RULES-5: a rule induction algorithm for classification problems involving continuous attributes

D T Pham[*], **S Bigot** and **S S Dimov**
Intelligent Systems Laboratory, Manufacturing Engineering Centre, School of Engineering, Cardiff University, Cardiff, Wales, UK

Abstract: This paper presents RULES-5, a new induction algorithm for effectively handling problems involving continuous attributes. RULES-5 is a 'covering' algorithm that extracts IF–THEN rules from examples presented to it. The paper first reviews existing methods of rule extraction and dealing with continuous attributes. It then describes the techniques adopted for RULES-5 and gives a step-by-step example to illustrate their operation. The paper finally gives the results of applying RULES-5 and other algorithms to benchmark problems. These clearly show that RULES-5 generates rule sets that are more accurate than those produced by its immediate predecessor RULES-3 Plus and by a well-known commercially available divide-and-conquer machine learning algorithm.

Keywords: machine learning, rule induction, covering algorithm, continuous attributes

NOTATIONS

A^i	ith attribute in an example
c	number of values appearing in T for a continuous attribute
C_E	class value in example E
CE	example closest to SE not belonging to the target class
$Cond_R^i$	condition in rule R for the ith attribute
d	number of possible values for a discrete attribute
m	number of attributes in an example
n	number of negative examples covered by the newly formed conjunction
N	total number of negative examples (examples not belonging to the target class)
p	number of positive examples covered by the newly formed conjunction
P	total number of positive examples (examples belonging to the target class)
$PRSET$	partial rules set
$PRSET\_size$	maximum number of rules in $PRSET$
SE	seed example
t^i	cutting point in a decision tree test for the ith continuous attribute
T	set of training examples
TC_R	target class value in rule R
T_PRSET	temporary partial rule set
V_E^i	value of the ith attribute in example E
$V_{\max}^i$	maximum known value of the ith continuous attribute
$V_{\min}^i$	minimum known value of the ith continuous attribute
V_R^i	discrete value employed in rule R to form a condition on the ith discrete attribute
Vc_k^i	kth continuous value in T of the ith attribute
Vd_k^i	kth discrete value of the ith attribute
$Vmax_R^i$	lower bound employed in rule R to form a condition on the ith continuous attribute
$Vmin_R^i$	lower bound employed in rule R to form a condition on the ith continuous attribute

1 INTRODUCTION

Knowledge-based systems have many applications in mechanical and manufacturing engineering {**1**, **2**}. The development of knowledge-based systems is often hampered by what is termed the knowledge-acquisition bottleneck—the difficulty in obtaining the knowledge required for these systems to operate. Machine learning

The MS was received on 11 April 2003 and was accepted after revision for publication on 4 August 2003.
[*] *Corresponding author: Intelligent Systems Laboratory, Manufacturing Engineering Centre, School of Engineering, Cardiff University, PO Box 925, Cardiff CF24 0YF, Wales, UK.*

has been suggested as a way of clearing the bottleneck by enabling knowledge to be extracted automatically from examples. With reference to automatic knowledge acquisition, Mitchell has introduced the ideas of 'concept learning' and 'general-to-specific ordering' [3]. 'Concept learning' can be viewed as performing a search over a space of possible hypotheses ordered from the most general to the most specific. The majority of inductive learning methods are based on this idea and have been categorized by Quinlan [4] into divide-and-conquer and covering methods. The main difference between these techniques is that they use different types of knowledge representation, namely decision trees and rule sets, and adopt significantly different types of search.

Divide-and-conquer methods construct sets of hypotheses in the form of decision trees. Because of the popularity of this representation technique, many divide-and-conquer algorithms have been developed. Perhaps the best-known divide-and-conquer algorithm is ID3 [5]. Since its creation, ID3 has been improved several times by a number of researchers. The most recent versions of this algorithm are C4.5 [6] and C5 [7], the latter being integrated into a commercially available software package.

Like the ID3 family of inductive learning algorithms, all divide-and-conquer algorithms employ the same general procedure that was first introduced by Hunt *et al.* [8] for constructing decision trees. There are two possible outcomes when this procedure is applied to a set of training examples T:

1. If T satisfies a particular stopping criterion, the decision tree for T is a leaf labelled with the most frequent class in the set.
2. If the stopping criterion is not satisfied, an attribute is selected, using a specific heuristic measure, to partition T into subsets of examples. The procedure is repeated on these new subsets until all the resultant subsets satisfy the stopping criterion.

Originally all algorithms used the event 'the subset contains examples belonging to a single class' as the *stopping criterion*, but in more recent algorithms this criterion has been refined in order to resolve the problem of over-fitting [6, 7]. Divide-and-conquer algorithms now differ in their choice of a stopping criterion and in the technique employed to select the attribute tests that will partition the training set and therefore guide the search through the hypothesis space.

Unlike decision-tree-based methods, covering methods represent classification knowledge in the form of a set of rules to describe each class. A number of covering algorithms are available, e.g. CN2 [9, 10], RIPPER [11] and AQ [12, 13] and its most recent version AQ19 [14]. All covering algorithms extract rules from a training set of examples employing the same general method as used for the first time in the AQ algorithm. This method adopts the following search process to form the rules for each class in the training set T:

While the *Stopping Criterion* is not satisfied:

1. Construct a new rule to cover examples belonging to a target class employing the *Rule Forming Process*.
2. Add this rule to the rule set.
3. Remove all examples from T covered by this new rule.

This search process continues until rules for all classes are formed.

This process is used in the majority of covering algorithms; however, many variations can be found. For instance, in the RULES family on which this work was based [15, 16] a simple technique is employed that does not need the training examples to be separated class by class, while in RIPPER [11], when the final class to be considered is reached, the process stops and this class is fixed as the default.

In the last decade, much research has been carried out to address different aspects of inductive learning methods. This paper discusses two important issues associated with covering algorithms, in particular search strategies for concept formation and continuous attribute handling. The paper starts with a review of existing techniques addressing these issues and then proposes ways to improve the rule forming and continuous attribute handling procedures in the RULES family. The paper concludes with a description of RULES-5, a new member of the family that incorporates the proposed improvements. Compared to its predecessors, RULES-5 employs a more efficient rule forming technique and a more advanced method of dealing with continuous attributes.

2 TECHNIQUES FOR RULE SEARCHING AND CONTINUOUS ATTRIBUTES HANDLING

2.1 Search strategies for concept formation

The most important part of the search strategies implemented in covering algorithms is the *Rule Forming Process*. The aim of this process is to create the 'best' rule. The notion of 'best' depends on the heuristic measure used. For instance, it could relate to the consistency and/or coverage of a rule. To form a rule, the *Rule Forming Process* searches for the best conjunction of conditions.

This process is the key element in all covering algorithms. From a computational viewpoint, it is very costly to consider all possible conjunctions of conditions. Therefore, covering methods employ different techniques within their specialization (rule forming) process to reduce the search space without sacrificing the quality of the resultant rule sets. These different search methods are based on the general-to-specific ordering approach [3] and characterize a covering algorithm.

The specialization process is composed of three main elements: the selection of conditions, the search heuristics and the search method:

1. *Selection of conditions*. In order to select which conditions can be used to form a conjunction of conditions, a number of methods exist. Some algorithms consider all the conditions available using all attribute–value pairs (see, for instance, CN2 [9, 10] or RIPPER [11]). Some others, such as AQ [12, 13] or the RULES family [15, 16], use the seed example (SE) method, where conditions are formed based only on the attribute–value pairs of the selected seed example.
2. *Search heuristics*. The search heuristics used in the specialization process are generally statistical measures that evaluate the quality of a particular refinement obtained by adding a condition to an initial conjunction of conditions. Two types of heuristic can be found in the specialization process. The first type is used to select the best conjunctions of conditions to be recorded for further refinement; this is called the specialization heuristic. For instance, the RULES family employs a parameter called the H measure [15]. The second type is used in order to evaluate the conjunction of conditions formed so far and to decide if the specialization should be stopped; this is called the stopping heuristic. In many covering algorithms, for instance in the RULES family, the specialization stops when the conjunction of conditions covers examples belonging to a single class. Pruning methods [17] are then used to resolve the problem of over-fitting and noise handling.
3. *Search method*. The search method defines the technique used to search through the hypothesis space. The simplest method is called 'hill climbing'. This method adds conditions incrementally to a single conjunction until the stopping heuristic reaches a particular value. This method is employed in RIPPER [11]. The main disadvantage of this search method is that the conditions for further specialization of the conjunctions are selected based only on the values of the statistical measure used as the specialization heuristic. The quality of the resultant rules is therefore highly dependent on the performance of this measure, which can vary with different data sets. Hence, during the rule forming process, the search might not always create optimum rules.

   The so-called 'beam search' method addresses this problem by considering not only the best conjunction formed so far for further specialization but also a fixed number of alternative conjunctions. In this way, a wider hypothesis space is explored, which ultimately leads to better results than for the 'hill-climbing' method. However, the results are still very much dependent on the statistical measure adopted. This method has been applied successfully in different algorithms, for instance RULES-3 Plus [15], AQ [12, 13] and CN2 [9, 10].

   A variation on the beam search method is known as 'best first search'. Here, instead of storing a fixed number of conjunctions, all possible candidate conjunctions not removed by a particular quality criterion are stored. This method can be computationally very costly.

   Another strategy for concept formation is a stochastic search, which allows the algorithm to specialize a conjunction of conditions not only by appending one condition at a time but also by randomly selecting more than one condition for further specialization.

The choice of these three elements (condition selection method, search heuristic and search technique) is critical to the performance of an algorithm. More information on them can be found in references [18] and [19].

2.2 Continuous attributes handling

It is important for machine learning algorithms to be able to deal with continuous attributes efficiently because real problems all involve such attributes. Two types of continuous attribute handling method are generally used: pre-processing discretization and on-line methods.

2.2.1 *Pre-processing discretization methods*

Originally, most algorithms (for instance the first versions of ID3 and AQ) were able to deal only with discrete attributes. Therefore, a number of pre-processing discretization techniques that transform continuous into discrete attributes have been developed. These techniques split the ranges of each continuous attribute into a fixed number of smaller intervals that are then regarded as discrete values. Unfortunately, this simplification can seriously affect the quality of the resultant rule sets. This is the case in RULES-3 Plus [15], where the user defines quantization levels to divide each attribute range into equal intervals. The number of these quantization levels depends on the training data and is difficult to specify without experimentation. For instance, if the user selects a small number of quantization levels, some of the rules might not be consistent due to the large intervals involved. On the contrary, if the user specifies a large number of intervals, the number of rules would increase considerably and the resultant rule set would be too specific (overspecialized).

Several attempts have been made at resolving this problem. Some algorithms use pre-processing discretization techniques where the number of intervals and their length are defined by the algorithm itself. With these techniques [20], the discretization is initialized by putting each training example into its own interval. A statistical measure is then used to decide whether or not two adjacent intervals should be merged.

The problem with pre-processed discretization is that continuous attribute intervals are fixed before the rule forming process. However, there is additional information available during rule formation. In addition, different rules might require different intervals. Thus, ideally the intervals should not be fixed beforehand but rather created 'on-line', in parallel with rule formation.

2.2.2 *On-line methods*

On-line methods for handling continuous attributes were developed originally in relation to divide-and-conquer techniques. They are generally similar in all algorithms.

The problem associated with continuous attributes occurs when selecting a test to split a set of examples T. If the attribute chosen for the partitioning is discrete, such tests are simply carried out using the d values of the discrete attribute to obtain a subtree with d branches, e.g. the tests $[A^i = Vd_1^i], \ldots, [A^i = Vd_k^i], \ldots, [A^i = Vd_d^i]$, where Vd_k^i is the kth possible discrete value of the ith attribute A^i. However, if the attribute is continuous, it has an infinite number of possible values; therefore the test takes the form $[A^i < t^i]$, with the outcomes true or false. The main difficulty is in selecting the best threshold t^i (cutting point). Divide-and-conquer algorithms adopt the threshold that maximizes a particular heuristic called the splitting criterion. For example, in C4.5, t^i is chosen among the following list:

$$\{t_1^i, \ldots, t_k^i = (Vc_k^i + Vc_{k+1}^i)/2, \ldots, t_c^i\}$$

where $\{Vc_1^i, \ldots, Vc_k^i, \ldots, Vc_c^i\}$ is a list containing the c possible continuous values of the ith attribute, appearing in T, stored in increasing order and k is an integer belonging to $[1, c]$.

The identification of the correct splitting criterion for the evaluation of potential cutting points has been the focus of a number of researchers. For instance, the C4.5 splitting criterion was originally based on a measure called the gain ratio [6]. Later, Dougherty examined different discretization methods [21] and found that the method giving the best results was entropy discretization, a method first introduced by Catlett [22] and then improved by Fayyad and Irani [23, 24]. Dougherty also noted that the full potential of these techniques was not realized. To address this issue in C4.5, Quinlan proposed a new technique [25], inspired by the minimum description length principle [26].

The main advantage of these techniques, in comparison to pre-processing discretization, is that the condition ranges are created in parallel with the rule forming process. However, most of the methods require the evaluation of a high number of potential cutting points, which could result in high computational costs.

2.3 Disadvantages of existing methods

Thus, it can be seen that, in inductive learning, statistical and probabilistic measures are commonly used as criteria for the creation and selection of the best conditions to form rules. These measures provide useful but limited information about the quality of conditions. They are employed for arbitrary decisions, but their effect on the learning process is not fully understood and their performance often varies depending on the application domain. In addition, data sets also contain neglected information that could be used to guide the learning process before statistical measures are applied as a last resort. Therefore, the new methods developed in this research have focused on the use of such information in order to reduce the dependence of the rule forming process on such arbitrary measures.

3 ALGORITHM DESCRIPTION

3.1 Preliminaries

This section presents RULES-5, a new covering algorithm that has overcome some of the deficiencies of the RULES-3 Plus algorithm [15]. In particular, it employs a new method for handling continuous attributes, and simple and more efficient techniques for extracting IF–THEN rules from examples.

Data are presented to RULES-5 in the form of a collection of objects, each belonging to one of a number of given classes. These objects together with their associated classes constitute a set of training examples (T) from which the algorithm induces a model. Each example E is described by its class value C_E and by a vector of m attributes $(A^1, \ldots, A^i, \ldots, A^m)$. Each attribute value V_E^i is either discrete or continuous. In the case of a continuous attribute, $[V_{\min}^i \leqslant V_E^i \leqslant V_{\max}^i]$, where $V_{\min}^i$ is the minimum known value for the ith attribute and $V_{\max}^i$ its maximum known value. An example E is therefore formally defined as follows:

$$E = (A^1 = V_E^1, \ldots, A^i = V_E^i, \ldots, A^m = V_E^m, \\ \text{class} = C_E)$$

Like its predecessors, RULES-5 forms a new rule by starting from an example not covered by previously created rules, the seed example (SE). The algorithm employs a specialization process that searches for consistent rules that are as general as possible. The result is a rule set that correctly classifies all or most of the training examples.

A rule set is a list of IF–THEN rules. Each rule R is described by a conjunction of conditions on each attribute $(Cond_R^i)$ and by a target class value (TC_R). A rule R can be formally defined as

$$Cond_R^1 \wedge \cdots \wedge Cond_R^i \wedge \cdots \wedge Cond_R^m \rightarrow TC_R$$

If $Cond_R^i$ exists, it could be an attribute–value pair $[A^i = V_R^i]$ or a range of values $[Vmin_R^i \leqslant A^i \leqslant Vmax_R^i]$ for discrete and continuous attributes respectively, where V_R^i is a discrete value and $Vmin_R^i$ and $Vmax_R^i$ are continuous values included in the ith continuous attribute range $[V_{\min}^i, V_{\max}^i]$.

The new method for continuous attribute handling does not require the data to be pre-processed. It does not need any particular user input. Automatically, the proposed method defines the intervals for each rule in the attribute space by analysing the example distribution. In addition, RULES-5 uses a new technique based on the beam search approach that selects conditions for concept formation also by first analysing the distribution of examples in the attribute space. Only then is a statistical measure employed to select the best condition. Consequently, the dependence of the concept formation process on the statistical measure is reduced and less variability is achieved in the performance of the algorithm with respect to different data sets.

The following sections present the key ideas underlying the algorithm.

3.2 Condition selection and continuous attribute handling

As with other covering methods, RULES-5 searches for rules that cover as many examples as possible from the target class and at the same time exclude examples belonging to other classes. The specialization process used in the RULES-3 Plus algorithm considers all conditions extractable from SE, starting with the condition with the highest information content to form a rule. In RULES-5, the rule formation procedure only takes account of conditions excluding the closest example (CE) not belonging to the target class and covered by the rule formed so far. The assumption is that this also leads to the exclusion of the maximum number of other examples not belonging to the target class.

To find CE, a measure is used to assess the distance between any two examples. Because the data set could contain continuous and discrete attributes, this measure should be able to handle both types at the same time. In RULES-5, the distance measure between an example $E1$ and an example $E2$ is defined as follows:

$$Distance\_E1-E2 = \sqrt{\sum_c \left(\frac{V_{E1}^i - V_{E2}^i}{V_{\max}^i - V_{\min}^i}\right)^2 + \sum_d d\_distance} \qquad (1)$$

where $\sum_c$ is the sum for continuous attributes, $\sum_d$ is the sum for discrete attributes, V_{E1}^i is the value of the ith attribute in example $E1$, V_{E2}^i is the value of the ith attribute in example $E2$, $V_{\max}^i$ is the maximum known value of the ith continuous attribute, $V_{\min}^i$ is the minimum known value of the ith continuous attribute and $d\_distance$ is defined for each discrete attribute by applying the following rule:

If $V_{E1}^i = V_{E2}^i$ Then $d\_distance = 0$
Else $d\_distance = 1$

Applying this distance measure, CE can be found at each step of the specialization and the rule forming procedure considers appending to the rule only conditions that exclude CE. This leads to a reduction of the search space because not all conditions are examined. In case more than one condition excludes the closest example, a statistical measure is used to identify the best ones.

Thus, the algorithm takes CE and creates candidate conditions to exclude it. These conditions are formed using attributes having different values for SE and CE $(V_{SE}^i \neq V_{CE}^i)$. In particular:

1. In the case of a discrete attribute, the created condition will have the following format:

   $[A^i = V_{SE}^i]$

2. In the case of a continuous attribute, the format of the formed condition will be

   $[V_{CE}^i < A^i]$ if $V_{CE}^i < V_{SE}^i$

   and

   $[A^i < V_{CE}^i]$ if $V_{CE}^i > V_{SE}^i$

In the case $Distance\_SE\text{-}CE$ is null, which means that SE and CE are identical but belong to different classes, the algorithm cannot create any condition that includes SE and excludes CE. Therefore, the current CE is ignored during the rule formation process and another CE is selected.

By following this procedure, the algorithm can easily handle continuous attributes and there is no need to pre-process the data in order to discretize them. The algorithm identifies splitting points for each continuous attribute range during the learning process, without using any particular statistical measure.

3.3 Rule forming process

A simple example will be used to illustrate the rule forming process. Table 1 shows a sample training set used for the development of a process planning expert system [16]. SE is the first example in Table 1:

$$\begin{aligned} SE = (&\text{Heat Treatment} = \text{Yes}, \\ &\text{Material} = \text{Steel\_3135},\ \text{Tolerance} = 10, \\ &\text{Finish} = \text{Medium},\ \text{Route} = \text{R2}) \end{aligned}$$

The search process starts with the default most general rule '→ Route = R2'. This rule is not consistent. Therefore it is specialized to cover only examples from the target class, R2. As explained in the previous section, a search is carried out to find the closest example (CE) not belonging to the target class that, at the same time, is covered by the rule formed so far. The closest example is

$$\begin{aligned} CE = (&\text{Heat Treatment} = \text{Yes}, \\ &\text{Material} = \text{Steel\_1045},\ \text{Tolerance} = 9, \\ &\text{Finish} = \text{Medium},\ \text{Route} = \text{R1}) \end{aligned}$$

The candidate conditions for specialization of the default rule are those that exclude this example. The following two conditions are considered:

$$[\text{Material} = \text{Steel\_3135}] \quad \text{and} \quad [\text{Tolerance} > 9]$$

Using these two conditions, two rules can be formed:

IF [Material = Steel_3135] THEN Route = R2
IF [Tolerance > 9] THEN Route = R2

If any of the formed rules are consistent, they are taken as candidate rules and the search process stops. Otherwise, if the formed rules pertain to more than one class, they are added to a set called the partial rules set ($PRSET$) as in the case of RULES-3 Plus. The maximum number of rules in $PRSET$ is specified by the user and determines how many alternatives are considered in each pass. In other words, this number is the width of the beam search implemented in RULES-3 Plus. These rules are specialized further by appending new conditions to them.

As with RULES-3 Plus, RULES-5 uses the H measure to assess the information content of each newly formed rule (specialization heuristic). This is done for continuity although there is potential for further improvements in performance by employing other measures. The H measure is composed by two specific terms. The first one represents the generality of the rule and can be written as [27]

$$G = \sqrt{\frac{p+n}{P+N}} \tag{2}$$

The second term represents the accuracy of the rule and can be expressed as [27]

$$A = 2 - 2\sqrt{\frac{p}{p+n}\frac{P}{P+N}} - 2\sqrt{\left(1 - \frac{p}{p+n}\right)\left(1 - \frac{P}{P+N}\right)} \tag{3}$$

Thus, for any particular rule, the H measure is defined as

$$H = \sqrt{\frac{p+n}{P+N}}\left[2 - 2\sqrt{\frac{p}{p+n}\frac{P}{P+N}} - 2\sqrt{\left(1 - \frac{p}{p+n}\right)\left(1 - \frac{P}{P+N}\right)}\right] \tag{4}$$

where P is the total number of positive examples (examples belonging to the target class), N is the total number of negative examples (examples not belonging to the target class), p is the number of positive examples covered by the newly formed rule and n is the number of negative examples covered by the newly formed rule.

During the rule forming process, the rules in $PRSET$ are ordered according to their H measure. If the H measure of a newly formed rule is higher than the H measure of any rule in $PRSET$, the new rule replaces the rule having the lowest H measure.

The specialization process could lead to the following three outcomes:

1. *No candidate rule*. All rules in $PRSET$ are specialized further by repeating the same process.
2. *Only one candidate rule*. The rule is added to the rule set and the search stops.
3. *More than one candidate rule*. The rule covering the highest number of examples not already covered by rules formed so far is added to the rule set and the search stops.

The search process continues until all examples in the training set are covered.

Table 1 Training set [16]

Heat treatment	Material	Tolerance	Finish	Route
Yes	Steel_3135	10	Medium	R2
No	Aluminium	12	Medium	R3
Yes	Steel_3135	8	Medium	R2
No	Steel_1045	14	Low	R3
No	Aluminium	7	High	R4
Yes	Steel_1045	9	Medium	R1
No	Aluminium	9	Medium	R3
No	Aluminium	10	High	R4
No	Aluminium	10	Low	R3
Yes	Steel_1045	7	Medium	R1
No	Steel_1045	7	Low	R3

3.4 Rule post-processing

During the rule formation process, the rules are created independently of one another and can contain conditions for different attributes. This leads to the presence of 'overlapping' rules that require further processing. Consider the following two rules:

If Finish = Low Then Route = R3

If Material = Steel_3135 Then Route = R2

These rules overlap because they cover areas in the attribute space that are common for both rules. For instance, the unclassified examples below are covered by both rules:

(Heat Treatment = Yes, Material = Steel_3135, Tolerance = 9, Finish = Low, Route = ?)

(Heat Treatment = No, Material = Steel_3135, Tolerance = 7, Finish = Low, Route = ?)

Rule overlapping is a feature of all covering methods. In fact, allowing the presence of overlapping permits the creation of simpler and more general rules. A real problem occurs when unseen examples are covered simultaneously by rules pointing to different classes. To resolve such cases, appropriate classification techniques should be used to select the best classifying rule, as will be illustrated in the next section.

The new method for handling continuous attributes also contributes to the generation of overlapping rules. In contrast to classical discretization methods, this method tends to produce more general rules that cover areas in the attribute space not represented in the training data. In addition, different intervals are created for each continuous attribute condition during the rule formation process, which also increases the possibility of overlapping. To illustrate these potential problems, consider the set of training data shown in Fig. 1.

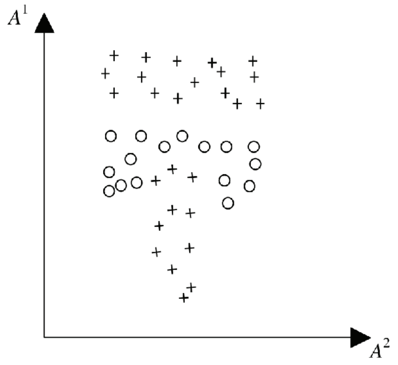

Fig. 1 Graphical representation of a training set with two continuous attributes

By applying the proposed discretization procedure to this data set, very general rules are created. The graphical representation of two of the generated rules shown in Fig. 2 demonstrates clearly the two problems mentioned above, namely the presence of overlapping areas between rules 1 and 2 and the coverage of an 'unknown' area (an area without examples in the training set) by rule 1. To avoid such excessively general rule sets, in RULES-5, after each iteration the rule coverage is limited to areas that are represented in the training set. This is achieved by reducing the coverage of some continuous attribute conditions to the training data only. For instance, by applying this procedure, the coverage of the rules in Fig. 2 would be limited to the training examples shown in Fig. 3. Only the 'known' areas of the attribute space are covered and the possibility of overlapping is reduced.

3.5 Illustrative problem

The new rule forming procedure of RULES-5 is summarized in Fig. 4. To illustrate how continuous values are handled in RULES-5, the training data shown in Fig. 1 is used. The rule forming procedure is

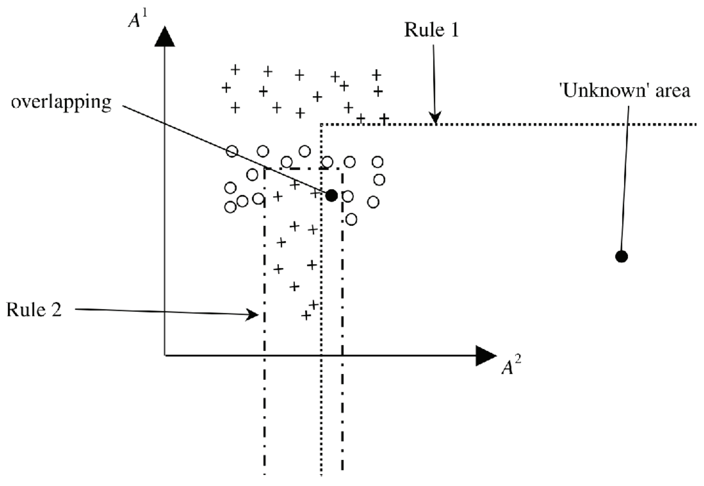

Fig. 2 Graphical representation of the rule set

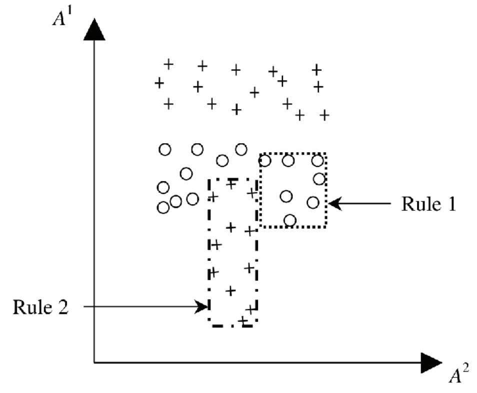

Fig. 3 The resultant reduction of the rule set coverage

explained with reference to the steps given in Fig. 4. For the given data set, the maximum number of rules in *PRSET* is set to 1 (*PRSET_size* = 1).

The following explains how a rule is created for the first uncovered example. This example, *SE*, is shown in Fig. 5.

Step 1. The initial rule with no conditions is formed. *PRSET* and *best_rule* are initialized:

$$best\_rule = \textbf{IF } [\text{no condition}] \textbf{ THEN } \text{Class} + \text{ and } PRSET = \{\textbf{IF } [\text{no condition}] \textbf{ THEN } \text{Class} +\}$$

Step 2. *best_rule* is not consistent and therefore a temporary list of rules (*T_PRSET*) is created. The initial rule is taken from *PRSET* for further specialization:

$$rule\_to\_specialise = \textbf{IF } [\text{no condition}] \textbf{ THEN } \text{Class} +$$

Step 3. The first closest example *CE*1 is identified (Fig. 5).

Step 4. The first attribute values of *CE*1 and *SE* differ ($V^1_{CE1} > V^1_{SE}$) and therefore the following rule can be formed:

$$new\_rule = \textbf{IF } [A^1 < V^1_{CE1}] \textbf{ THEN } \text{Class } +, \quad \text{H} = 0.00086953$$

Step 5. The created new rule is again not consistent (Fig. 6). The new rule is stored in *T_PRSET*:

$$T\_PRSET = \{\textbf{IF } [A^1 < V^1_{CE1}] \textbf{ THEN } \text{Class } +\}, \quad \text{H} = 0.00086953$$

Then the procedure returns to step 4 to form a new condition for another attribute.

Step 4. The values of the second attribute of *CE*1 and *SE* are also different ($V^2_{CE1} < V^2_{SE}$). Hence, a new rule can be formed using a condition created for this attribute:

$$new\_rule = \textbf{IF } [V^2_{CE1} < A^2] \textbf{ THEN } \text{Class } +, \quad \text{H measure} = 0.00610018$$

Step 5. The newly formed rule is also not consistent (Fig. 7). In addition, the number of rules in *T_PRSET* is equal to the pre-set value of 1 (*PRSET_size*) and the H measure of *new_rule* is higher than the H measure of the rule already stored in *T_PRSET*. Therefore, *new_rule* replaces it:

$$T\_PRSET = \{\textbf{IF } [V^2_{CE1} < A^2] \textbf{ THEN } \text{Class } +\}$$

There are no more attributes to be considered nor remaining rules in *PRSET*.

Step 6. The rule in *T_PRSET* is copied into *PRSET* and the procedure returns to step 2.
Step 2. *T_PRSET* is initialized and the first rule in *PRSET* is taken for further specialization:

$$rule\_to\_specialise = \{\textbf{IF } [V^2_{CE1} < A^2] \textbf{ THEN } \text{Class} +\}$$

Step 3. The procedure identifies the new closest example *CE*2 (Fig. 8).
Steps 4 and 5. Two new rules are considered (Fig. 9). The rule with the highest H measure, Rule 1, is selected and stored in *T_PRSET*.
Step 6. The rule in *T_PRSET* is copied into *PRSET*. The procedure returns to step 2.

The rule forming process continues until the rule shown in Fig. 10 is generated. Then, by applying the rule set post-processing procedure, the coverage of the rule is limited to the training data. As a result, the rule in Fig. 11 is formed. The final result of applying RULES-5 to the training data shown in Fig. 1 is given in Fig. 12.

4 MISSING ATTRIBUTE VALUES

Missing attribute values can seriously affect the performance of inductive learning algorithms. Several methods have been developed to overcome this problem. For instance, the following techniques can be applied in RULES-5 when an example contains an unknown attribute value [**28**]:

1. Ignore the example.
2. Treat the example as though it has the most common value of the attribute.
3. Consider the unknown value as a separate value for the attribute.

RULES-5 Rule Forming Procedure

```
        Take one example uncovered by the rule set formed so far (SE)

STEP1
        Initialise PRSET (empty list)
        Form an initial rule with no conditions to classify SE
        Store this rule in PRSET and copy it into best_rule

STEP2
        WHILE best_rule is not consistent DO
            Initialise T_PRSET (empty list)
            FOR each rule in PRSET DO
                rule_to specialise = the rule taken from PRSET
STEP3
                CE = an example misclassified by rule_to specialise and the closest to SE
                IF all attribute values of CE and SE are identical THEN ignore the current CE and select a new CE
                FOR $i = 1$ to $i = \mathrm{m}$ DO
STEP4
                    IF $V_{SE}^{i} \neq V_{CE}^{i}$ THEN
                        new_rule = rule_to_specialise
                        IF continuous attribute THEN
                            IF $V_{SE}^{i} < V_{CE}^{i}$ THEN
                                Append the condition [$A^{i} < V_{CE}^{i}$] to new_rule
                            ELSE
                                Append the condition [$A^{i} > V_{CE}^{i}$] to new_rule
                        ELSE
                                Append the condition [$A^{i} = V_{SE}^{i}$] to new_rule
STEP5
                        IF new_rule is consistent
                        AND new_rule covers more uncovered examples than best_rule THEN
                                Replace best_rule with new_rule

                        IF new_rule is not consistent THEN
                                IF number of rules in T_PRSET < PRSET_size THEN
                                        Store new_rule into T_PRSET
                                ELSE
                                IF the new_rule H measure is higher than the H measure of any rule in T_PRSET THEN
                                        Replace the rule with the lowest H measure in T_PRSET with new_rule
                    END FOR
                END FOR
STEP6
            copy T_PRSET into PRSET
        END WHILE

        IF best_rule contains continuous attribute conditions THEN
            Constrain their coverage to training examples

        Add best_rule to the rule set
```

Where: *PRSET*: a list of rules to be specialised; *T_PRSET*: a temporary list of rules; *PRSET_size*: Maximum number of rules in *PRSET* and *T_PRSET* (defined by the user)

Fig. 4 RULES-5 rule forming procedure

These techniques are not used in RULES-5, although they could be applied in pre-processing data before it is presented to RULES-5. It is recognized that an example with a missing value does carry useful information and this information should be utilized by the algorithm. Therefore, the algorithm should be able to handle missing values and use the information contained in the affected examples while extracting rules. In particular, the following procedures are implemented in RULES-5:

1. *Compute the distance between two examples*. If an attribute value is missing in an example, the

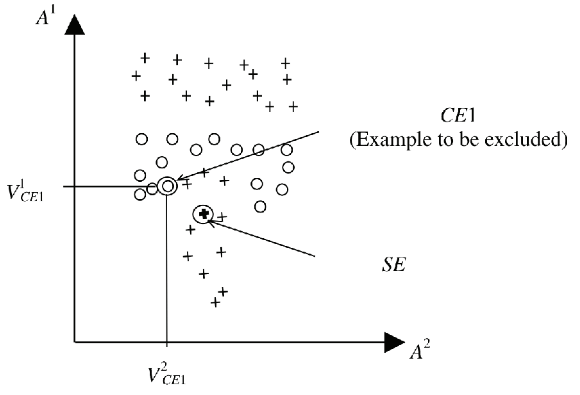

Fig. 5 Identification of the closest misclassified example

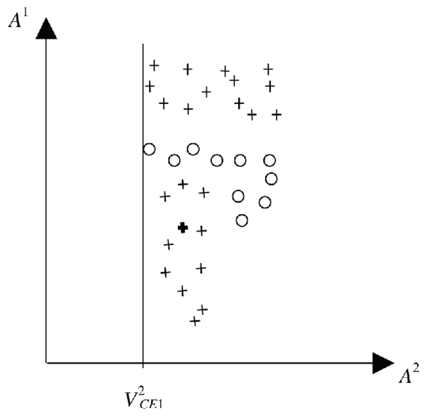

Fig. 7 Graphical representation of the coverage of the second rule

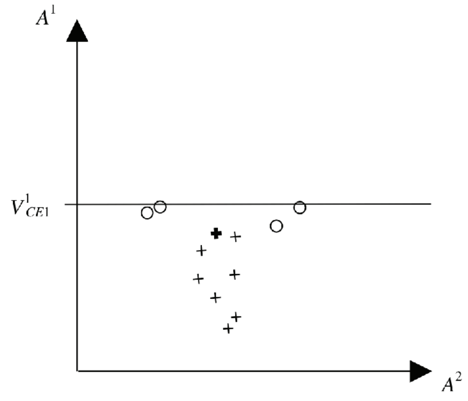

Fig. 6 Graphical representation of the coverage of the first rule

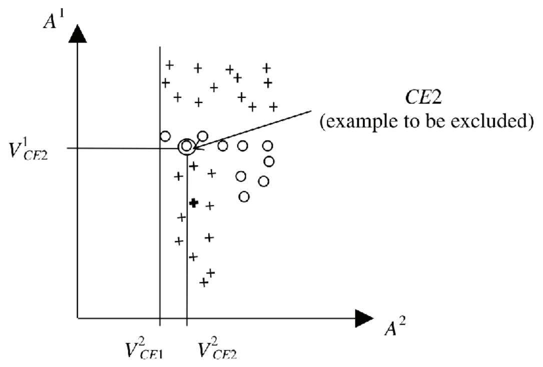

Fig. 8 The closest example misclassified by *rule_to_specialise*

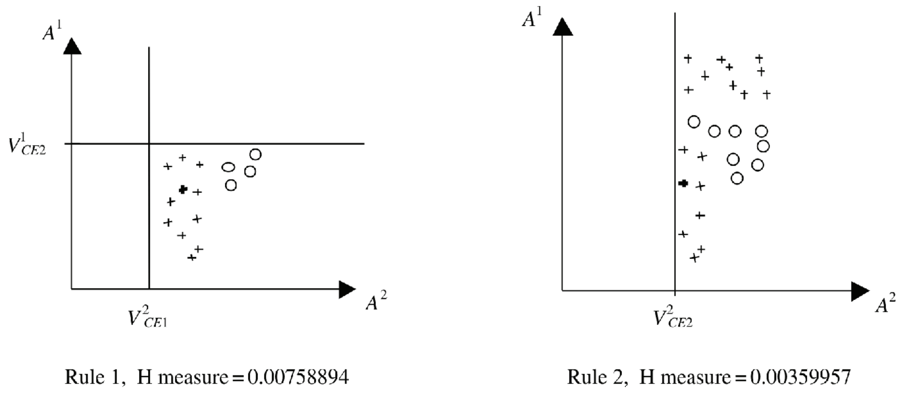
A^1
V^1_{CE2}
V^2_{CE1}
A^2
Rule 1, H measure = 0.00758894
A^1
V^2_{CE2}
A^2
Rule 2, H measure = 0.00359957

Fig. 9 Graphical representation of the coverage of the two new rules

corresponding term *d_distance* or $(V^i_{E1} - V^i_{E2})/(V^i_{\max} - V^i_{\min})$ in equation (1) is given a value of 1, which represents the maximum distance. In this way, consideration of examples with missing values in the specialization process will be delayed.

2. *Create a new condition to exclude CE*. If *SE* has missing attribute values, no conditions are created for these attributes. If *CE* has a missing attribute value, conditions can be created only if the attribute is discrete. This is because, to form a condition for a continuous attribute, both the attribute values extracted from *SE* and *CE* are necessary.
3. *Check if an example is covered by a rule*. If an example has missing attribute values for which conditions exist in a rule, the example is considered to be not covered by the rule.

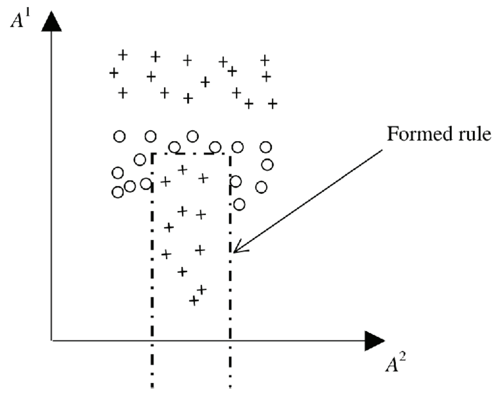

Fig. 10 The result of the rule forming procedure

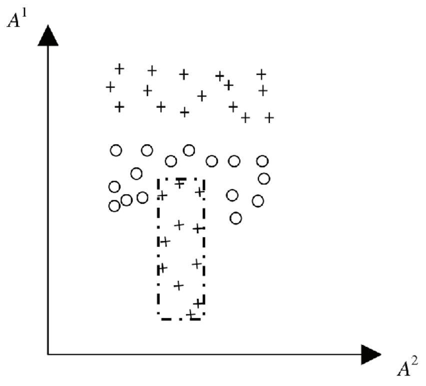

Fig. 11 Constraining the rule coverage to the training data only

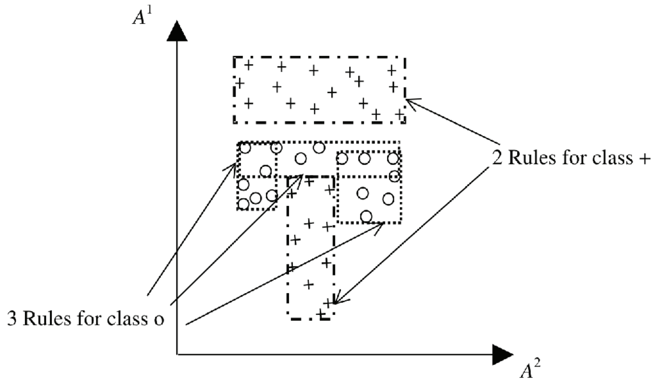

Fig. 12 Final rule set

The implementation of these procedures in RULES-5 allows the algorithm to handle missing values, but they do not exclude the use of the pre-processing techniques mentioned above.

5 RULES-5 CLASSIFICATION TECHNIQUE

As already seen, RULES-5 creates a set of rules whose coverage is limited to the training examples only. Therefore, when a set of rules is used as a classification model, it is possible that some new examples will not be covered by it. A common solution to this problem is to consider all the examples not classified by the rule set as belonging to a default class. This simple approach is not appropriate in cases where such examples carry useful information that could be used to improve the classification performance of the model. For instance, by assessing the position of an example relative to the areas covered by the rule set in the attribute space, the best rule to classify it can be identified.

Another specific characteristic of the rule sets generated by RULES-5 is that these sets consist of independent unordered rules. As a result, some examples can be covered by more than one rule. In such cases, it is necessary to assess the information content of all covering rules in order to select the best one to classify any particular example. This problem does not exist in the case of decision tree models because there is no overlapping. Neither does it in the case of an ordered set of rules, because the rules are classified automatically during the rule formation process and when classifying a new example the first covering rule in the rule set will be used because it is considered better than any following ones. To resolve this problem, specific to models in the form of unordered rule sets, RULES-5, in common with its predecessors in the RULES family, uses the H measure as the criterion to select the best rule (the one with the highest H measure) to predict the class of an unknown example.

The classification technique can be summarized as follows. There are three possible outcomes when using rule sets formed by RULES-5 to classify an example:

1. *Only one rule covers the example.* The example belongs to the class of the covering rule.
2. *More than one rule covers the example.* The rule with the highest H measure is used to classify the example. For instance, Rule 2 in Fig. 13 is selected to classify the example.
3. *No rules cover the example.* The rule 'closest' to the example in the attribute space is employed to classify it. To find the 'closest' rule, the distance between a rule R and an example E is defined as follows:

$$Distance\ R/E = \sqrt{\sum_c c\_distance + \sum_d d\_distance} \tag{5}$$

where $\sum_c$ is the sum for continuous attributes and *c_distance* is defined for each continuous attribute as follows:

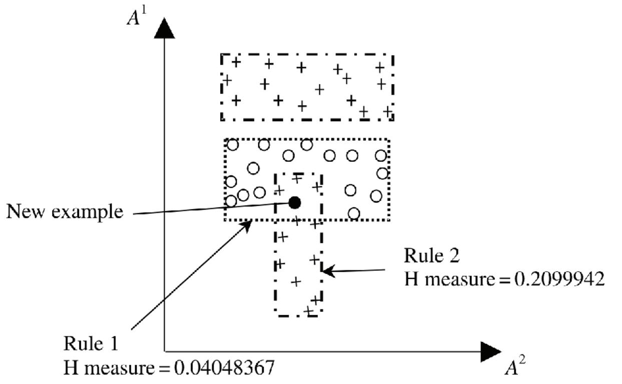

Fig. 13 Classification of an example when it is covered by more than one rule

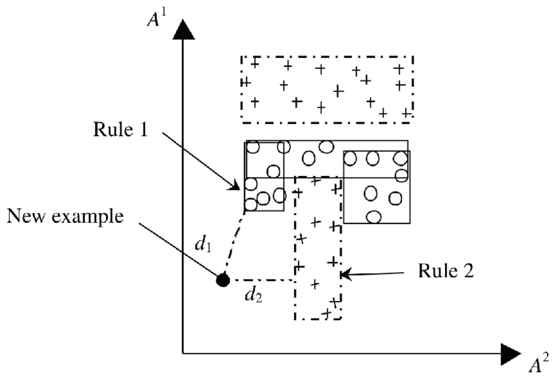

Fig. 14 Classification of an example not covered by the rule set ($d_1 = 13.7$ and $d_2 = 13$)

If the value is outside the condition range of the attribute

$$c\_\,distance = \left(\frac{\min(|V_E^i - Vmax_R^i|, |V_E^i - Vmin_R^i|)}{V_{\max}^i - V_{\min}^i}\right)^2$$

Else

$$c\_\,distance = 0$$

$\sum_d$ is the sum for discrete attributes and *d_distance* is defined for each discrete attribute by applying the following rule:

If $V_E^i = V_R^i$ Then $d\_\,distance = 0$
Else $d\_\,distance = 1$

For instance, after applying this distance measure to the rule set shown in Fig. 14, Rule 2 is employed to classify the example because $d_2 < d_1$.

6 TESTS AND ANALYSIS OF RESULTS

RULES-5 has been tested against RULES-3 Plus and C5 on 15 data sets, which were extracted from the University of California, Irvine (UCI) repository of machine learning [29]. These databases were contributed by many researchers, mostly from the field of machine learning. A simple method was used for dividing the data into training and test sets: approximately 70 per cent of each data set was adopted for the training set and the remaining 30 per cent were employed for testing. The splitting also made sure that the same proportion of each class was present in both sets. The test results are given in Table 2. It should be noted that default parameters were used for C5. In addition, process times have been recorded but only for RULES-3 Plus and RULES-5. This is because the same programmer developed these two algorithms, while C5 is a commercial algorithm written by a different programming team and therefore a fair comparison of process time with this algorithm would not be possible.

Compared to RULES-3 Plus, RULES-5 generates more compact rule sets (with on average 25.2 per cent fewer rules). At the same time, the classification accuracy of these rule sets is higher (by an average of 4.6 per cent). In addition to this improvement, the specialization method used in RULES-5 is more efficient than in RULES-3 Plus. For most of the data sets, the computing time is reduced by as much as half of that required in RULES-3 Plus. In addition, it should be noted that these improvements occur not only with data sets containing continuous attributes but also with data sets with discrete attributes or combinations of both types.

Compared to C5, the classification accuracy of the rule sets is on average much higher. However, the rule sets that RULES-5 generates contain more rules. This can be explained by the search mechanism employed in RULES-5, which produces rule sets to cover the training data completely. As a result, some of the rules can be 'overspecialized' and will thus affect the capability of the algorithm to handle noisy data. At the same time, it should be noted that the smaller and sometimes less accurate rule sets generated by C5 can be attributed to the pruning techniques employed. Through the use of such techniques, the number of rules can be reduced by making some of them more general and even inconsistent. As a result, rule sets could be formed that are less accurate but at the same time more robust to noise.

However, an example that demonstrates the problems associated with the use of pruning techniques is the rule set formed by C5 when applied to the Haberman data set. In this case, the rule set generated contains only one all-inclusive rule for one class ('IF anything THEN survival'), which fails to represent any interesting patterns within the data. In spite of the lower test accuracy of the rule set generated by RULES-5, it is more likely that this rule set will contain information about existing patterns in the data set.

Thus, if noise or the number of rules is an issue and some accuracy could be sacrificed, a pruning method

Table 2 Test results

	C5 results			Rules-3 Plus results				RULES-5 results			
Data set name	Rules	Training (%)	Test (%)	Rules	Training (%)	Test (%)	Time	Rules	Training (%)	Test (%)	Time
balance_scale	34	90.37	79.89	216	100.00	82.54	10 s	93	100	86.24	4 s
breast_cancer	11	97.95	91.94	43	100.00	95.73	7 s	33	100	96.21	10 s
wdbc	10	97.98	95.35	53	100.00	97.67	25 s	35	100	95.35	22 s
wpbc	13	90.51	77.05	46	100.00	60.66	8 s	35	100	73.77	7 s
car	131	96.30	94.10	275	100.00	100.00	1 min 50 s	246	100	100.00	43 s
credit screening	43	90.46	79.22	148	98.76	81.73	1 min 35 s	117	100	87.02	30 s
cylinder-band	2	71.54	66.87	218	100.00	62.58	51 s	230	100	66.87	18 s
dermatology	8	98.42	96.46	48	100.00	95.58	41 s	43	100	94.69	5 s
diabetes	17	81.34	75.43	224	93.84	65.09	55 s	160	100	76.29	1 min 10 s
ecoli	16	17.32	15.24	83	95.24	77.14	12 s	53	100	80.00	7 s
glass	6	100.00	100.00	26	100.00	94.03	2 s	17	100	94.03	2 s
haberman	1	73.71	73.12	53	80.75	77.42	2 s	72	99.5305	72.04	5 s
iris	4	99.02	95.83	12	98.04	91.67	1 s	8	100	95.83	1 s
liver	11	80.68	62.32	86	81.64	55.80	8 s	61	100	60.87	6 s
tic-tac-toe	75	93.73	85.76	140	100.00	95.14	55 s	32	100	100.00	5 s

such as that described by Fürnkranz [17] could be adopted in RULES-5. However, these methods should provide users with a means to control the level of generalization and thus to avoid problems associated with overpruning. An example that highlights this problem is the rule set formed when C5 is applied to the Haberman data set.

7 CONCLUSION

This paper has presented RULES-5, a new inductive learning algorithm that employs a simple and efficient rule search mechanism. In comparison to other algorithms in the RULES family, RULES-5 generates fewer rules, requires less training time and produces more accurate rule sets. Additionally, RULES-5 employs a simple method to handle continuous attributes during the rule formation process, which does not require any data pre-processing. The test results obtained with RULES-5 have shown that the rule sets extracted are more accurate than those produced using its immediate predecessor RULES-3 Plus and the well-known divide-and-conquer algorithm C5. Future improvements to RULES-5 will include a bespoke pruning technique and an appropriate specialization heuristic.

ACKNOWLEDGEMENTS

This work was carried out within ESPRIT Project 29114 'INFOMAN' and the ERDF Project 'Innovation in Manufacturing Centre'. The authors would also like to thank Dr Z. Salem and Dr Z. Cai for their help in the literature survey and technical discussions.

REFERENCES

1 **Teti, R.** (Ed.) Intelligent computation in manufacturing engineering. In Proceedings of the 3rd CIRP International Seminar, Ischia, Italy, 2002.

2 **Pham, D. T.** and **Alcock, R. J.** *Smart Inspection Systems: Techniques and Applications of Intelligent Vision*, 2002 (Academic Press and Elsevier Science, New York and Oxford).

3 **Mitchell, T. M.** *Machine Learning*, 1997 (McGraw-Hill, New York).

4 **Quinlan, J. R.** Learning efficient classification procedures and their applications to chess-end-games. In *Machine Learning—An Artificial Intelligence Approach*, 1983, pp. 463–482 (Los Altos, California).

5 **Quinlan, J. R.** Induction of decision trees. In *Machine Learning*, 1986, Vol. 1, pp. 81–106 (Kluwer, Boston, Massachusetts).

6 **Quinlan, J. R**. *C4.5: Programs for Machine Learning*, 1993 (Morgan Kaufmann, San Mateo, California).

7 *Data Mining Tools See5 and C5* (Rulequest Research). Available from http://www.rulequest.com/see5-info.html (accessed 1 February 2003).

8 **Hunt, E. B., Marin, J.** and **Stone, P. J.** *Experiments in Induction*, 1966 (Academic Press, New York).

9 **Clark, P.** and **Niblett, T.** The CN2 induction algorithm. *Mach. Learning*, 1989, **3**, 261–284.

10 **Clark, P.** and **Boswell, R.** Rule induction with CN2: some recent improvements. In Proceedings of the 5th European Working Session on *Learning*, Porto, Portugal, 1991, pp. 151–163.

11 **Cohen, W. W.** Fast effective rule induction. In Proceedings of the 12th International Conference on *Machine Learning*, Tahoe City, California, 1995, pp. 115–123.

12 **Michalski, R. S.** On the quasi-minimal solution of the general covering problem. In Proceedings of the 5th International Symposium on *Information Processing* (*FCIP 69*), Bled, Yugoslavia, 1969, Vol. A3, pp. 125–128.

13 **Michalski, R. S.** Pattern recognition as rule-guided inductive inference. *IEEE Trans. Pattern Analysis and Mach. Intell.*, 1980, **2**, 349–361.

14 **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, Virginia, 2001; www.mli.gmu.edu/

15 **Pham, T.** and **Dimov, S. S.** An efficient algorithm for automatic knowledge acquisition. *Pattern Recognition*, 1996, **30**(7), 1137–1143.

16 **Pham, D. T.** and **Dimov, S. S.** An algorithm for incremental inductive learning. *Proc. Instn Mech. Engrs, Part B: J. Engineering Manufacture*, 1997, **211**(B3), 239–249.

17 **Fürnkranz, J.** Pruning algorithms for rule learning. *Mach. Learning*, 1996, **27**(2), 139–171.

18 **Fürnkranz, J.** Separate-and-conquer rule learning. *Artif. Intell. Rev.*, 1999, **13**(1), 3–54.

19 **Pham, D. T., Afify, A. A.** and **Dimov, S. S.** Machine learning in manufacturing. In Proceedings of the 3rd CIRP International Seminar on *Intelligent Computation in Manufacturing Engineering* (*ICME 2002*), Ischia, Italy, 2002, pp. III–XII.

20 **Kerber, R.** Chimerge: discretisation of numeric attributes. In Proceedings of the 10th National Conference on *Artificial Intelligence*, San Jose, California, 1992, pp. 123–128.

21 **Dougherty, J., Kohavi, R.** and **Sahami, M.** Supervised and unsupervised discretisation of continuous features. In Proceedings of the 12th International Conference on *Machine Learning*, Los Altos, California, 1995, pp. 194–202.

22 **Catlett, J.** On changing continuous attributes into ordered discrete attributes. In Proceedings of the European Working Session on *Learning*, Berlin, Germany, 1991, pp. 164–178.

23 **Fayyad, U. M.** and **Irani, K. B.** On the handling of continuous-valued attributes in decision tree generation. *Mach. Learning*, 1992, **8**, 87–102.

24 **Fayyad, U. M.** and **Irani, K. B.** Multi-interval discretisation of continuous-value attributes for classification learning. In Proceedings of the 13th International Joint Conference on *Artificial Intelligence*, San Francisco, California, 1993, pp. 1022–1027.

25 **Quinlan, J. R.** Improved use of continuous attributes in C4.5. *J. Artif. Intell. Res.*, 1996, **4**, 77–90.

26 **Rissanen, J.** A universal prior for integers and estimation by minimum description length. *Ann. Statistics*, 1983, **11**, 416–431.

27 **Lee, C.** Generating classification rules from databases. In Proceedings of the 9th Conference on *Applications of AI in Engineering*, Malvern, Pennsylvania, 1994, pp. 205–212.

28 **Quinlan, J. R.** Unknown attribute values in induction. In Proceedings of the 6th International Workshop on *Machine Learning*, Los Altos, California, 1989, pp. 164–168.

29 *UCI Machine Learning Repository* (UCI). Available from http://www1.ics.uci.edu/~mlearn/MLRepository.html (accessed 1 February 2003).