Pergamon

Expert Systems With Applications, Vol. 8, No. 1, pp. 59–65, 1995

0957-4174/95 $9.50 + .00

RULES: A Simple Rule Extraction System

D. T. PHAM AND M. S. AKSOY

Intelligent Systems Research Laboratory, School of Electrical, Electronic and Systems Engineering, University of Wales College of Cardiff, Cardiff, UK

Abstract—*We present RULES, a simple inductive learning algorithm for extracting IF–THEN rules from a set of training examples. We also describe the application of RULES to a range of problems, each involving a different number of attributes, values, and classes. The results obtained demonstrate that in spite of its simplicity RULES is at least comparable to other inductive learning algorithms, if not more accurate and more general.*

1. INTRODUCTION

WHEREAS THE DEVELOPMENT of expert systems can be considerably facilitated through the use of tools that provide ready-made inferencing and user interface facilities, the acquisition of the expert domain knowledge to form the knowledge base often remains a bottleneck task. Thus, a major topic in expert systems research has been how to ease the knowledge acquisition process. From this research and from the area of machine learning in general, several inductive learning algorithms have been proposed for automating this process. These algorithms are all aimed at enabling a system to extract knowledge in a suitable format from a set of training examples. Notable inductive learning algorithms include CLS (Hunt, Maria, & Stone, 1966), ID3 (Quinlan, 1983) and its derivatives, for example, ID4 (Schlimmer & Fisher, 1986) and ID5 (Utgoff, 1988), CN2 (Clark and Boswell, 1991), BCT (Chan, 1989), and AQ11 (Michalski & Larson, 1978).

Perhaps the best known inductive algorithm to date has been ID3, due to the relatively simple way in which it can be implemented. ID3 essentially employs heuristic hill-climbing and non-backtracking search to find a decision tree for a given classification task. A problem with ID3 is that the decision tree produced might not be as general as it could be. As demonstrated later, this is caused by the fact that decision rules in ID3 can sometimes involve unnecessary or irrelevant conditions (Cheng, Fayyad, Irani, & Qian, 1988). Another problem with ID3 relates to the fact that for applications involving a large set of examples which cannot be kept in the memory of the computer at once, the algorithm can only examine a reduced set. This technique, called "windowing" (Quinlan, 1983), cannot be guaranteed to yield the same decision tree as would be obtained from the original complete set of examples, nor even to be able to generate a decision tree that would classify all examples correctly (Wirth & Catlett, 1988; O'Keefe, 1983).

We present a simple heuristic inductive algorithm called RULES (for RULe Extraction System). RULES produces IF–THEN rules from a set of training examples. The size of the set is only limited by the capacity of the auxiliary memory of the computer used. Also, as discussed later, the extracted rules contain only relevant conditions. We describe the application of RULES to a range of problems demonstrating the strong performance of the algorithm.

2. ALGORITHM DESCRIPTION

RULES is a simple algorithm for extracting a set of classification rules for a collection of objects belonging to a given set of classes. An object must be described in terms of a fixed set of attributes, each with its own set of possible values. For example "Weather" and "Temperature" might be attributes with sets of possible values {rainy, sunny, snowy} and {low, average, high}, respectively.

In RULES, an attribute–value pair constitutes a condition. If the number of attributes is n_a, a rule may contain between one and n_a conditions, each of which must be a different attribute–value pair. Only the conjunction of conditions is permitted in a rule, and therefore the attributes must all be different if the rule comprises more than one condition. The attributes and the values associated with them in a collection of objects form an array of attributes and values. The total number of elements of the array is the total number of all possible values. For example, if there are four attributes

Requests for reprints should be sent to D. T. Pham, Intelligent Systems Research Laboratory, School of Electrical, Electronic and Systems Engineering, University of Wales College of Cardiff, P.O. Box 917, Cardiff CF2 1XH, UK.

TABLE 1
Training Set for Season Classification Problem

Example	Weather	Trees	Temperature	Season (Class)
1	rainy	yellow	average	*autumn*
2	rainy	leafless	low	*winter*
3	snowy	leafless	low	*winter*
4	sunny	leafless	low	*winter*
5	rainy	leafless	average	*autumn*
6	rainy	green	high	*summer*
7	rainy	green	average	*spring*
8	sunny	green	average	*spring*
9	sunny	green	high	*summer*
10	sunny	yellow	average	*autumn*
11	snowy	green	low	*winter*

with 3, 4, 2, and 5 values, respectively, the total number of elements is 14.

The rule-forming procedure may require at most n_a iterations. The first iteration produces rules with one condition, and the second iteration results in rules with two conditions, etc. In the first iteration, each element of the array of attributes and values is examined to decide whether it can form a rule with that element as the condition. For the whole set of examples, if a given element applies to only one class, then it is a candidate for forming a rule. If it pertains to more than one class, it is passed over and the next element is examined. When all elements of the array have been looked at, the whole set of examples is checked for any example that cannot be classified by the candidate rules. If there are no unclassified examples, the procedure terminates. Otherwise, a new array is constructed that comprises attributes and values contained in all of the unclassified examples. In the second iteration, elements of the array are examined in pairs to determine whether they apply to only one class in the whole set of examples. As before, for those pairs of elements that pertain to unique classes, candidate rules are obtained. If there are still unclassified examples at the end of this iteration, a new array is formed and the next iteration is initiated. This procedure continues until all examples are correctly classified or the number of iterations (the number of conditions) is equal to n_a. In the latter case, all remaining unclassified examples are taken as rules. For each iteration after the first, candidate rules extracted in the current iteration are checked against previously obtained rules. Candidate rules that do not contain irrelevant conditions are added to the rule set, and the others are ignored. This check is not required for the first iteration, as each rule can only have one condition. The procedure is summarised in the flowchart shown in Figure 1.

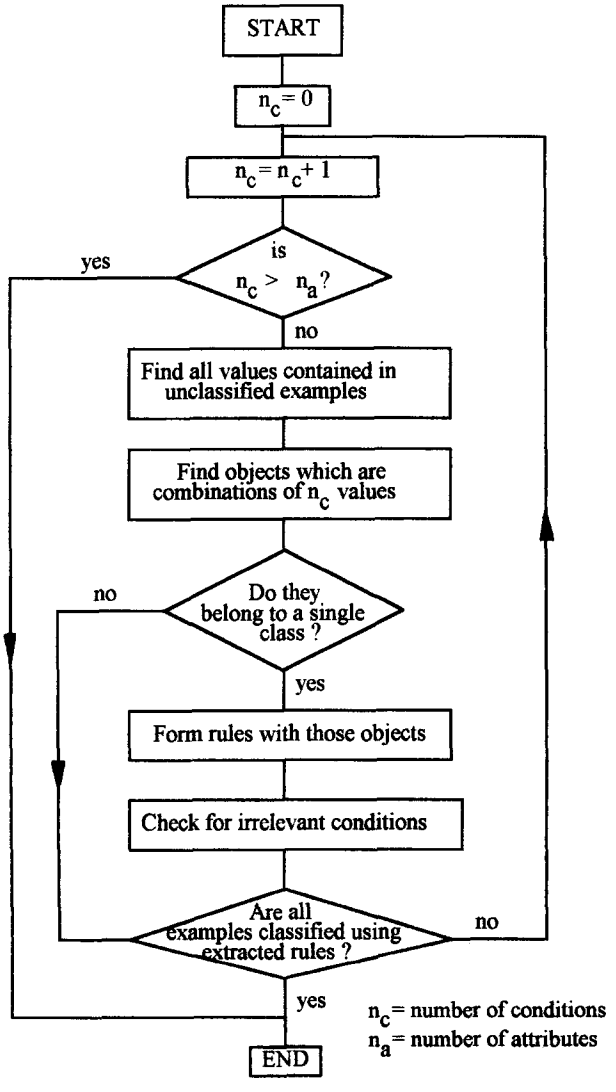

FIGURE 1. Flow chart of RULES.

As an illustration of the operation of RULES, consider the Season Classification Problem, the training

TABLE 2
Initial Array of Attributes and Values for Season Classification Problem

Attribute	Value
Weather	rainy, sunny, snowy
Trees	green, yellow, leafless
Temperature	low, average, high

TABLE 3
Array of Attributes and Values for Season Classification Problem in the Second Iteration

Attribute	Value
Weather	rainy, sunny
Trees	green, leafless
Temperature	average

TABLE 4
Human Identification Problem (Quinlan, 1983)

Training Set			
Height	Hair	Eyes	Class
short	blond	blue	+
tall	blond	brown	−
tall	red	blue	+
short	dark	blue	−
tall	dark	blue	−
tall	blond	blue	+
tall	dark	brown	−
short	blond	brown	−

Attribute	Relevant values
Height	short, tall
Hair	blond, red, dark
Eyes	blue, brown

set for which is given in Table 1. Each object in the training set is described in terms of the following attributes: Weather, with values {*rainy, sunny, snowy*}, Trees, with values {*green, yellow, leafless*}, and Temperature, with values {*low, average, high*}. Each object belongs to one of four classes, *winter, summer, autumn,* or *spring.* The array of attributes and values for the first iteration is shown in Table 2. In the first iteration, the four subcollections corresponding to {Weather: *snowy*}, {Trees: *yellow*}, {Temperature: *low*} and {Temperature: *high*} contain objects of only a single class. This means four rules can be extracted as shown below:

Rule 1.
IF Weather *IS snowy THEN* Class *IS winter*
Rule 2.
IF Trees *ARE yellow THEN* Class *IS autumn*
Rule 3.
IF Temperature *IS low THEN* Class *IS winter*
Rule 4.
IF Temperature *IS high THEN* Class *IS summer.*

Applying these four rules to the set of training examples enables examples 1, 2, 3, 4, 6, 9, 10, and 11 to be correctly classified. Three examples (5, 7, and 8) remain unclassified. These three examples contain the following attributes and values: {Weather: *rainy*}, {Weather: *sunny*}, {Trees: *leafless*}, {Trees: *green*}, and {Temperature: *average*}. The new array is shown in Table 3. The number of elements of the array is 5 and is smaller than that of the previous array, which is 9. As there are still unclassified examples, and the number of conditions for the rules to be extracted in the next iteration (2) is smaller than the number of attributes (3) the procedure continues. As mentioned above, only combinations of different attributes are allowed, therefore, the possible pairs of elements in this step are {Weather: *rainy,* Trees: *green*}, {Weather: *rainy,* Trees: *leafless*}, {Weather: *rainy,* Temperature:

TABLE 5
Golf Playing Problem (Quinlan, 1988)

Training Set				
Outlook	Temperature (F)	Humidity (%)	Windy	Decision
rainy	>69 ≤ 75	>80	true	*don't play*
rainy	≤69	≤80	true	*don't play*
overcast	>69 ≤ 75	>80	true	*play*
overcast	>75	≤80	false	*play*
rainy	>69 ≤ 75	≤80	false	*play*
overcast	≤69	≤80	true	*play*
sunny	>69 ≤ 75	≤80	true	*play*
sunny	>75	>80	true	*don't play*
sunny	>75	>80	false	*don't play*
overcast	>75	≤80	false	*play*
rainy	≤69	≤80	false	*play*
rainy	>69 ≤ 75	>80	false	*play*
sunny	>69 ≤ 75	>80	false	*don't play*
sunny	≤69	≤80	false	*play*

Attributes	Relevant values
Outlook	rainy, overcast, sunny
Temperature (F)	≤69, >69 ≤ 75, >75
Humidity (%)	>80, ≤80
Windy	true, false

TABLE 6
Cheese Advisor Problem (Kottai & Bahill, 1989)

Training Set

Course	Dessert	Savouriness	Consistency	Cheese (Class)
appetizer	no	mild	soft	*montrachet*
appetizer	no	flavourful	soft	*montrachet*
appetizer	no	pungent	soft	*gorgonzola*
appetizer	no	mild	firm	*stilton*
appetizer	no	flavourful	firm	*stilton*
appetizer	no	pungent	firm	*kasseri*
salad	no	mild	soft	*montrachet*
salad	no	flavourful	soft	*montrachet*
salad	no	pungent	soft	*gorgonzola*
salad	no	mild	firm	*stilton*
salad	no	flavourful	firm	*stilton*
salad	no	pungent	firm	*kasseri*
none	yes	mild	soft	*camembert*
none	yes	flavourful	soft	*camembert*
none	yes	pungent	soft	*tallegio*
none	yes	mild	firm	*italian fontina*
none	yes	flavourful	firm	*italian fontina*
none	yes	pungent	firm	*appenzeller*
appetizer	yes	flavourful	soft	*brie*
appetizer	yes	pungent	soft	*chevres*
appetizer	yes	mild	firm	*gouda*
appetizer	yes	flavourful	firm	*gouda*
appetizer	yes	pungent	firm	*asiage*
salad	yes	mild	soft	*brie*
salad	yes	flavourful	soft	*brie*
salad	yes	pungent	soft	*chevres*
salad	yes	mild	firm	*gouda*
salad	yes	flavourful	firm	*gouda*
salad	yes	pungent	firm	*edam*

Attribute	Relevant values
Course	appetizer, salad, none
Dessert	yes, no
Savouriness	mild, flavourful, pungent
Consistency	soft, medium, firm

average}, {Weather: *sunny,* Trees: *green*}, {Weather: *sunny,* Trees: *leafless*}, {Weather: *sunny,* Temperature: *average*}, {Trees: *green,* Temperature: *average*}, {Trees: *leafless,* Temperature: *average*}. In the given set of examples the subcollections corresponding to the pairs {Weather: *sunny,* Trees: *leafless*}, {Trees: *green,* Temperature: *average*}, and {Trees: *leafless,* Temperature: *average*} contain objects of only a single class. The rules that can be extracted in the second iteration are thus:

Rule 5.
IF Weather *IS sunny AND* Trees *ARE leafless THEN* Class *IS winter*
Rule 6.
IF Trees *ARE green AND* Temperature *IS average THEN* Class *IS spring*
Rule 7.
IF Trees *ARE leafless AND* Temperature *IS average THEN* Class *IS autumn.*

The new rules can be used to classify all remaining examples. Thus, there are no more unclassified examples, and the procedure ends, in spite of the number of conditions being smaller than the number of attributes.

3. RESULTS AND DISCUSSION

The rule-forming procedure just described works, provided that there are no objects belonging to different classes but having identical values for each attribute. Because with each new iteration a new array is constructed that normally has fewer elements than for the array in the previous iteration, not all of the possible combinations of attribute values have to be searched. Combined with the fact that the algorithm only involves the simple matching of attribute value pairs with the training set of examples rather than complex com-

TABLE 7
Flower Planning Guide Problem (Kottai & Bahill, 1989)

Training Set

Life Span	Colour	Season	Height (in)	Sun or Shade	Flower Type
perennial	red	spring	12	shade	Berginia
perennial	blue	spring	12	shade	Blueberry
perennial	red	summer	12	shade	Bleed-heart
perennial	any_colour	summer	6	part-shade	Viola
annual	white	spring	6	sun	Candytuft
perennial	violet	spring	6	part-shade	Ajuga
annual	violet	spring	24	part-shade	Honesty
perennial	any_colour	summer	30	sun	Delphinium
perennial	violet	summer	12	sun	Lavender
perennial	red	summer	30	any	Daylily
perennial	yellow	summer	30	any	Daylily
annual	white	summer	4	sun	Alyssum
perennial	yellow	summer	24	sun	Peony
perennial	red	summer	8	sun	Dianthus
annual	white	summer	6	sun	Eng-daisy
perennial	yellow	summer	28	sun	Yarrow
annual	yellow	summer	10	sun	Gazania
perennial	any_colour	summer	24	sun	Columbine
perennial	any_colour	summer	24	part-shade	Columbine
perennial	blue	summer	30	sun	Japan-iris
perennial	white	summer	30	sun	Japan-iris
perennial	blue	spring	10	shade	Lungwort
perennial	yellow	summer	30	sun	Bl-eyed-su
perennial	any_colour	autumn	16	sun	Hardy-aster
perennial	red	summer	18	part-shade	Cranesbill
perennial	blue	autumn	30	part-shade	Lobelia
perennial	red	autumn	20	part-shade	Lobelia
perennial	white	autumn	36	part-shade	Jap-anemon
perennial	red	summer	30	sun	Poppy

Attributes	Relevant values
Life Span	perennial, annual
Colour	red, blue, white, violet, yellow, any_colour
Season	spring, summer, autumn
Height (in)	4, 6, 8, 10, 12, 16, 18, 20, 24, 28, 30, 36
Sun or shade	shade, part-shade, sun, any

putations, this means that the training time can be very short. It can easily be derived that in the worst case, the maximum number of matching operations for the algorithm is:

$$\sum_{i=1}^{n_a} n_v!/(n_v - \mathrm{i})!\mathrm{i}!$$

n_v = total number of values.
n_a = number of attributes.

RULES has been implemented in Quick Basic Version 4.5 on a PC 486 50 MHz computer. Five different example problems (Tables 1, 4, 5, 6, and 7) have been used to highlight aspects of RULES concerning the training time, the relevance of conditions in the extracted rules, the need or otherwise for windowing, and the ability to classify unseen examples. Those problems involve different numbers of attributes, values, and classes, in order to show that RULES can be employed for a wide range of situations. Table 8 summarises the features of the different problems and the results obtained.

3.1. Training Time

In Table 8, it can be seen that the training times for all the problems considered are only in the order of seconds.

3.2. Windowing

Windowing was not necessary for any of the problems considered. Indeed, this technique is not required at all by RULES, as there is no need to keep all training

TABLE 8
Summary of the Features of the Problems Tested

Example	No. of Attributes	No. of Values	No. of Classes	No. of Examples	Training Time (s)	No. of Rules
Season	3	9	4	11	0.16	7
Human	3	7	2	8	0.11	4
Golf	4	10	2	14	0.39	13
Cheese	4	11	13	29	3.01	24
Flower	5	27	25	29	5.85	56

examples in the main memory (RAM) of the computer. The algorithm is written such that the rule set can be extracted from the whole set of examples by considering only one example at a time. Therefore, the size of the problem (example set) is only limited by the size of the auxiliary memory (for example, hard disk) of the computer.

3.3. Relevant Conditions

Because of the rule-checking procedure adopted, the rules obtained involve only relevant conditions. This can be easily verified for the set of rules produced for the Season Classification Problem. For comparison purposes, the rules extracted by ID3 for the same problem are:

Rule 1.
IF Temperature *IS low THEN* Class *IS winter*
Rule 2.
IF Temperature *IS high THEN* Class *IS summer*
Rule 3.
IF Temperature *IS average AND* Trees *ARE green THEN* Class *IS spring*
Rule 4.
IF Temperature *IS average AND* Trees *ARE yellow THEN* Class *IS autumn*
Rule 5.
IF Temperature *IS average AND* Trees *ARE leafless THEN* Class *IS autumn*

Clearly Rule 4 contains an unnecessary condition, as it is obvious from the set of training examples that when trees are yellow then the season must be autumn. As shown in Table 9, using ID3, irrelevant conditions occur in four of the five tested problems.

3.4. Ability to Classify Unseen Examples

Because RULES does not employ irrelevant conditions, the number of conditions in extracted rules is likely to be fewer than for other algorithms that suffer from this problem. This has been demonstrated for all of the problems tested (Table 9). When an unseen example is given, as the number of conditions is smaller, the probability of classifying the example is higher. For instance, in the Season Classification Problem, if an unseen object has the attribute–value pair {Trees: *yellow*} but has no value for Temperature, it will be classified correctly by RULES but not by ID3. In general, it is preferable to have fewer rules, but if their abilities to classify unseen examples are poor because they are too specific, the performance is low. Using the proposed algorithm, the number of rules may be greater than for other algorithms. However, a high level of performance is achieved because all rules are relevant.

TABLE 9
Irrelevant Conditions

Problem	Number of Attributes	Number of Used Attributes	Number of Rules	Number of Irrelevant Conditions	Algorithm
Season	3	3	5	0	RULES
	3	2	5	1	ID3
Human	3	2	4	0	RULES
	3	2	4	1	ID3
Golf	4	4	13	0	RULES
	4	3	5	0	ID3
Cheese	4	4	24	0	RULES
	4	4	23	8	ID3
Flower	5	5	56	0	RULES
	5	4	28	8	ID3

4. CONCLUSION

We have discussed RULES, a simple algorithm for symbolic inductive learning. RULES has been applied to different problems, and the results obtained have shown that in all cases the training time required is low and the generality or ability to classify new examples is good. The latter feature is due to the fact that RULES does not suffer from the irrelevant-condition problem. As RULES does not require all examples to be kept in the main memory of the computer at once, it is economical in memory space while not even needing to use windowing. Research is continuing to add two new features to RULES, which will widen its application potential. The first feature is the capacity for dealing with incomplete examples, that is examples in which the values of some attributes are unknown. The second feature is the automatic ability to handle attributes with numerical values without these values having to be converted manually into labels as was done for the Golf Playing Problem shown in Table 5.

Acknowledgements—We thank Sakarya University (Turkey) and the UK Committee of Vice Chancellors and Principals for the financial support given to M. S. Aksoy.

REFERENCES

Chan, P.K. (1989). "Inductive Learning With BCT", Proc. Sixth International Workshop on Machine Learning. Cornell University Ithaca, New York, pp. 104–108.

Cheng, J., Fayyad, U. M., Irani, K. B., Qian, Z. (1988). Improved decision tree: A generalised version of ID3. *Proceedings of the Fifth International Conference on Machine Learning* (pp. 100–106). Ann Arbor, MI: University of Michigan.

Clark, P., & Boswell, R. (1991). Rule induction with CN2: Some recent improvements. In J. Siekmann (Ed.). Lecture Notes in Artificial Intelligence (pp. 151–163). Berlin: Springer-Verlag.

Hunt, E. B., Marin, J., & Stone, P. J. (1966). Experiments in induction. New York, London: Academic Press.

Kottai, R. M., & Bahill, A. T. (1989). Expert systems made with neural networks, *International Journal of Neural Networks*, **1**, 211–226.

Michalski, R. S., & Larson, J. B. (1978). Selection of most representative training examples and incremental generation of VL1 hypothesis: The underlying methodology and the descriptions of programs ESEL and AQ11 (Report No. 867). Urbana, Illinois: Department of Computer Science, University of Illinois.

O'Keefe, R. A. (1983). Concept formation from very large training sets (pp. 479–481). *Proceedings of the Eighth International Joint Conference on Artificial Intelligence*. Karlsruhe, West Germany: Morgan Kaufmann.

Quinlan, J. R. (1983). Learning efficient classification procedures and their application to chess end games. In R. S. Michalski, J. G. Carbonell, & T. M. Mitchell, *Machine learning, an artificial intelligence approach* (pp. 463–482). Palo Alto, CA: Tioga.

Quinlan, J. R. (1988). Induction, knowledge and expert systems. In J. S. Gero & R. Stanton, *Artificial intelligence developments and applications* (pp. 253–271). Amsterdam: Elsevier (North-Holland).

Schlimmer, J. C. & Fisher, D. (1986). A case study of incremental concept induction. *Proceedings of the Fifth National Conference on Artificial Intelligence* (pp. 496–501). Philadelphia, PA: Morgan Kaufmann.

Utgoff, P. E. (1988). ID5: An incremental ID3. *Proceedings of the Fifth International Conference on Machine Learning* (pp. 107–120). Ann Arbor, MI: University of Michigan.

Wirth, J., & Catlett, J. (1988). Experiments on the costs and benefits of windowing in ID3. *Proceedings of the Fifth International Conference on Machine Learning* (pp. 87–99). Ann Arbor, MI: University of Michigan.