RULES-6: A Simple Rule Induction Algorithm for Supporting Decision Making

D T Pham
Manufacturing Engineering Centre
School of Engineering, Cardiff University
Cardiff, Queen's Buildings, The Parade,
Newport Road, Cardiff CF24 3AA
phamdt@cf.ac.uk

A A Afify
Manufacturing Engineering Centre
School of Engineering, Cardiff University
Cardiff, Queen's Buildings, The Parade,
Newport Road, Cardiff CF24 3AA
afifyae@cf.ac.uk

***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 for handling continuous attributes and noisy data are needed. This paper presents a new rule induction algorithm called RULES-6, 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 continuous attributes handling. A detailed empirical evaluation of the algorithm is reported in the paper. The results presented demonstrate the strong performance of the algorithm.**

1. INTRODUCTION

Recently, there has been substantial attention devoted to the use of machine learning techniques as tools for decision support. These methods have been applied to a wide variety of problems in engineering [1-3] because of their ability to discover patterns from data. The integration of these methods with conventional decision support systems can provide a means for significantly improving the quality of decision making. A decision support system can employ machine learning techniques to derive knowledge directly from prior decision examples and to refine this knowledge continually.

Inductive learning is perhaps the most widely used machine learning technique. Inductive learning algorithms are simple and fast. Another advantage is that they generate models that are easy to understand. Finally, inductive learning algorithms are more accurate compared with other machine learning techniques.

Inductive learning techniques can be divided into two main categories, namely, decision tree induction and rule induction [4]. RULES (RULe Extraction System) is a family of inductive learning algorithms that follow the rule induction approach. The first three algorithms in the RULES family of algorithms (RULES-1, 2 and 3) were developed by Pham and Aksoy [5-7]. Later, the rule forming procedure of RULES-3 was improved by Pham and Dimov [8] and the new algorithm was called RULES-3 Plus. Compared to 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 [9] for selecting and sorting candidate rules according to their generality and accuracy. The first incremental learning algorithm in the RULES family was RULES-4 [10]. 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. [11] 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 [12], the mapping of manufacturing information to design features [12] and the classification of defects in automated visual inspection [13]. 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[1]. 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 discretised using a simplistic equal-width approach [14] before data is passed to the learning system. This discretisation 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 organised 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

[1] Instances of the target class (the class of the training instance under consideration) in the training set are called positive instances. Instances in the training set that do not belong to the target class are called negative instances.

0-7803-9252-3/05/$20.00 ©2005 IEEE

data. Section 5 concludes the paper and provides suggestions for future work.

2. RULES-3 PLUS

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 the examples covered by previously formed rules are only marked in order to stop RULES-3 Plus from repeatedly finding the same rule. However, these examples are used to guide the specialisation process and to assess the accuracy and generality of each newly formed rule.

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 specialises it in steps considering only conditions extractable from the selected seed example. The aim of specialisation 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.

A pseudo-code description of RULES-3 Plus and a simple example clearly illustrating its operation can be found in [8].

3. RULES-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.

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

```
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

```
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 specialised and *NewPartialRules*: a new list of rules to be used for further specialisations.

RULES-3 Plus, 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 specialisation process. Only the rules in this list are considered for further specialisation. *NewPartialRules* is used to save valid partial rules obtained by specialising the rules in *PartialRules*. The learning of single rules starts with the most general rule whose body is empty (step (1) in Figure 2) and specialises it by incrementally adding conditions to its body (step (3) in Figure 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, an off-line discretisation 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 Figure 2). The new rule is then inserted into the *NewPartialRules* list (step (9) in Figure 2) unless one of the conditional tests (step (5), (6) or (7) in Figure 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 as ineffective and additional specialisations 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 specialisations of the same parent rule (step (8) in Figure 2). Thus, the *NewPartialRules* list only contains useful rules that can be employed for further specialisation. This process is repeated until there are no remaining rules to be specialised 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 specialisation step is identified. Rules that satisfy the conditional test at step (10) are removed from the *NewPartialRules* list (step (11) in Figure 2), again, 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 Figure 2). All *InvalidValues* are then deleted from the corresponding set of *ValidValues* for each rule in the *NewPartialRules* list (step (13) in Figure 2). Such values cannot lead to a viable specialisation from any point in the search space that can be reached via identical sets of specialisations and thus excluding them will prevent the unnecessary construction of ineffective specialisations at subsequent specialisation 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 Figure 2). The comparison between rules is based on the quality measure defined in the next section. 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 specialisation process is then repeated until the *PartialRules* list becomes empty (step (2) in Figure 2) due 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 specialisation, the best rule obtained is stored and returned at the end of the procedure. In RULES-3 Plus, the specialisation 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. On the other hand, if the search process continues, more general rules might be created.

The following sections discuss the key ideas underlying RULES-6 in more 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 optimise 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 existing algorithms. These include *purity*, *information content, entropy*, the metric applied in RIPPER, *accuracy*, *Laplace* and the *m-probability-estimate* [15, 16].

The performance of the *H* measure and the seven quality measures mentioned above when used with the RULES-6 algorithm was evaluated empirically [17]. 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 to select the best rule (step (4) in Figure 2) and to decide on the best specialisations to retain (step (14) in Figure 2) after each specialisation step.

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 organised 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 [18]. This structure implies that all specialisations of a rule cover a monotonically decreasing number of positive and negative instances. This organisation 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 maximise the quality measure, thus significantly speeding up the search process. Since the rules 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. These pruning rules and how their effect is maximised are detailed in [17].

3.3. Discretisation Method

Since most real-world applications of classification learning involve continuous-valued attributes, properly handling these attributes is important. Discretisation of the data is one possibility. The usual approach to discretisation of continuous-valued attributes is to perform this discretisation off-line, prior to the learning process [19]. First, all continuous attributes in the data are discretised to obtain a discrete data set. Then learning algorithms are applied to this discretised data set.

Several off-line discretisation methods have been developed. Four different state-of-the-art off-line discretisation procedures are the *equal-width* method proposed by Wong and Chiu [14], the *1R Discretizer* proposed by Holte [20], the *entropy-based discretisation* method by Fayyad and Irani [21] and the *"optimal"* discretisation method of Cai [22]. These methods are representative of the existing discretisation techniques and widely used in other comparative studies.

Table 1. Search-space pruning rules employed by RULES-6
r′ is any specialisation of rule r and Prune (r) indicates that the children of r should not be searched.

1. **If** Covered_Positives (r) ≤ MinPositives **Then** Prune (r)
2. **If** Covered_Negatives (r) – Covered_Negatives (r′) ≤ MinNegatives **Then** Prune (r′)
3. **If** Consistency (r) = 100% **Then** Prune (r)
4. **If** Optimistic_Score (r) ≤ Score (BestRule) **Then** Prune (r)

The experimental results of many studies [22, 23] have indicated that the choice of a discretisation method depends on both the data to be discretised and the learning algorithm. The performance of the four discretisation methods mentioned above when used with the RULES-6 algorithm was evaluated empirically [17]. 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 discretised using the entropy method. As a result, this discretisation method is adopted for use with RULES-6.

4. EMPIRICAL EVALUATION OF RULES-6

This section presents an empirical evaluation to assess the performance of the RULES-6 algorithm. RULES-6 was compared to its immediate predecessor RULES-3 Plus and to the well-known inductive learner C5.0 [24] 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. All execution times were obtained on a Pentium IV computer with a 2.4 GHz processor, 512 MB of memory and the Windows NT 4.0 operating system.

In order to draw reliable conclusions about the behaviour of the learning algorithms, 40 data sets were considered. All data sets were obtained from the University of California at Irvine (UCI) repository of machine learning databases [25].

In the experiments conducted in this study, the hold-out approach was used to partition the data into training and test data [26]. 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 above procedure was repeated ten times, and the results were averaged.

RULES-6 and C5.0 each has a number of parameters whose values determine the quality of their induced rule sets. RULES-6 was tested using values of 4, 2 and 1 for beam width, *MinPositives* and *MinNegatives* respectively. C5.0 was run with the default settings.

4.1. Comparison with RULES-3 Plus

This section describes an empirical study to compare RULES-6 against RULES-3 Plus. Table 2 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-organisation, 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 4 data sets, equivalent results were obtained. It can also be seen from the table that RULES-6 produced much more compact rule sets than

Table 2. Results for RULES-3 Plus and RULES-6

	RULES-3 Plus					RULES-6				
Data Set Name	Acc. (%)	# Rules	# Conditions	# Rules explored	CPU Time (s)	Acc. (%)	# Rules	# Conditions	# Rules explored	CPU Time (s)
Abalone	18.5	313	1947	26853	28	**25.3**	**21**	**49**	**1012**	**1**
Adult	77.5	6686	70144	1986685	29938	**83.1**	**118**	**395**	**16193**	**415**
Anneal	**99.7**	37	119	10119	3	93.3	**16**	**45**	**1912**	**1**
Australian	83.9	148	807	26301	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	176109	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	30277	4	79.5	**34**	**119**	**3624**	**1**
Diabetes	66.8	190	739	12399	2	**71.5**	**12**	**25**	**305**	**0**
German	70.9	247	1043	57120	13	**75.7**	**77**	**289**	**8402**	**3**
German-organisation	66.4	252	1381	90770	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	12526	1	**80.9**	**31**	**105**	**3902**	1
Hypothyroid	94.9	138	1743	88221	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	13709	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	798350	1943	**82.8**	**196**	**666**	**42926**	**155**
Segment	**90.5**	172	1198	51880	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	154065	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	46253	13	82.0	**29**	**82**	**3953**	**1**
Splice	91.8	239	1127	209203	340	**92.7**	**135**	**474**	**66354**	**118**
Tic-tac-toe	94.7	89	374	7970	1	**97.8**	**29**	**101**	**1757**	**0**
Tokyo	**91.3**	83	478	48551	27	89.4	**19**	**49**	**3673**	**2**
Vehicle	59.6	214	875	42013	7	**68.1**	**31**	**125**	**4032**	**1**
Vote	**97.0**	33	134	4999	0	95.6	**10**	**22**	**464**	0
Total	3271.0	11654	101787	3966753	33294	**3343.9**	**1342**	**4361**	**204652**	**805**

Table 3. Results for C5.0 and RULES-6

	C5.0		RULES-6	
Data Set Name	Accuracy (%)	No. of Rules	Accuracy (%)	No. 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-organisation	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**

RULES-3 Plus. The total number of rules decreased by 88.5% from 11654 to 1342. Also, the total number of conditions dropped by 95.7% from 101787 to 4361. The reduction in the number of rules and number of conditions for the Adult data set is 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 8. The total number of evaluations fell by 94.8% from 3966753 to 204652 and this was accompanied by a total 97.6% reduction in the execution time from 33294 seconds to 805 seconds. These results confirm that RULES-6 is more robust to noise and more accurate than RULES-3 Plus.

4.2. Comparison with C5.0

C5.0 has a facility to generate a set of pruned production rules from a decision tree. Table 3 presents the performance results of RULES-6 and C5.0. In each case, the accuracy on 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 one was assigned to the default rule.

It is clear from Table 3 that the accuracy obtained by RULES-6 was in total higher than that of C5.0. In addition, RULES-6 achieved the higher accuracy for 19 out of 40 data sets, while C5.0 achieved better accuracy for 17 out of 40 data sets. Both algorithms achieved similar accuracies for the remaining 4 data sets. It is also clear from the table that in total RULES-6 created fewer rules than C5.0. However, with RULES-6, the number of rules was lower for 10 data sets and higher in 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 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 specialisation 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 specialisations 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 RULES-6. Additional rule-space pruning strategies could be considered to increase the speed of the learning algorithm further. Post-pruning techniques could also be used to reduce the error and complexity of the learned

rule set in a post-processing phase. Finally, a method for discretisation of continuous-valued attributes during the learning process could be considered. Incorporating discretisation 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.

6. ACKNOWLEDGEMENTS

An extended version of this paper will be published in the Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science.

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: Phases 1 and 2) and within the project "Innovative Production Machines and Systems" (I*PROMS).

7. REFERENCES

[1] Braha, D. Data Mining for Design and Manufacturing: Methods and Applications. Kluwer Academic Publishers, Boston, 2001.

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

[3] Pham, D.T. and Afify, A.A. Machine learning techniques and their applications in manufacturing. Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture, 2005, 219 (5), 395-412.

[4] Pham, D.T. and Afify, A.A. Machine learning: Techniques and trends. Proceedings of the 9th International Workshop on Systems, Signals and Image Processing (IWSSIP-02), Manchester, UK, 2002, 12-36.

[5] Pham, D.T. and Aksoy, M.S. An algorithm for automatic rule induction. Artificial Intelligence in Engineering, Elsevier Science Limited, 1993, 227-282.

[6] Pham, D.T. and Aksoy, M.S. RULES: A simple rule extraction system. Expert Systems with Applications, 1995, 8 (1), 59-65.

[7] Pham, D.T. and Aksoy, M.S. A new algorithm for inductive learning. Journal of Systems Engineering, 1995, 5, 115-122.

[8] Pham, D.T. and Dimov, S.S. An efficient algorithm for automatic knowledge acquisition. Pattern Recognition, 1997, 30 (7), 1137-1143.

[9] Lee, C. Generating classification rules from databases. Proceedings of the 9th Conference on Application of Artificial Intelligence in Engineering, PA, USA, 1994, 205-212.

[10] Pham, D.T. and Dimov, S.S. An algorithm for incremental inductive learning. Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture, 1997, 211, 239-249.

[11] Pham, D.T., Bigot, S. and Dimov, S.S. RULES-5: A rule induction algorithm for problems involving continuous attributes. Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science, 2003, 217 (12), 1273-1286.

[12] Pham, D.T. and Dimov, S.S. An approach to concurrent engineering. Proceedings of the Institution of Mechanical Engineers Part B: Journal of Engineering Manufacture, 1998, 212, 13-27.

[13] Jennings, N.R. Automated Visual Inspection of Engine Valve Stem Seals. Internal Report, University of Wales Cardiff, Cardiff, UK, 1996.

[14] Wong, A.K.C. and Chiu, D.K.Y. Synthesizing statistical knowledge from incomplete mixed-mode data. IEEE Transactions on Pattern Analysis and Machine Intelligence, 1987, 9 (6), 796-805.

[15] Fürnkranz, J. and Flach, P.A. An analysis of rule evaluation metrics. Proceedings of the 20th International Conference on Machine Learning, Washington, DC, USA, AAAI Press, 2003, 202-209.

[16] Pham, D.T. and Afify, A.A. SRI: A scalable rule induction algorithm. Submitted to Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science, 2004.

[17] Afify, A.A. Design and Analysis of Scalable Rule Induction Systems. Ph.D. thesis, University of Wales Cardiff, School of Engineering, Systems Engineering Division, Cardiff, UK, 2004.

[18] Mitchell, T.M. Machine Learning. McGraw Hill, New York, 1997.

[19] Liu, H., Hussain, F., Tan, C. L. and Dash, M. Discretization: An enabling technique. Data Mining and Knowledge Discovery, 2002, 6, 393-423.

[20] Holte, R.C. Very simple classification rules perform well on most commonly used data sets. Machine Learning, 1993, 11, 63-90.

[21] Fayyad, U.M. and Irani, K.B. Multi-interval discretization of continuous-valued attributes for classification. Proceedings of the 13th International Joint Conference on Artificial Intelligence, Chambery, France, 1993, 1022-1027.

[22] Cai, Z. Technical Aspects of Data Mining. Ph.D. thesis, University of Wales Cardiff, Cardiff, UK, 2001.

[23] Pham, D.T. and Afify, A.A. On-line discretisation of continuous-valued attributes in rule induction. Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science, 2005 (In press).

[24] 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.

[25] 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.

[26] Efron B. and Tibshirani R. An Introduction to the Bootstrap. Chapman & Hall, USA, 1993.