use of global discretization. In Proceedings of the 15th International Conference on *Machine learning*, Madison, Wisconsin, USA, 1998, pp. 152–160.
51 **Trautzsch, S.** and **Perner, P.** Multi-interval discretization methods for decision tree learning. In *Advances in pattern recognition* (Eds A. Amin, D. Dori, P. Pudil, and H. Freeman), 1998, **1451**, pp. 475–482 (Springer-Verlag, Germany).
52 **An, A.** and **Cercone, N.** Discretisation of continuous attributes for learning classification rules. In Proceedings of the Third Pacific-Asia Conference on *Knowledge discovery and data mining (PAKDD-99)*, Kyoto, Japan, 1999, pp. 509–514.
53 **Wong, A. K. C.** and **Chiu, D. K. Y.** Synthesizing statistical knowledge from incomplete mixed-mode data. *IEEE Trans. Pattern Anal. Mach. Intell.*, 1987, **9**(6), 796–805.
54 **Holte, R. C.** Very simple classification rules perform well on most commonly used data sets. *Mach. Learn.*, 1993, **11**, 63–90.
55 **Catlett, J.** On changing continuous attributes into ordered discrete attributes. In Proceedings of the European Working Session on *Learning*, Porto, Portugal, 1991, pp. 164–178 (Springer-Verlag, Germany).
56 **Fayyad, U. M.** and **Irani, K. B.** Multi-interval discretization of continuous-valued attributes for classification. In Proceedings of the 13th International Joint Conference on *Artificial intelligence*, Chambery, France, 1993, pp. 1022–1027.
57 **Fayyad, U. M.** and **Irani, K. B.** On the handling of continuous-valued attributes in decision tree generation. *Mach. Learn.*, 1992, **8**, 87–102.
58 **Maass, W.** Efficient agnostic PAC-learning with simple hypotheses. In Proceedings of the Seventh Annual ACM Conference on *Computational learning theory*, New Brunswick, New Jersey, USA, 1994, pp. 67–75.
59 **Auer, P., Holte, R. C.,** and **Maass, W.** Theory and application of agnostic PAC-learning with small decision trees. In Proceedings of the 12th International Conference on *Machine learning*, Lake Tahoe City, California, USA, 1995, pp. 21–29 (Morgan Kaufmann, San Francisco, CA).
60 **Fulton, T., Kasif, S.,** and **Salzberg, S.** Efficient algorithms for finding multi-way splits for decision trees. In Proceedings of the 12th International Conference on *Machine learning*, Lake Tahoe City, California, USA, 1995, pp. 244–251 (Morgan Kaufmann, San Francisco CA).
61 **Birkendorf, A.** On fast and simple algorithms for finding maximal subarrays and applications on computational learning theory. In *Lecture notes in artificial intelligence*, 1997, **1208**, pp. 198–209 (Springer-Verlag, Heidelberg).
62 **Rousu, J.** *Efficient range partitioning in classification learning*. PhD Thesis, Department of Computer Science, University of Helsinki, Finland, 2001.
63 **Cai, Z.** *Technical aspects of data mining*. PhD Thesis, University of Wales Cardiff, Cardiff, UK, 2001.
64 **Dougherty, J., Kohavi, R.,** and **Sahami, M.** Supervised and unsupervised discretization of continuous features. In Proceedings of the 12th International Conference on *Machine learning*, Lake Tahoe City, California, USA, 1995, pp. 194–202 (Morgan Kaufmann, San Francisco, CA).
65 **Ventura, D.** and **Martinez, T. R.** An empirical comparison of discretization methods. In Proceedings of the 10th International Symposium on *Computer and information sciences*, Aydin, Turkey, 1995, pp. 443–450.
66 **Kohavi, R.** and **Sahami, M.** Error-based and entropy-based discretization of continuous features. In Proceedings of the second International Conference on *Knowledge discovery in databases* (Eds E. Simondis, U. Fayyad), 1996, pp. 114–119 (AAAI Press, Menlo Park, CA).
67 **Blake, C. L.** and **Merz, C. J.** UCI repository of machine learning databases. University of California, Department of Information and Computer Science, Irvine, CA, 1998, available from http://www.ics.uci.edu/~mlearn/MLRepository.html, accessed 1 February 2003.
68 **Devijver, P. A.** and **Kittler, J.** *Pattern recognition: a statistical approach*, 1982 (Prentice Hall, Englewood Cliffs, London).
69 **Efron, B.** and **Tibshirani, R.** *An introduction to the bootstrap*, 1993 (Chapman and Hall, USA).

APPENDIX

Notation

A_i	the ith attribute in an example
C_t	the target class (the class to be learned)
k	the number of classes in a data set
m	a domain-dependent parameter
MDL	minimum description length
n_{class}	the number of positive instances covered by a given rule
$n_{covered}$	the total number of instances covered by a given rule
n_{C_t}	the number of instances in the target class C_t
N	the total number of instances in the training data set
$P_0(C_t)$	the *a priori* probability of the target class C_t
r'	any specialization of rule r
RULES	RULe Extraction System
RULES-3 Plus	RULe Extraction System – Version 3 Plus
RULES-6	RULe Extraction System – Version 6
s	a seed example
v_{is}	the value of attribute A_i in the seed example s
w	the beam width